ESCUELA POLITECNICA NACIONAL

ESCUELA DE INGENIERIA

PAQUETE DE SOFTWARE DIDÁCTICO PARA LA ENSEÑANZA- APRENDIZAJE DE MÉTODOS NUMÉRICOS

PROYECTO PREVIO A LA OBTENCIÓN DEL TITULO DE INGENIERO EN ELECTRONICA Y TELECOMUNICACIONES

MILTON NAPOLEON TIPÁN SIMBAÑA

DIRECTOR: Msc. MIGUEL HINOJOSA

Quito, Noviembre del 2003

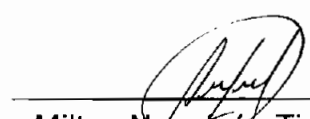
DEDICATORIA

El presente trabajo esta dedicado a mis padres quienes con su esfuerzo y apoyo han logrado que culmine mis estudios con éxito. A mis hermanas, hermano y sobrinos por su apoyo incondiconal que siempre me han brindado.

DECLARACIÓN

Yo Milton Napoleón Tipán Simbaña, declaro que el presente trabajo aquí descrito es de mi autoría; que no ha sido previamente presentada para ningún grado o calificación profesional; y, que he consultado las referencias bibliográficas que se incluyen en este documento.

La Escuela Politécnica Nacional, puede hacer uso de los derechos correspondientes a este trabajo, según lo establecido por la Ley de Propiedad Intelectual, por su Reglamento y por la normatividad institucional vigente



Milton Napoleón Tipán Simbaña

RESUMEN

El presente Proyecto de Titulación contiene el desarrollo de un Paquete de Software Didáctico para la Enseñanza –Aprendizaje de Métodos Numéricos (P.S.D.E.A.M.N), el mismo que fue desarrollado utilizando una metodología orientada a objetos iterativa.

Este Proyecto consta de cinco capítulos los cuales nos llevan por todo el proceso de desarrollo de software, los mismos que son:

1. PLANEACIÒN Y ELABORACIÓN
2. ANÁLISIS
3. DISEÑO
4. CONSTRUCCIÓN
5. PRUEBAS

El capítulo 1 abarca lo referente a la adquisición de los requerimientos que debe cumplir el software, el cual es luego plasmado en el documento de Requerimientos del Paquete de Software Didáctico para la Enseñanza-Aprendizaje de Métodos Numéricos.

En el capítulo 2 se investiga sobre el problema, sobre los conceptos relacionados que surgen al identificar los casos de uso para el sistema, lo que permite la construcción del diagrama de casos de uso y los diagramas de secuencia, que dan una visión de los diferentes aspectos relacionados con el software, culminado con la construcción de los contratos de operación que nos indican el comportamiento esperado del sistema.

El capítulo 3 describe el diseño del conjunto de objetos relacionados (clusters) y sus interfaces. Se define la arquitectura del sistema, se diseña la base de datos y se elaboran los diagramas de diseño de clase.

El capítulo 4 se refiere a la implementación del software, eligiendo previamente una herramienta de desarrollo adecuada en el caso del Paquete de Software Didáctico para la Enseñanza-Aprendizaje de Métodos Numéricos se ha optado por Visual Basic 6.0 que nos brinda todas las facilidades para la implementación del software.

En el capítulo 5 se desarrollan las pruebas del software, las mismas que tienden a verificar si es correcto lo que se ha implementado, usando para tal efecto las pruebas de unidad y de sistema.

El desarrollo del Paquete de Software Didáctico para la Enseñanza-Aprendizaje de Métodos Numéricos permitirá, tanto para los alumnos que tomen esta materia, como para los profesores que la dicta en las diferentes carreras de la Escuela Politécnica Nacional, ser una herramienta que facilite el aprendizaje.

PRESENTACIÓN

Los métodos numéricos son técnicas mediante las cuales es posible formular problemas de tal forma que puedan resolverse usando operaciones aritméticas. Los métodos numéricos llevan a cabo un buen número de tediosos cálculos aritméticos, por lo que hacerlos a mano llevaría mucho tiempo. Con el desarrollo de lenguajes de alto nivel existentes hoy en día se puede programar a la computadora, para servirnos de ésto y realizar cálculos que de otra manera serían imposibles de resolver analíticamente o manualmente, usar la computadora para obtener soluciones directamente se puede aproximar los cálculos sin tener que recurrir a suposiciones o simplificaciones o técnicas deficientes.

Los métodos numéricos son herramientas extremadamente poderosas para la solución de problemas, son capaces de manejar sistemas de ecuaciones grandes, no linealidades y geometrias complicadas comunes en la práctica de ingeniería, con el uso de una herramienta computacional que nos permita realizar estos cálculos se dispondría de más tiempo para aprovechar las habilidades creativas personales y así dar más importancia a la formulación de un problema, a la interpretación de la solución y a su incorporación al sistema total.

Esta herramienta computacional es la que se desarrolla en este Proyecto de Titulación a la que se la denominó Paquete de Software Didáctico para la Enseñanza-Aprendizaje de Métodos Numéricos, la misma que contiene cuatro módulos, un módulo de contenido teórico, un módulo de graficación, un módulo de algoritmos y un módulo de evaluación, tratando con ello de reproducir el modelo pedagógico seguido para la enseñanza de esta materia

Esta herramienta desarrollada no pretende ser en ningún sentido un sustituto del profesor que dicta esta materia, al contrario el objetivo es el de ser una ayuda para los alumnos que toman esta materia o para las personas que deseen aprender sobre la misma

CONTENIDO

CAPÍTULO 4: CONSTRUCCIÓN

CAPÍTULO 1

PLANEACIÓN Y ELABORACIÓN

1.1 SOFTWARE EDUCATIVO

1.1.1 CONCEPTOS GENERALES

1.1.1.1 Educación

"La educación es la formación y desarrollo de la personalidad del hombre en cuanto trata del desarrollo y las capacidades intelectuales, físicas, espirituales del individuo y fomenta en él, elevados sentimientos humanos gustos estéticos, convierte los principios ideológicos, políticos y morales en convicciones personales y hábitos de conducta diaria. Es decir forman un hombre **culto, libre, apto**, para vivir y participar activa y concientemente en la formación de una sociedad igualitaria"[10].

La educación busca formar a todo el pueblo en una concepción científica del mundo. El objetivo de la educación consiste en el desarrollo multifacético de la personalidad del hombre y la concepción científica del mundo.

1.1.1.2 Pedagogía

"Es el estudio de principios, normas, medios, métodos, técnicas, formas, procedimientos, población y medio ambiente que se encuentran involucrados en el proceso de enseñanza-aprendizaje" [10].

La misión de la pedagogía se concentra en la actividad del educador y del educando con sentido científico.

En organizar y dirigir en forma dinámica el conjunto complejo de actividades sistemáticas para una educación integral.

1.1.1.3 Didáctica

"La didáctica es la disciplina pedagógica de carácter práctico y normativo que tiene por objeto específico la técnica de la enseñanza, esto es la técnica de incentivar y orientar eficazmente a los alumnos en su aprendizaje.

Es el conjunto sistemático de principios, normas, recursos y procedimientos específicos que todo profesor debe conocer y saber aplicar para orientar con seguridad a sus alumnos en el aprendizaje de las materias de los programas, teniendo en vista sus objetivos educativos" [10].

La didáctica es la única, entre las ciencias pedagógicas que estudia la técnica de enseñar en todos sus aspectos prácticos y operativos, estableciendo la recta razón de la actuación educativa.

1.1.1.4 Proceso de Enseñanza-Aprendizaje

1.1.1.4.1 Enseñanza

1. Enseñar es la acción del profesor en relación a la dirección del aprendizaje.
2. Enseñar es impartir o transmitir conocimientos, instruir, producir o causar aprendizaje.
3. Enseñar a una persona significa introducir en ella algún cambio.

1.1.1.4.2 Aprendizaje

1. Una persona ha aprendido cuando ha modificado algún aspecto de su conducta.
2. Son los cambios permanentes de la conducta obtenidos como consecuencia de la práctica o la experiencia.
3. Aprender no significa solo retener en la memoria conocimientos, sino adquirir en y para la acción experiencias y, en general cierto nuevo modo de comportamiento en la vida, ello es, modificar en lo deseable la conducta del educando.

1.1.1.4.3 Método Didáctico

Es la organización racional y práctica de los recursos y los procedimientos del profesor, con el propósito de dirigir el aprendizaje de los alumnos, hacia resultados previsto y deseados. El método pretende hacer que los alumnos alcancen los objetivos propuestos lo mejor posible.

En el sentido estricto de la palabra solo existe un método que es el método científico el cual es aceptado y válido en todo nivel.

Desde el punto de vista pedagógico podemos clasificar a los métodos en dos, que son:

- Inductivo
- Deductivo

Existen otros llamados "métodos" como son: argumentación, experiencia directa, lectura, investigación, discusión, clases, experimental de aprendizaje holístico, entre otros. De los cuales el último es el que propone la última reforma curricular educativa.

1.1.1.4.3.1 Método Inductivo

Consiste en que a través de casos particulares o fenómenos aislados pretende que el alumno defina o establezca reglas, principios o leyes que han de regir a estos mismos fenómenos.

1.1.1.4.3.2 Método Deductivo

La deducción se presenta cuando partiendo de generalidades o leyes universales, llega el alumno a comprender que lo que es propio de un género determinado lo es de las especies que lo integran, que lo que es característico de un conjunto dado, lo es también de los fenómenos aislados que lo constituyen.

1.1.1.4.3.3 Modelo Experimental de Aprendizaje Holístico

"El Dr. David Kolb (y sus colaboradores Irwin Ruben y James McIntyre) del Instituto Tecnológico de Massachusetts, luego de profundas investigaciones en el área de enseñanza-aprendizaje, identificó cuatro maneras o estilos de cómo las personas aprenden: Unas lo hacen a través de métodos activos o experimentales y otras por métodos teóricos y reflexivos; los primeros incluyen la experiencia concreta y la experimentación activa y los segundos la observación-reflexión y la conceptualización abstracta. Constató también que cada individuo aprende más a través de uno de estos estilos, pero es susceptible de aprender, aunque en menor grado, con las otras maneras o estilos" [11].

El Dr. Kolb desarrolló su modelo metodológico que aplica a la enseñanza los cuatro modos o estilos de aprendizaje: de tal manera de que en un grupo, todos sus integrantes, aprenden sin rezagarse, ni aburrirse, por cuanto en algún momento de la clase, se está usando su estilo predominante de aprendizaje.
El ciclo de enseñanza-aprendizaje se inicia con una experiencia concreta en la cual el estudiante experimenta, vive, ve, escucha, siente, palpa, una situación: sobre la cual en el segundo paso observa y reflexiona buscando las causas y consecuencias, los sentimientos y reacciones que esa situación tiene y origina.

Como consecuencia de este proceso reflexivo el estudiante busca una explicación teórica y conceptual que puede ser producto de la reflexión. Todo esto quedaría en un nivel cognoscitivo, si no encuentra el estudiante la manera de aplicarlo prácticamente o de probar que la teoría funciona en la vida. Así el modelo completo se denomina Modelo Experiencial el mismo que se muestra la Fig.1.1

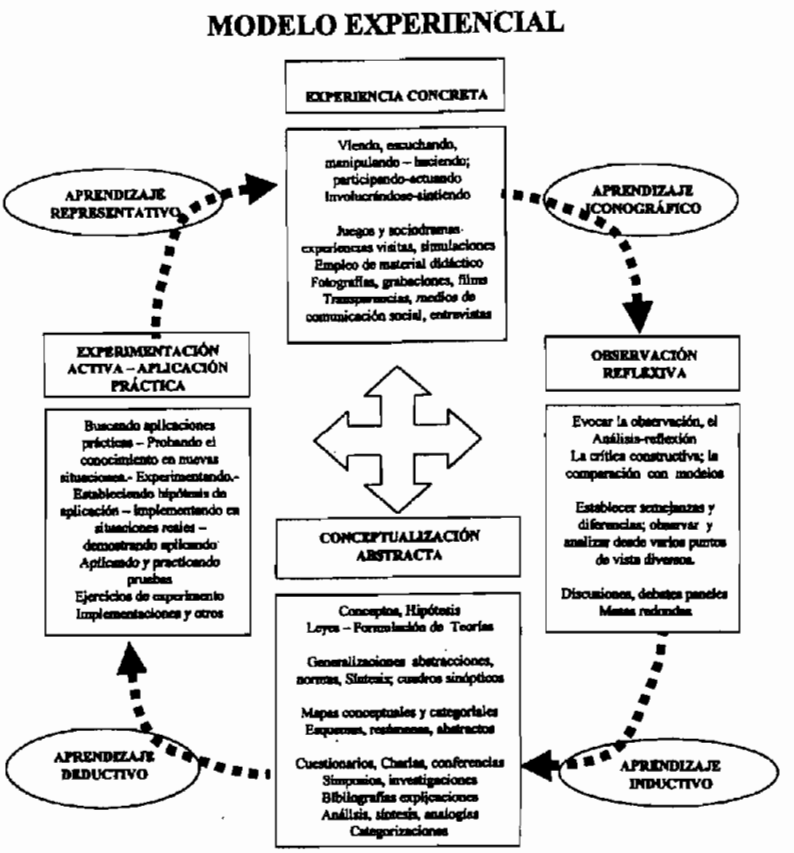


Fig. 1.1 Modelo Experiencial

La metodología de enseñanza-aprendizaje desarrollada por el Dr. Kolb ha mejorado el proceso educativo en las aulas escolares y universitarias, su influencia ha sido significativa en el área del comportamiento humano, por cuanto es el área donde la relación teórica-práctica se hace evidente y necesaria; pues de nada vale tener una teoría, si ésta no es debidamente reflexionada y explicada, para lograr un cambio de actitud, el fortalecimiento de otra o el desarrollo de una nueva conducta.

1.1.1.5 Técnicas didácticas

Las técnicas didácticas son maneras prefijadas de enseñar, que han sido comprobadas por la experiencia como eficientes para conducir el proceso de aprendizaje.

Las técnicas no son algo concreto como los materiales didácticos, en realidad son caminos de tipo intelectual que orientan al profesor sobre "Como enseñar" que le indican la ruta a seguir a través de fases o etapas y que propician una rápida obtención de los objetivos educativos propuestos.

1.1.1.5.1 Clasificación de las técnicas didácticas

a) Técnicas dinámicas para conducir el aprendizaje[11]

Exposición	Sociodrama o dramatización	Hoja de notas
Interrogatorio	Panel	Palabra clave
Demostración	Mapa conceptual	Crucigrama
Investigación bibliográfica	Mesa redonda	Acróstico
Investigación de campo	Discusión dirigida	Foro
Corrillo	Simposio	Trabajo de laboratorio
Philips 66	Móvil	Poder, querer y saber
Lluvia de ideas	Discusión libre	Rutas de aprendizaje
Palabra clave	Comisión	Collage
Arbolgrama	Debate	Taller pedagógico
Espina de pez	Ordenador de ideas	Juego atómico
Bingo	Mentefactos	6 ideas y 6 argumentos
Tarjetas de Cotejo	Rompecabezas didáctico	Lectura de párrafos

Tabla. 1.1 Técnicas dinámicas para conducir el aprendizaje

b) Técnicas de trabajo

Base	Expresión Escrita	Expresión Oral
Lectura	Apuntes	Cuchicheo
Subrayado	Esquemas	Saturno
	Resumen	
	Monografías	
	Diagrama de Estudios	
	Cuadro Sinóptico	
	Fichas	
	Informe	

Tabla. 1.2 Técnicas de Trabajo

c) Técnicas de integración grupal

Binas y Cuartas	Comunicación en uno y otro sentido	Sociometría
Caballos	Fiesta de Presentación	Teléfono Descompuesto
Construcción de una Torre	Rompecabezas	Tres Experiencias

Tabla. 1.3. Técnicas de integración grupal

d) Técnicas de investigación

Censo	Entrevista	Observación
Encuesta	Fichaje	Test

Tabla. 1.4 Técnicas de investigación

1.1.1.6 Evaluación del aprendizaje

La evaluación del aprendizaje es el proceso continuo que permite al maestro determinar el nivel en que cada alumno logra los objetivos de un nivel, curso, unidad o sección.
Esto es: detectar si se ha registrado los cambios de conducta en cualquiera de los dominios de la personalidad (Cognoscitivo, afectivo y psicomotor) por lo que el concepto de evaluación es más amplio que el de medición.

Por lo tanto:
La evaluación es un " proceso integral, permanente, sistemático y científico" (Art.290 Cap. XII, reglamento General de la Ley de Educación) inmerso en el proceso enseñanza- aprendizaje, como un elemento fundamental del mismo. Constituye un proceso por cuanto es algo sistematizado, con fases o etapas que se deben cumplir. Por lo tanto no es algo improvisado al contrario se trata de un acto intencional planificado por el maestro, como lo determina el Art. 295 del citado Reglamento.

1.1.1.6.1 Fases de la evaluación

La evaluación de acuerdo a los propósitos o momentos en los que se realice la evaluación puede ser:

- Evaluación Diagnóstica
- Evaluación Formativa
- Evaluación Sumativa

1.1.1.6.1.1 Evaluación Diagnóstica

Se realiza al iniciar un año lectivo, un semestre, una unidad, entre otros. Siempre antes de iniciar algún proceso pedagógico.
La evaluación inicial es la exploración diagnóstica que consiste en un conjunto de actividades de auscultación, mediante las cuales el maestro detectará el nivel de

aprendizaje anterior al proceso de los alumnos, lo que permitirá seleccionar los recursos necesarios para lograr los objetivos propuestos.

1.1.1.6.1.2 Evaluación Formativa

Se desarrolla durante el proceso de enseñanza-aprendizaje en forma continua y permanente.

Esta evaluación permite detectar logros, dificultades y obstáculos que se presentan en el proceso de aprendizaje, para ofrecer la recuperación y la ayuda necesaria antes de finalizar el proceso pedagógico y con ello asegurar la eficacia de la enseñanza y del aprendizaje.

1.1.1.6.1.3 Evaluación Sumativa

Esta se efectúa al finalizar el proceso de enseñanza aprendizaje. Esta evaluación permite conocer el aspecto global del proceso, en los campos cognoscitivo, afectivo y psicomotriz, dándonos información acerca del éxito o fracaso del proceso pedagógico al final de este.

1.1.1.6.2 Momentos evaluativos

La Reforma Curricular, para la evaluación establece tres momentos evaluativos con sus respectivos agentes o responsables de ejecución o participación [9]:

- Autoevaluación
- Coevaluación
- Heteroevaluación

1.1.1.6.2.1 Autoevaluación

Es imprescindible que el estudiante desarrolle comportamientos de autoevaluación de sus propias actuaciones. La autoevaluación consiste en la calificación de sí mismo por parte del alumno, por medio de este momento

evaluativo tanto el agente (alumno) de la evaluación como el objeto (temas, contenidos, informaciones, asignaturas) de la evaluación se unifican e identifican.

1.1.1.6.2.2 Coevaluación

El alumno no se encuentra aislado y solitario en el ambiente social y escolar; por lo tanto debe compartir sus deberes, responsabilidades y actividades en su nivel de realización, con el fin de recibir criterio, observaciones, reflexiones, recomendaciones y correcciones.

La coevaluación consiste en la evaluación mutua, conjunta de una actividad o grupo de trabajo que puede realizarse en pares para luego hacerlo en grupos pequeños.

El maestro genera trabajos de grupos, en equipos, sobre temas o actividades que deben ser previamente planificada, en las cuales cada uno valora lo que le parece más interesante de los otros.

1.1.1.6.2.3 Heteroevaluación

Es la evaluación tradicional y que siempre la realiza el maestro, consiste en que una persona (maestro) evalúa a otra (alumno) sobre su trabajo, actuación, entre otros. Esta práctica requiere del profesor una solvente preparación y conocimiento de las técnicas e instrumentos de evaluación para eliminar una serie de dificultades y problemas que frecuentemente suelen presentarse y luego derivan en un antagonismo a la disciplina de estudio, apatía al mismo docente y probablemente una desidia por los estudios.

1.1.1.6.3 Técnicas e instrumentos de evaluación

No puede seguir la vigencia del "examen" como el único medio o instrumento de recolección de información y validación de los aprendizajes. Se hace imprescindible ampliar y completar con otras técnicas e instrumentos para recoger e interpretar los datos procesales, que permitan valorar todo lo escencial en la formación armónica del estudiante.

La subjetividad es un punto importante a ser tomado en cuenta en el momento de la evaluación ya que se podrá cuestionar que la observación es eminentemente subjetiva, por lo que sus resultados no son confiables, sin embargo la influencia de la subjetividad se da tanto, o más, cuando el resultado se expresa con frases ya que la plasmación numérica no garantiza su objetividad.
Para eliminar o disminuir la subjetividad, existen formas o medios que ayudan a este objetivo y constituyen la utilización de instrumentos técnicamente elaborados e inteligentemente aplicados.

Cada técnica permite evaluar una o varias destrezas (cognitivas, afectivas, psicomotoras) en el estudiante por lo que el maestro deberá usarlas dependiendo de la materia o asignatura que éste dicte, ya que cada campo de la ciencia desarrolla distintas destreza en los alumnos.
En la Fig. 1.2 se presenta el campo o dominio del alumno que puede ser evaluado con la técnica más adecuada para el efecto [9].

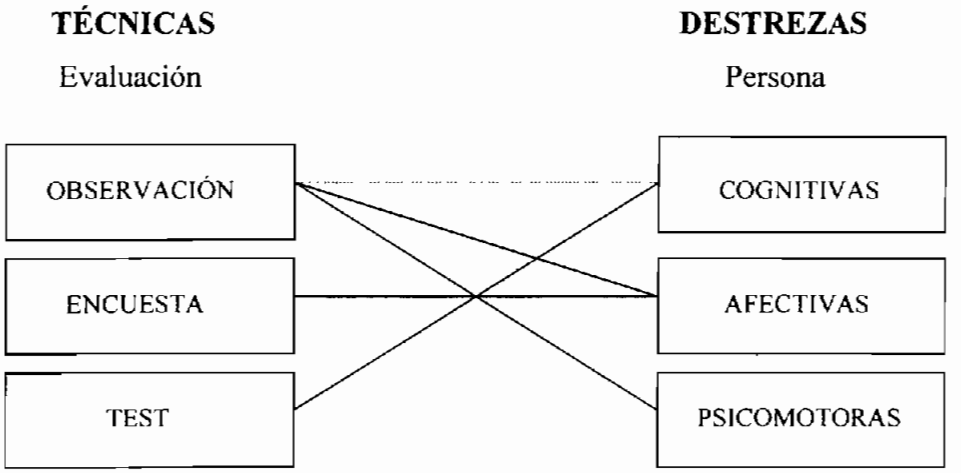


Fig. 1.2 Técnicas y destrezas

A continuación se presentan en la Fig. 1.3 las respectivas Técnicas e Instrumentos del Modelo Evaluativo [9].

Método: Inductivo-descriptivo

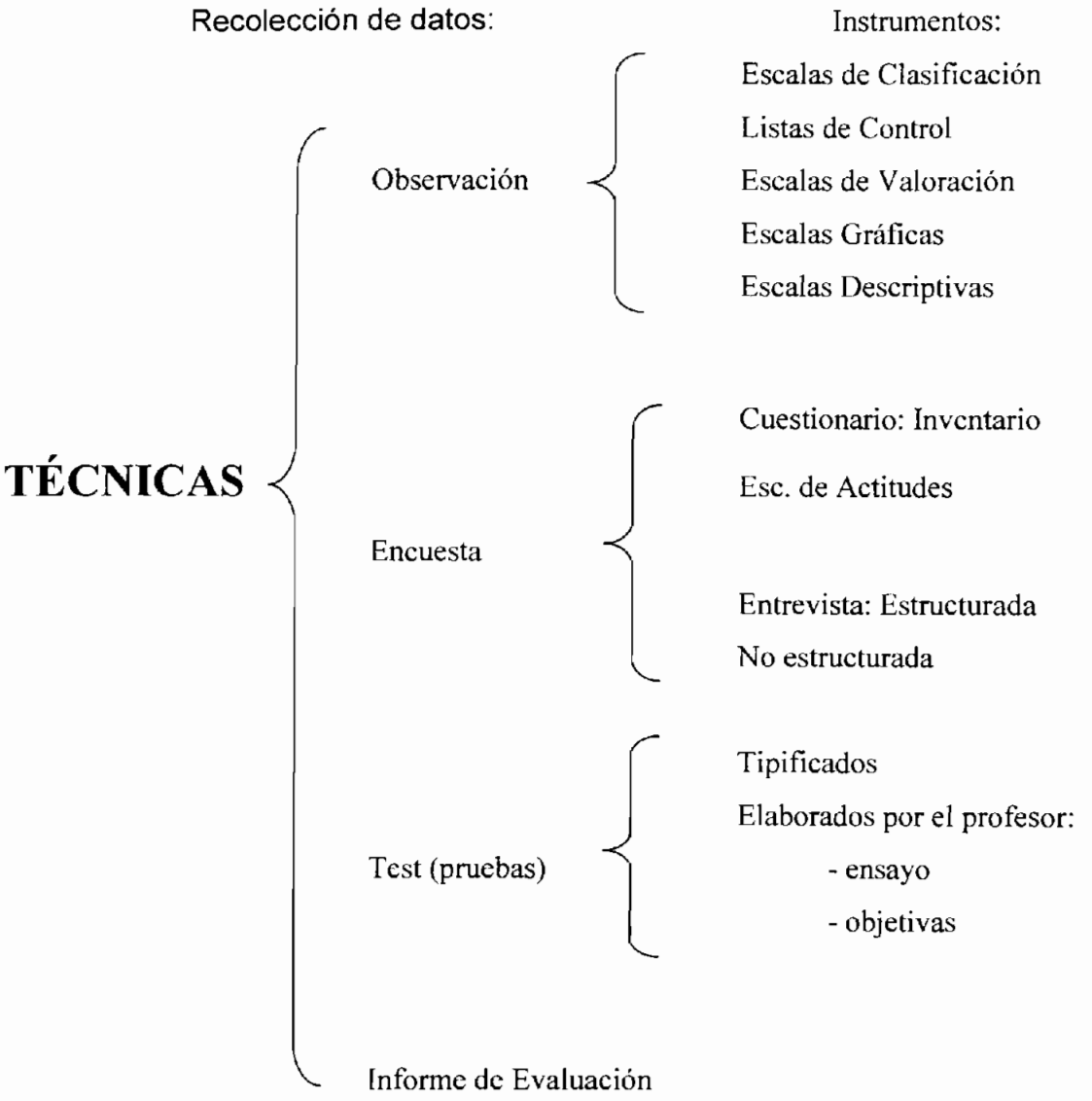


Fig. 1.3 Técnicas e Instrumentos del Modelo Educativo

1.1.2 La Educación y la Tecnología

Aprender es ante todo educarse, formar el propio ser. Y este es un proceso que se desarrolla de forma permanente a lo largo de nuestras vidas. La introducción de las tecnologías de la información y de la comunicación en los procesos de aprendizaje ha significado la creación de un nuevo espacio educativo, un espacio con nuevas reglas y que exige nuevos roles, pero, en definitiva, un espacio en el que es posible aprender. Las tecnologías construyen los marcos de aprendizaje, y las personas nos insertamos en ellos como tales, con nuestros sentimientos, emociones y objetivos por realizar.

La educación actual afronta múltiples retos. Uno de ellos es dar respuesta a los profundos cambios sociales, económicos y culturales que se prevén para la "sociedad de la información". Existe la convicción generalizada de que las instituciones tradicionales, de ladrillos y cemento, no serán suficientes para responder al desafío en materia de formación inicial y permanente inherente a "la sociedad de la información", por lo tanto es menester desarrollar nuevas modalidades educativas a lo largo del ciclo vital ajustadas a las necesidades y posibilidades de un público adulto que no puede desplazarse hasta los centros de formación por sus obligaciones familiares, laborales o personales. La formación debe flexibilizarse para acomodarse a necesidades crecientemente diversificadas y temporalmente críticas (formación a la carta "just in time", en el puesto de trabajo, etc.).

La disponibilidad generalizada de las nuevas tecnologías interactivas de la información y la comunicación abre una inmensa cantidad de posibilidades que se concretan en el desarrollo de nuevos modelos pedagógicos en la formación a distancia, tradicionalmente basada en la actividad del alumno sobre materiales impresos estandarizados. En los últimos años, la enseñanza abierta y a distancia ha despertado un considerable interés a todos los niveles, las nuevas tecnologías enriquecen la formación a distancia con la posibilidad no sólo de difundir

información de modo barato y eficiente, sino de dotar a los participantes (profesores, alumnos, expertos, etc.) de herramientas hardware/software para la comunicación personal y grupal que refuercen la acción tutorial y el aprendizaje colaborativo.

Durante estos años, la evolución del concepto de formación se ha visto influido por la aparición y consolidación de las nuevas tecnologías de la información, lo cual nos permite dibujar tres etapas claramente definidas: primero fue la informática y su utilización en las denominadas EAO (*Enseñanza Asistida por Ordenador*) donde se combinaban texto e imágenes estáticas conjuntamente con una interactividad limitada. Posteriormente fue la *multimedia* y el aumento de posibilidades como consecuencia de la incorporación de imágenes en movimiento y sonido, además de una aumento considerable de la interactividad. Por último y más recientemente el fenómeno *Internet* y su plasmación en la World Wide Web.

La aparición de Internet como medio de comunicación ha supuesto que el acceso a la información sea sencilla y rápida, además la aparición de un nuevo concepto: la educación en línea, es decir, los procesos educativos cuyo medio de comunicación fundamental son las redes informáticas. El concepto fundamental implícito en las últimas experiencias de educación en línea en internet es el de "aula virtual": un intento de implementar mediante aplicaciones telemáticas la calidad de la comunicación de la formación presencial en la educación a distancia. Las aulas virtuales son la manera de incorporar los efectos didácticos de las aulas reales a contextos en los que no es posible reunir físicamente a los participantes en un proceso de enseñanza/aprendizaje.

La mayor parte de esta información en internet reside en las conocidas páginas Web, que suelen presentar texto e imágenes en dos dimensiones. El mundo real es tridimensional, por lo que al reducir el "mundo" web a sólo dos dimensiones se está perdiendo información, de ahí la conveniencia de la integración de una tercera dimensión que permita, por ejemplo, recorrer las instalaciones de un museo o de una universidad hasta llegar a la información que interese al visitante.

Esto ya es una realidad que puede conseguirse a través de un lenguaje de modelado de realidad virtual como VRML (*Virtual Reality Modeling Language*).

La Realidad Virtual es una tecnología especialmente adecuada para la enseñanza, debido a su facilidad para captar la atención de los estudiantes mediante su inmersión en mundos virtuales relacionados con las diferentes ramas del saber, lo cual puede ayudar en el aprendizaje de los contenidos de cualquier materia.

Las aplicaciones de realidad virtual consiguen un efecto llamado "inmersión", según el cual "los estudiantes pueden interactuar completamente con el ambiente artificial utilizando los sentidos del tacto, el oído, y la vista mediante dispositivos especiales que están conectados al computador, tales como "guantes de datos" y pequeños monitores de vídeo dentro de un casco. Estos aparatos tienen sensores que detectan el movimiento de forma precisa, repercutiendo en el mundo virtual en el que los estudiantes están inmersos"

Las nuevas tecnologías de la información y la comunicación tienen una presencia consolidada en el campo de la educación. A diferencia de lo ocurrido con otros medios, como puede ser el caso del cine o de la televisión, cuya presencia efectiva en las aulas ha resultado en gran parte fallida, en la medida que estos medios no han resuelto problemas curriculares específicos; el caso de las denominadas nuevas tecnologías aplicadas a la educación es distinto. Fundamentalmente, porque la industria ha visto claro que el futuro está en habituar a los nuevos usuarios en estas tecnologías desde edades tempranas. Y esto se viene propiciando tanto en el ámbito escolar como en el familiar. Además, la investigación educativa, viene profundizando en la incidencia que herramientas como la multimedia pueden tener en los procesos de aprendizaje.

Uno de los campos en el que la aplicación de las nuevas herramientas multimedia ha avanzado más, es conocido por la expresión anglosajona *edutainment*. Que es una contracción de las palabras *education* y *entertainment*, que identifican aquellas propuestas con soporte informático que combinan educación y

entretenimiento. Las nuevas herramientas multimedia permiten el desarrollo de programas educativos multimedia denominados ***softwares educativos***, los mismos que son un recurso didáctico complementario que se debe usar adecuadamente en los momentos adecuados y dentro de un proyecto docente amplio.

Cada situación educativa concreta puede aconsejar, o desaconsejar, la utilización de determinados programas educativos multimedia como generadores de actividades de aprendizaje para los estudiantes y, por otra parte, un mismo programa puede convenir utilizarlo de manera distinta en contextos educativos diferentes.

Como norma general se puede decir que convendrá utilizar un determinado programa cuando su empleo aporte más ventajas que la aplicación de otros medios didácticos alternativos. Y en cuanto a la forma de utilización, nuevamente será la que proporcione más ventajas.

El **software educativo** llega a las entidades educativas usualmente por la vía comercial. Y ya sabemos lo que esto significa: hay siempre un vendedor como agente inmediato entre los "productos" y el comprador (la entidad educativa). Y los vendedores se dedican a vender y a nada más; sabemos que muchas veces desean vender "como sea", el asunto es vender. Pero los diseñadores de software suelen ser ingenieros de sistemas; evalúan los aspectos computacionales, algunos de los referidos a la relación hombre máquina y a la facilidad o funcionalidad de uso del programa y no evalúan la parte pedagógica que dicho producto debe tener, por lo que es necesario la participación de un evaluador educativo o pedagógico en estos procesos.

El hecho de que sean herramientas que indistintamente pueden utilizarse en el ámbito escolar o en el familiar, debe propiciar una mayor comunicación y coordinación entre padres y educadores. Por lo tanto, la permanente exigencia de actualización que es consustancial a la actividad profesional de los docentes,

también debe extrapolarse al papel de los padres. La educación es un trabajo de todos. Y todo trabajo debe ejercerse responsablemente.

1.2 PROCESO DE DESARROLLO DE SOFTWARE

Cuando se va a construir un sistema de software es necesario conocer un lenguaje de programación, pero con eso no basta. Si se quiere que el sistema sea robusto y mantenible es necesario que el problema sea analizado y la solución sea cuidadosamente diseñada. Se debe seguir un proceso robusto, que incluya las actividades principales. Si se sigue un proceso de desarrollo que se ocupa de plantear cómo se realiza el análisis y el diseño, y cómo se relacionan los productos de ambos, entonces la construcción de sistemas software va a poder ser planificable y repetible, y la probabilidad de obtener un sistema de mejor calidad al final del proceso aumenta considerablemente, especialmente cuando se trata de un equipo de desarrollo formado por varias personas.

1.2.1 CICLO DE VIDA DEL SOFTWARE

En la ingeniería informática el término de ciclo de vida viene citado intrínsecamente en la definición de la ingeniería del software que tiene como objeto el proceso de desarrollo de aplicaciones informáticas. Este proceso comprende la definición de requisitos, la especificación funcional, la descripción del diseño, la realización de los programas, los métodos de prueba, y el mantenimiento. Es decir, este proceso comprende el ciclo de vida.

Los métodos de desarrollo de software aceptados desde los años 70 son:

- Cascada o lineal (ciclo de vida clásico)
- Prototipeo
- Espiral
- Evolucionario o Iterativo

1.2.1.1 Método de desarrollo en cascada o lineal

El método de desarrollo en cascada (waterfall en inglés), involucraba a los usuarios sólo en la etapa de análisis y especificación de los requerimientos. Sólo al finalizar el proyecto, los usuarios podían revisar si el sistema entregado satisfacía sus requerimientos, generalmente no los cumplía. La Fig. 1.4

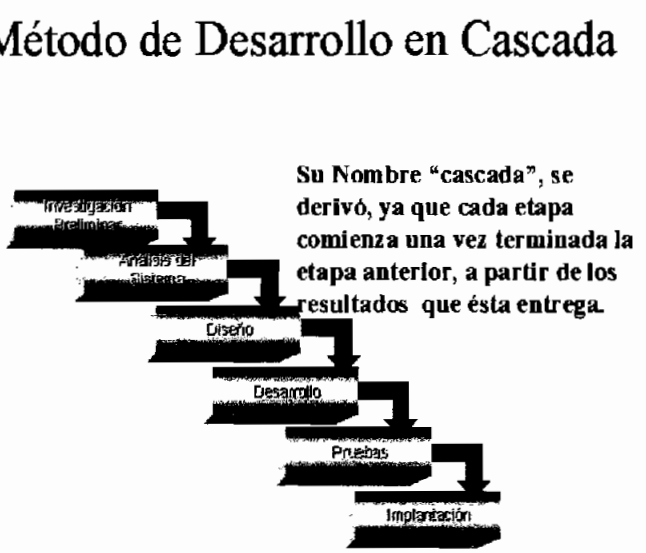


Fig. 1.4 Método de desarrollo en cascada

1.2.1.2 Método de desarrollo de Prototipeo

El método de desarrollo de prototipeo buscó involucrar mas a los usuarios, mediante su participación en la construcción previa de un prototipo del sistema final. Como un prototipo "soporta todo" se prometían una funcionalidad que luego no se podía implementar.

Los prototipos constituyen un nuevo concepto en el modelo de objetos, que es utilizado en lenguajes como Self u Omega en sustitución de las clases para la construcción de objetos. La estructura y comportamiento de un conjunto de

objetos no se describe mediante una clase, sino a través de un prototipo, que no es más que un objeto prefabricado, con una estructura, contenido y comportamiento predefinido, que se utiliza para crear nuevos objetos mediante un proceso de copia. En la Fig. 1.5 se muestra el proceso del prototipeo

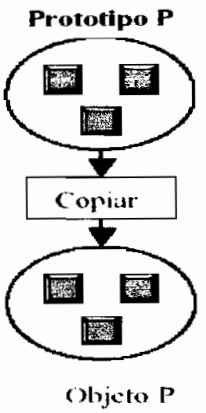


Fig. 1.5 Método de desarrollo de prototipeo

1.2.1.3 Método de desarrollo en espiral

La Fig. 1.5 presenta el esquema del método de desarrollo en espiral el mismo que incluyó lo mejor de los dos paradigmas anteriores e introdujo el concepto de evaluación del riesgo del proyecto, el cual incluía el riesgo de culminar en el tiempo programado, de no gastar mas de lo presupuestado y de lograr satisfacer los requerimientos de los usuarios. Mas no se entregaba nada concreto a los usuarios sino hasta la finalización del proyecto.

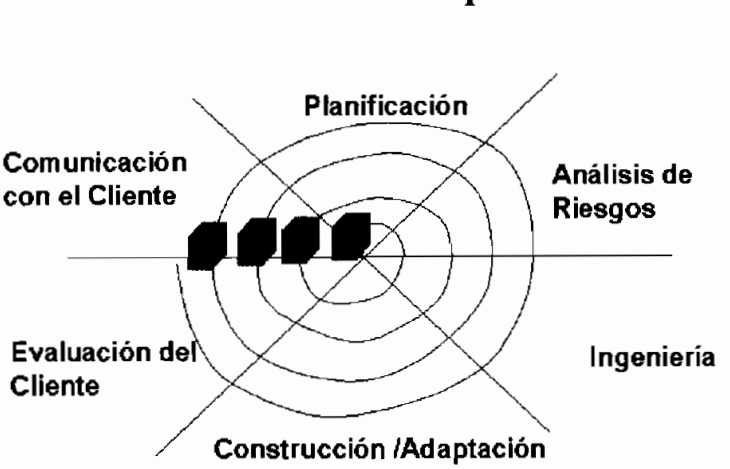


Fig. 1.5 Método de desarrollo en espiral

1.2.1.4 Método de desarrollo evolucionario, incremental o iterativo

El método de desarrollo evolucionable planteó que el desarrollo debía ser evolutivo en la medida que se entregaba algo concreto y ejecutable para los usuarios. Las fases de este método son mostradas en la Fig. 1.6 las cuales son: concepción o incepción, elaboración, construcción y transición. La concepción es definir el alcance del proyecto (Análisis). La elaboración es proyectar un plan, definir las características y cimentar la arquitectura (Diseño). La construcción es crear el producto (Código y Pruebas) y la transición es transferir el producto a sus usuarios.

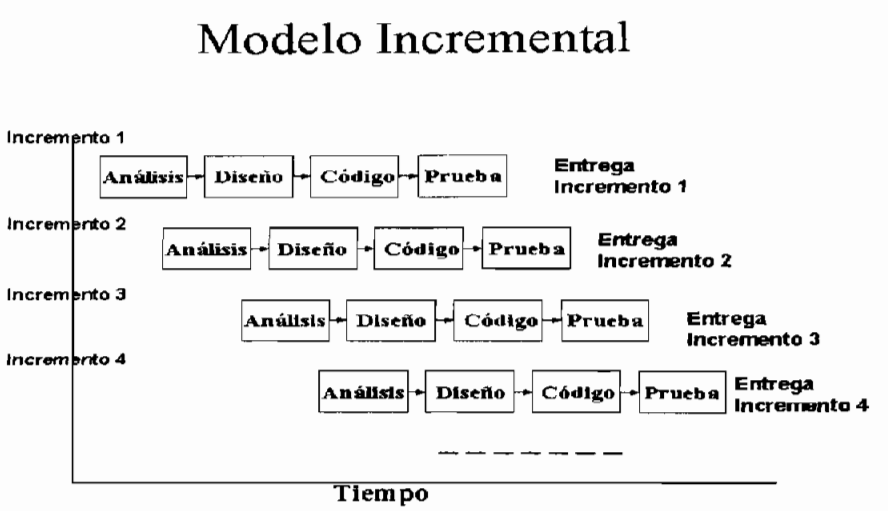


Fig. 1.6 Método de desarrollo evolucionable, incremental o iterativo

1.2.2 PARADIGMAS DE MODELAMIENTO

En la actualidad existen dos tipos de paradigmas que son:

- Paradigma de programación estructurada
- Paradigma orientado a objetos

En cualquier caso, el paradigma orientado a objetos ha sufrido una evolución similar al paradigma de Programación Estructurada (PE): primero se empezaron a utilizar los lenguajes de programación estructurados, que permiten la descomposición modular de los programas; esto condujo a la adopción de técnicas de diseño estructuradas y de ahí se pasó al análisis estructurado. El paradigma orientado a objetos ha seguido el mismo camino: el uso de la Programación Orientada a Objetos (POO) ha modificado las técnicas de diseño para adaptarlas a los nuevos lenguajes y ahora se están empezando a utilizar técnicas de análisis basadas en este nueva forma de desarrollar software.

1.2.2.1 Paradigma de programación estructurada

La idea básica de este paradigma es: **procesos + datos.**

Se basa en los conceptos separados de proceso (aspecto activo) y dato (aspecto pasivo) es decir que "los procesos actúan sobre los datos". El modelo del sistema se organiza en términos de funciones o procesos, presenta funcionalidad al entorno que coincide con la composición del sistema, la coordinación tiende a ser centralizada y externa a los procesos y los datos.

La Fig. 1.7 muestra como la descomposición del sistema se lleva a cabo, generalmente al descomponer los procesos en subprocesos. La agregación del sistema se da al agrupar estos subprocesos juntos porque constituyen un proceso de más alto nivel o más agregado.

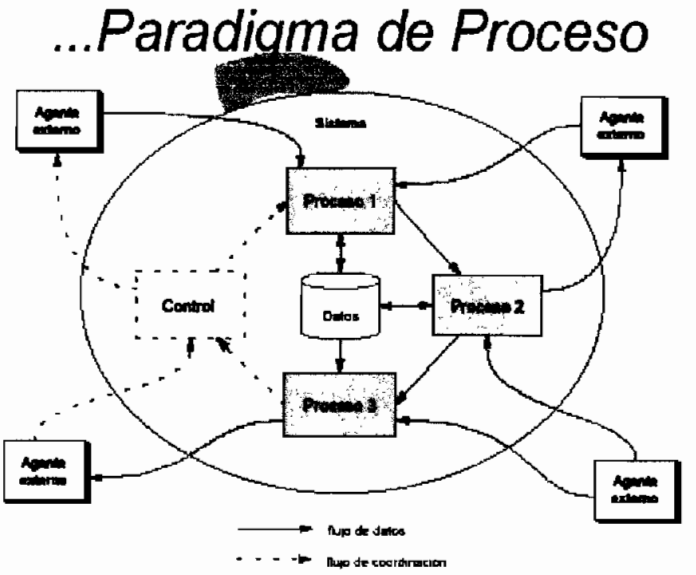


Fig. 1.7 Paradigma de Proceso

1.2.2.2 Paradigma orientado a objetos

La idea básica de este paradigma es: **objetos**

Se basa en el concepto de objeto que encapsula (agrupa) atributos (datos) y operaciones (procesos). El modelo del sistema se organiza en términos de clases y objetos, presenta funcionalidad que no coincide con la composición del sistema; la coordinación, atributos y operaciones son propios de los objetos. Descomposición del sistema puede darse indirectamente sobre la base de objetos que se organizan en:

- jerarquías de generalización/especialización
- estructuras todo/parte

La Fig. 1.8 muestra como la agregación del sistema se da al agrupar las operaciones juntas porque operan sobre el mismo conjunto de atributos, conformando así un objeto.

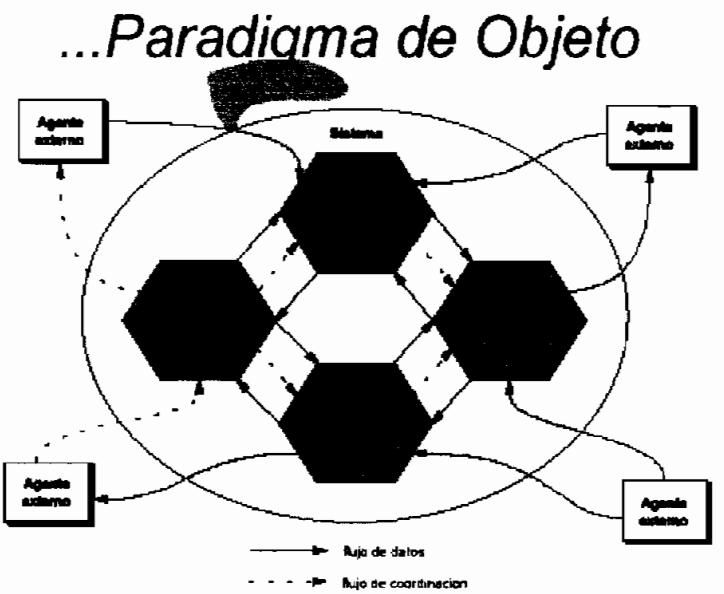


Fig. 1.8 Paradigma de Objeto

1.2.2.3 Comparación de los paradigmas de modelamiento

El paradigma orientado a objetos (OO) es más que una forma de programar. Es una forma de pensar acerca de un problema en términos del mundo real en vez de en términos de un ordenador. El análisis orientado a objetos (AOO) permite analizar mejor el dominio del problema, sin pensar en términos de implementar el sistema en un ordenador. El AOO permite pasar directamente el dominio del problema al modelo del sistema.

El concepto OO es más simple y está menos relacionado con la informática que el concepto de flujo de datos. Esto permite una mejor comunicación entre el analista y el experto en el dominio del problema (es decir, el cliente).

Los objetos encapsulan tanto atributos como operaciones. Debido a esto, el AOO reduce la distancia entre el punto de vista de los datos y el punto de vista del proceso, dejando menos lugar a inconsistencias o disparidades entre ambos modelos.

El paradigma OO utiliza la herencia para expresar explícitamente las características comunes de una serie de objetos. Estas características comunes quedan escondidas en el paradigma de programación estructurada y llevan a duplicar entidades en el análisis y código en los programas. Sin embargo, el paradigma OO pone especial énfasis en la reutilización, y proporciona mecanismos efectivos que permiten reutilizar aquello que es común, sin impedir por ello describir las diferencias.

Los cambios en los requisitos afectan notablemente a la funcionalidad de un sistema, por lo que afectan mucho al software desarrollado con métodos estructurados. Sin embargo, los cambios afectan en mucha menor medida a los objetos que componen o maneja el sistema, que son mucho más estables. Las modificaciones necesarias para adaptar una aplicación basada en objetos a un cambio de requisitos suelen estar mucho más localizadas.

Aparte de la reutilización interna, basada en la expresión explícita de características comunes, el paradigma OO desarrolla modelos mucho más próximos al mundo real, con lo que aumentan las posibilidades de reutilización. Es

probable que en futuras aplicaciones nos encontremos con objetos iguales o similares a los de la actual.

1.2.3 METODOLOGÍAS ORIENTADAS A OBJETOS

1.2.3.1 Conceptos generales

Las técnicas orientadas a objetos se basan en organizar el software como una colección de objetos discretos que incorporan tanto estructuras de datos como comportamiento. Esto contrasta con la programación convencional, en la que las estructuras de datos y el comportamiento estaban escasamente relacionadas.
Las características principales del enfoque orientado a objetos son:

- **Identidad.**

Los datos se organizan en entidades discretas y distinguibles llamadas objetos. Estos objetos pueden ser concretos o abstractos, pero cada objeto tiene su propia identidad. Dicho de otra forma: dos objetos son distintos incluso aún en el caso de que los valores de todos sus atributos (p. ej. nombre y tamaño) coincidan. Dos manzanas pueden ser totalmente idénticas pero no por eso pierden su identidad: nos podemos comer una u otra.

- **Clasificación.**

Los objetos que tengan los mismos atributos y comportamiento se agrupan en *clases*. Todas las manzanas tienen una serie de atributos comunes: tamaño, peso, grado de maduración, y un comportamiento común: podemos coger una manzana, moverla o comerla. Los valores de los atributos podrán ser distintos para cada una de ellas, pero todas comparten los mismos atributos y comportamiento (las operaciones que se pueden realizar sobre ellas). Una clase es una abstracción que describe propiedades (atributos y comportamiento) relevantes para una aplicación determinada, ignorando el resto. La elección de clases es arbitraria, y depende del dominio del problema.

Según esto, una clase es una abstracción de un conjunto posiblemente infinito de objetos individuales. Cada uno de estos objetos se dice que es una *instancia o ejemplar* de dicha clase. Cada instancia de una clase tiene sus propios valores para sus atributos, pero comparte el nombre de estos atributos y las operaciones con el resto de instancias de su clase.

- **Polimorfismo.**

El polimorfismo permite que una misma operación pueda llevarse a cabo de forma diferente en clases diferentes. Por ejemplo, la operación *mover*, es distinta para una pieza de ajedrez que para una ficha de parchís, pero ambos objetos pueden ser movidos. Una operación es una acción o transformación que realiza o padece un objeto. La implementación específica de una operación determinada en una clase determinada se denomina *método*.

Según lo dicho, una operación es una abstracción de un comportamiento similar (pero no idéntico) en diferentes clases de objetos. La semántica de la operación debe ser la misma para todas las clases. Sin embargo, cada método concreto seguirá unos pasos procedimentales específicos.

- **Herencia.**

El concepto de herencia se refiere a la compartición de atributos y operaciones basada en una relación jerárquica entre varias clases. Una clase puede definirse de forma general y luego refinarse en sucesivas subclases. Cada clase hereda todas las propiedades (atributos y operaciones) de su superclase y añade sus propiedades particulares.

La posibilidad de agrupar las propiedades comunes de una serie de clases en una superclase y heredar estas propiedades en cada una de las subclases es lo que permite reducir la repetición de código en el paradigma OO y es una de sus principales ventajas.

1.2.3.2 Comparación de las principales metodologías orientadas a objetos

En cualquier desarrollo de software la elección de la metodología a usar es una etapa importante que no conviene infravalorar. Una metodología inadecuada puede dar lugar a especificaciones poco precisas o completamente inútiles. Es por esto por lo que se hace necesario examinar algunas de las metodología disponibles para conocer hacia que tipo de software está enfocado y que herramientas proporciona al desarrollador.

Hay varias metodologías existentes; entre las más populares se incluyen las siguientes:

- *Catalysis*: Un método orientado a objetos que fusiona mucho del trabajo reciente en métodos orientados a objetos, y además ofrece técnicas específicas para modelar componentes distribuidos.
- *Objetory*: Un método de Caso de Uso guiado para el desarrollo, creado por Ivar Jacobson.
- *Shlaer/Mellor*: El método para diseñar sistemas de tiempo real, puesto en marcha por Sally Shlaer y Steven Mellor en dos libros de 1991, Ciclos de vida de Objetos, modelando el Mundo en Estados y Ciclos de vida de Objetos, Modelando el mundo en Datos (Prentice Hall). Shlaer/Mellor continúan actualizando su método continuamente (la actualización más reciente es el OOA96 report), y recientemente publicaron una guía sobre cómo usar la notación UML con Shlaer/Mellor.
- *Fusion*: Desarrollado en Hewlett Packard a mediados de los noventa como primer intento de un método de diseño orientado a objetos estándar. Combina OMT y Booch con tarjetas CRC y métodos formales.
- *OMT*: La Técnica de Modelado de Objetos fue desarrollada por James Rumbaugh y otros, y publicada en el libro de gran influencia "Diseño y Modelado Orientado a Objetos" (Prentice Hall, 1991). Un método que propone análisis y diseño 'iterativo', más centrado en el lado del análisis.

- *Booch*: Parecido al OMT, y también muy popular, la primera y segunda edición de "Diseño Orientado a Objetos, con Aplicaciones" (Benjamin Cummings, 1991 y 1994), (Object-Oriented Design, With Applications), detallan un método ofreciendo también diseño y análisis 'iterativo', centrándose en el lado del diseño.
- Craig Larman: Usa un modelo de análisis y diseño iterativo ayudado de lenguaje modelador UML este modelo es presentado en el Libro de "Análisis y Diseño Orientado a Objetos con UML" (Prentice Hall, 1999).

De estas metodologías anteriormente enumeradas las últimas cuatro serán de las que se extraerán sus cualidades más representativas y se determinará cual es la más apropiada para el desarrollo este proyecto.

BOOCH

La Booch es una metodología de propósito general en la que se parte de que cada etapa no es un proceso aislado si no que ha de interactuar con sus siguientes y precedentes en una especie de bucle del que se sale cuando se esté satisfecho con el modelo conseguido. En un principio se tienen una serie de objetos y clases que forman el sistema, a continuación se construye el modelo de interfaz y se examinan las relaciones entre las clases lo que, a su vez, genera la adición de nuevas interfaces que generarán nuevas relaciones iterándose hasta llegar al estado de refinamiento deseado. El método Booch proporciona un conjunto de herramientas gráficas y notaciones que ayudan a representar visualmente los modelos definidos en las fases de análisis y diseño. Algunas de ellas son:

- ***Diagramas de clase***.

Se trata de una variación de los diagramas de entidad relación en los que se añaden nuevos tipos de relaciones como la herencia, instanciación y uso. Además permite agrupar las clases y relaciones en categorías para diagramas demasiado complejos.

- ***Diagramas de objeto***.

En este tipo de gráfico se muestran los objetos y sus relaciones de forma dinámica mostrando la forma en la que los objetos se pasan mensajes entre ellos. Así mismo, en esto diagramas es posible representar la visibilidad de los objetos siendo ésta la que determina que objetos se pueden comunicar con otros.

- ***Diagramas temporales***.

Muestran la secuencia temporal de creación y destrucción de objetos. Suelen ir acompañados de pseudocódigo en el que se explica el flujo de mensajes de control entre los objetos del sistema.

- ***Diagramas de transición de estados.***

Permiten definir como las instancias de las clases pasan de un estado a otro a causa de ciertos eventos y que acciones se desencadenan de esos cambios de estado.

- ***Diagramas de módulo y proceso***.

En Booch, en la fase de implementación, es posible representar mediante estos gráficos la parte física del sistema, es decir, podemos mostrar como se van a almacenar internamente las clases y objetos, relaciones entre módulos en tiempo de compilación, procesos, dispositivos y las comunicaciones entre ellos.

Ventajas

- Es una metodología de propósito general.
- Herramientas y notaciones comprensibles.
- Proporciona una gran cantidad de información sobre las semánticas de los objetos.

Inconvenientes

- Notaciones poco precisas.
- Poca ayuda para el desarrollador novel.
- Herramientas orientadas al texto más que a los gráficos.

OBJECT MODELING TECHNIQUE (OMT)

La OMT, al contrario que la Booch, divide el proceso de desarrollo en tres partes aisladas: análisis, diseño e implementación. A su vez cada una de estas fases consta de otras subtareas como son los modelos de objetos, dinámico y funcional del análisis y el de sistema y objetos en el diseño.

- ***Modelo de objetos***

En esta primera parte del análisis se forma una primera imagen del modelo de clases del sistema con sus atributos y las relaciones entre ellas, usando para ello un diagrama entidad relación modificado en el que además de las clases y sus relaciones se pueden representar también los métodos.

- ***Modelo dinámico.***

El modelo dinámico usa un grafo para representar el comportamiento dinámico de cada clase, es decir, el comportamiento de estas ante cada evento que se produce en el sistema. Un evento desencadenará un cambio de estado en la clase que se traducirá en una modificación de los atributos o relaciones de ésta.

- ***Modelo funcional***.

Muestra que es lo que el sistema hará mediante un diagrama de flujo de datos, sin entrar en la secuencia temporal en la que los procesos se ejecutan. El modelo funcional puede revelar nuevos objetos y métodos que se pueden incorporar en los dos modelos anteriores. Por eso se dice que el método OMT es iterativo.

- ***Diseño del sistema.***

Se centra en la parte física del sistema como la descomposición de éste en subsistemas, el tipo de entorno en el que se va a ejecutar, el manejo de recursos y el almacenamiento de datos.

- ***Diseño de los objetos***.

Determina que operaciones van a realizar los métodos y profundiza incluso, en cuales algoritmos se van a usar. Se escogen los distintos tipo de representación de datos y se subdivide todo en módulos que pasarán a formar parte de la implementación.

- ***Implementación.***

En esta fase se convierte finalmente el diseño de objetos en código.

Ventajas

- Proporciona una serie de pasos perfectamente definidos al desarrollador.
- Tratamiento especial de la herencia.
- Facilita el mantenimiento dada la gran cantidad de información que se genera en el análisis.

Inconvenientes

- Hay pocos métodos para encontrar inconsistencias en los modelos.
- Interacción de objetos no soportada explícitamente en ninguna herramienta gráfica.
- Al ser un análisis iterativo es difícil de saber cuando comenzar con el diseño.

MÉTODO FUSION

El método Fusión toma los mejores aspectos de cada uno de las metodologías anteriores. Por ejemplo de la OMT toma los modelos de objetos, de la Booch sus gráficos de visibilidad, incluso toma los cuadros de pre y postcondición de las metodologías estructuradas. Además, aunque se pueda aplicar a cualquier tipo de desarrollo, está especialmente diseñado para proporcionar técnicas y herramientas al analista de sistemas industriales y de producción. Las distintas tareas de las que consta, tanto en el análisis como en el diseño son:

- ***Modelo de objetos***.

Es similar al usado en la OMT.

- ***Modelo operacional***.

Es análogo al modelo funcional de la OMT, sin embargo se ha prescindido de los diagramas de flujo de datos debido a que en la práctica estos se han demostrado inapropiados y se han adoptado las especificaciones de precondición y postcondición, propias de los métodos formales, para especificar que es lo que el sistema hace.

- ***Modelo de ciclo de vida***.

Se usan notaciones textuales de gramáticas para representar la operaciones que se realizan durante la vida de una entidad.

- ***Gráficos de interacción de objetos.***

Muestra como los objetos se comunican entre ellos mediante el paso de mensajes.

- ***Gráficos de visibilidad.***

Determina como un objeto conoce la existencia de otro, y si es exclusiva o compartida.

- ***Descriptores de clase***

Se trata de un refinamiento de las definiciones anteriores de las clases. Ahora se detallan sus atributos, junto con los tipos de datos de que se tratan, y los métodos para manejar esos atributos.

- ***Gráficos de herencia.***

Mediante unos gráficos sencillos se muestra las especializaciones y generalizaciones de cada una de las clases que intervienen en el sistema.

Ventajas

- Proporciona una gran cantidad de herramientas para comprobar la validez y consistencia de los modelos realizados en el análisis y diseño.
- El proceso de modelado es sistemático están definidos todos sus pasos.
- Es posible modelar el flujo de control y la creación y destrucción dinámica de objetos.
- No sólo es apropiado para sistemas de gestión.

Inconvenientes

- A veces es un método excesivamente riguroso.
- No es un proceso iterativo.

MÉTODO DE CRAIG LARMAN

Craig Larman propone un modelo de desarrollo de software iterativo, ayudándose para el modelado de los diferentes diagramas de un lenguaje modelador denominado UML.

UML (*Unified Modeling Language*) es un lenguaje que permite modelar, construir y documentar los elementos que forman un sistema software orientado a objetos. Se ha convertido en el estándar de facto de la industria, debido a que ha sido concebido por los autores de los tres métodos más usados de orientación a objetos: Grady Booch, Ivar Jacobson y Jim Rumbaugh.

Estos autores fueron contratados por la empresa Rational Software Co. para crear una notación unificada en la que se basa la construcción de sus herramientas CASE. En el proceso de creación de UML han participado, no obstante, otras empresas de gran peso en la industria como: Microsoft, Hewlett-Packard, Oracle o IBM, así como grupos de analistas y desarrolladores.

Esta notación ha sido ampliamente aceptada debido al prestigio de sus creadores y debido a que incorpora las principales ventajas de cada uno de los métodos particulares en los que se basa: Booch, OMT y OOSE. UML ha puesto fin a las llamadas "guerras de métodos" que se han mantenido a lo largo de los 90, en las que los principales métodos sacaban nuevas versiones que incorporaban las técnicas de los demás. Con UML se fusiona la notación de estas técnicas para formar una herramienta compartida entre todos los ingenieros software que trabajan en el desarrollo orientado a objetos.

Las tres fases al nivel más alto que propone Craig Larman son las siguientes:

- ***Planificación y Especificación de Requisitos:***

Planificación, definición de requisitos, construcción de prototipos, etc.

- ***Construcción***:

La construcción del sistema. Las fases dentro de esta etapa son la siguientes:

- **Análisis**: Se analiza el problema a resolver desde la perspectiva de los usuarios y de las entidades externas que van a solicitar servicios al sistema.
- **Diseño**: El sistema se especifica en detalle, describiendo cómo va a funcionar internamente para satisfacer lo especificado en el análisis.
- **Implementación**: Se lleva lo especificado en el diseño a un lenguaje de programación.
- **Pruebas**: Se llevan a cabo una serie de pruebas para corroborar que el software funciona correctamente y que satisface lo especificado en la etapa de Planificación y Especificación de Requisitos.

- **Instalación**:

La puesta en marcha del sistema en el entorno previsto de uso.

Ventajas

- El proceso de modelado es sistemático.
- Es un modelo iterativo.
- Recopila lo más sobresaliente de las otras metodologías gracias a UML.

Inconvenientes

- La fase de construcción es la que consume la mayor parte del esfuerzo.
- El formato del documento de Especificación de Requisitos no está definido en UML.

Luego de la exposición realizada de cada uno de los métodos y observando las ventajas y desventajas que cada uno tiene se opta por elegir como metodología para el desarrollo del Paquete Didáctico para la Enseñanza-Aprendizaje de Métodos Numéricos la propuesta por Craig Larman.

Este proceso no fija una metodología estricta, sino que define una serie de actividades que pueden realizarse en cada fase, las cuales deben adaptarse

según las condiciones del proyecto que se esté llevando a cabo. Se ha escogido seguir este proceso debido a que aplica los últimos avances en Ingeniería del Software, y a que adopta un enfoque eminentemente práctico, portando soluciones a las principales dudas y/o problemas con los que se enfrenta el desarrollador. Su mayor aportación consiste en atar los cabos sueltos que anteriores métodos dejan.

Debido a que este método presenta tres fases de alto nivel (planificación y especificación de requisitos, construcción e instalación) se dividirán estas fases entre los capítulos del presente proyecto de titulación así: la planificación y especificación de requisitos se lo realizara en el presente capítulo, la fase de construcción que se subdivide en análisis, diseño, implementación y pruebas; cada una de estas etapas se realizarán en un capítulo por separado de tal manera que en el capítulo II se realizará la etapa de análisis, en el capítulo III la etapa de diseño, en el capítulo IV la etapa de implementación y finalmente en el capítulo V la etapa de pruebas conjuntamente con la fase de instalación.

1.3 FASE DE PLANIFICACIÓN Y ESPECIFICACIÓN DE REQUISITOS

Esta fase se corresponde con la especificación de requisitos tradicional ampliada con un borrador de modelo conceptual y con una definición de casos de uso de alto nivel.

Las actividades de esta fase son las siguientes:

1. Definir el Plan-Borrador.
2. Crear el Informe de Investigación Preliminar.
3. Definir los Requisitos.
4. Registrar Términos en el Glosario. (*continuado en posteriores fases*)
5. Implementar un Prototipo. (*opcional*)
6. Definir Casos de Uso (de alto nivel y esenciales).
7. Definir el Modelo Conceptual-Borrador. (*puede retrasarse hasta una fase posterior*)

8. Definir la Arquitectura del Sistema-Borrador. (*puede retrasarse hasta una fase posterior*)
9. Refinar el Plan.

El orden propuesto es el que parece más lógico, y en él los pasos 5 y 7 pueden estar en posiciones distintas. De todos modos, el orden no es estricto, lo normal es que las distintas actividades se solapen en el tiempo. Esto sucede también en las actividades de las fases de Análisis y de Diseño, que se verán más adelante.

En este capítulo las actividades anteriormente enumeradas que no se van a realizar son: la que corresponde a definir el plan-borrador ya que este se creó al presentar el Plan de Proyecto de Titulación ante la Comisión Académica para su aprobación.

No se van tampoco a definir el modelo conceptual-borrador ni la arquitectura del sistema ya que éstas se pueden retrasar hasta fases posteriores, tampoco se implementará un prototipo por ser éste opcional, ya que éste sirve para recabar información sobre los requisitos del software y luego es desechado, como sustituto se utilizarán otras técnica de adquisición de información para sustituir al prototipo.

.

1.3.1 INFORME DE INVESTIGACIÓN PRELIMINAR DEL PAQUETE DE SOFTWARE DIDÁCTICO PARA LA ENSEÑANZA-APRENDIZAJE DE MÉTODOS NUMÉRICOS.

El planteamiento preliminar realizado en el plan-borrador nos da una primera idea acerca del paquete de software a desarrollar, para clarificar ésta se realiza una investigación tomando en cuenta a las personas involucradas en este desarrollo, como resultado de esta investigación se plasma el siguiente documento.

INFORME DE INVESTIGACIÓN PRELIMINAR DEL PAQUETE DE SOFTWARE DIDÁCTICO PARA LA ENSEÑANZA-APRENDIZAJE DE MÉTODOS NUMÉRICOS.

Propósito y Alcance.

El informe de investigación preliminar del paquete de software didáctico para la enseñanza-aprendizaje de métodos numéricos se realiza con el fin de recabar información, la misma que nos permitirá situarnos en el dominio del problema.
Este informe contiene el resultado del análisis de la información obtenida tanto de documentos (libros, folletos, programas) de métodos y análisis numéricos, como de encuestas y entrevistas realizadas a profesionales expertos en el tema, los mismos que prestan sus servicios en diferentes carreras de la Escuela Politécnica Nacional.

Principales problemas identificados.

A continuación se detallan los problemas hallados en el transcurso de la investigación.

- Los temas que abarcan los programas de las materias de métodos numérico y análisis numérico tiene diferencias sustanciales y varían entre las carreras.
- No se cuenta con un documento que englobe la explicación de todos los algoritmos que abarcan los programas tanto de métodos numéricos como de análisis numérico.
- Existen libros en los que el análisis de los algoritmos es muy riguroso y otros en los que éste es muy pobre, y son muy pocos los que dan una visión práctica del uso de los algoritmos.
- El software que viene junto con los libros de métodos numéricos o análisis numérico solamente implementan a lo más un solo algoritmo de cada uno de los temas que el texto abarca.
- El software de los libros se encuentra escrito en lenguajes de programación como Fortran, C, Basic, los cuales tiene interfaces de usuario muy pobres,

esto imposibilitan su manipulación en nuevas plataformas operativas como por ejemplo Windows en sus versiones 9X, XP, o NT con las cuales los estudiantes están familiarizados.

- El estudiante no cuenta con otro medio para evaluar su conocimiento que no sea las pruebas o exámenes realizados por los profesores.

Planteamiento de requerimientos adicionales de los usuarios.

Los requerimientos adicionales que el paquete de software debería cumplir de acuerdo a los expertos se detallan a continuación.

Módulo de contenido teórico

1. Deberá contener los siguientes temas:

- Conceptos básicos.
- Raíces de funciones no lineales.
- Manipulación de polinomios.
- Sistemas de ecuaciones lineales y no lineales.
- Regresión e interpolación polinomial.
- Diferenciación e integración numérica.
- Resolución de ecuaciones diferenciales ordinarias.
- Resolución de ecuaciones diferenciales parciales.

2. Debe permitir realizar búsquedas por temas.
3. El acceso a los diferentes tópicos no debe ser secuencial.
4. Su contenido no debe ser extenso.
5. Debe contener el fundamento matemático de los algoritmos.
6. En cada tópico se debe dar una descripción de las ventajas y desventajas respecto a otros algoritmos.

Módulo de algoritmos

1. Se implementarán los siguientes algoritmos:

- Raíces de funciones no lineales: Aproximaciones sucesivas, Bisección, Falsa posición, Primer orden, Newton-Raphson, Secante.

Limitaciones: para cualquier f(x), especificando condiciones de convergencia o limitando el número de iteraciones

- Manipulación de polinomios: Regla de Horner, Newton-Horner, Newton-Bairstow, Aplicación de Newton-Bairstow.

Limitaciones: polinomios de hasta décimo grado con opción a variar el límite
Evaluación de polinomios hasta la primera derivada

- Sistemas de ecuaciones lineales y no lineales: Eliminación gaussiana, Gauss-Jordan, Factorización (Doolittle, Cholesky), Jacobi, Gauss-Seidel, para S.E.L. Jacobi, Gauss-Seidel, Newton para S.E.N.L.

Limitaciones: Hasta sistemas de 20 ecuaciones.

- Regresión e interpolación polinomial: Regresión polinomial para regresión, Técnica matricial de Vandermonde, Polinomio de Interpolación de Lagrange, Polinomio de interpolación de Newton, Interpolación Trigonométrica, Interpolación segmentaria (Spline).

Limitaciones: Ingreso de 50 pares de datos con opción a variar dicho límite, polinomios de hasta quinto orden.

- Diferenciación e integración numérica: Fórmulas de diferenciación (primera y segunda derivada) para diferenciación, Trapecio, Punto medio, Simpson, Gaussiana, para integración.

Limitaciones: primera y segunda derivada, compuesta del trapecio, 3/8 de Simpson, Gaussiana hasta 8 nodos, limitando el número de iteraciones.

- Resolución de ecuaciones diferenciales ordinarias: Euler, Euler modificado, Runge-Kutta, Tiro simple para E.D.O.

Limitantes: Ecuaciones de hasta tercer orden
Runge-Kutta para S.E.D.O.

Limitantes: Hasta tres ecuaciones de primer orden.

- Resolución de ecuaciones diferenciales parciales: elípticas, parabólicas, hiperbólicas.

Limitantes: Ecuación de ondas unidimencional, ecuación del calor unidimencional, Ecuación de Laplace, fronteras regulares rectangulares, condiciones de contorno mixtas, red de puntos en diferencias finitas de 10x10.

2. Debe contener información del algoritmo: tipos de datos que se ingresan y limitantes
3. Los algoritmos podrán ser usados, aún cuando no se hayan aprobado las evaluaciones de los tópicos a los que pertenecen.

Módulo de graficación

1. Permitirá graficar 3 funciones simultáneamente.
2. Se podrá cambiar las escalas de graficación.
3. Mostrará las coordenadas de cualquier punto de la función graficada con la ayuda del mouse.
4. Permitirá ampliación (zoom) de cualquier parte de la función graficada.

Módulo de evaluación

1. La evaluación se realizara por temas.
2. La evaluación parcial contendrá 20 preguntas.
3. El tiempo de evaluación parcial será de una hora.
4. Deberá tener una opción de evaluación total.
5. La evaluación total contendrá 30 preguntas.
6. El tiempo de evaluación total será de una hora y treinta minutos.
7. Las evaluaciones serán objetivas usando para ello selección múltiple.
8. Una vez completada la evaluación se dará una indicación de aprobado si por lo menos el 70% de respuestas son correctas.
9. No debe permitir regresar a la parte de teoría hasta que haya finalizado la evaluación.

En caso de que falten de contestar algunas preguntas se deben dar los mensajes adecuados para que el usuario complete contestando las preguntas que falten.

Planteamiento de suposiciones críticas

- Los lineamientos con respecto a la evaluación se mantengan
- Los programas tanto de métodos numéricos como de análisis numérico no tengan cambios sustanciales.

Recursos Requeridos

- 1 computadora
- Software para el desarrollo de programas en lenguaje de alto nivel,
- Documentación técnica del software.
- Documentación de métodos numéricos y análisis numérico
- Acceso a Internet

Recomendaciones

- Es necesario crear un documento que permita aprender los algoritmos de una manera fácil y rápida sin que el contenido sea muy extenso.
- Dar una visión de la utilidad práctica de los algoritmos que son objeto de estudio.
- Complementar los requerimientos del plan-borrador con los temas y algoritmos adicionales propuestos por los expertos.
- Utilizar como plataforma para el sistema operativo Windows.

1.3.2 REQUISITOS DEL PAQUETE DE SOFTWARE DIDÁCTICO PARA LA ENSEÑANZA-APRENDIZAJE DE MÉTODOS NUMÉRICOS.

Un requisito es una descripción de necesidades o aspiraciones respecto a un producto. El objetivo principal de la actividad de definición de requisitos consiste en identificar qué es lo que realmente se necesita. Esto se hace en un modo que sirva de comunicación entre el cliente y el equipo de desarrollo.

Es aconsejable que un documento de Especificación de Requisitos tenga los siguientes puntos:

- . Propósito.
- . Ámbito del Sistema, Usuarios.
- . Funciones del Sistema.
- . Atributos del Sistema.

El formato del documento de Especificación de Requisitos no está definido en UML por lo que es necesario recurrir a otro estándar que permita la realización del mismo sin que éste afecte al modelo de desarrollo de software que seguimos.

Por lo que seguiremos las recomendaciones que nos plantea Alan M. Davis en su libro titulado "Software Requirements Objects, Functións and States" en el cual se describen una serie de técnicas, las mismas que nos permiten realizar las diferentes actividades que nos propone este autor. Según Alan M. Davis hay dos tipos de actividades que ocurren durante la fase de requerimientos: análisis del problema y descripción del producto como se observa en la Fig. 1.9.

Durante el análisis del problema, el analista consume el tiempo: pensando, entrevistando a personas quienes tiene más conocimiento acerca del problema e identifica todos los principales conflictos que pueden surgir durante solución del problema. Al mismo tiempo existe una considerable expansión de información y conocimiento del problema la misma que debe ser organizada por el analista.

Durante la descripción del producto el analista prepara un documento que describe las funciones externas esperadas del producto a ser construido, el mismo que resolverá un problema ahora conocido. Es momento también de organizar ideas, resolver conflictos encontrados, eliminar inconsistencias y ambigüedades.

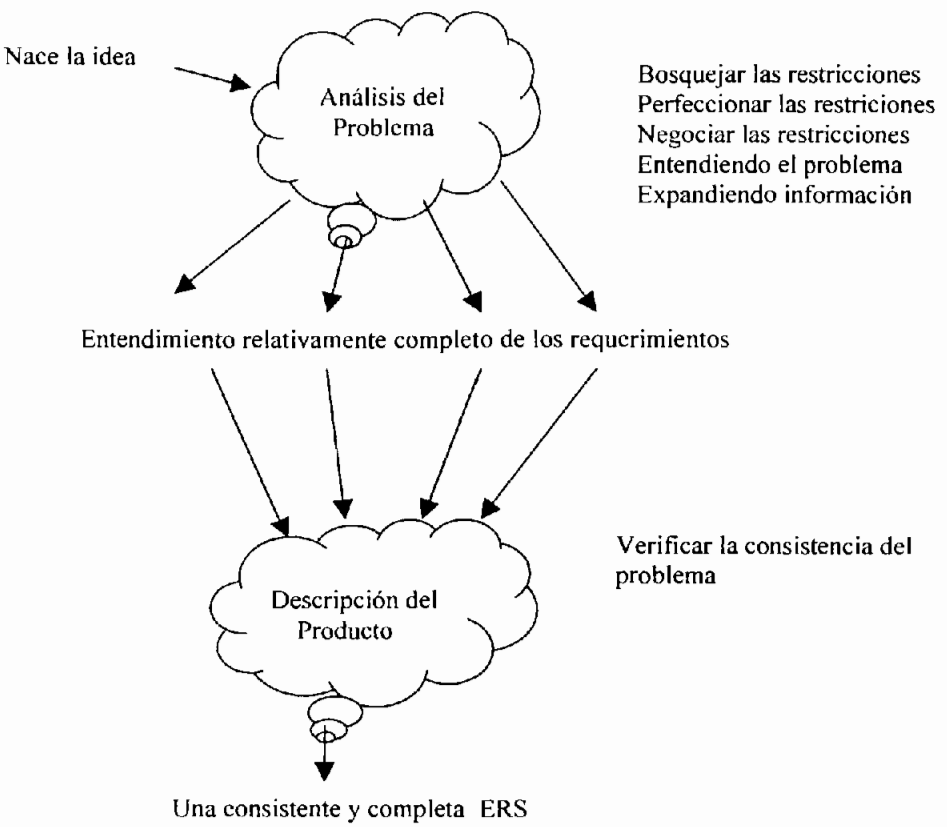


Fig. 1.9 Actividades durante la fase de requerimientos

Este documento que prepara el analista se denomina Especificación de Requerimientos de Software (ERS), en ingles Software Requirements Specification (SRS), este documento contiene una completa descripción de lo que el software debería hacer sin describir como éste lo hará.

Para el análisis del problema se usara la técnica propuesta por Yeh y Zave durante el análisis del problema, ellos sugieren dividir fundamentalmente en tres principales estructuras: partición, abstracción y proyección. La partición captura la "agregación/parte de " la relación estructural entre objetos, funciones y estados en el dominio del problema la misma que es mostrada en la Fig.1.10. La abstracción captura la "generalización/especificación " o "ejemplo de ", o " instancia de " la relación estructural entre objetos, funciones y estados en el dominio del problema, ésta es mostrada en la Fig. 1.11. La proyección captura la "visión de "la relación estructural entre objetos, funciones y estados en el dominio del problema como se observa en la Fig. 1. 12.

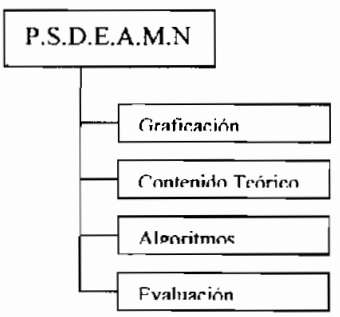


Fig. 1.10 Estructura de partición para el Paquete de Software Didáctico para la Enseñanza-Aprendizaje de Métodos Numéricos.

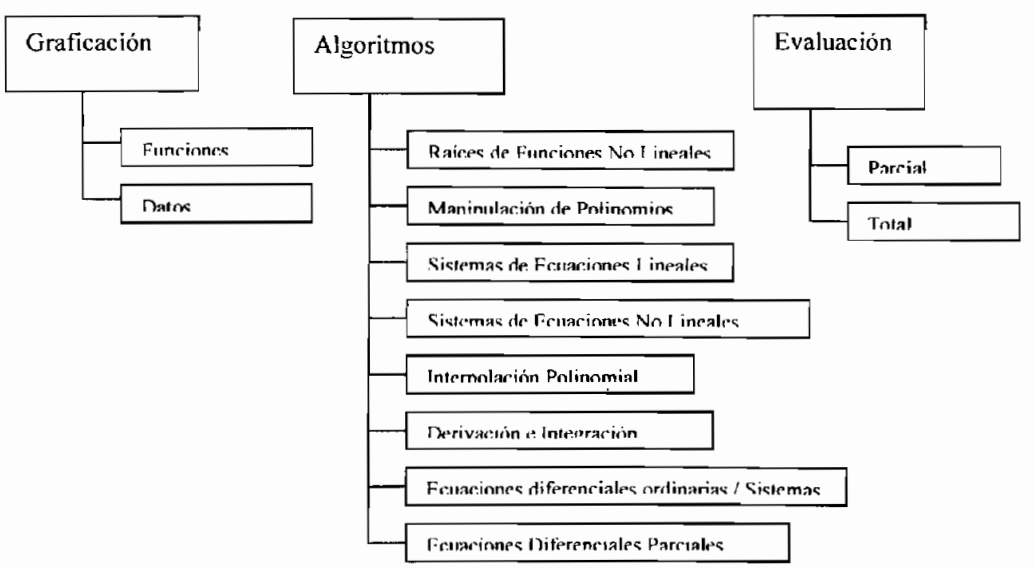


Fig. 1.11 Estructura de abstracción para el Paquete de Software Didáctico para la Enseñanza-Aprendizaje de Métodos Numéricos.

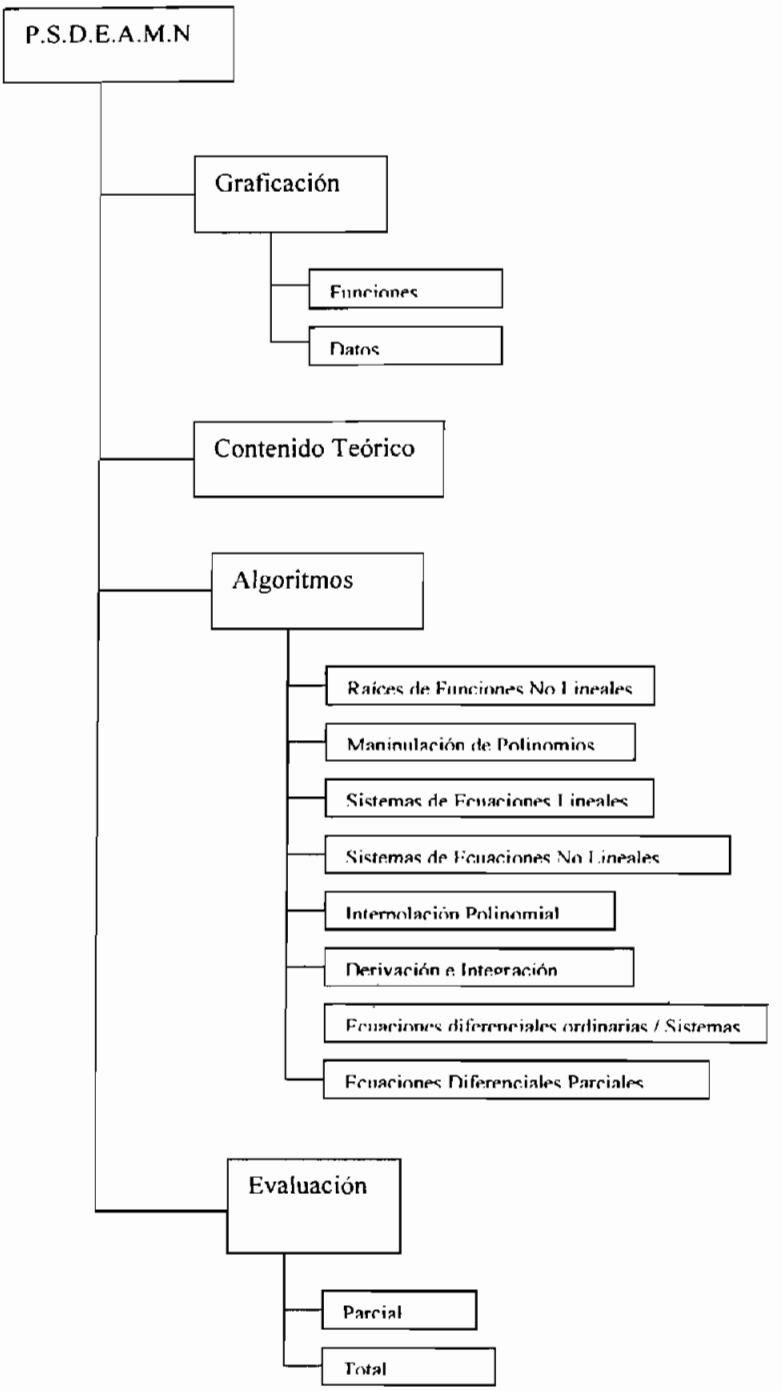


Fig. 1.12 Estructura de proyección para el Paquete de Software Didáctico para la Enseñanza-Aprendizaje de Métodos Numéricos.

Para la especificación de los requerimientos funcionales se hará uso de la técnica Orientada al estado denominada Statecharts que fue propuesta por Harel como una extensión, a las máquinas de estado finito para el modelado de sistemas en tiempo real, ésta describe que la transición del estado S1 al S2 podría darse cuando el estímulo i es recibido y la condición C es verdadera en la Fig. 1.13 se muestra esta técnica y desde la Fig. 1.14 a la Fig. 1.19 se aplica al análisis del problema.

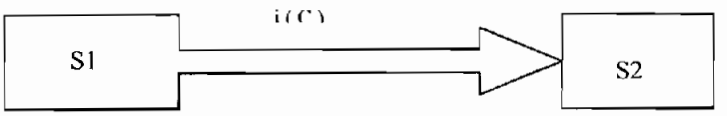


Fig. 1.13 Notación de la técnica Statecharts

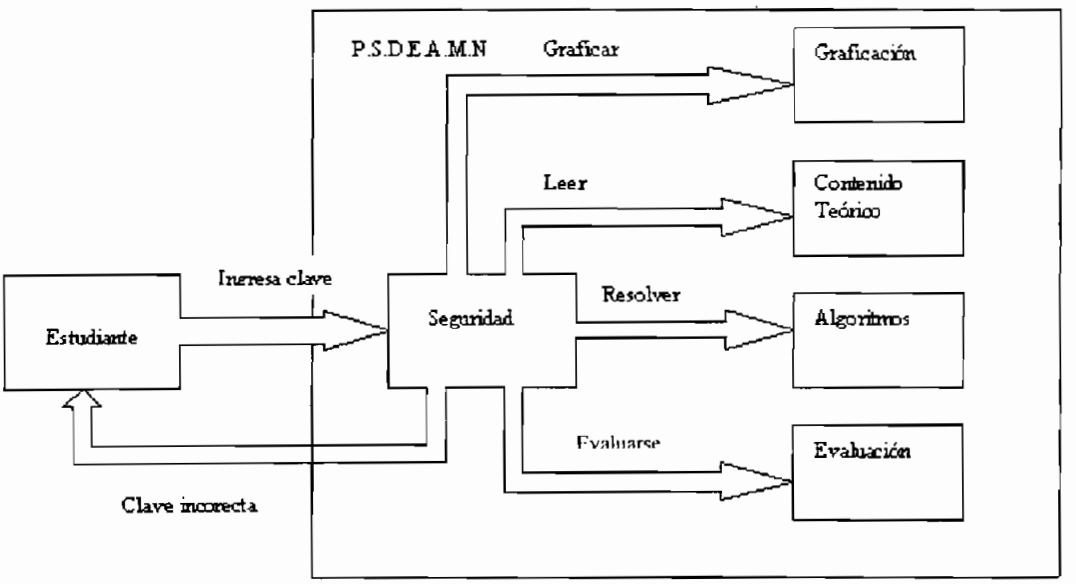


Fig. 1.14 Uso de Statecharts en el análisis de los requerimientos funcionales del Sistema

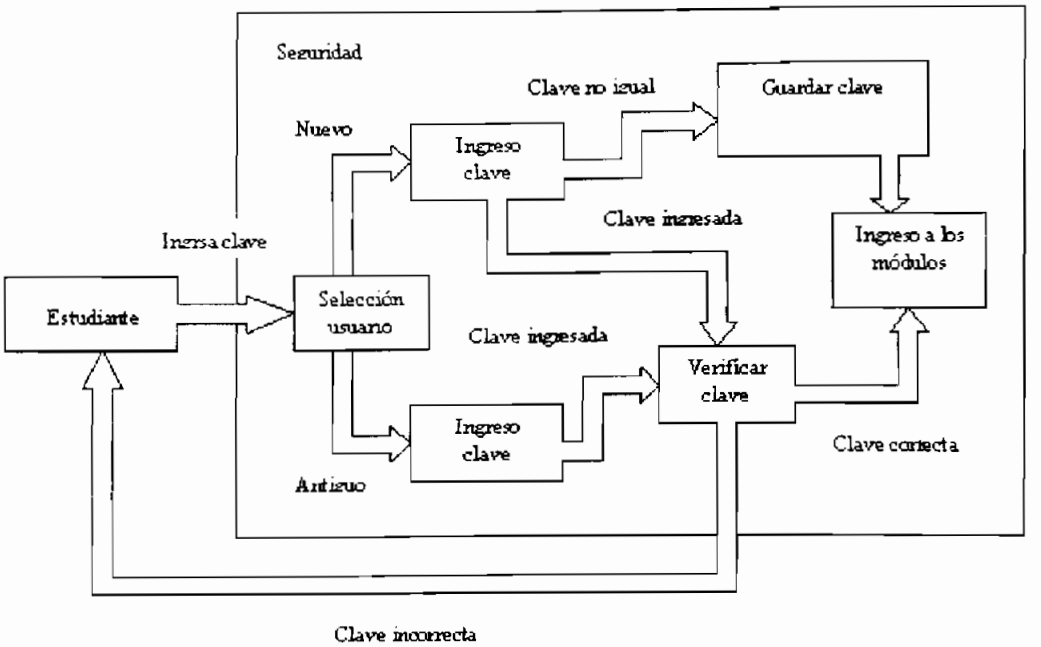


Fig. 1.15 Uso de Statecharts en el análisis de los requerimientos funcionales del Módulo de Seguridad

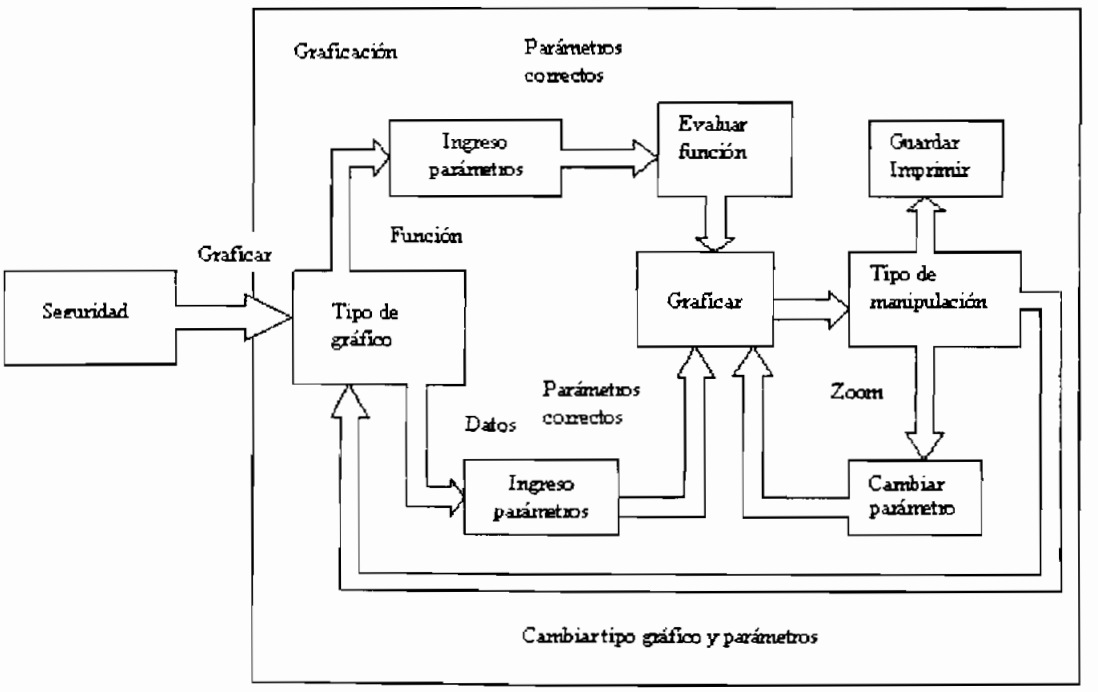


Fig. 1.16 Uso de Statecharts en el análisis de los requerimientos funcionales del Módulo de Graficación.

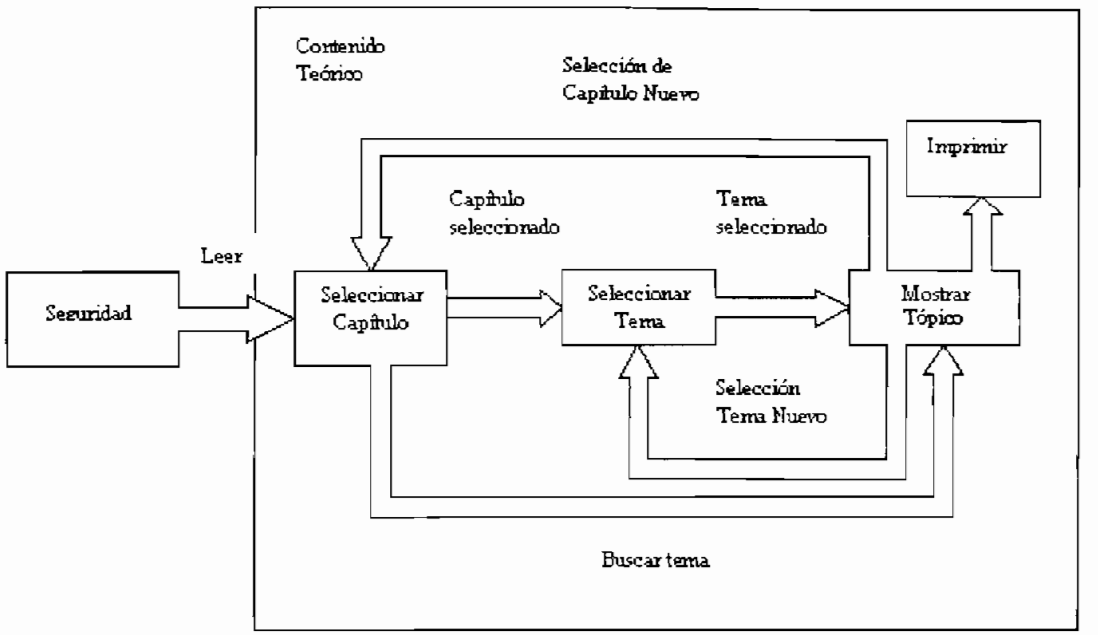


Fig. 1.17 Uso de Statecharts en el análisis de los requerimientos funcionales del Módulo de Contenido Teórico

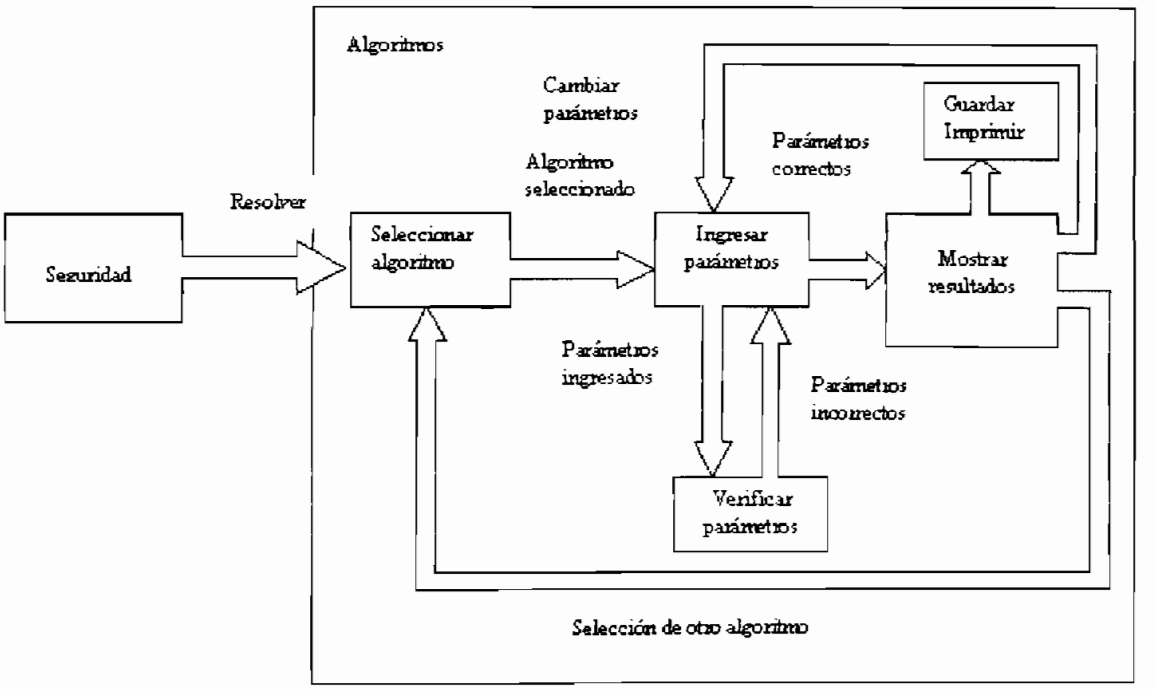


Fig. 1.18 Uso de Statecharts en el análisis de los requerimientos funcionales del Módulo de Algoritmos

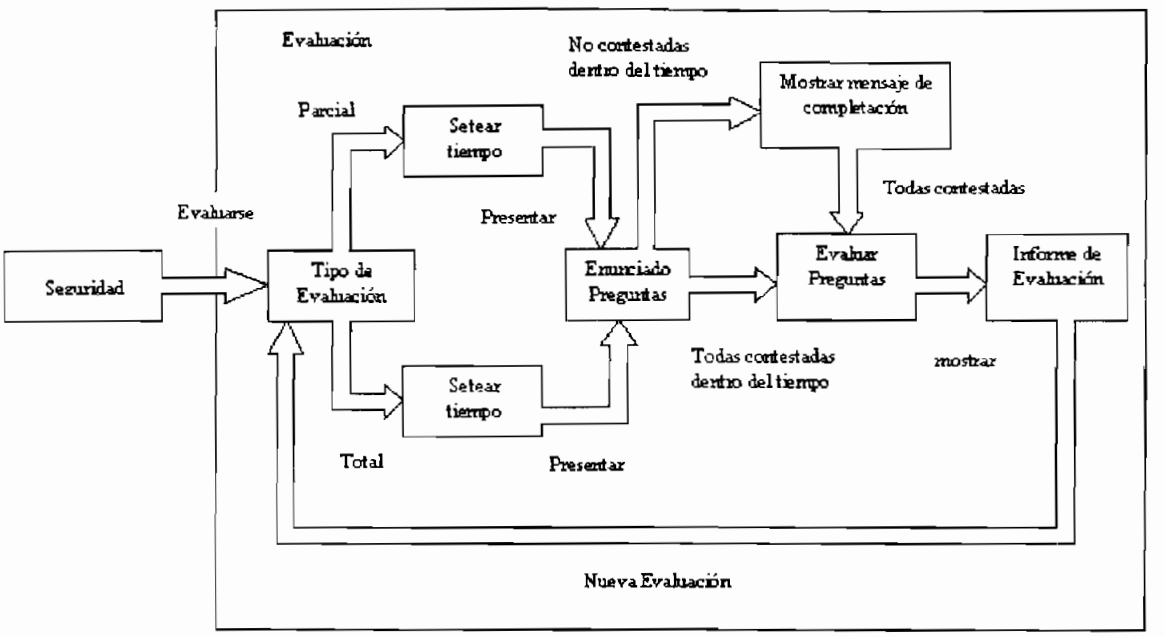


Fig. 1.19 Uso de Statecharts en el análisis de los requerimientos funcionales del Módulo de Evaluación

Estas dos últimas técnicas descritas también se encuentran sugeridas por Alan M. Davis en su libro [1]. A continuación se muestra el ERS basado en el estándar ANSI/IEEE STD-830-1984.

Para la escritura del ERS se seguirá el estándar ANSI/IEEE STD-830-1984 el mismo que es sugerido por Alan M. Davis en su libro. La misma que se detalla a continuación.

Especificación de Requerimientos del Paquete de Software Didáctico para la Enseñanza-Aprendizaje de Métodos Numéricos

1. Introducción

1.1 Propósito de la especificación de requerimientos

La elaboración de esta especificación de requerimientos tiene como fin el documentar los requerimientos que debe cumplir el Paquete de Software Didáctico para la Enseñanza –Aprendizaje de Métodos Numéricos, además el de ser utilizado como fuente básica de comunicación entre los usuarios finales(profesores, estudiantes), y todo aquel involucrado en la implementación del sistema.

1.2 Alcance del producto

Este documento contiene una descripción completa de las necesidades y funcionalidades del Paquete de Software Didáctico para la Enseñanza - Aprendizaje de Métodos Numéricos; describe la forma como hará sus funciones para lo cual se definen los requerimientos funcionales y los no funcionales (atributos). Al mismo tiempo se definen todos los requerimientos de hardware y software, diagramas, modelos del sistema y cualquier otra información que sirva de soporte y guía para fases posteriores.

1.3 Definiciones, acrónimos y abreviaciones

Palabra	Ente lingüístico	Descripción
Acceso	Verbo	Ingresar al P.S.D.E.A.M.N
Algoritmo	Sustantivo	Parte del módulo de algoritmos
Ampliación	Verbo	Incrementar la escala del gráfico
Aprobado	Verbo	
Base de datos	Sustantivo	Conjunto de archivos en los que se guarda el nombre y password del estudiante
Base de texto	Sustantivo	Conjunto de archivos que corresponden a los tópicos del módulo del contenido teórico
Búsqueda	Verbo	Permite localizar de manera rápida un tópico del contenido teórico
Capturar	Verbo	Retiene en variables el nombre y el password del estudiante
Conceptos básicos	Sustantivo	Grado de conocimiento mínimo que debe tener un estudiante
Condiciones de convergencia	Sustantivo	Indican si un algoritmo llegara o no a la aproximación deseada de la respuesta
Contener	Verbo	Tener algo en su interior

Contenido Teórico	Sustantivo	Módulo del P.S.D.E.A.M.N
Contestar	Verbo	Responder a las preguntas del módulo de evaluación
Coordenadas	Sustantivo	Valores x, y que indican la posición de un punto en la gráfica de una función
Diferenciación numérica	Sustantivo	Tema del Módulo de Algoritmos
Ecuaciones diferenciales ordinarias	Sustantivo	Tema del Módulo de Algoritmos
Ecuaciones diferenciales parciales	Sustantivo	Tema del Módulo de Algoritmos
Error	Sustantivo	Indicación de la existencia de conflictos en la ejecución del programa
Estudiante	Sustantivo	Persona que desea aprender Métodos Numéricos
Evaluación	Sustantivo	Módulo de P.S.D.E.A.M.N
Extenso	Sustantivo	Que abarca gran cantidad de información
Función	Sustantivo	Representación matemática de la variación de una variables respecto a otra
Graficar	Verbo	Mostrar en forma gráfica los valores que toma una función
Guardar	Verbo	Escribir en un archivo
Icono	Sustantivo	Imagen pequeña

Implementar	Verbo	Llevar a cabo la codificación del P.S.D.E.A.M.N. en un lenguaje de programación
Imprimir	Verbo	Trasladar a una hoja de papel lo presentado en pantalla
Informe de evaluación	Sustantivo	Pantalla que presenta el resultado de la evaluación realizada
Integración numérica	Sustantivo	Tema del Módulo de Algoritmos
Interfaz	Sustantivo	Pantalla que se presenta al usuario, objeto que permite comunicarse con otro software
Interpolación polinomial	Sustantivo	Tema del Módulo de Algoritmos
Iteraciones	Sustantivo	Número de operaciones matemáticas básicas que comprende un algoritmo para resolver un problema
Límite	Sustantivo	Restricción impuesta
Manipulación de polinomios	Sustantivo	Tema del Módulo de Algoritmos
Manipular	Verbo	Realizar alguna operación
Mensaje	Verbo	Texto presentado en pantalla

Métodos Numéricos	Sustantivo	Area del conocimiento que aproxima mediante operaciones matemáticas básicas matemáticas complicadas
Módulo de algoritmos	Sustantivo	Parte del P.S.D.E.A.M.N. que se encarga del manejo de los diferentes algoritmos
Módulo de contenido teórico	Sustantivo	Parte del P.S.D.E.A.M.N. que se encarga del manejo de los diferentes temas y tópicos que contiene el tutorial
Módulo de evaluación	Sustantivo	Parte del P.S.D.E.A.M.N. que se encarga del manejo de las diferentes evaluaciones de los temas y tópicos que contiene el tutorial
Módulo de graficación	Sustantivo	Parte del P.S.D.E.A.M.N. que se encarga del manejo de la presentación gráfica de las funciones y datos
Módulo de seguridad	Sustantivo	Parte del P.S.D.E.A.M.N. que se encarga del manejo del ingreso al mismo
Mostrar	Verbo	Presentar en pantalla
Opción	Sustantivo	Una de varias posibilidades

Pantalla de algoritmos	Sustantivo	Interfaz de usuario del módulo de algoritmos
Pantalla de graficación	Sustantivo	Interfaz de usuario del módulo de algoritmos
Paquete de software	Sustantivo	Conjunto de Instrucciones, funciones, módulos que cumplen un determinado fin
Parámetros de graficación	Sustantivo	Conjunto de valores que permiten realizar un gráfico de una función o datos
Parámetros del algoritmo	Sustantivo	Conjunto de valores que permiten realizar un gráfico de una función o datos
Pares de datos	Sustantivo	Dos valores que son correspondientes
Password	Sustantivo	Palabra clave que permite el acceso al P.S.D.E.A.M.N.
Permitir	Verbo	Dejar hacer algo
Preguntas de selección múltiple	Sustantivo	Grupo de enunciados que permiten elegir entre varias respuestas
Preguntas de verdadero falso	Sustantivo	Grupo de enunciados que permiten elegir entre dos respuestas una verdadera y otra falsa
Profesional	Sustantivo	Persona que a acreditado un titulo a nivel superior

Profesor	Sustantivo	Persona que a acreditado un título a nivel superior y enseña un determinado campo de la ciencia como Métodos Numéricos
Raíces de funciones no lineales	Sustantivo	Tema del Módulo de Algoritmos
Realizar	Verbo	Hacer
Regresión numérica	Sustantivo	Tema del Módulo de Algoritmos
Resultado de las iteraciones	Sustantivo	Conjunto de datos producto de aplicar un algoritmo a un problema
Secuencial	Sustantivo	Que se ingrese uno a continuación del otro sin permitir ir al siguiente sin haber pasado por el anterior
Sistemas de ecuaciones diferenciales ordinarias	Sustantivo	Tema del Módulo de Algoritmos
Sistemas de ecuaciones lineales	Sustantivo	Tema del Módulo de Algoritmos
Sistemas de ecuaciones no lineales	Sustantivo	Tema del Módulo de Algoritmos
Temas	Sustantivo	Conjunto de tópicos que tiene características comunes
Tiempo establecido	Sustantivo	Límite de tiempo impuesto
Tipos de algoritmos	Sustantivo	Grupo de algoritmos
Tópicos	Sustantivo	Parte de un tema del Módulo de Contenido Teórico

Usuario	Sustantivo	Persona que ingresa al P.S.D.E.A.M.N.
Variar	Verbo	Cambiar
Ventana	Sustantivo	Interfaz de usuario

2. Descripción general

2.1 Perspectiva del producto

El Paquete de Software Didáctico para la Enseñanza - Aprendizaje de Métodos Numéricos es un sistema independiente es decir no es un componente o aporte de un sistema o un proyecto más grande que lo contenga, por lo que no posee interfaz externa alguna con otro sistema de software.

El Paquete de Software Didáctico para la Enseñanza - Aprendizaje de Métodos Numéricos está destinado básicamente a estudiantes de educación media, superior y a profesionales que se interesen por este campo de estudio, los mismos que poseen conocimientos básicos de las matemáticas que se desarrollan en esta área. Esto no significa que se restrinja su uso a personas que no tengan conocimientos matemáticos amplios y que deseen aprender la materia de métodos numéricos ya que el contenido teórico dará una visión tal que permita aprender de una manera fácil y rápida los temas tratados.

2.2 Funciones del producto

El Paquete de Software Didáctico para la Enseñanza - Aprendizaje de Métodos Numéricos se describe como un software educativo con las siguientes funciones principales:

- Enseñanza de definiciones, conceptos y algoritmos en la materia de métodos numéricos.
- Posibilidad de la impresión del contenido de la enseñanza.

- Evaluación, presentación de informes de la evaluación y recomendaciones al usuario al terminar las evaluaciones para mejorar su aprendizaje.
- Utilización de elementos multimedia tales como: texto, hipertexto e imágenes que facilitan la enseñanza.

2.3 Características de los usuarios

Los siguientes usuarios son las principales personas y entidades que intervienen en las fases de operación y mantenimientos del ciclo de vida del Paquete de Software Didáctico para la Enseñanza - Aprendizaje de Métodos Numéricos.

Profesor

Persona que enseña la materia de métodos numéricos, debe poseer un titulo de nivel superior al menos, este tipo de usuario no requiere de alguna experiencia en particular salvo haber manejado alguna aplicación que se ejecute en Windows.

Estudiante

Persona que constituye la razón de ser del Paquete de Software Didáctico para la Enseñanza - Aprendizaje de Métodos Numéricos, es aquel usuario que aprende todos y cada uno de los algoritmos mediante la lectura y su posterior utilización en el paquete de software.

Para entender los métodos numéricos mediante el uso del Paquete de Software Didáctico para la Enseñanza - Aprendizaje de Métodos Numéricos solamente se necesita ser estudiantes de educación media, superior y o profesionales que se interesen por este campo de estudio. Este tipo de usuario no necesita tener experiencia alguna en la materia de métodos numéricos, mas bien el paquete de software trata de enseñarle desde un nivel cero hasta un nivel aceptable para que con los conocimientos adquiridos pueda resolver problemas relacionados con los temas expuestos que encontrará en el transcurso de su vida estudiantil o profesional.

Personal de mantenimiento

Es la persona o grupo de personas que cuando el Paquete de Software Didáctico para la Enseñanza - Aprendizaje de Métodos Numéricos se encuentre en ejecución podrán dar el adecuado mantenimiento tanto adaptivo o correctivo y mejoramiento al sistema. Las mencionadas personas deben ser por lo menos ingenieros, analistas en sistemas o personas que hayan desarrollado varias aplicaciones orientadas a objetos, además haber realizado mantenimiento tanto adaptivo, correctivo y mejoramiento de aplicaciones en producción.

2.4 Restricciones generales

- Desarrollo bajo la plataforma Microsoft Windows
- Los tiempos de respuesta deben variar de 10 a 20 segundos desde que el usuario ejecuta un comando y el paquete de software muestra la respuesta correspondiente.
- Existen interfaces con software o hardware especializado.
- No se maneja multiprocesamiento.
- No se maneja funcionamiento en red
- No existe el manejo multiusuario
- El paquete de software no es parte o módulo de otro sistema mayor.
- La implementación será en un lenguaje visual y orientado a objetos
- Utilización de elementos multimedia
- No existen algoritmos especiales para el manejo de seguridad.

2.5 Suposiciones y dependencias

- Cambio a otro sistema operativo que no sea otro que Microsoft Windows.
- Aumento o disminución de los tipos de usuario del sistema.
- Aumento o disminución del número de datos o funciones a resolver.

- Soporte para red.
- Soporte multiusuario.
- Manejo de multiprocesamiento.
- Manejo de algoritmos para la seguridad.
- Soporte tanto de hardware y software específicos.
- Cambios en los lineamientos, políticas y tácticas de la enseñanza de la materia de métodos numéricos.
- Aumento o disminución de los requerimientos del sistema.
- Cambio en la perspectiva del producto de independencia o parte de otro sistema mayor.

3. Especificación de requerimientos

3.1 Requerimientos funcionales

Modulo de seguridad

Ref. #	Función	Categoría
R1.1	Registrar la condición (nuevo o antiguo) del estudiante que ingresa al paquete de software.	Evidente
R1.2	Permitir el ingreso del nombre y el password del estudiante	Evidente
R1.3	Verificar que el número de caracteres tanto del nombre como del password del estudiante sean los correctos.	Oculta
R1.4	Guardar el nombre y el password del estudiante	Oculta
R1.5	Verificar que el nombre y el password del estudiante correspondan a los de la base de datos	Oculta
R1.6	Mostrar mensaje que indique que el nombre o el password es incorrecto	Evidente

Módulo de contenido teórico

Ref. #	Función	Categoría
R2.1	Mostrar el contenido teórico del tutorial	Evidente
R2.2	Mostrar el texto del tema seleccionado	Evidente
R2.3	Permitir el ingreso de la o las palabras del tema a buscar	Evidente
R2.4	Buscar en la base de texto temas afines a la o las palabras ingresadas para la búsqueda.	Evidente
R2.5	Mostrar el o los temas afines encontrados como resultado de la búsqueda.	Evidente
R2.6	Mostrar el tema siguiente	Evidente
R2.7	Mostrar el tema anterior	Evidente
R2.8	Imprimir el tema activo	Evidente

Módulo de graficación

Ref. #	Función	Categoría
R3.1	Registrar tipo de gráfico a realizar.	Evidente
R3.2	Registrar número de funciones a graficar.	Evidente
R3.3	Registrar el número de pares de datos a graficar.	Evidente
R3.4	Permitir el ingreso de la expresión de la o las funciones a graficar.	Evidente
R3.5	Capturar la expresión de la o las funciones a graficar.	Oculta
R3.6	Evaluar la expresión de la o las funciones a ser graficadas.	Oculta

R3.7	Permitir el ingreso de los pares de datos a ser graficados.	Evidente
R3.8	Capturar los valores de los pares de datos ingresados para ser graficados	Oculta
R3.9	Capturar los valores resultantes de la evaluación de la o las funciones a ser graficadas.	Oculta
R3.10	Permitir el ingreso de las escalas de graficación.	Evidente
R3.11	Capturar los valores de las escalas de graficación.	Oculta
R3.12	Verificar que los parámetros de graficación estén completos y dentro de los limites establecidos.	Oculta
R3.13	Mostrar un mensaje indicando que faltan parámetros de graficación por ingresar.	Evidente
R3.14	Permitir cambiar las escalas de graficación, funciones, pares de datos en pantalla	Evidente
R3.15	Mostrar el plano de graficación con las escalas de graficación y los ejes coordenados.	Evidente
R3.16	Mostrar el gráfico de la o las funciones o los pares de datos en el plano de graficación.	Evidente
R3.17	Permitir seleccionar con el mouse la parte del gráfico a ser ampliada.	Evidente
R3.18	Mostrar el gráfico ampliado	Evidente
R3.19	Imprimir la pantalla del módulo de graficación.	Evidente
R3.20	Permitir guardar el gráfico de la función, la expresión de la función o los pares de datos en un archivo.	Evidente

Módulo de Evaluación

Ref. #	Función	Categoría
R4.1	Registrar el tipo de evaluación.	Evidente
R4.2	Registrar el tema a ser evaluado.	Evidente
R4.3	Seleccionar de la base de datos las preguntas correspondientes al tema a ser evaluado.	Oculta
R4.4	Mostrar las preguntas en pantalla de modo que se pueda navegar entre ellas.	Evidente
R4.5	Setear el tiempo de la evaluación.	Oculta
R4.6	Verificar el tiempo transcurrido de la evaluación.	Oculta
R4.7	Permitir contestar las preguntas.	Evidente
R4.8	Evaluar las respuestas cuando el usuario indique que ha terminado la evaluación, o cuando el tiempo haya expirado y el estudiante haya completado de responder las preguntas.	Oculta
R4.9	Verificar que todas las preguntas han sido contestadas	Oculta
R4.10	Mostrar un mensaje para que el estudiante complete las preguntas que le faltan cuando el tiempo para la prueba haya expirado.	Evidente
R4.11	Verificar las respuestas del estudiante con las correctas de la base de datos.	Oculta
R4.12	Mostrar informe del resultado de la evaluación	Evidente

Módulo de algoritmos

Ref. #	Función	Categoría
R5.1	Registrar el tema al que el algoritmo pertenece	Evidente
R5.2	Registrar el tipo de algoritmo a usar.	Evidente
R5.3	Permitir el ingreso de los parámetros que el algoritmo requiere para su funcionamiento.	Evidente
R5.4	Capturar el valor o expresiones que contengan los parámetros del algoritmo.	Oculta
R5.5	Verificar que los parámetros del algoritmo estén completos y dentro de los limites establecidos.	Oculta
R5.6	Mostrar un mensaje indicando que no están completos los parámetros del algoritmo.	Evidente
R5.7	Permitir cambiar los parámetros del algoritmo en pantalla.	Evidente
R5.8	Calcular los resultados con ayuda del algoritmo implementado.	Oculta
R5.9	Mostrar los resultados principales en pantalla	Evidente
R5.10	Mostrar mensaje de fracaso si el algoritmo no pudo resolver el problema.	Evidente
R5.11	Mostrar los resultados de las iteraciones en pantalla como opcional.	Evidente
R5.12	Permitir usar el módulo de graficación en la misma pantalla en la cual se muestran los parámetros del algoritmo y sus resultados principales.	Evidente
R5.13	Guarda los parámetros del algoritmo, sus resultados y el gráfico en un archivo	Evidente
R5.14	Imprimir la pantalla donde se encuentran tanto los parámetros del algoritmo como la de los resultados de las iteraciones	Evidente

3.2 Requerimientos de la interfaz externa

3.2.1 Interfaz de usuario

Requerimientos Generales

1. Las interfaces de usuario serán interactivas y fáciles de manipular mediante el uso del teclado o el ratón (mouse) para lo cual se utilizarán ventanas.
2. En la ventana de bienvenida se mostrará el nombre del programa y del autor.
3. La ventana principal del paquete didáctico deberá tener cinco botones que permitan los primeros cuatro ingresar a los diferentes módulos (contenido teórico, algoritmos, evaluación y graficación) del paquete de software y el otro que permita salir de la aplicación.
4. Cada elemento que contenga la ventana (botones, etiquetas, barra de menús, etc.) deberá estar plenamente identificados mediante un nombre o ícono que indique la función que éste realiza.
5. Existirán cuadros de mensajes que le indiquen al usuario que ha existido un error (se indicará su causa y su posible solución), que ingrese algún dato (ingreso del password, selección del algoritmo, etc.) o le mostrará resultado de algún proceso (indicación de la aprobación de la evaluación).
6. Todas las ventanas del paquete didáctico tendrán acceso a un archivo de ayuda al cual se accederá mediante un botón ubicado en la barra de menú.

Requerimientos de la Interfaz del Módulo de Contenido Teórico

1. La ventana mostrará cada una de las tareas a realizar (atrás, imprimir, etc.) mediante botones.
2. En la misma ventana el usuario podrá observar el tópico donde se encuentra mediante un esquema jerárquico y el contenido del tópico.
3. Se podrá alternar entre el esquema jerárquico del contenido y la opción de búsqueda mediante el uso de un manejador de fichas.

4. Permitirá con la ayuda del ratón, observar el contenido de otro tópico que tenga el mismo sentido semántico de una palabra que se encuentre en un tópico en el cual estemos anteriormente.

Requerimientos de la Interfaz del Módulo de Algoritmos

1. El módulo de algoritmos contendrá una ventana principal en la que estarán especificados los temas generales (raíces de funciones no lineales, integración y diferenciación numérica, etc.) para los cuales se han implementado los algoritmos. Los mismos que tendrán enlaces a cada una de las ventanas de los algoritmos implementados, también contendrá un botón que le permita enlazarse con la ventana principal del paquete didáctico.
2. Cada ventana de algoritmos dispondrá de un botón en la barra de menú que permitirá elegir las diferentes variantes del algoritmo si las tuviere.

3. Las ventanas de los algoritmos mostrarán elementos que permitan:
   - El ingreso de los parámetros de entrada
   - La salida de datos respectivos
   - El regreso hacia la pantalla principal del módulo de algoritmos.
4. En cada ventana de los algoritmos se dispondrá de un área para graficación de la función o datos que tenga toda la potencialidad del módulo de graficación (variación de las escalas de graficación, zoom).

Requerimientos de la Interfaz del Módulo de Evaluación

1. Se presentará al usuario una ventana en la cual se le pedirá que ingrese su nombre y password como paso obligatorio antes de ingresar a la ventana principal del módulo de evaluación.
2. Se dispondrá de una ventana principal del módulo de evaluación en la cual se indicarán todos los tópicos que contiene para la evaluación los mismos que coincidirán con los temas del módulo teórico, los mismos que se enlazarán con las ventanas de evaluación respectivas. Además se contarán con

opciones como la de realizar una evaluación total y la de regresar a la ventana principal del paquete didáctico.

3. Las ventanas de evaluación permitirán que el usuario pueda moverse por las distintas preguntas mediante el uso de una barra de desplazamiento.
4. Cada ventana de evaluación contendrá dos botones el primero para que le indique al paquete didáctico el momento en el que ha concluido de contestar las preguntas y otro para regresar a la ventana principal del módulo de evaluación, el cual se activará solo cuando haya finalizado la evaluación.

Requerimientos de la Interfaz del Módulo de Graficación

1. Se mostrará una ventana en la cual se disponga de elementos para el ingreso de las escalas de graficación, fórmulas o datos, con las cuales se procederá al realizar el gráfico correspondiente.
2. Se dispondrá también dos botones que permita el uno realizar la ampliación del gráfico (zoom) y el otro para retornar a la pantalla principal del paquete didáctico.

3.2.2 Interfaz de hardware

El Paquete de Software Didáctico para la Enseñanza - Aprendizaje de Métodos Numéricos para su funcionamiento completo necesita manejar interfaces con los siguientes elementos de hardware.

DISPOSITIVO	DESCRIPCIÓN DE LA INTERFAZ
Monitor SVGA o superior	Permite la presentación en pantalla de las ventanas que el paquete de software posee.
Mouse serie estándar y teclado estándar	Permite al usuario interactuar con el paquete de software
Impresora matricial o superior	Permite imprimir el tema, informes o pantallas del paquete de software.
Disco duro	Permite la grabación de los archivos tanto de ejecución como de manejo del sistema.
Unidad de CD-Rom	Permite la lectura de todos los archivos que permiten la instalación del sistema.
Unidad de discos flexibles	Permite la portabilidad de datos a otro ambiente.

3.2.3 Interfaz de software

El Paquete de Software Didáctico para la Enseñanza - Aprendizaje de Métodos Numéricos requiere interfaces con los siguientes productos de software para su funcionamiento y ejecución.

Nombre: Windows 98 C
Versión: 4.00 o posterior
Fuente: Microsoft Corporation
Se requiere su instalación: Si
Descripción de la interfaz: permite la ejecución del sistema así como el manejo y acceso al hardware necesario, es uno de los posibles sistemas operativos que se pueden utilizar. Se necesita que la interfaz del sistema operativo sea la que añada el Internet Explorer 4.0 o posterior.

Nombre: Windows NT
Versión: 4.00 o posterior
Fuente: Microsoft Corporation
Se requiere su instalación: Si
Descripción de la interfaz: permite la ejecución del sistema así como el manejo y acceso al hardware necesario, es uno de los posibles sistemas operativos que se pueden utilizar. Se necesita que la interfaz del sistema operativo sea la que añada el Internet Explorer 4.0 o posterior.
Los anteriores productos de software constituyen las posibles plataformas de ejecución por lo tanto se requiere uno de las dos.

Nombre: Microsoft Access
Versión: 7.00 o posterior
Fuente: Microsoft Corporation
Se requiere su instalación: Si
Descripción de la interfaz: permite el manejo y almacenamiento de los datos de usuario, las preguntas y respuestas que se utilizaran en la evaluación.

Nombre: Internet Explorer
Versión: 4.00 o posterior
Fuente: Microsoft Corporation
Se requiere su instalación: Si
Descripción de la interfaz: permite el manejo de búsqueda dinámica de los hipertextos del contenido teórico.

Nombre: Microsoft Excel 97
Versión: 7.00 o posterior
Fuente: Microsoft Corporation
Se requiere su instalación: Si
Descripción de la interfaz: permite la evaluación de las funciones ingresadas por teclado para su posterior graficación.

Los siguientes son los productos de software que permiten el control del hardware necesario para el paquete de software, cada uno debe encontrase correctamente instalado en el sitio de ejecución, estas interfaces no son proporcionadas directamente por el paquete de software sino son parte del equipo donde se ejecutará el paquete.

- Controladores de disco duro.
- Controladores de la unidad de CD-ROM
- Controladores de audio y vídeo.

3.2.4 Interfaz de comunicación

No existe requerimiento alguno en la parte concerniente a comunicaciones.

3.3 Requerimientos de desempeño

El Paquete de Software Didáctico para la Enseñanza - Aprendizaje de Métodos Numéricos tendrá la siguiente especificación de desempeño:

- Instalación en cualquier número de terminales.
- No existe multiusuarios, número de usuarios simultáneos uno.
- Tamaño máximo de archivos de base de datos 1 gigabyte.
- Tamaño máximo de una tabla 1 gigabyte.
- Máximo número de preguntas posibles aproximadamente 200.
- El número máximo de formularios disponibles solamente se limita por las capacidades del sistema.

3.4 Restricciones de diseño

Restricciones en el módulo de contenido teórico

- El tema expuesto tendrá el número y título del tema.
- Se dispondrá de hipervínculos para enlazar temas afines.
- Se utilizarán sub-ventanas para mostrar algún concepto que se desee aclarar.
- Se presentara en la misma ventana el contenido teórico a la derecha y el contenido del tema a la izquierda.
- Se podrá cambiar de la vista del contenido teórico al sistema de búsqueda de temas mediante solapas.

Restricciones en el módulo de graficación

- En la misma ventana se presentarán los parámetros de graficación, las funciones y el gráfico.
- Se permitirá graficar máximo 3 funciones a la vez.
- Se permitirá ingresar máximo 50 pares de valores para ser graficados.

Restricciones en el módulo de algoritmos

- En una sola ventana se presentarán los parámetros del algoritmo, sus resultados, los parámetros de graficación, las funciones y el gráfico.
- En una ventana aparte se mostrarán el resultado de las iteraciones.
- Para los algoritmos de raíces de funciones no lineales se cumplirá que sea cualquier función f(x), especificando condiciones de convergencia o limitando el número de iteraciones.
- Para los algoritmos de manipulación de polinomios se cumplirá que los polinomios sean hasta de décimo grado y la evaluación del polinomio hasta la primera derivada.

- Para los sistemas de ecuaciones lineales y no lineales se cumplirá que sean hasta sistemas de 20 ecuaciones.
- Para los algoritmos de regresión e interpolación polinomial se cumplirá que sea el ingreso de hasta 50 pares de datos y polinomios de hasta quinto orden.
- Para los algoritmos de diferenciación e integración numérica se cumplirá que sea hasta la tercera derivada.
- Para los algoritmos de resolución de ecuaciones diferenciales ordinaria se cumplirá que sean hasta ecuaciones de tercer orden y hasta tres ecuaciones de primer orden.
- Para los algoritmos de resolución de ecuaciones diferenciales parciales se cumplirá que se resuelvan ecuaciones unidimensionales de onda y del calor, ecuación de Laplace y fronteras regulares, red de puntos de 10 x 10.

Restricciones en el módulo de evaluación

- La ventana de evaluación presentará: un encabezado con el tipo de evaluación y el tema que se está evaluando, las preguntas de modo que se pueda ir de una a la otra, un botón que permita la inicialización de la evaluación de las preguntas contestada.
- La ventana de informe presentará un encabezado con el titulo del tema evaluado; en el detalle se incorporará: el nombre del usuario que se evalúa, porcentaje de aciertos, sugerencia que indique en que parte ha acertado y en cuales no, y si ha pasado o no la evaluación.
- La evaluación parcial contendrá 20 preguntas y el tiempo para esta evaluación será de una hora.
- La evaluación total contendrá 30 preguntas y el tiempo para la evaluación será de una hora y treinta minutos,
- Se dará por aprobada las evaluaciones si por lo menos el 70% de las respuestas son correctas

Restricciones de Hardware

El Paquete de Software Didáctico para la Enseñanza - Aprendizaje de Métodos Numéricos necesita de los siguientes requisitos de hardware mínimos para poder ser ejecutable.

- Un monitor SVGA
- Un disco duro
- Unidad de CD-ROM
- Unidad de disco flexible
- Microprocesador 80486
- 8 MB de Ram
- Una tarjeta de vídeo
- Una tarjeta de sonido.
- Parlantes.

3.5 Atributos

Disponibilidad

- El sistema se encontrará disponible para la ejecución por parte del usuario después de realizar la instalación respectiva en una computadora que posea todos los requerimientos de software y hardware especificados en este documento.
- El sistema se ejecutará sin restricción alguna hasta el momento en el que el usuario detenga la ejecución expresamente.
- El usuario se encuentra habilitado de detener la ejecución del programa en cualquier momento.
- Después de detener la ejecución del sistema el usuario puede volver a ejecutarlo sin ninguna restricción.
- El sistema dejará de estar disponible únicamente si se ejecuta la utilidad de desinstalación que provee el sistema.

Seguridad

Los siguientes elementos especifican los requisitos de seguridad que debe manejar el Paquete de Software Didáctico para la Enseñanza - Aprendizaje de Métodos Numéricos.

- No existirá el manejo de técnicas de inscripción.
- Se guardará el nombre y password del usuario.
- El número de caracteres del nombre y el password del usuario será: mínimo 3 y máximo 11.
- Se guardará el resultado de las evaluaciones del usuario.

La siguiente lista muestra los posibles factores que pueden afectar la seguridad del sistema haciendo que se ejecute el sistema incorrectamente o no se ejecute.

- Si se modifican los archivos del sistema
- Si se destruyen los archivos del sistema.
- Si se mueven los archivos del sistema a otra localización diferente en la cual los ubicó la utilidad de instalación.
- Si se eliminan algunos o todos los archivos del sistema.
- Suspención de la energía eléctrica.
- Daño en la ubicación física de los archivos del sistema
- Daño en un dispositivo de hardware necesario.

Todo control sobre algún evento especial adicional como por ejemplo: mal manejo de un dispositivo, desbordamiento de pila, fallas inesperadas en un dispositivo, mal manejo de memoria por parte de otros productos de software se lo otorga al sistema operativo que soporta la aplicación.

Mantenibilidad

Para facilitar su mantenimiento el Paquete de Software Didáctico para la Enseñanza - Aprendizaje de Métodos Numéricos debe poseer los siguientes requisitos.

Implementación mediante el uso de una herramienta visual orientada a objetos.
Uso de todas las virtudes de la programación orientada a objetos como son: objetos, clases, encapsulamiento, herencia, poliformismo y persistencia.
Todas las variables y funciones deben ser implementadas respondiendo a un estándar que sea fácilmente legible y entendible.
Cada servicio de un objeto debe poseer un pequeño comentario que explique que y como realiza su trabajo.

Se debe utilizar en lo posible algoritmos simples que por tanto faciliten su modificación para soportar futuros requerimientos.
El código debe ser formateado de tal manera que sea interpretado sin ambigüedades y fácilmente por las personas que realicen el mantenimiento, estos formatos pueden incluir por ejemplo: utilización de espacios en blanco, identación de los niveles para evitar que el código sea ilegible.
En lo posible evitar la utilización de estructuras complejas en el código.

Transferencia / conversión
El Paquete de Software Didáctico para la Enseñanza - Aprendizaje de Métodos Numéricos para su portabilidad a otro ambiente necesita de los siguientes procedimientos.

- Una computadora con los requisitos de software y hardware especificados en este documento.
- Respaldar la base de datos en un disco flexible instalar la aplicación en el otro ambiente y finalmente restablecer (sobreescribir) la base de datos en el segundo ambiente, esto permite al estudiante portar el software sin necesidad de volver al aprendizaje desde el principio en otro ambiente.

3.6 Otros requerimientos

3.6.1 Base de datos

El sistema utilizará Microsoft Access 7.0 como herramienta de desarrollo de software que permite el soporte de los requerimientos de administración de datos, dichos requerimientos se especifican a continuación.

Almacenará la siguiente información: el nombre, password, las notas de las evaluaciones de los estudiantes que interactuan con el Paquete de Software Didáctico para la Enseñanza - Aprendizaje de Métodos Numéricos, las preguntas tanto de selección múltiple como lógicas con sus respectivas respuestas.
Será necesaria su utilización en los siguientes casos: El estudiante ingresa al sistema, cuando el estudiante sale del módulo de evaluación.
El usuario no tendrá ningún acceso directo a la base de datos, es decir no se le permitirá agregar, modificar, o eliminar ninguna información manualmente.

1.1.1 GLOSARIO DE TÉRMINOS DEL PAQUETE DE SOFTWARE DIDÁCTICO PARA LA ENSEÑANZA-APRENDIZAJE DE MÉTODOS NUMÉRICOS.

El glosario de términos se encuentra detallado en la especificación de requerimientos del Paquete de Software Didáctico para la Enseñanza - Aprendizaje de Métodos Numéricos en el numeral 1.3 correspondiente a Definiciones, acrónimos y abreviaciones.

1.3.3 CASOS DE USO DEL PAQUETE DE SOFTWARE DIDÁCTICO PARA LA ENSEÑANZA-APRENDIZAJE DE MÉTODOS NUMÉRICOS.

1.3.3.1 Casos de Uso

Un Caso de Uso es un documento narrativo que describe la secuencia de eventos de un actor (una agente externo) que usa un sistema para completar un proceso. Es una historia o una forma particular de usar un sistema. Los casos de uso no son exactamente requisitos ni especificaciones funcionales, pero ilustran e implican requisitos en las historias que cuentan.

Nótese que UML no define un formato para describir un caso de uso. Tan sólo define la manera de representar la relación entre actores y casos de uso en un diagrama (Diagrama de Casos de Uso).

El formato textual que se va a usar en este proyecto de titulación para definir los caso de uso se va a definir a continuación.

El significado de cada apartado de este formato es como sigue:

- Caso de Uso: ***Nombre del Caso de Uso***
- Actores: Lista de actores (agentes externos), indicando quién inicia el caso de uso. Los actores son normalmente roles que un ser humano desempeña, pero puede ser cualquier tipo de sistema.
- Propósito: Intención del caso de uso.
- Visión General: Repetición del caso de uso de alto nivel, o un resumen similar.
- Tipo: 1. primario, secundario u opcional.
  2. esencial o real.
- Referencias: Casos de uso relacionados y funciones del sistema que aparecen en los requisitos.
- Curso Típico de Eventos: Descripción de la interacción entre los actores y el sistema mediante las acciones numeradas de cada uno. Describe la secuencia más común de eventos, cuando todo va bien y el proceso se completa

satisfactoriamente. En caso de haber alternativas con grado similar de probabilidad se pueden añadir secciones adicionales a la sección principal.

- Cursos Alternativos: Puntos en los que puede surgir una alternativa, junto con la descripción de la excepción.

En un primer momento interesa abordar un caso de uso desde un nivel de abstracción alto, es lo que se denomina Caso de Uso de Alto Nivel.

Casos de Uso de Alto Nivel

El siguiente Caso de Uso de Alto Nivel describe el proceso de sacar dinero cuando se está usando un cajero automático:

- Caso de Uso: *Realizar Reintegro*
- Actores: Cliente
- Tipo: primario
- Descripción: Un Cliente llega al cajero automático, introduce la tarjeta, se identifica y solicita realizar una operación de reintegro por una cantidad específica. El cajero le da el dinero solicitado tras comprobar que la operación puede realizarse. El Cliente coge el dinero y la tarjeta y se va.

En un caso de uso descrito a alto nivel, la descripción es muy general, normalmente se condensa en dos o tres frases. Es útil para comprender el ámbito y el grado de complejidad del sistema.

Casos de Uso Expandidos

Los casos de uso que se consideren los más importantes y que se considere que son los que más influencian al resto, se describen a un nivel más detallado: en el formato expandido.

La principal diferencia con un caso de uso de alto nivel está en que incluye un apartado de *Curso Típico de Eventos*, pero también incluye otros apartados como se ve en el siguiente ejemplo:

- Caso de Uso: **Realizar Reintegro**
- Actores: Cliente (iniciador)
- Propósito: Realizar una operación de reintegro de una cuenta del banco.
- Visión General: Un Cliente llega al cajero automático, introduce la tarjeta, se identifica y solicita realizar una operación de reintegro por una cantidad específica. El cajero le da el dinero solicitado tras comprobar que la operación puede realizarse. El Cliente coge el dinero y la tarjeta y se va.
- Tipo: primario y esencial
- Referencias: *Funciones:* R1.3, R1.7
- Curso Típico de Eventos:

Acción del Actor	Respuesta del Sistema
1. Este caso de uso empieza cuando un Cliente introduce una tarjeta en el cajero.	2. Pide la clave de identificación.
3. Introduce la clave.	4. Presenta las opciones de operaciones disponibles.
5. Selecciona la operación de Reintegro.	6. Pide la cantidad a retirar.
7. Introduce la cantidad requerida.	8. Procesa la petición y eventualmente, da el dinero solicitado. Devuelve la tarjeta y genera un recibo.
9. Recoge la tarjeta.	
10. Recoge el recibo.	
11. Recoge el dinero y se va.	

Cursos Alternativos:

Línea 4: La clave es incorrecta. Se indica el error y se cancela la operación.

Línea 8: La cantidad solicitada supera el saldo. Se indica el error y se cancela la operación.

1.3.3.2 Descripción de los casos de uso del paquete de software didáctico para la enseñanza-aprendizaje de métodos numéricos.

A continuación se detallan los casos de uso tanto los de alto nivel como los expandidos.

Casos de uso de alto nivel

Caso de uso: **Identificar estudiante.**
Actores: Estudiante
Tipo: primario
Descripción: Un estudiante abre el paquete de software en la computadora, se identifica como usuario antiguo, ingresa su nombre y password; finalmente tiene acceso a los módulos que componen el paquete de software.

Caso de uso: **Desplegar tema del contenido teórico.**
Actores: Estudiante
Tipo: primario
Descripción: Un estudiante abre el paquete de software en la computadora, se identifica y accede al módulo de contenido teórico. Elige el tema en el contenido, lo lee y posteriormente cierra el módulo de contenido teórico y finalmente sale del paquete de software.

Caso de uso: **Graficar una función**
Actores: Estudiante
Tipo: primario
Descripción: Un estudiante abre el paquete de software en la computadora, se identifica y accede al módulo de graficación. Ingresa la expresión de las funciones y los parámetros de graficación; grafica la función y cierra el módulo de graficación, finalmente sale del paquete de software.

Caso de uso: **Graficar pares de datos**

Actores: Estudiante

Tipo: primario

Descripción: Un estudiante abre el paquete de software en la computadora, se identifica y accede al módulo de graficación. Ingresa los pares de datos y los parámetros de graficación; grafica la función y cierra el módulo de graficación, finalmente sale del paquete de software.

Caso de uso: **Realizar evaluación parcial**

Actores: Estudiante

Tipo: primario

Descripción: Un estudiante abre el paquete de software en la computadora, se identifica y accede al módulo de evaluación. Elige evaluación parcial y el tema para la evaluación, contesta todas las preguntas y se evalúa; lee el informe de evaluación y cierra el módulo de evaluación, finalmente sale del paquete de software.

Caso de uso: **Realizar evaluación Total**

Actores: Estudiante

Tipo: primario

Descripción: Un estudiante abre el paquete de software en la computadora, se identifica y accede al módulo de evaluación. Elige evaluación total y el tema para la evaluación, contesta todas las preguntas y se evalúa; lee el informe de evaluación y cierra el módulo de evaluación, finalmente sale del paquete de software.

Caso de uso: **Encontrar raíces de funciones no lineales**

Actores: Estudiante

Tipo: primario

Descripción: Un estudiante abre el paquete de software en la computadora, se identifica y accede al módulo de algoritmos. Elige raíces de funciones no lineales y el algoritmo que desea ejecutar; ingresa los parámetros del algoritmo enseguida

calcula los resultados y cierra el módulo de algoritmos, finalmente sale del paquete de software.

Caso de uso: **Encontrar raíces de un polinomio**
Actores: Estudiante
Tipo: primario
Descripción: Un estudiante abre el paquete de software en la computadora, se identifica y accede al módulo de algoritmos. Elige manipulación de polinomios y el algoritmo para encontrar raíces de un polinomio que desea ejecutar; ingresa los parámetros del algoritmo enseguida calcula los resultados y cierra el módulo de algoritmos, finalmente sale del paquete de software.

Caso de uso: **Resolver sistemas de ecuaciones lineales**
Actores: Estudiante
Tipo: primario
Descripción: Un estudiante abre el paquete de software en la computadora, se identifica y accede al módulo de algoritmos. Elige sistemas de ecuaciones lineales y el algoritmo que desea ejecutar; ingresa los parámetros del algoritmo enseguida calcula los resultados y cierra el módulo de algoritmos, finalmente sale del paquete de software.

Caso de uso: **Resolver sistemas de ecuaciones no lineales**
Actores: Estudiante
Tipo: primario
Descripción: Un estudiante abre el paquete de software en la computadora, se identifica y accede al módulo de algoritmos. Elige sistemas de ecuaciones no lineales y el algoritmo que desea ejecutar; ingresa los parámetros del algoritmo enseguida calcula los resultados y cierra el módulo de algoritmos, finalmente sale del paquete de software.

Caso de uso: **Realizar una regresión polinomial**

Actores: Estudiante

Tipo: primario

Descripción: Un estudiante abre el paquete de software en la computadora, se identifica y accede al módulo de algoritmos. Elige regresión polinomial y el algoritmo que desea ejecutar; ingresa los parámetros del algoritmo enseguida calcula los resultados y cierra el módulo de algoritmos, finalmente sale del paquete de software.

Caso de uso: **Realizar una interpolación polinomial**

Actores: Estudiante

Tipo: primario

Descripción: Un estudiante abre el paquete de software en la computadora, se identifica y accede al módulo de algoritmos. Elige interpolación polinomial y el algoritmo que desea ejecutar; ingresa los parámetros del algoritmo enseguida calcula los resultados y cierra el módulo de algoritmos, finalmente sale del paquete de software.

Caso de uso: **Calcular la derivada de una función**

Actores: Estudiante

Tipo: primario

Descripción: Un estudiante abre el paquete de software en la computadora, se identifica y accede al módulo de algoritmos. Elige diferenciación numérica y el algoritmo que desea ejecutar; ingresa los parámetros del algoritmo enseguida calcula los resultados y cierra el módulo de algoritmos, finalmente sale del paquete de software.

Caso de uso: **Calcular la Integral de una función**

Actores: Estudiante

Tipo: primario

Descripción: Un estudiante abre el paquete de software en la computadora, se identifica y accede al módulo de algoritmos. Elige integración numérica y el algoritmo que desea ejecutar; ingresa los parámetros del algoritmo enseguida

calcula los resultados y cierra el módulo de algoritmos, finalmente sale del paquete de software.

Caso de uso: **Resolver ecuaciones diferenciales ordinarias**
Actores: Estudiante
Tipo: primario
Descripción: Un estudiante abre el paquete de software en la computadora, se identifica y accede al módulo de algoritmos. Elige ecuaciones diferenciales ordinarias y el algoritmo que desea ejecutar; ingresa los parámetros del algoritmo enseguida calcula los resultados y cierra el módulo de algoritmos, finalmente sale del paquete de software.

Caso de uso: **Resolver un sistema de ecuaciones diferenciales ordinarias**
Actores: Estudiante
Tipo: primario
Descripción: Un estudiante abre el paquete de software en la computadora, se identifica y accede al módulo de algoritmos. Elige sistema de ecuaciones diferenciales ordinarias; ingresa los parámetros del algoritmo enseguida calcula los resultados y cierra el módulo de algoritmos, finalmente sale del paquete de software.

Caso de uso: **Resolver ecuaciones diferenciales parciales**
Actores: Estudiante
Tipo: primario
Descripción: Un estudiante abre el paquete de software en la computadora, se identifica y accede al módulo de algoritmos. Elige ecuaciones diferenciales parciales y el algoritmo que desea ejecutar; ingresa los parámetros del algoritmo enseguida calcula los resultados y cierra el módulo de algoritmos, finalmente sale del paquete de software.

Casos de uso expandidos

Caso de Uso: **Identificar estudiante**

Actores: Estudiante

Propósito: Identificar el tipo de estudiante que ingresa al paquete de software para regístrarlo.

Visión General: Un estudiante abre el paquete de software en la computadora, se identifica como usuario antiguo, ingresa su nombre y password; finalmente tiene acceso a los módulos que componen el paquete de software

Tipo: primario y esencial

Referencias: *Funciones:* R1.1a la R1.6

Curso Típico de Eventos:

Acción del Actor	Respuesta del Sistema
1. Este caso de uso empieza cuando el estudiante abre el paquete de software.	2. Pide el tipo de estudiante
3. Elige estudiante antiguo	4. Pide el nombre y el password
5. Introduce el nombre y el password	6. Permite el ingreso al los módulos.

Cursos Alternativos:

- Línea 3: Elige estudiante nuevo. Ingresa el nombre y password para su identificación posterior y se le permite el ingreso a los módulos.
- Línea 6: El nombre o el password es incorrecto. Se indica el error y se pide el ingreso nuevamente.

Caso de Uso: **Leer un tema del contenido teórico**

Actores: Estudiante

Propósito: Acceder al contenido de texto de un tema

Visión General: Un estudiante abre el paquete de software en la computadora, se identifica y accede al módulo de contenido teórico. Elige el tema en el contenido, lo lee y posteriormente cierra el módulo de contenido teórico y finalmente sale del paquete de software.

Tipo: primario y esencial

Referencias: *Funciones:* R2.1a la R2.8; *Caso de uso*: Leer tema del contenido teórico

Curso Típico de Eventos:

Acción del Actor	Respuesta del Sistema
1. Este caso de uso empieza cuando el estudiante abre el paquete de software.	2. Pide identificación
3. Se identifica	4. Permite el ingreso a los módulos
5. Elige módulo de contenido teórico	6. Presenta el contenido del tutorial y la primera hoja del mismo.
7. Selecciona el tema a leer	8. Presenta el texto del tema elegido.
9. Cierra el módulo de contenido teórico	10. Presenta pantalla con los diferentes módulos.
11. Sale del paquete de software.	

Cursos Alternativos:

- Línea 7: accede a la opción de búsqueda, ingresa el o las palabras a buscar y selecciona el tema.
- Línea 9: Selecciona imprimir el tema activo, cierra el módulo de contenido teórico y sale del paquete de software.

Caso de Uso: **Graficar una función**

Actores: Estudiante

Propósito: Graficar una función ingresada por teclado.

Visión General: Un estudiante abre el paquete de software en la computadora, se identifica y accede al módulo de graficación. Ingresa la expresión de la función y los parámetros de graficación; grafica la función y cierra el módulo de graficación, finalmente sale del paquete de software.

Tipo: primario y esencial

Referencias: *Funciones:* R3.1a la R3.20; *Caso de uso*: Identificar estudiante

Curso Típico de Eventos:

Acción del Actor	Respuesta del Sistema
1. Este caso de uso empieza cuando el estudiante abre el paquete de software.	2. Pide identificación
3. Se identifica	4. Permite el ingreso a los módulos
5. Elige módulo de Graficación	6. Presenta la pantalla de graficación para el ingreso de los parámetros de graficación.
7. Ingresa los parámetros de graficación Y selecciona graficar	8. Muestra el gráfico de la función.

9. Selecciona el tipo de operación:
   a. Cambiar parámetros
   b. Ampliar gráfico.
   c. Guardar el gráfico a un archivo.
   d. Imprimir pantalla

10. Realiza la operación

11. Cierra el módulo de graficación

12. Presenta pantalla con los diferentes módulos.

13. Sale del paquete de software.

Cursos Alternativos:

- Línea 8: Parámetros de graficación incompletos. Muestra mensaje de completación de parámetros

- Sección: Ampliar Gráfico

Acción del Actor	Respuesta del Sistema
1. Selecciona con el mouse la parte del gráfico a ser ampliada.	2. Muestra el gráfico ampliado

Caso de Uso: **Realizar evaluación parcial**

Actores: Estudiante

Propósito: Realizar una evaluación de los conocimientos adquiridos.

Visión General: Un estudiante abre el paquete de software en la computadora, se identifica y accede al módulo de evaluación. Elige evaluación parcial y el tema para la evaluación, contesta todas las preguntas y se evalúa; lee el informe de evaluación y cierra el módulo de evaluación, finalmente sale del paquete de software.

Tipo: primario y esencial

Referencias: *Funciones:* R4.1a la R4.12; *Caso de uso*: Identificar estudiante

Curso Típico de Eventos:

Acción del Actor	Respuesta del Sistema
1. Este caso de uso empieza cuando el estudiante abre el paquete de software.	2. Pide identificación
3. Se identifica	4. Permite el ingreso a los módulos
5. Elige módulo de evaluación	6. Pide elegir el tipo de evaluación.
7. Elige evaluación parcial	8. Muestra las preguntas.
9. Contesta las preguntas: a. Dentro del tiempo permitido. b. Fuera del tiempo permitido.	
	10. Muestra el informe de evaluación
11. Cierra el módulo de contenido teórico	12. Presenta pantalla con los diferentes módulos.
13. Sale del paquete de software.	

Cursos Alternativos:

- Línea 11: imprime informe de evaluación y luego cierra el módulo de contenido teórico y sale del paquete de software.

- Sección: Dentro del tiempo permitido

Acción del Actor	Respuesta del Sistema
1. Contesta todas las preguntas dentro del tiempo permitido para la evaluación	2. Muestra el informe de evaluación.

Cursos Alternativos:

- Línea 1: no contesta todas las preguntas. Se presenta un mensaje para que el estudiante responda a las preguntas que le faltan por contestar.

- Sección: Fuera del tiempo permitido

Acción del Actor	Respuesta del Sistema
1. Contesta todas las preguntas fuera del tiempo permitido para la evaluación	2. Muestra el informe de evaluación.

Cursos Alternativos:

- Línea 1: no contesta todas las preguntas dentro del tiempo permitido. Se presenta un mensaje para que el estudiante responda a las preguntas que le faltan por contestar.

Caso de Uso: **Encontrar raíces de funciones no lineales**
Actores: Estudiante
Propósito: Encontrar las raíces de una función no lineal ingresada por teclado
Visión General: Un estudiante abre el paquete de software en la computadora, se identifica y accede al módulo de algoritmos. Elige raíces de funciones no lineales y el algoritmo que desea ejecutar; ingresa los parámetros del algoritmo enseguida calcula los resultados y cierra el módulo de algoritmos, finalmente sale del paquete de software.

Tipo: primario y esencial

Referencias: *Funciones:* R5.1a la R5.14; *Caso de uso*: Identificar estudiante

Curso Típico de Eventos:

Acción del Actor	Respuesta del Sistema
1. Este caso de uso empieza cuando el estudiante abre el paquete de software.	2. Pide identificación
3. Se identifica	4. Permite el ingreso a los módulos
5. Elige módulo de algoritmos	6. Pide el elegir el tema del algoritmo.
7. Elige raíces de funciones no lineales	8. Muestra los tipos de algoritmos.
9. Selecciona el tipo de algoritmo: a. Aproximaciones sucesivas. b. Bisección. c. Falsa posición. d. Primer orden. e. Newton-Raphson f. Secante.	
	10. Muestra la pantalla de algoritmos para que ingrese los parámetros tanto del algoritmo seleccionado como los de graficación.
10. Ingresa los parámetros del algoritmo y los de graficación.	
	12. Muestra los resultados del algoritmo y el gráfico.

13. Selecciona el tipo de operación
    a. Cambiar parámetros del algoritmo.
    b. Cambiar parámetros de graficación.
    c. Ampliar el gráfico.
    d. Mostrar resultado de las iteraciones.
    e. Guardar los resultados y el gráfico.
    f. Imprimir pantalla.

14. Realiza la operación

15. Cierra el módulo de contenido teórico

16. Presenta pantalla con los diferentes módulos.

13. Sale del paquete de software.

Cursos Alternativos:

- Línea 11: Parámetros del algoritmo o de graficación incompletos. Muestra mensaje de completación de parámetros
- Sección: Mostrar resultados de las iteraciones.

Acción del Actor	Respuesta del Sistema
1. Selecciona mostrar resultados de las Iteraciones.	2. Muestra en pantalla los resultados de las iteraciones.

Caso de Uso: **Resolver sistemas de ecuaciones lineales**

Actores: Estudiante

Propósito: Encontrar la solución de un sistema de ecuaciones lineales.

Visión General: Un estudiante abre el paquete de software en la computadora, se identifica y accede al módulo de algoritmos. Elige sistemas de ecuaciones no lineales y el algoritmo que desea ejecutar; ingresa los parámetros del algoritmo enseguida calcula los resultados y cierra el módulo de algoritmos, finalmente sale del paquete de software.

Tipo: primario y esencial

Referencias: *Funciones:* R5.1a la R5.14; *Caso de uso*: Identificar estudiante

Curso Típico de Eventos:

Acción del Actor	Respuesta del Sistema
1. Este caso de uso empieza cuando el estudiante abre el paquete de software.	2. Pide identificación
3. Se identifica	4. Permite el ingreso a los módulos
5. Elige módulo de algoritmos	6. Pide el elegir el tema del algoritmo.
7. Elige sistemas de ecuaciones lineales	8. Muestra los tipos de algoritmos.
9. Selecciona el tipo de algoritmo: a. Eliminación gaussiana. b. Gauss-Jordan c. Factorización Criterio de Doolittle. d. Factorización Criterio de Choleski. e. Jacobi. f. Gauss-Seidel.	

10. Muestra la pantalla de algoritmos para que ingrese los parámetros tanto del algoritmo seleccionado como los de graficación.

11. Ingresa los parámetros del algoritmo y los de graficación.

12. Muestra los resultados del algoritmo y el gráfico.

14. Selecciona el tipo de operación

a. Cambiar parámetros del algoritmo.
b. Cambiar parámetros de graficación.
c. Ampliar el gráfico.
d. Mostrar resultado de las iteraciones.
e. Guardar los resultados y el gráfico.
f. Imprimir pantalla.

14. Realiza la operación

15. Cierra el módulo de contenido teórico

16. Presenta pantalla con los diferentes módulos.

13. Sale del paquete de software.

Cursos Alternativos:

- Línea 11: Parámetros del algoritmo o de graficación incompletos. Muestra mensaje de completación de parámetros

- Sección: Mostrar resultados de las iteraciones.

Acción del Actor	Respuesta del Sistema
1. Selecciona mostrar resultados de las Iteraciones.	2. Muestra en pantalla los resultados de las iteraciones.

1.3.3.3 Diagrama de Casos de uso.

Un diagrama de casos de uso explica gráficamente un conjunto de casos de uso de un sistema, los actores y la relación entre estos y los casos de uso.

En la Fig. 1.20 a y b se muestra el diagrama de casos de uso para el Paquete de Software Didáctico para la Enseñanza-Aprendizaje de Métodos Numéricos, el mismo que se ha dividido en dos por su amplitud y las recomendaciones indican que en un diagrama de casos de uso no deben representarse mas de 15 casos de uso.

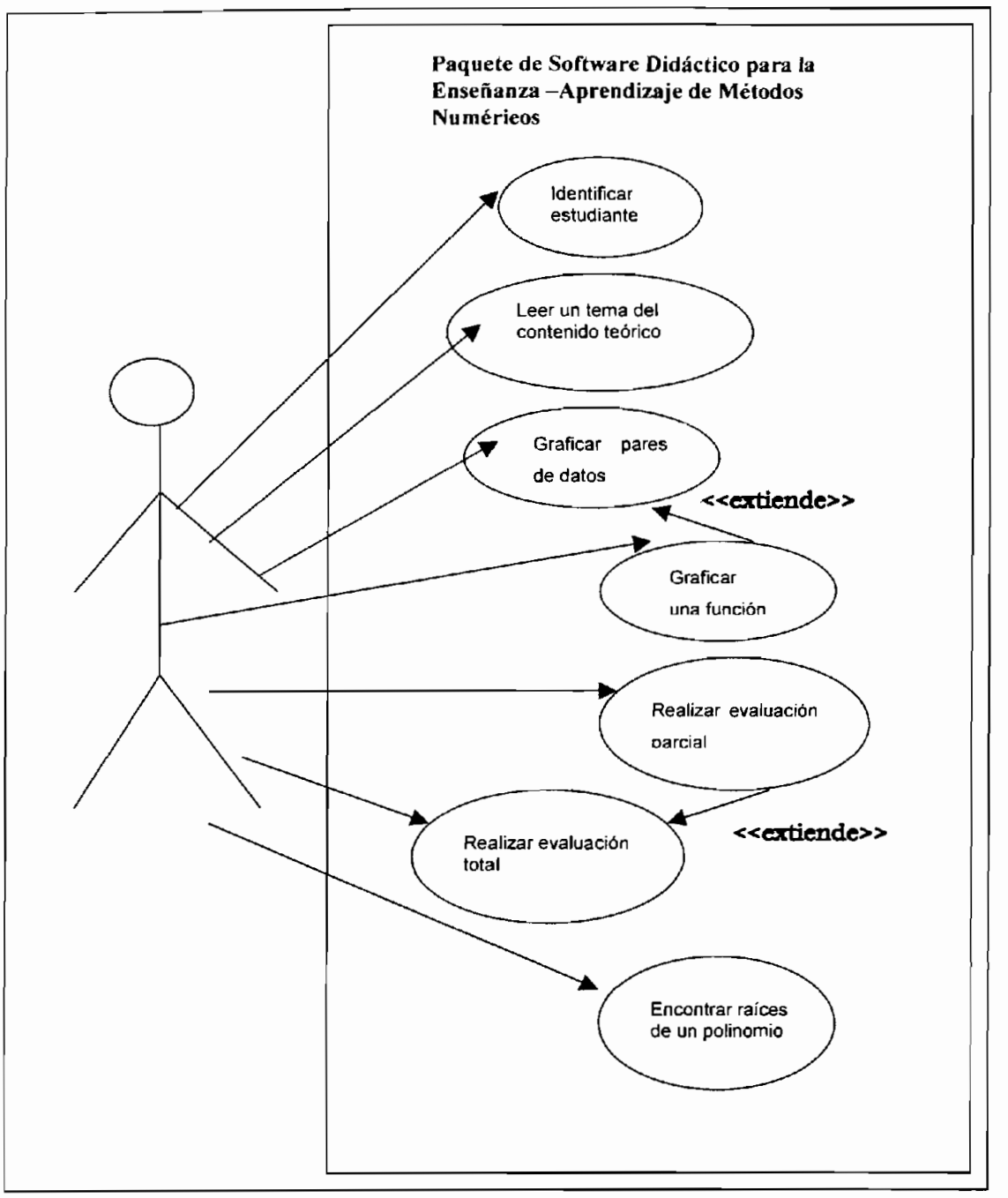


Fig. 1.20 a Diagrama de Casos de Uso del Paquete de Software Didáctico

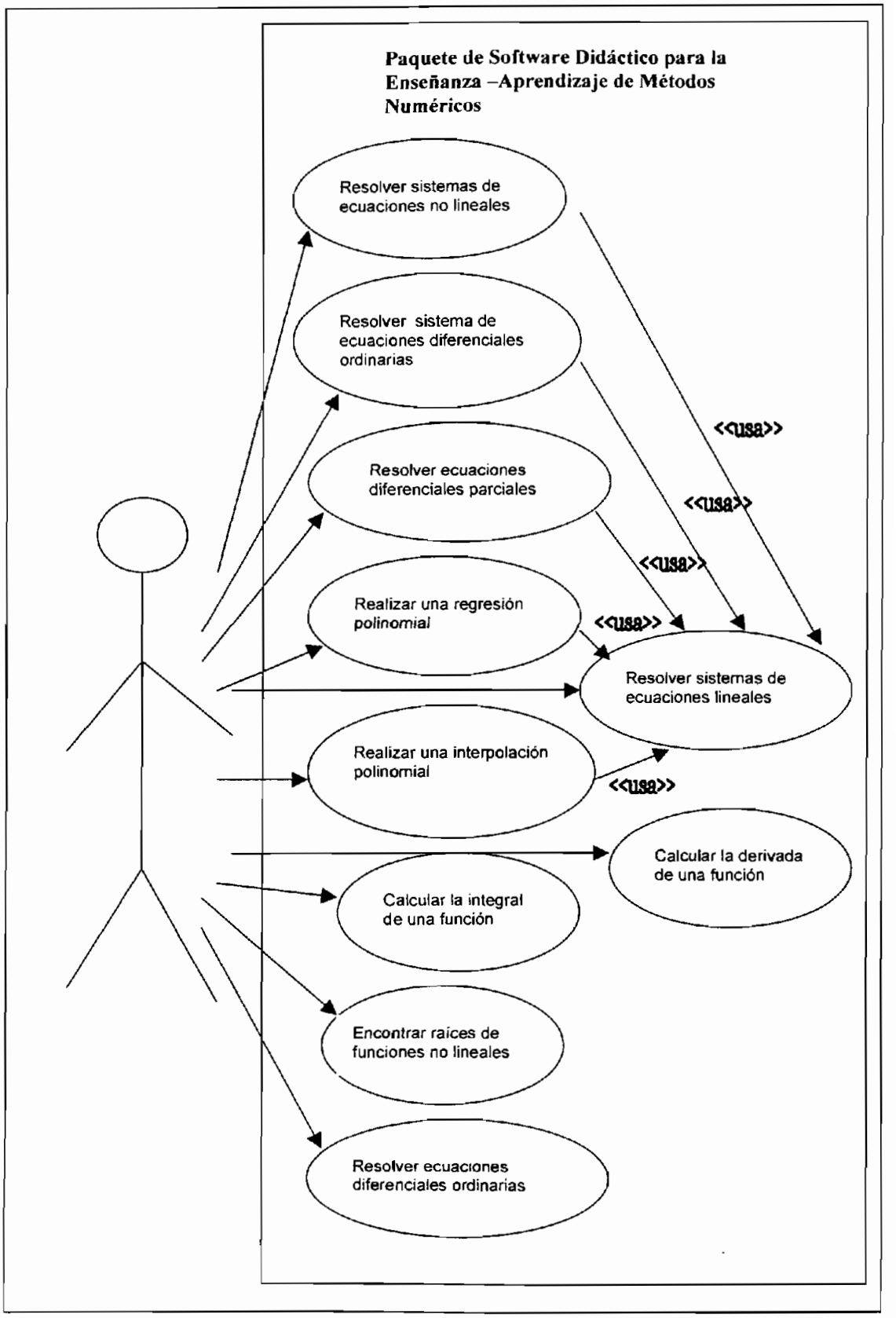


Fig. 1.20 b Diagrama de Casos de Uso del Paquete de Software Didáctico

1.3.3.4 Planificación de los casos de uso según ciclos de desarrollo.

Para tomar la decisión de qué casos de uso se van a tratar primero es necesario ordenarlos según prioridad. Las características de un caso de uso específico que van a hacer que un caso de uso tenga una prioridad alta son las siguientes:

a. Impacto significativo en el diseño de la arquitectura. Por ejemplo, si aporta muchas clases al modelo del dominio o requiere persistencia en los datos.
b. Se obtiene una mejor comprensión del diseño con un nivel de esfuerzo relativamente bajo.
c. Incluye funciones complejas, críticas en el tiempo o de nivel elevado de riesgo.
d. Implica bien un trabajo de investigación significante, o bien el uso de una tecnología nueva o arriesgada.
e. Representa un proceso de gran importancia en el sistema a modelar.

Para realizar la clasificación se asignará a cada caso de uso una valoración numérica de 1 a 5 para cada una de las características anteriores, para conseguir una puntuación total se aplicará una ponderación del 1 al 5 a cada característica.

En la siguiente tabla se muestra clasificación de los casos de uso:

Ponderación para cada Característica	3	2	4	1	3	**Suma**
Caso de uso	a	b	c	d	e	
Identificar estudiante	5	4	1	1	5	43
Desplegar un tema del contenido teórico	5	4	4	4	5	58
Graficar una función	4	3	5	5	4	55
Graficar pares de datos	4	3	3	5	4	47
Realizar evaluación parcial	5	4	3	5	5	55
Realizar evaluación total	5	4	3	3	5	53
Encontrar raíces de funcione no lineales	4	5	3	2	3	45
Encontrar raíces de un polinomio	4	5	4	3	3	50

Resolver sistemas de ecuaciones no lineales	5	3	5	4	3	54
Resolver sistemas de ecuaciones lineales	5	4	5	5	3	57
Realizar una regresión polinomial	4	3	3	3	3	42
Realizar una interpolación polinomial	4	3	4	3	3	46
Calcular la derivada de una función	4	4	4	3	3	48
Calcular la integral de una función	4	3	3	4	3	43
Resolver ecuaciones diferenciales ordinarias	4	2	4	4	3	45
Resolver un sistema de ecuaciones diferenciales ordinarias	5	1	4	5	3	47
Resolver ecuaciones diferenciales parciales	4	2	4	4	3	45

En la siguiente lista se ordena los casos de uso de mayor a menor de acuerdo a la tabla anterior:

- Desplegar un tema del contenido teórico
- Resolver sistemas de ecuaciones lineales
- Graficar una función
- Realizar evaluación parcial
- Resolver sistemas de ecuaciones no lineales
- Realizar evaluación total
- Encontrar raíces de un polinomio
- Calcular la derivada de una función
- Graficar pares de datos
- Resolver sistema de ecuaciones diferenciales ordinarias
- Realizar una interpolación polinomial
- Identificar estudiante
- Resolver ecuaciones diferenciales ordinarias
- Resolver ecuaciones diferenciales parciales

- Encontrar raíces de funciones no lineales
- Calcular la integral de una función
- Realizar una regresión polinomial

Siempre se va a tener un caso de uso *inicialización*, aunque no tenga una prioridad alta en la clasificación anterior pero interesa que éste sea desarrollado desde el principio, por lo que nuestro caso de inicialización será identificar estudiante. Luego de completado el ciclo de desarrollo para este caso de uso se seguirá con los tres casos de uso siguientes según la lista anterior para el nuevo ciclo de desarrollo y se seguirá así hasta completar todos los casos de uso de la lista.

El ciclo de desarrollo para cada uno de los casos de uso será detallado en los capítulos siguientes que abarcan las etapas de análisis, diseño, implementación y pruebas.

CAPÍTULO 2

ANÁLISIS

2.1 ACTIVIDADES DE LA FASE DE ANÁLISIS

En el capítulo anterior que corresponde a la Fase de Planeación y Elaboración, se obtuvo el documento de especificación de requerimientos del Paquete de Software Didáctico para la Enseñanza-Aprendizaje de Métodos Numéricos, además se planificó los casos de uso para cada ciclo de desarrollo que serán abordados en este y los posteriores capítulos. La Fig. 2.1 muestra una visión global del desarrollo del software y el cuadro pintado de celeste indica en que momento de este nos encontramos.

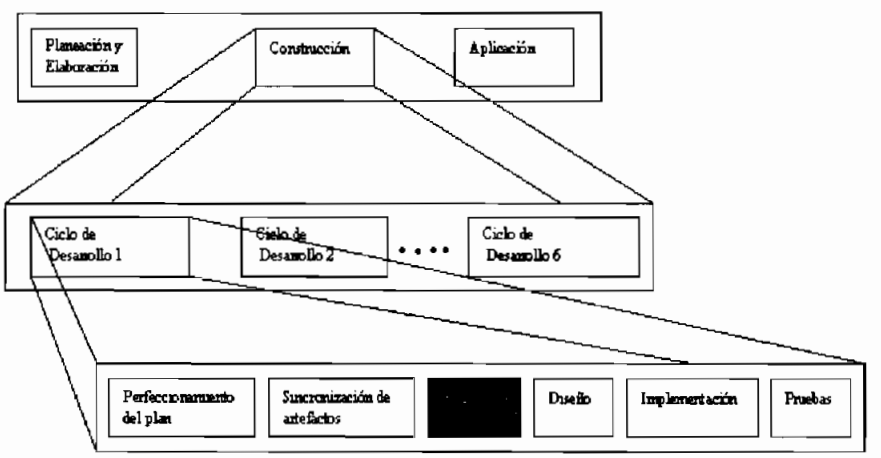


Fig. 2.1 Visión global del desarrollo de software iterativo

La Fig. 2.1 muestra claramente lo iterativo del método de desarrollo escogido y se observa que la fase de construcción se forma de varios ciclos de desarrollo en nuestro caso son seis, para cada ciclo de desarrollo es necesario realizar una serie de actividades como son Perfeccionamiento del plan, sincronización de artefactos, análisis, diseño, implementación, pruebas.

Las dos primeras corresponden al hecho de ir realizando para cada ciclo de desarrollo nuevo un mejoramiento en los diagramas que se generan en cada etapa de análisis y diseño, los que han variado por la forma como estos han sido implementados.

En la fase de Análisis de un ciclo de desarrollo se *investiga* sobre el problema, sobre los conceptos relacionados con el subconjunto de casos de uso que se esté tratando. Se intenta llegar a una buena comprensión del problema por parte del equipo de desarrollo, sin entrar en cómo va a ser la solución en cuanto a detalles de implementación.

Cuando el ciclo de desarrollo no es el primero, antes de la fase de Análisis hay una serie de actividades de planificación. Estas actividades consisten en actualizar los modelos que se tengan según lo que se haya implementado, pues siempre se producen desviaciones entre lo que se ha analizado y diseñado y lo que finalmente se construye. Una vez se tienen los modelos acordes con lo implementado se empieza el nuevo ciclo de desarrollo con la fase de Análisis.

En esta fase se trabaja con los modelos de Análisis construidos en la fase anterior, ampliándolos con los conceptos correspondientes a los casos de uso que se traten en el ciclo de desarrollo actual.

Las actividades de la fase de Análisis son las siguientes:

1. Definir Casos de Uso Esenciales en formato expandido. (*si no están definidos*)
2. Refinar los Diagramas de Casos de Uso.
3. Refinar el Modelo Conceptual.
4. Refinar el Glosario (Anexo)
5. Definir los Diagramas de Secuencia del Sistema.
6. Definir Contratos de Operación.

7. Definir Diagramas de Estados.

Para modelar el comportamiento del sistema pueden usarse los Diagramas de Estados, éstos se puede aplicar al comportamiento de los siguientes elementos:

- Una clase software.
- Un concepto.
- Un caso de uso.

El uso de Diagramas de Estados es opcional. Tan solo se usa cuando nos ayudan a expresar mejor el comportamiento del elemento descrito, por lo tanto en vista de que el uso de este diagrama es opcional y no ayuda en mejorar mayormente la comprensión del sistema no se lo realizará.

2.2 CASOS DE USO ESCENCIALES EN FORMATO EXPANDIDO

Tomando en consideración que los casos de uso: Identificar estudiante, desplegar un tema del contenido teórico, Resolver sistemas de ecuaciones lineales, Graficar una función, Realizar evaluación parcial, Resolver sistemas de ecuaciones no lineales ya han sido realizados en formato expandido, se procederá a realizar los formatos expandidos para el resto de los casos de uso de acuerdo al ciclo de desarrollo en el cual están siendo analizados.

Caso de Uso: **Realizar evaluación parcial**
Actores: Estudiante
Propósito: Realizar una evaluación de los conocimientos adquiridos.
Visión General: Un estudiante abre el paquete de software en la computadora, se identifica y accede al módulo de evaluación. Elige evaluación parcial y el tema para la evaluación, contesta todas las preguntas y se evalúa; lee el informe de evaluación y cierra el módulo de evaluación, finalmente sale del paquete de software.

Tipo: primario y esencial

Referencias: *Funciones:* R4.1a la R4.12; *Caso de uso*: Identificar estudiante

Curso Típico de Eventos:

Acción del Actor	Respuesta del Sistema
1. Este caso de uso empieza cuando el estudiante abre el paquete de software.	2. Pide identificación
3. Se identifica	4. Permite el ingreso a los módulos
5. Elige módulo de evaluación	6. Pide elegir el tipo de evaluación.
7. Elige evaluación Total	8. Muestra las preguntas.
9. Contesta las preguntas: a. Dentro del tiempo permitido. b. Fuera del tiempo permitido.	
	10. Muestra el informe de evaluación
11. Cierra el módulo de contenido teórico	12. Presenta pantalla con los diferentes módulos.
13. Sale del paquete de software.	

Cursos Alternativos:

- Línea 11: imprime informe de evaluación y luego cierra el módulo de contenido teórico y sale del paquete de software.

- Sección: Dentro del tiempo permitido

Acción del Actor	Respuesta del Sistema
1. Contesta todas las preguntas dentro del tiempo permitido para la evaluación	2. Muestra el informe de evaluación.

Cursos Alternativos:

- Línea 1: no contesta todas las preguntas. Se presenta un mensaje para que el estudiante responda a las preguntas que le faltan por contestar.

- Sección: Fuera del tiempo permitido

Acción del Actor	Respuesta del Sistema
1. Contesta todas las preguntas fuera del tiempo permitido para la evaluación	2. Muestra el informe de evaluación.

Cursos Alternativos:

Línea 1: no contesta todas las preguntas dentro del tiempo permitido. Se presenta un mensaje para que el estudiante responda a las preguntas que le faltan por contestar.

Caso de Uso: **Encontrar raíces de un polinomio**

Actores: Estudiante

Propósito: Encontrar las raíces de un polinomio ingresado por teclado

Visión General: Un estudiante abre el paquete de software en la computadora, se identifica y accede al módulo de algoritmos. Elige manipulación de polinomios y el algoritmo para encontrar raíces de un polinomio que desea ejecutar; ingresa los parámetros del algoritmo enseguida calcula los resultados y cierra el módulo de algoritmos, finalmente sale del paquete de software.

Tipo: primario y esencial
Referencias: *Funciones:* R5.1a la R5.14; *Caso de uso*: Identificar estudiante
Curso Típico de Eventos:

Acción del Actor	Respuesta del Sistema
1. Este caso de uso empieza cuando el estudiante abre el paquete de software.	2. Pide identificación
3. Se identifica	4. Permite el ingreso a los módulos
5. Elige módulo de algoritmos	6. Pide el elegir el tema del algoritmo.
7. Elige raíces de polinomio	8. Muestra los tipos de algoritmos.
9. Selecciona el tipo de algoritmo: a. Horner. b. Newton-Horner. c. Newton-Bairstow. d. Aplicación de Newton-Bairstow.	

10. Muestra la pantalla de algoritmos para que ingrese los parámetros tanto del algoritmo seleccionado como los de graficación.

11. Ingresa los parámetros del algoritmo y los de graficación.

12. Muestra los resultados del algoritmo y el gráfico.

13. Selecciona el tipo de operación
    a. Cambiar parámetros del algoritmo.
    b. Cambiar parámetros de graficación.
    c. Ampliar el gráfico.
    d. Mostrar resultado de las iteraciones.
    e. Guardar los resultados y el gráfico.
    f. Imprimir pantalla.

14. Realiza la operación

15. Cierra el módulo de contenido teórico

16. Presenta pantalla con los diferentes módulos.

13. Sale del paquete de software.

Cursos Alternativos:

- Línea 11: Parámetros del algoritmo o de graficación incompletos. Muestra mensaje de completación de parámetros

- Sección: Mostrar resultados de las iteraciones.

Acción del Actor	Respuesta del Sistema
1. Selecciona mostrar resultados de las Iteraciones.	2. Muestra en pantalla los resultados de las iteraciones.

Caso de Uso: **Calcular la derivada de una función**
Actores: Estudiante
Propósito: Calcular la derivada de una función ingresado por teclado
Visión General: Un estudiante abre el paquete de software en la computadora, se identifica y accede al módulo de algoritmos. Elige diferenciación numérica y el algoritmo que desea ejecutar; ingresa los parámetros del algoritmo enseguida calcula los resultados y cierra el módulo de algoritmos, finalmente sale del paquete de software.
Tipo: primario y esencial
Referencias: *Funciones:* R5.1a la R5.14; *Caso de uso*: Identificar estudiante
Curso Típico de Eventos:

Acción del Actor	Respuesta del Sistema
1. Este caso de uso empieza cuando el estudiante abre el paquete de software.	2. Pide identificación
3. Se identifica	4. Permite el ingreso a los módulos
5. Elige módulo de algoritmos	6. Pide el elegir el tema del algoritmo.
7. Elige raíces de polinomio	8. Muestra los tipos de algoritmos.

9. Selecciona el tipo de algoritmo:

a. Diferencias centradas
b. Diferencias progresivas
c. Diferencias regresivas
d. Derivada del polinomio de interpolación de Newton.

10. Muestra la pantalla de algoritmos para que ingrese los parámetros tanto del algoritmo seleccionado como los de graficación.

11. Ingresa los parámetros del algoritmo y los de graficación.

12. Muestra los resultados del algoritmo y el gráfico.

13. Selecciona el tipo de operación

a. Cambiar parámetros del algoritmo.
b. Cambiar parámetros de graficación.
c. Ampliar el gráfico.
d. Mostrar resultado de las iteraciones.
e. Guardar los resultados y el gráfico.
f. Imprimir pantalla.

14. Realiza la operación

15. Cierra el módulo de contenido teórico

16. Presenta pantalla con los diferentes módulos.

17. Sale del paquete de software.

Cursos Alternativos:

- Línea 11: Parámetros del algoritmo o de graficación incompletos. Muestra mensaje de completación de parámetros

- Sección: Mostrar resultados de las iteraciones.

Acción del Actor	Respuesta del Sistema
1. Selecciona mostrar resultados de las Iteraciones.	2. Muestra en pantalla los resultados de las iteraciones.

Caso de Uso: **Graficar pares de datos**

Actores: Estudiante

Propósito: Graficar pares de datos ingresados por teclado.

Visión General: Un estudiante abre el paquete de software en la computadora, se identifica y accede al módulo de graficación. Ingresa los pares de datos y los parámetros de graficación; gráfica la función y cierra el módulo de graficación, finalmente sale del paquete de software.

Tipo: primario y esencial

Referencias: *Funciones:* R3.1a la R3.20; *Caso de uso*: Identificar estudiante

Curso Típico de Eventos:

Acción del Actor	Respuesta del Sistema
1. Este caso de uso empieza cuando el estudiante abre el paquete de software.	2. Pide identificación
3. Se identifica	4. Permite el ingreso a los módulos
5. Elige módulo de Graficación	6. Presenta la pantalla de graficación para el ingreso de los parámetros de graficación.
7. Ingresa los parámetros de graficación, pares de datos y selecciona graficar	8. Muestra el gráfico de la función.
9. Selecciona el tipo de operación: a. Cambiar parámetros b. Ampliar gráfico. c. Guardar el gráfico a un archivo. d. Imprimir pantalla	10. Realiza la operación
11. Cierra el módulo de graficación	12. Presenta pantalla con los diferentes módulos.
13. Sale del paquete de software.	

Cursos Alternativos:

- Línea 8: Parámetros de graficación incompletos. Muestra mensaje de completación de parámetros

- Sección: Ampliar Gráfico

Acción del Actor	Respuesta del Sistema
1. Selecciona con el mouse la parte del gráfico a ser ampliada.	2. Muestra el gráfico ampliado

Caso de Uso: **Resolver sistema de ecuaciones diferenciales ordinarias**
Actores: Estudiante
Propósito: Resolver un sistema de ecuaciones diferenciales ordinarias ingresadas por teclado.
Visión General Un estudiante abre el paquete de software en la computadora, se identifica y accede al módulo de algoritmos. Elige sistema de ecuaciones diferenciales ordinarias; ingresa los parámetros del algoritmo enseguida calcula los resultados y cierra el módulo de algoritmos, finalmente sale del paquete de software.
Tipo: primario y esencial
Referencias: *Funciones:* R5.1a la R5.14; *Caso de uso*: Identificar estudiante
Curso Típico de Eventos:

Acción del Actor	Respuesta del Sistema
1. Este caso de uso empieza cuando el estudiante abre el paquete de software.	2. Pide identificación
3. Se identifica	4. Permite el ingreso a los módulos
5. Elige módulo de algoritmos	6. Pide el elegir el tema del algoritmo.
7. Elige raíces de polinomio	8. Muestra los tipos de algoritmos.
9. Selecciona el algoritmo	

10. Muestra la pantalla de algoritmos para que ingrese los parámetros tanto del algoritmo como los de graficación.

11. Ingresa los parámetros del algoritmo y los de graficación.

12. Muestra los resultados del algoritmo y el gráfico.

13. Selecciona el tipo de operación

a. Cambiar parámetros del algoritmo.
b. Cambiar parámetros de graficación.
c. Ampliar el gráfico.
d. Mostrar resultado de las iteraciones.
e. Guardar los resultados y el gráfico.
f. Imprimir pantalla.

14. Realiza la operación

15. Cierra el módulo de contenido teórico

16. Presenta pantalla con los diferentes módulos.

17. Sale del paquete de software.

Cursos Alternativos:

- Línea 11: Parámetros del algoritmo o de graficación incompletos. Muestra mensaje de completación de parámetros

- Sección: Mostrar resultados de las iteraciones.

Acción del Actor	Respuesta del Sistema
1. Selecciona mostrar resultados de las Iteraciones.	2. Muestra en pantalla los resultados de las iteraciones.

Caso de Uso: **Realizar una interpolación polinomial**

Actores: Estudiante

Propósito: Realizar una interpolación polinomial de pares de datos ingresado por teclado

Visión General: Un estudiante abre el paquete de software en la computadora, se identifica y accede al módulo de algoritmos. Elige interpolación polinomial y el algoritmo que desea ejecutar; ingresa los parámetros del algoritmo enseguida calcula los resultados y cierra el módulo de algoritmos, finalmente sale del paquete de software.

Tipo: primario y esencial

Referencias: *Funciones:* R5.1a la R5.14; *Caso de uso*: Identificar estudiante

Curso Típico de Eventos:

Acción del Actor	Respuesta del Sistema
1. Este caso de uso empieza cuando el estudiante abre el paquete de software.	2. Pide identificación
3. Se identifica	4. Permite el ingreso a los módulos
5. Elige módulo de algoritmos	6. Pide el elegir el tema del algoritmo.
7. Elige raíces de polinomio	8. Muestra los tipos de algoritmos.

9. Selecciona el tipo de algoritmo:

a. Técnica Matricial
b. Polinomio de Lagrange
c. Polinomio de Newton
d. Trigonométrica.
e. Segmentaria

10. Muestra la pantalla de algoritmos para que ingrese los parámetros tanto del algoritmo seleccionado como los de graficación.

11. Ingresa los parámetros del algoritmo y los de graficación.

12. Muestra los resultados del algoritmo y el gráfico.

13. Selecciona el tipo de operación

a. Cambiar parámetros del algoritmo.
b. Cambiar parámetros de graficación.
c. Ampliar el gráfico.
d. Mostrar resultado de las iteraciones.
e. Guardar los resultados y el gráfico.
f. Imprimir pantalla.

14. Realiza la operación

15. Cierra el módulo de contenido teórico	16. Presenta pantalla con los diferentes módulos.
17. Sale del paquete de software.	

Cursos Alternativos:

- Línea 11: Parámetros del algoritmo o de graficación incompletos. Muestra mensaje de completación de parámetros
- Sección: Mostrar resultados de las iteraciones.

Acción del Actor	Respuesta del Sistema
1. Selecciona mostrar resultados de las Iteraciones.	2. Muestra en pantalla los resultados de las iteraciones.

Caso de Uso: **Resolver ecuaciones diferenciales ordinarias**

Actores: Estudiante

Propósito: Resolver ecuaciones diferenciales ordinarias ingresado por teclado

Visión General: Un estudiante abre el paquete de software en la computadora, se identifica y accede al módulo de algoritmos. Elige ecuaciones diferenciales ordinarias y el algoritmo que desea ejecutar; ingresa los parámetros del algoritmo enseguida calcula los resultados y cierra el módulo de algoritmos, finalmente sale del paquete de software.

Tipo: primario y esencial

Referencias: *Funciones:* R5.1a la R5.14; *Caso de uso*: Identificar estudiante

Curso Típico de Eventos:

Acción del Actor	Respuesta del Sistema
1. Este caso de uso empieza cuando el estudiante abre el paquete de software.	2. Pide identificación
3. Se identifica	4. Permite el ingreso a los módulos
5. Elige módulo de algoritmos	6. Pide el elegir el tema del algoritmo.
7. Elige raíces de polinomio	8. Muestra los tipos de algoritmos.
9. Selecciona el tipo de algoritmo: a. Euler b. Euler modificado c. Heun d. Runge-Kutta e. Adams-Bashforth-Moulton f. Milene-Simpson	
	10. Muestra la pantalla de algoritmos para que ingrese los parámetros tanto del algoritmo seleccionado como los de graficación.
11.Ingresa los parámetros del algoritmo y los de graficación.	12. Muestra los resultados del algoritmo y el gráfico.

13. Selecciona el tipo de operación

a. Cambiar parámetros del algoritmo.
b. Cambiar parámetros de graficación.
c. Ampliar el gráfico.
d. Mostrar resultado de las iteraciones.
e. Guardar los resultados y el gráfico.
f. Imprimir pantalla.

14. Realiza la operación

15. Cierra el módulo de contenido teórico

16. Presenta pantalla con los diferentes módulos.

17. Sale del paquete de software.

Cursos Alternativos:

- Línea 11: Parámetros del algoritmo o de graficación incompletos. Muestra mensaje de completación de parámetros
- Sección: Mostrar resultados de las iteraciones.

Acción del Actor	Respuesta del Sistema
1. Selecciona mostrar resultados de las Iteraciones.	2. Muestra en pantalla los resultados de las iteraciones.

Caso de Uso: **Resolver ecuaciones diferenciales parciales**

Actores: Estudiante

Propósito: Resolver ecuaciones diferenciales parciales ingresado por teclado

Visión General: Un estudiante abre el paquete de software en la computadora, se identifica y accede al módulo de algoritmos. Elige ecuaciones diferenciales parciales y el algoritmo que desea ejecutar; ingresa los parámetros del algoritmo enseguida calcula los resultados y cierra el módulo de algoritmos, finalmente sale del paquete de software.

Tipo: primario y esencial

Referencias: *Funciones:* R5.1a la R5.14; *Caso de uso*: Identificar estudiante

Curso Típico de Eventos:

Acción del Actor	Respuesta del Sistema
2. Este caso de uso empieza cuando el estudiante abre el paquete de software.	2. Pide identificación
3. Se identifica	4. Permite el ingreso a los módulos
5. Elige módulo de algoritmos	6. Pide el elegir el tema del algoritmo.
7. Elige raíces de polinomio	8. Muestra los tipos de algoritmos.
9. Selecciona el tipo de algoritmo: a. Diferencias finitas b. Explícito c. Implícito d. Crank-Nicholson	

10. Muestra la pantalla de algoritmos para que ingrese los parámetros tanto del algoritmo seleccionado como los de graficación.

11. Ingresa los parámetros del algoritmo y los de graficación.

12. Muestra los resultados del algoritmo y el gráfico.

13. Selecciona el tipo de operación

a. Cambiar parámetros del algoritmo.
b. Cambiar parámetros de graficación.
c. Ampliar el gráfico.
d. Mostrar resultado de las iteraciones.
e. Guardar los resultados y el gráfico.
f. Imprimir pantalla.

14. Realiza la operación

15. Cierra el módulo de contenido teórico

16. Presenta pantalla con los diferentes módulos.

17. Sale del paquete de software.

Cursos Alternativos:

- Línea 11: Parámetros del algoritmo o de graficación incompletos. Muestra mensaje de completación de parámetros
- Sección: Mostrar resultados de las iteraciones.

Acción del Actor	Respuesta del Sistema
1. Selecciona mostrar resultados de las Iteraciones.	2. Muestra en pantalla los resultados de las iteraciones.

Caso de Uso: **Encontrar raíces de funciones no lineales**
Actores: Estudiante
Propósito: Encontrar raíces de funciones no lineales ingresado por teclado
Visión General: Un estudiante abre el paquete de software en la computadora, se identifica y accede al módulo de algoritmos. Elige raíces de funciones no lineales y el algoritmo que desea ejecutar; ingresa los parámetros del algoritmo enseguida calcula los resultados y cierra el módulo de algoritmos, finalmente sale del paquete de software.
Tipo: primario y esencial
Referencias: *Funciones:* R5.1a la R5.14; *Caso de uso*: Identificar estudiante
Curso Típico de Eventos:

Acción del Actor	Respuesta del Sistema
1. Este caso de uso empieza cuando el estudiante abre el paquete de software.	2. Pide identificación

3. Se identifica

4. Permite el ingreso a los módulos

5. Elige módulo de algoritmos

6. Pide el elegir el tema del algoritmo.

7. Elige raíces de polinomio

8. Muestra los tipos de algoritmos.

9. Selecciona el tipo de algoritmo:

a. Aproximaciones Sucesivas
b. Bisección
c. Falsa Posición
d. Newton-Raphson
e. Secante

10. Muestra la pantalla de algoritmos para que ingrese los parámetros tanto del algoritmo seleccionado como los de graficación.

11. Ingresa los parámetros del algoritmo y los de graficación.

12. Muestra los resultados del algoritmo y el gráfico.

13. Selecciona el tipo de operación

a. Cambiar parámetros del algoritmo.
b. Cambiar parámetros de graficación.
c. Ampliar el gráfico.
d. Mostrar resultado de las iteraciones.
e. Guardar los resultados y el gráfico.
f. Imprimir pantalla.

14. Realiza la operación

15. Cierra el módulo de contenido teórico	16. Presenta pantalla con los diferentes módulos.
17. Sale del paquete de software.	

Cursos Alternativos:

- Línea 11: Parámetros del algoritmo o de graficación incompletos. Muestra mensaje de completación de parámetros
- Sección: Mostrar resultados de las iteraciones.

Acción del Actor	Respuesta del Sistema
1. Selecciona mostrar resultados de las Iteraciones.	2. Muestra en pantalla los resultados de las iteraciones.

Caso de Uso: **Calcular la integral de una función**

Actores: Estudiante

Propósito: Encontrar raíces de funciones no lineales ingresado por teclado

Visión General: Un estudiante abre el paquete de software en la computadora, se identifica y accede al módulo de algoritmos. Elige integración numérica y el algoritmo que desea ejecutar; ingresa los parámetros del algoritmo enseguida calcula los resultados y cierra el módulo de algoritmos, finalmente sale del paquete de software.

Tipo: primario y esencial

Referencias: *Funciones:* R5.1a la R5.14; *Caso de uso*: Identificar estudiante

Curso Típico de Eventos:

Acción del Actor	Respuesta del Sistema
1. Este caso de uso empieza cuando el estudiante abre el paquete de software.	2. Pide identificación
3. Se identifica	4. Permite el ingreso a los módulos
5. Elige módulo de algoritmos	6. Pide el elegir el tema del algoritmo.
7. Elige raíces de polinomio	8. Muestra los tipos de algoritmos.
9. Selecciona el tipo de algoritmo: a. Newton-Cotes cerrada b. Newton-Cotes abierta c. Regla extendida del Trapecio d. Regla extendida de Simpson e. Integración Gaussiana	
	10. Muestra la pantalla de algoritmos para que ingrese los parámetros tanto del algoritmo seleccionado como los de graficación.
11.Ingresa los parámetros del algoritmo y los de graficación.	
	12. Muestra los resultados del algoritmo y el gráfico.

13. Selecciona el tipo de operación

a. Cambiar parámetros del algoritmo.
b. Cambiar parámetros de graficación.
c. Ampliar el gráfico.
d. Mostrar resultado de las iteraciones.
e. Guardar los resultados y el gráfico.
f. Imprimir pantalla.

14. Realiza la operación

15. Cierra el módulo de contenido teórico

16. Presenta pantalla con los diferentes módulos.

17. Sale del paquete de software.

Cursos Alternativos:

- Línea 11: Parámetros del algoritmo o de graficación incompletos. Muestra mensaje de completación de parámetros
- Sección: Mostrar resultados de las iteraciones.

Acción del Actor	Respuesta del Sistema
1. Selecciona mostrar resultados de las Iteraciones.	2. Muestra en pantalla los resultados de las iteraciones.

Caso de Uso: **Realizar una regresión polinomial**

Actores: Estudiante

Propósito: Realizar una regresión polinomial de datos ingresados por teclado.

Visión General: Un estudiante abre el paquete de software en la computadora, se identifica y accede al módulo de algoritmos. Elige regresión polinomial y el algoritmo que desea ejecutar; ingresa los parámetros del algoritmo enseguida calcula los resultados y cierra el módulo de algoritmos, finalmente sale del paquete de software.

Tipo: primario y esencial

Referencias: *Funciones:* R5.1a la R5.14; *Caso de uso*: Identificar estudiante

Curso Típico de Eventos:

Acción del Actor	Respuesta del Sistema
1. Este caso de uso empieza cuando el estudiante abre el paquete de software.	2. Pide identificación
3. Se identifica	4. Permite el ingreso a los módulos
5. Elige módulo de algoritmos	6. Pide el elegir el tema del algoritmo.
7. Elige Regresión Polinomial	8. Muestra el algoritmo.
9. Selecciona Regresión Polinomial	
	10. Muestra la pantalla del algoritmo para que ingrese los parámetros tanto del algoritmo seleccionado como los de graficación.

11. Ingresa los parámetros del algoritmo y los de graficación.	12. Muestra los resultados del algoritmo y el gráfico.
13. Selecciona el tipo de operación a. Cambiar parámetros del algoritmo. b. Cambiar parámetros de graficación. c. Ampliar el gráfico. d. Mostrar resultado de las iteraciones. e. Guardar los resultados y el gráfico. f. Imprimir pantalla.	
	14. Realiza la operación
15. Cierra el módulo de contenido teórico	16. Presenta pantalla con los diferentes módulos.
17. Sale del paquete de software.	

Cursos Alternativos:

- Línea 11: Parámetros del algoritmo o de graficación incompletos. Muestra mensaje de completación de parámetros
- Sección: Mostrar resultados de las iteraciones.

Acción del Actor	Respuesta del Sistema
1. Selecciona mostrar resultados de las Iteraciones.	2. Muestra en pantalla los resultados de las iteraciones.

2.3 DIAGRAMAS DE CASO DE USO REFINADOS

La implementación del método de Craig Larman nos involucra en un desarrollo del software iterativo, el mismo que permite ir mejorando en cada ciclo de desarrollo los diagramas que se realizaron en la etapa.

Como consecuencia de aquello, el cambio en el diagrama de casos de uso deberá reflejarse en un diagrama para cada ciclo de desarrollo. Debido a que el proceso de desarrollo de las etapas Diseño e Implementación no han modificado el conjunto de casos de uso del sistema el Diagrama de Casos de Uso de la Fig. 1.20 a y Fig.1.20 b del capítulo 1 se mantiene.

2.4 MODELO CONCEPTUAL

Una parte de la investigación sobre el dominio del problema consiste en identificar los conceptos que lo conforman. Para representar estos conceptos se va a usar un Diagrama de Estructura Estática de UML, al que se va a llamar Modelo Conceptual.

En el Modelo Conceptual se tiene una representación de conceptos del mundo real, no de componentes software. El objetivo de la creación de un Modelo Conceptual es aumentar la comprensión del problema.

Para crear el Modelo Conceptual se siguen los siguientes pasos:

1. Hacer una lista de conceptos candidatos, búsqueda de sustantivos relacionados con los requisitos en consideración en este ciclo.
2. Representarlos en un diagrama.
3. Añadir las asociaciones necesarias para ilustrar las relaciones entre conceptos que es necesario conocer.

4. Añadir los atributos necesarios para contener toda la información que se necesite conocer de cada concepto.

Como se explicó en el capítulo 1, el Modelo de Craig Larman permite posponer la creación de un Modelo Conceptual Borrador a fases posteriores, por lo tanto es aquí, donde se realizará este Modelo Conceptual Borrador, ya que éste toma importancia por que permite tener una visión de los conceptos del mundo real que intervienen en el desarrollo del software y además servirá como base para la construcción de los modelos conceptuales de los diferentes ciclos de desarrollo.

Tomando en consideración la Especificación de Requerimientos del Paquete de Software Didáctico para la Enseñanza-Aprendizaje de Métodos Numéricos y las recomendaciones para crear el modelo conceptual enumeradas anteriormente se tiene el Modelo Conceptual Borrador mostrado en la Fig. 2.2 en el cual no se incluyen los atributos, las relaciones y asociaciones porque éstas serán analizadas para cada ciclo de desarrollo.

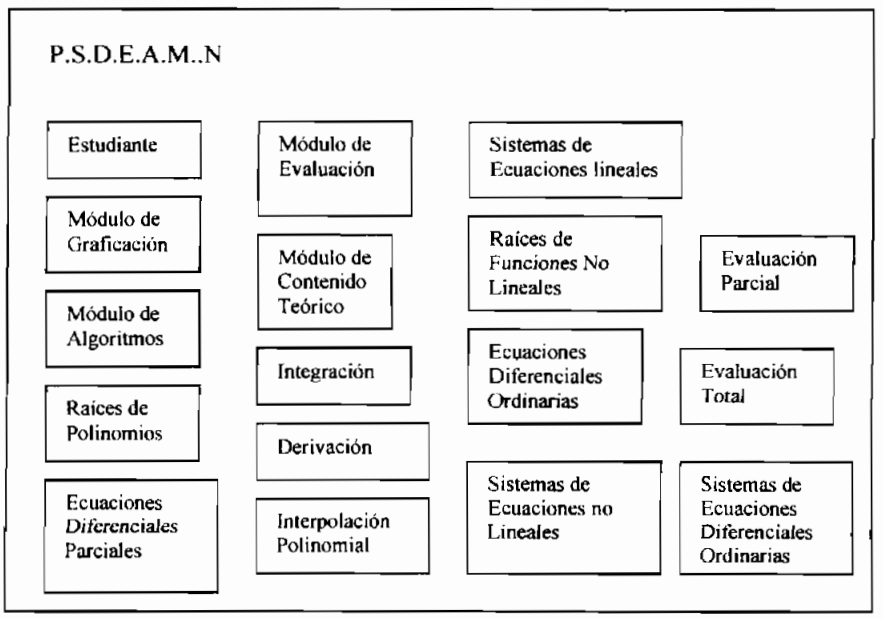


Fig. 2.2 Modelo Conceptual Borrador

2.4.1 CREACIÓN DEL MODELO CONCEPTUAL (1)

Para este ciclo de desarrollo en el cual el caso de uso Identificar estudiante es el analizado se tiene la siguiente lista de conceptos candidatos:

- Estudiante
- Nombre
- Password
- Módulos
- Tipo estudiante

Debido a que estudiante representa a una persona que desea aprender y en vista que el paquete puede ser usado por una persona experta en el tema se cambia por usuario para estar más acorde, del mismo modo sustantivo nombre y password también se cambia por clave ya que son atributos de ésto; tipo estudiante y módulos se desechan debido a que la primera es más un atributo del usuario y la segunda no tiene un significado trascendental en la creación del Modelo Conceptual.

Por lo que, el Modelo Conceptual será el siguiente:

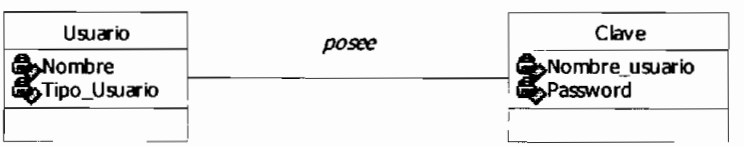


Fig. 2.3 Modelo Conceptual para caso de uso Identificar Estudiante

2.4.2 CREACIÓN DEL MODELO CONCEPTUAL (2)

En este ciclo de desarrollo se tiene la siguiente lista de conceptos candidatos de acuerdo a cada caso de uso:

Leer un tema del contenido teórico:

- Paquete de Software
- Estudiante
- Módulos
- Contenido Teórico
- Tópicos
- Texto
- Tutorial
- Capítulo

Paquete de Software, tutorial y módulos se desechan porque no tienen un significado trascendental en la creación del Modelo Conceptual, Estudiante se cambia por el análisis anterior y por lo tanto se acepta solamente Contenido Teórico ya que las demás son más atributos de éste.
Por lo que el Modelo Conceptual será el siguiente:

Fig.2.4 Modelo Conceptual para caso de uso Leer un Tema del Contenido Teórico

Resolver sistemas de ecuaciones lineales:

- Paquete de Software
- Estudiante
- Módulo
- Sistemas de ecuaciones lineales
- Tema del Algoritmo
- Parámetros del algoritmo

Paquete de Software, Módulos se desechan y Estudiante se cambia por el análisis anterior, además Tema del Algoritmo se cambia por tipo de algoritmo, por consiguiente se acepta solamente Sistemas de ecuaciones lineales ya que las demás son más atributos de éste.

Por lo que el Modelo Conceptual será el siguiente:

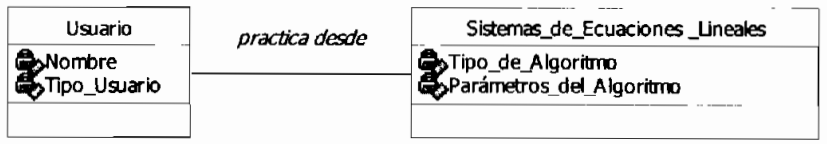


Fig.2.5 Modelo Conceptual para caso de uso Resolver Sistemas de Ecuaciones Lineales.

Graficar una función:

- Paquete de Software
- Estudiante
- Funciones
- Parámetros de Graficación
- Módulos
- Gráfico

Paquete de Software, Módulos se desechan y Estudiante se cambia por el análisis anterior, por consiguiente se acepta solamente Gráfico, pero se cambia a Graficación por que tiene un significado más extenso, los demás son más atributos de éste. Además como graficar una función extiende al caso de uso graficar pares de datos se realizaran conjuntamente en este ciclo de desarrollo.

Por lo que el Modelo Conceptual será el siguiente:

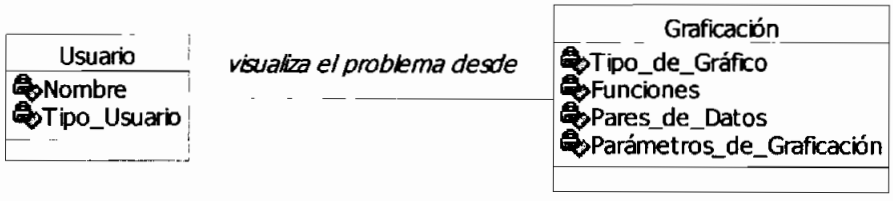


Fig.2.6 Modelo Conceptual para caso de uso Graficar una Función

2.4.3 CREACIÓN DEL MODELO CONCEPTUAL (3)

En este ciclo de desarrollo se tiene la siguiente lista de conceptos candidatos de acuerdo a cada caso de uso:

Realizar evaluación parcial

- Paquete de Software
- Estudiante
- Módulos
- Pregunta
- Informe de Evaluación

Paquete de Software, Módulos se desechan y Estudiante se cambia por el análisis anterior, por consiguiente se aceptan Pregunta e Informe de Evaluación. Como el caso de uso Realizar evaluación parcial y el caso de uso Realizar evaluación total

presentan listas de conceptos candidatos similares permite que éstas sean tomadas juntas ya que ambas serian procedimientos de un concepto mayor al que se le llamará evaluación.

Por lo que el Modelo Conceptual será el siguiente:

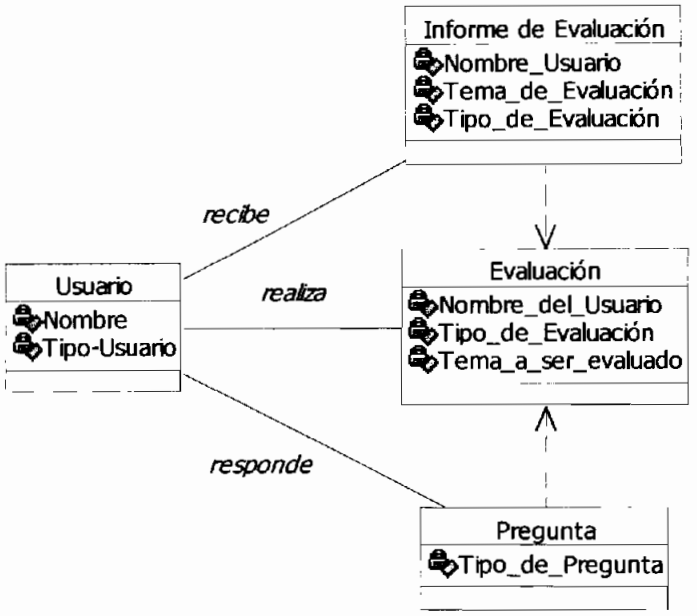


Fig.2.7 Modelo Conceptual para caso de uso Evaluación Parcial

Resolver sistemas de ecuaciones no lineales

- Paquete de Software
- Estudiante
- Módulo
- Sistemas de ecuaciones no lineales
- Tema del Algoritmo
- Parámetros del algoritmo

Paquete de Software, Módulos se desechan y Estudiante se cambia por el análisis anterior, además Tema del Algoritmo se cambia por tipo de algoritmo, por consiguiente se acepta solamente Sistemas de ecuaciones no lineales ya que las demás son más atributos de éste.

Por lo que el Modelo Conceptual será el siguiente:

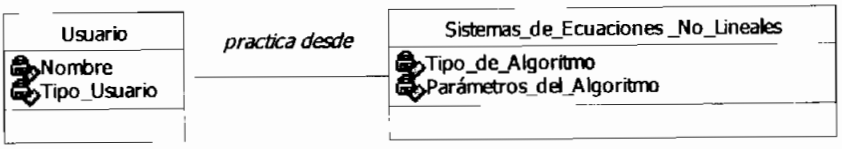


Fig.2.8 Modelo Conceptual para caso de uso Resolver Sistemas de Ecuaciones No Lineales.

2.4.4 CREACIÓN DEL MODELO CONCEPTUAL (4)

En este ciclo de desarrollo se tiene la siguiente lista de conceptos candidatos de acuerdo a cada caso de uso:

Encontrar raíces de un polinomio

- Paquete de Software
- Estudiante
- Módulo
- Raíces de un Polinomio
- Algoritmo de Horner
- Algoritmo de Newton-Horner
- Algoritmo de Newton-Baristow
- Aplicación de Newton-Bairstow
- Tema del Algoritmo
- Parámetros del algoritmo

Paquete de Software, Módulos se desechan y Estudiante se cambia por el análisis anterior, además Tema del Algoritmo se cambia por tipo de algoritmo, y Raíces de un Polinomio por Manipulación de Polinomios ya que no solo se encuentran raíces del polinomio sino que también se evalúan tanto en un argumento real como en uno

complejo, los diferentes tipos de algoritmos pueden ser tratados como procedimientos de este objeto. Parámetros del algoritmo es aceptado.
Por lo que el Modelo Conceptual será el siguiente:

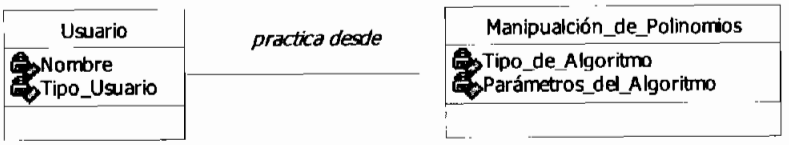


Fig.2.9 Modelo Conceptual para caso de uso Encontrar Raíces de un Polinomio

Calcular la derivada de una función

- Paquete de Software
- Estudiante
- Módulo
- Derivada de una función
- Diferencias Centrales
- Diferencias Progresivas
- Diferencias Regresivas
- Tema del Algoritmo
- Parámetros del algoritmo

Paquete de Software, Módulos se desechan y Estudiante se cambia por el análisis anterior, además Tema del Algoritmo se cambia por tipo de algoritmo, y derivada de una función por Derivación, los diferentes tipos de algoritmos pueden ser tratados como procedimientos de este objeto. Parámetros del algoritmo es aceptado.
Por lo que el Modelo Conceptual será el siguiente:

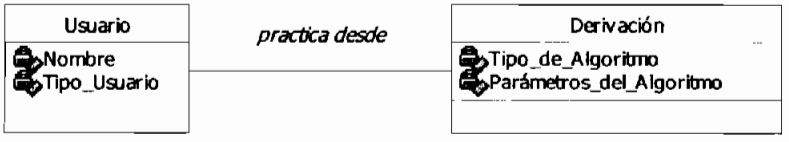


Fig.2.10 Modelo Conceptual para caso de uso Calcular la Derivada de una Función

2.4.5 CREACIÓN DEL MODELO CONCEPTUAL (5)

En este ciclo de desarrollo se tiene la siguiente lista de conceptos candidatos de acuerdo a cada caso de uso:

Resolver sistema de ecuaciones diferenciales ordinarias

- Paquete de Software
- Estudiante
- Módulo
- Sistemas de ecuaciones diferenciales ordinarias
- Runge-Kutta de cuarto orden
- Tema del Algoritmo
- Parámetros del algoritmo

Paquete de Software, Módulos se desechan y Estudiante se cambia por el análisis anterior, además Tema del Algoritmo se cambia por tipo de algoritmo, el algoritmo puede ser tratado como procedimiento de este objeto. Parámetros del algoritmo es aceptado.

Por lo que el Modelo Conceptual será el siguiente:

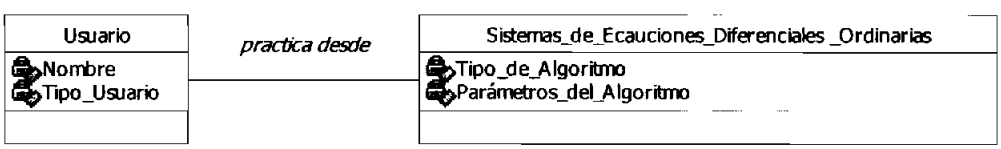


Fig.2.11 Modelo Conceptual para caso de uso Resolver Sistemas de Ecuaciones Diferenciales Ordinarias.

Realizar una interpolación polinomial

- Paquete de Software
- Estudiante
- Módulo
- Interpolación polinomial
- Técnica Matricial
- Polinomio de Lagrange
- Polinomio de Newton
- Trigonométrica
- Segmentaria
- Tema del Algoritmo
- Parámetros del algoritmo

Paquete de Software, Módulos se desechan y Estudiante se cambia por el análisis anterior, además Tema del Algoritmo se cambia por tipo de algoritmo, debido a que tiene afinidad la interpolación polinomial con la regresión y además ésta consta de un solo algoritmo se decide agregarla a este objeto. Los diferentes tipos de algoritmos pueden ser tratados como procedimientos de este objeto y el concepto Parámetros del algoritmo es aceptado.

Por lo que el Modelo Conceptual será el siguiente:

Fig.2.12 Modelo Conceptual para caso de uso Realizar una Interpolación Polinomial.

Resolver ecuaciones diferenciales ordinarias

- Paquete de Software
- Estudiante
- Módulo
- Ecuaciones diferenciales ordinarias
- Euler
- Euler Modificado
- Heun
- Runge-Kutta
- Adams-Bashforth-Moulton
- Milene Simpson
- Tema del Algoritmo
- Parámetros del algoritmo

Paquete de Software, Módulos se desechan y Estudiante se cambia por el análisis anterior, además Tema del Algoritmo se cambia por tipo de algoritmo, los diferentes tipos de algoritmos pueden ser tratados como procedimientos del objeto ecuaciones diferenciales ordinarias. Parámetros del algoritmo es aceptado.

Por lo que el Modelo Conceptual será el siguiente:

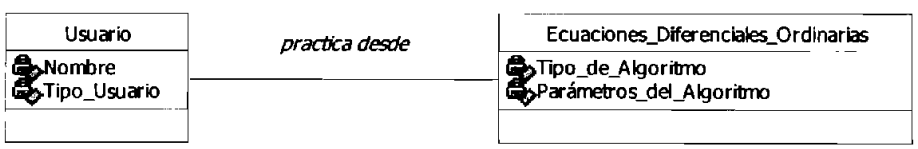


Fig.2.13 Modelo Conceptual para caso de uso Resolver Ecuaciones Diferenciales Ordinarias.

2.4.6 CREACIÓN DEL MODELO CONCEPTUAL (6)

En este ciclo de desarrollo se tiene la siguiente lista de conceptos candidatos de acuerdo a cada caso de uso:

Resolver ecuaciones diferenciales parciales

- Paquete de Software
- Estudiante
- Módulo
- Ecuaciones diferenciales ordinarias
- Diferencias finitas
- Método Explícito
- Método Implícito
- Aplicación de Crank-Nicholson
- Tema del Algoritmo
- Parámetros del algoritmo

Paquete de Software, Módulos se desechan y Estudiante se cambia por el análisis anterior, además Tema del Algoritmo se cambia por tipo de algoritmo, los diferentes tipos de algoritmos pueden ser tratados como procedimientos del objeto ecuaciones diferenciales parciales. Parámetros del algoritmo es aceptado.
Por lo que el Modelo Conceptual será el siguiente:

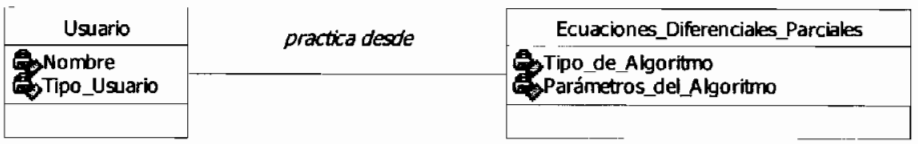


Fig.2.14 Modelo Conceptual para caso de uso Resolver Ecuaciones Diferenciales Parciales.

Encontrar raíces de funciones no lineales

- Paquete de Software
- Estudiante
- Módulo
- Raíces de funciones no lineales
- Aproximaciones sucesivas
- Bisección
- Falsa posición
- Newton Raphson
- Secante
- Tema del Algoritmo
- Parámetros del algoritmo

Paquete de Software, Módulos se desechan y Estudiante se cambia por el análisis anterior, además Tema del Algoritmo se cambia por tipo de algoritmo, los diferentes tipos de algoritmos pueden ser tratados como procedimientos del objeto Raíces de funciones no lineales. Parámetros del algoritmo es aceptado.

Por lo que el Modelo Conceptual será el siguiente:

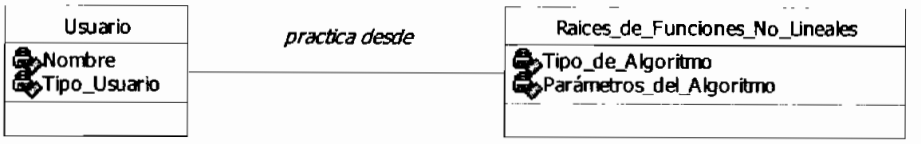


Fig.2.15 Modelo Conceptual para caso de Encontrar Raíces de Funciones No Lineales.

Calcular la integral de una función

- Paquete de Software
- Estudiante
- Módulo
- Integral de una función
- Newton-Cotes cerrada
- Newton-Cotes abierta
- Regla extendida del Trapecio
- Regla extendida de Simpson
- Integración Gaussiana
- Tema del Algoritmo
- Parámetros del algoritmo

Paquete de Software, Módulos se desechan y Estudiante se cambia por el análisis anterior, además Tema del Algoritmo se cambia por tipo de algoritmo, y Integración de funciones por Integración, los diferentes tipos de algoritmos pueden ser tratados como procedimientos de este objeto. Parámetros del algoritmo es aceptado.

Por lo que el Modelo Conceptual será el siguiente:

Fig.2.16 Modelo Conceptual para caso de uso Calcular la Integral de una Función.

Realizar una Regresión polinomial

- Paquete de Software
- Estudiante
- Módulo
- Regresión Polinomial
- Parámetros del algoritmo

Paquete de Software, Módulos se desechan y Estudiante se cambia por el análisis anterior. El algoritmo de Regresión Polinomial puede ser tratado como procedimientos del Objeto Regresión Polinomial Parámetros del algoritmo es aceptado.

Por lo que el Modelo Conceptual será el siguiente:

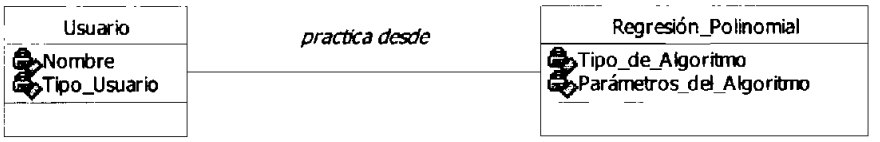


Fig.2.17 Modelo Conceptual para caso de uso Realizar una Regresión Polinomial.

Como consecuencia del refinamiento de los anteriores modelos conceptuales se tiene el modelo conceptual del sistema el mismo que se muestra en la Fig. 2.17.

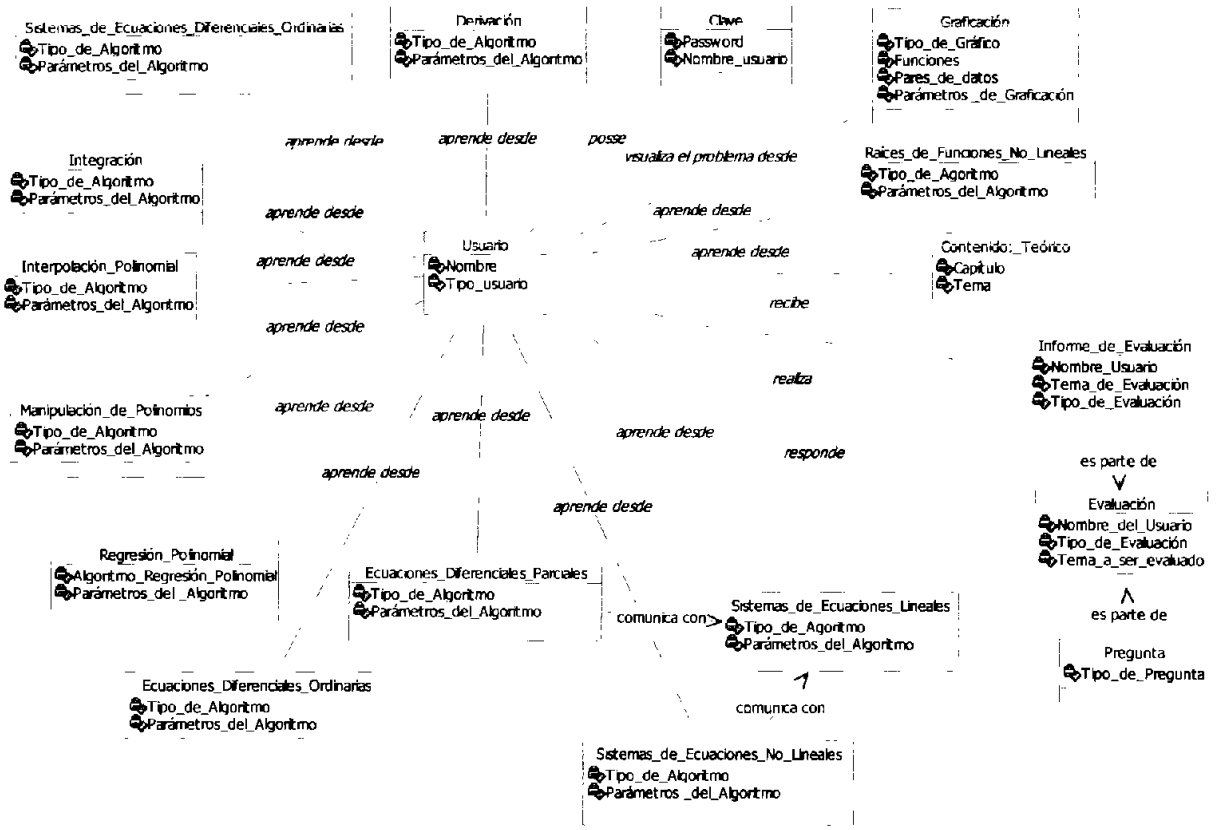


Fig. 2.18 Modelo Conceptual del Paquete de Software Didáctico para la Enseñanza- Aprendizaje de Métodos Numéricos.

2.5 DIAGRAMAS DE SECUENCIA

Además de investigar sobre los conceptos del sistema y su estructura, también es preciso investigar en el Análisis sobre el comportamiento del sistema, visto éste como una caja negra. Una parte de la descripción del comportamiento del sistema se realiza mediante los Diagramas de Secuencia del Sistema.

En cada caso de uso se muestra una interacción de actores con el sistema. En esta interacción, los actores generan eventos, solicitando al sistema operaciones. Por ejemplo, en el caso de una reserva de un billete de avión, el empleado de la agencia de viajes solicita al sistema de reservas que realice una reserva. El evento que supone esa solicitud inicia una operación en el sistema de reservas.

Un Diagrama de Secuencia de Sistema se representa usando la notación para diagramas de secuencia de UML. En él se muestra para un escenario particular de un caso de uso, los eventos que los actores generan, su orden, y los eventos que se intercambian entre sistemas.

Para cada caso de uso que se esté tratando se realiza un diagrama para el curso típico de eventos, y además se realiza un diagrama para los cursos alternativos de mayor interés.

En la Fig. 2.18 se muestra el Diagrama de Secuencia del Sistema para el caso de uso Realizar Reintegro de un cajero automático.

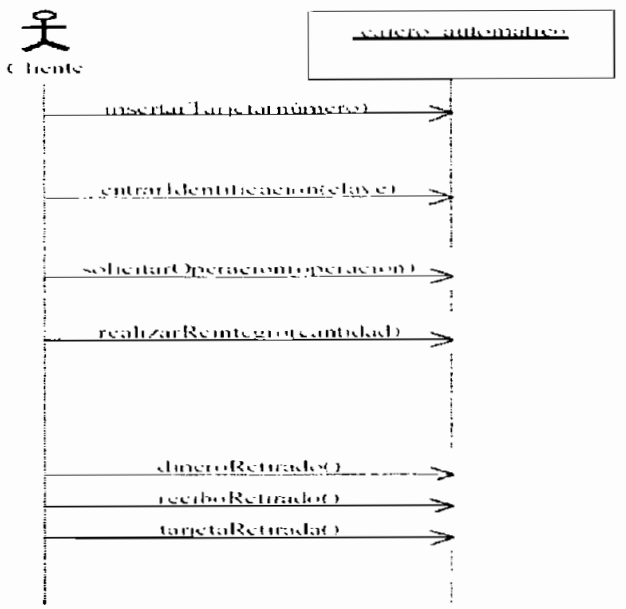


Fig. 2.19 Diagrama de Secuencia del Sistema para el caso de uso Realizar Reintegro de un cajero automático.

Construcción de un Diagrama de Secuencia del Sistema

Para construir un Diagrama de Secuencia del Sistema para el curso típico de eventos de un caso de uso, se siguen los siguientes pasos:

1. Representar el sistema como un objeto con una línea debajo.
2. Identificar los actores que directamente operan con el sistema, y dibujar una línea para cada uno de ellos.
3. Partiendo del texto del curso típico de eventos del caso de uso, identificar los eventos (externos) del sistema que cada actor genera y representarlos en el diagrama.

4. Opcionalmente, incluir el texto del caso de uso en el margen del diagrama.

Los eventos del sistema deberían expresarse en base a la noción de operación que representan, en vez de en base a la interfaz particular.

A continuación se desarrollan los diagramas de secuencia para cada caso de uso en su ciclo de desarrollo respectivo. No se coloca el texto del caso de uso al margen del diagrama por cuanto éste es muy extenso, además este requisito es opcional.

2.5.1 DIAGRAMAS DE SECUENCIA (1)

Identificar estudiante

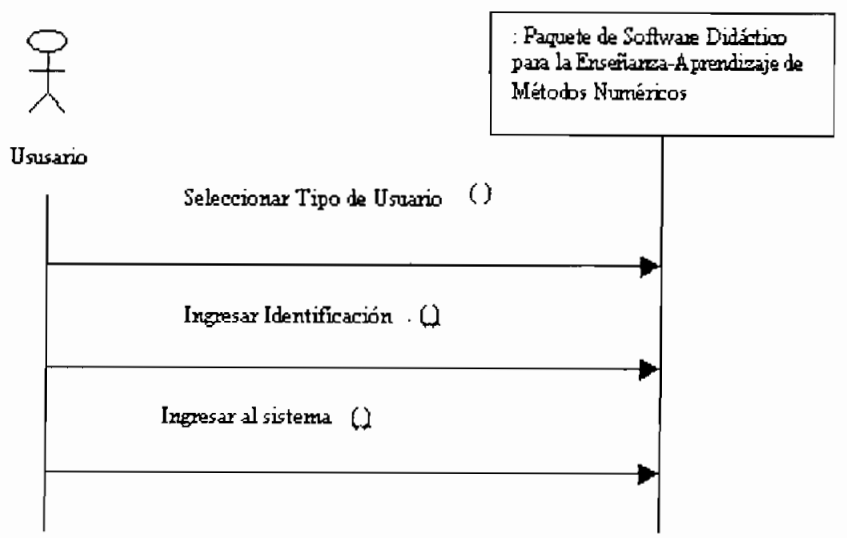


Fig. 2.20 Diagrama de Secuencia del Sistema para el caso de uso Identificar Estudiante.

2.5.2 DIAGRAMAS DE SECUENCIA (2)

Desplegar tema del contenido teórico

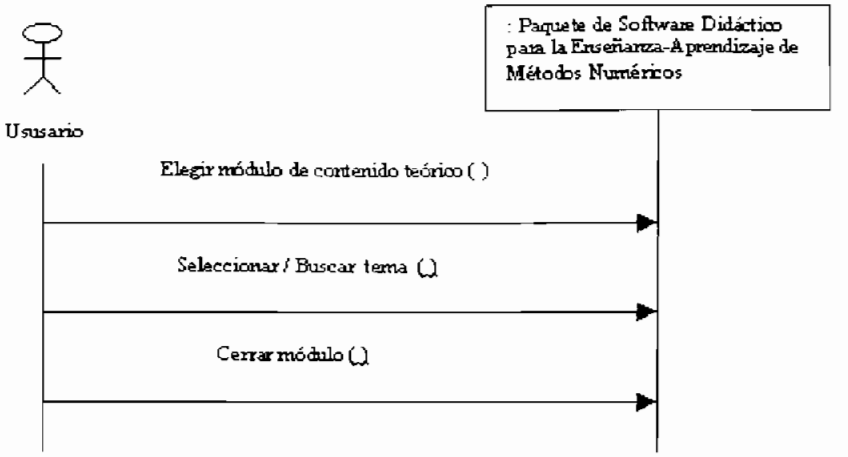


Fig. 2.21 Diagrama de Secuencia del Sistema para el caso de uso Desplegar Tema de Contenido Teórico

Resolver Sistemas de Ecuaciones Lineales

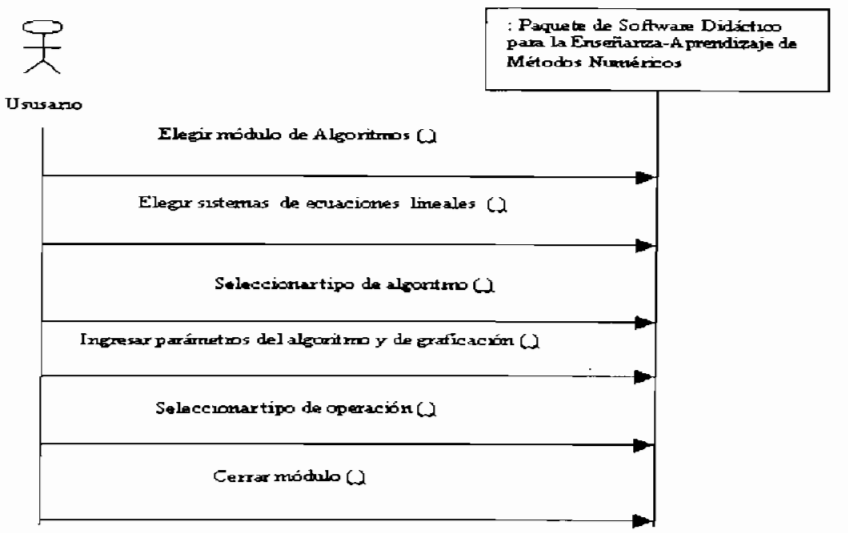


Fig. 2.22 Diagrama de Secuencia del Sistema para el caso de uso Resolver Sistemas de Ecuaciones Lineales

Graficar una función

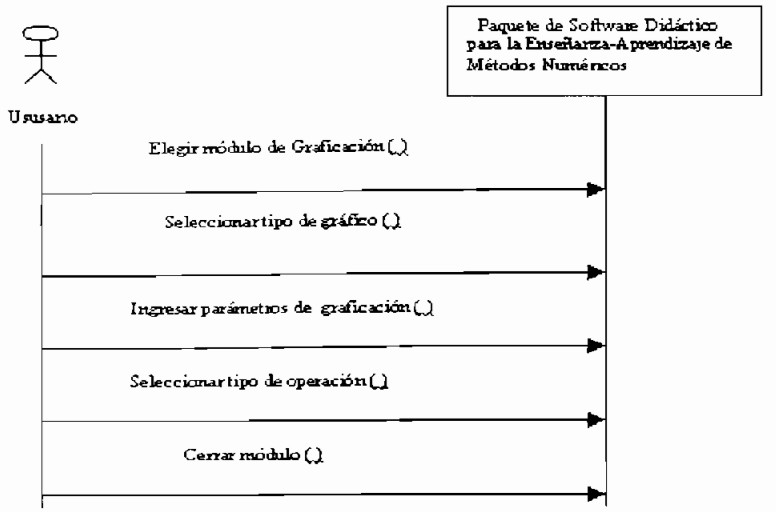


Fig. 2.23 Diagrama de Secuencia del Sistema para el caso de uso Graficar una Función

2.5.3 DIAGRAMAS DE SECUENCIA (3)

Realizar evaluación

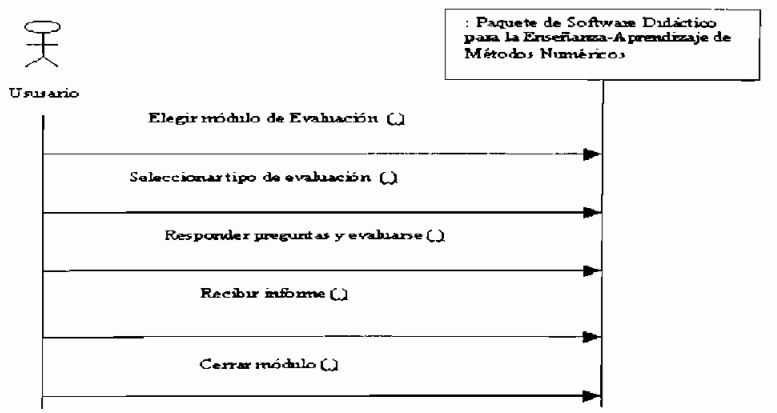


Fig. 2.24 Diagrama de Secuencia del Sistema para el caso de uso Realizar Evaluación

Resolver sistemas de ecuaciones no lineales

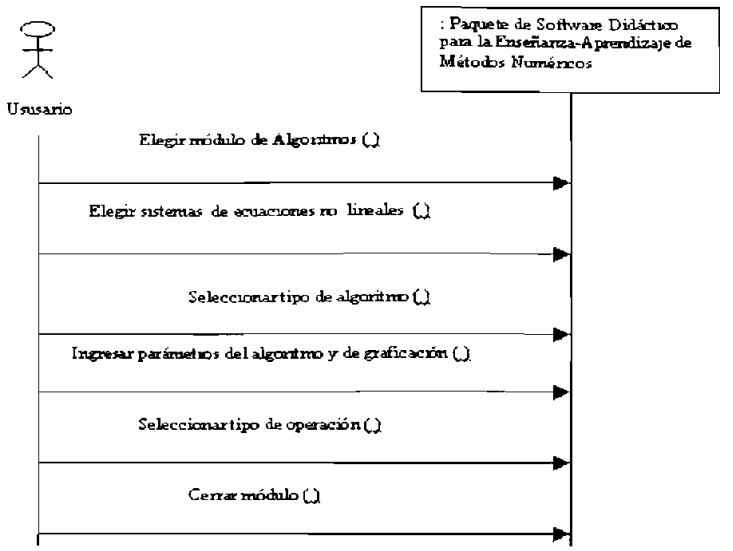


Fig. 2.25 Diagrama de Secuencia del Sistema para el caso de uso Resolver Sistemas de Ecuaciones No Lineales

2.5.4 DIAGRAMAS DE SECUENCIA (4)

Encontrar raíces de un polinomio

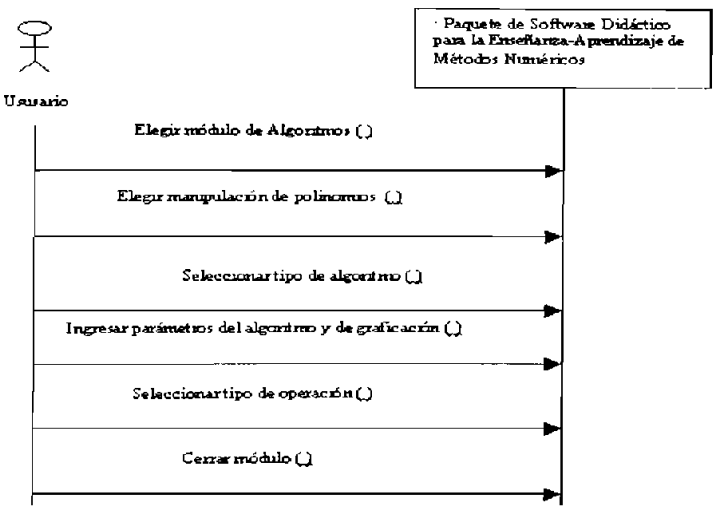


Fig. 2.26 Diagrama de Secuencia del Sistema para el caso de uso Encontrar Raíces de un Polinomio.

Calcular la derivada de una función

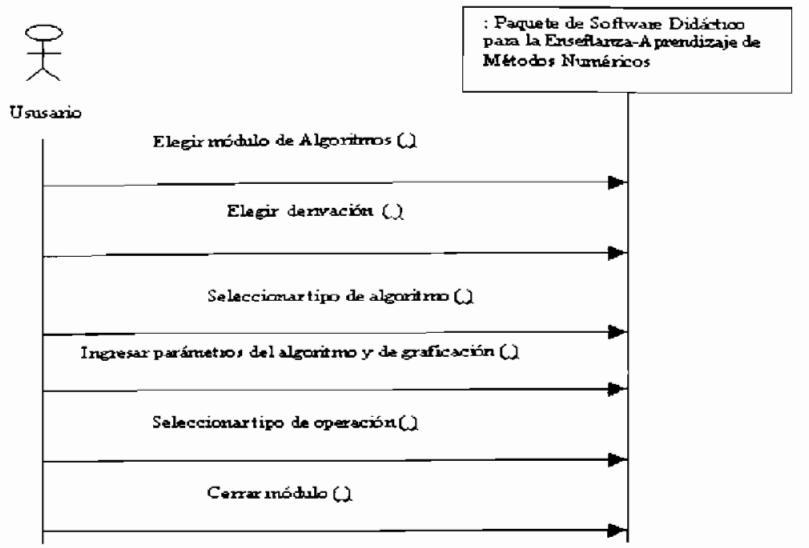


Fig. 2.27 Diagrama de Secuencia del Sistema para el caso de uso Calcular la derivada de una función

2.5.5 DIAGRAMAS DE SECUENCIA (5)

Resolver un sistema de ecuaciones diferenciales ordinarias

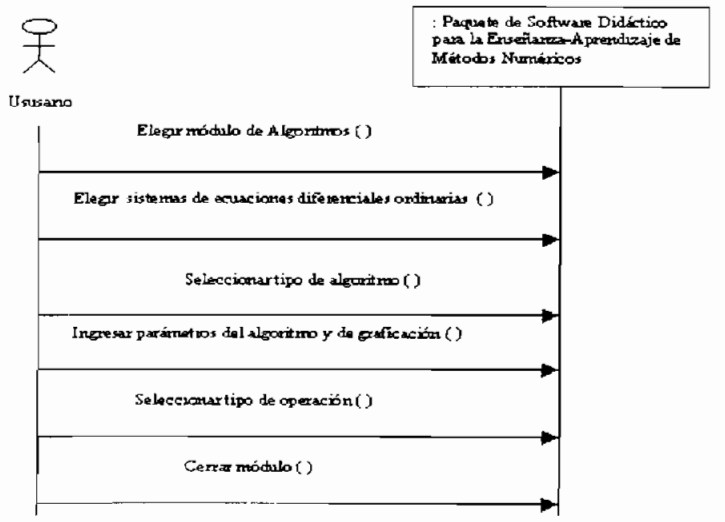


Fig. 2.28 Diagrama de Secuencia del Sistema para el caso de uso Resolver sistemas de ecuaciones diferenciales ordinarias

Realizar una interpolación polinomial

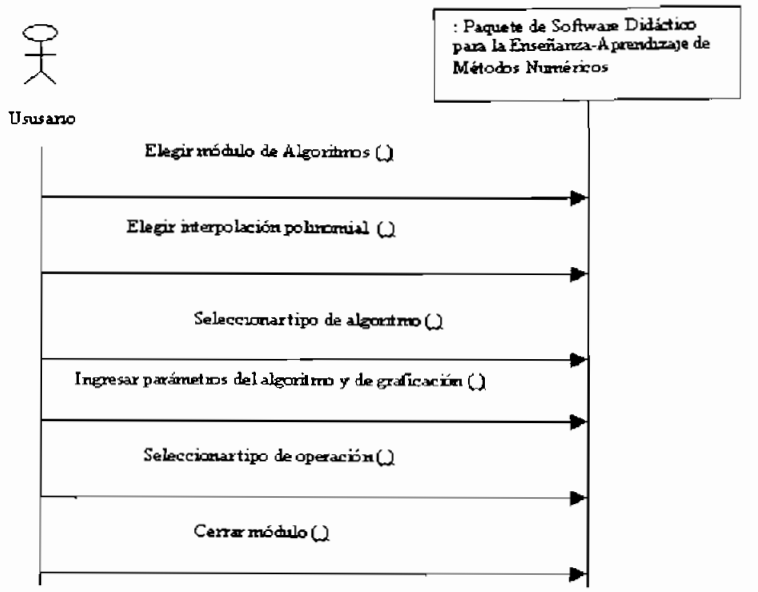


Fig. 2.29 Diagrama de Secuencia del Sistema para el caso de uso Realizar Una interpolación polinomial

Resolver ecuaciones diferenciales ordinarias

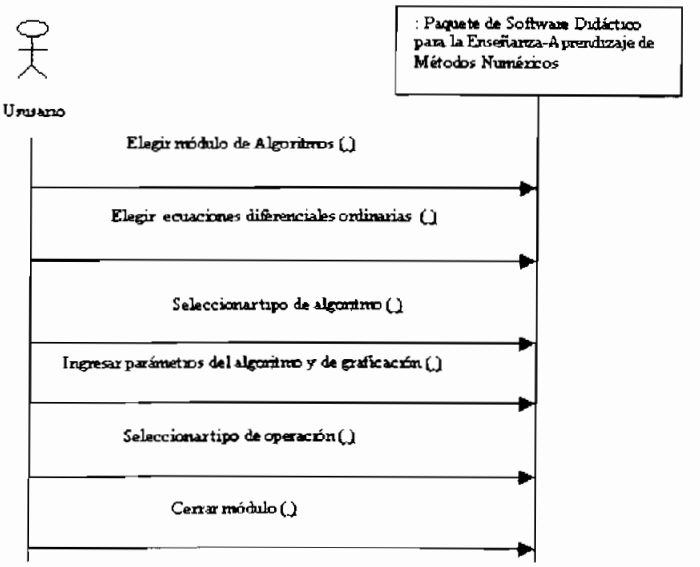


Fig. 2.30 Diagrama de Secuencia del Sistema para el caso de uso Resolver ecuaciones diferenciales ordinarias

2.5.6 DIAGRAMAS DE SECUENCIA (6)

Resolver ecuaciones diferenciales parciales

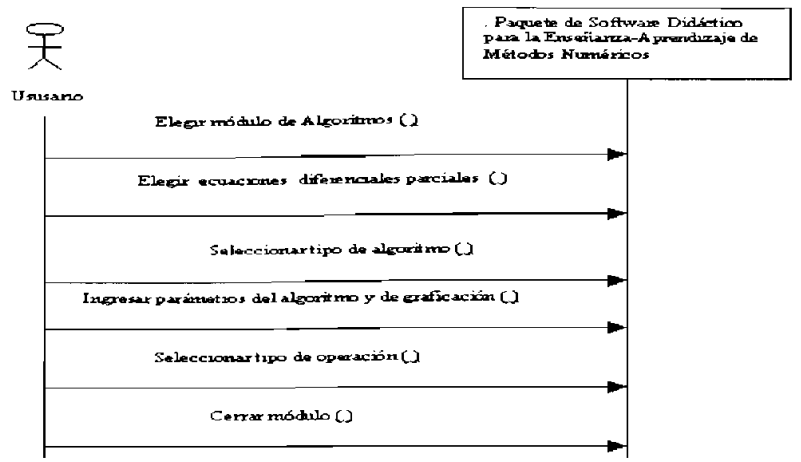


Fig. 2.31 Diagrama de Secuencia del Sistema para el caso de uso Resolver ecuaciones diferenciales parciales

Encontrar raíces de funciones no lineales

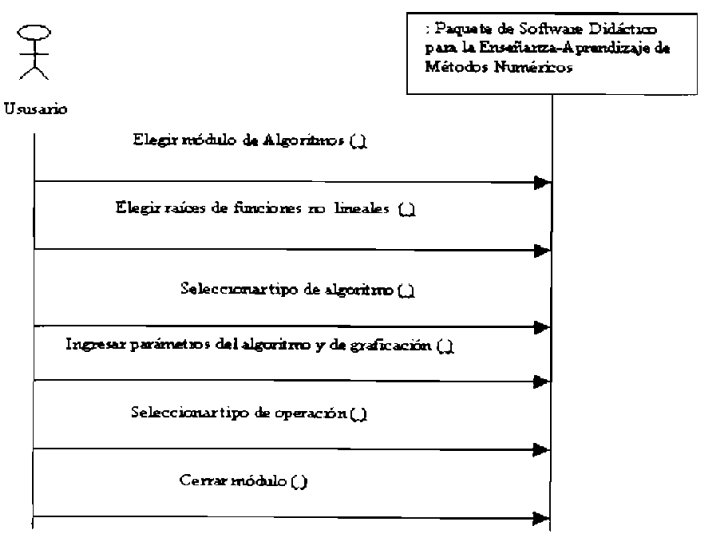


Fig. 2.32 Diagrama de Secuencia del Sistema para el caso de uso Encontrar raíces de funciones no lineales

Calcular la integral de una función

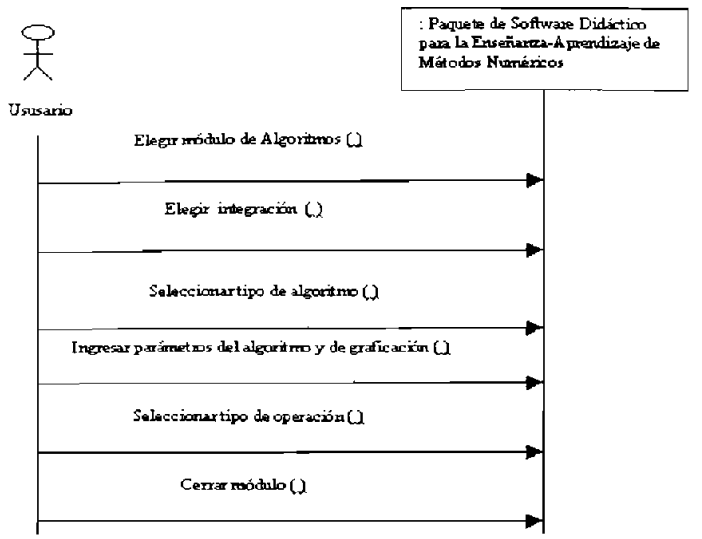


Fig. 2.33 Diagrama de Secuencia del Sistema para el caso de uso Calcular la integral de una función

Realizar una regresión polinomial

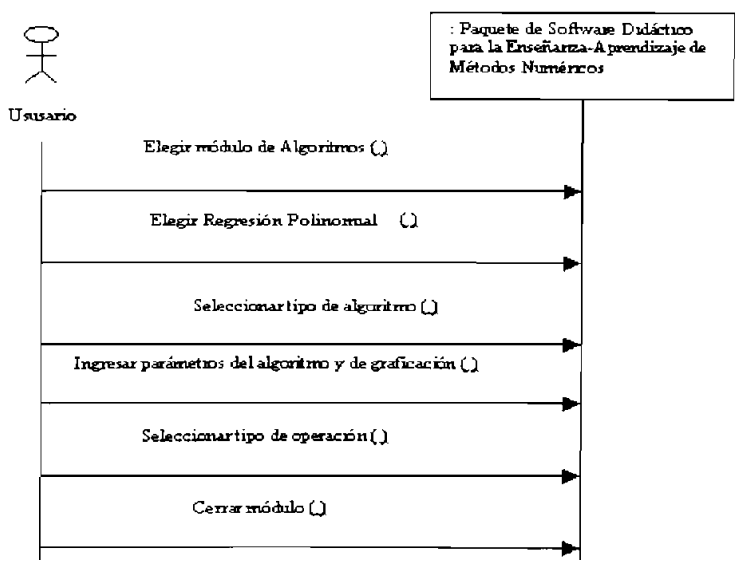


Fig. 2.34 Diagrama de Secuencia del Sistema para el caso de uso Realizar una regresión polinomial

2.6 CONTRATOS DE OPERACIÓN

Una vez que se tienen las Operaciones del Sistema identificadas en los Diagramas de Secuencia, se describe mediante contratos el comportamiento esperado del sistema en cada operación.

Un Contrato es un documento que describe qué es lo que se espera de una operación. Tiene una redacción en estilo declarativo, enfatizando en el *qué* más que en el *cómo*. Lo más común es expresar los contratos en forma de pre- y post-condiciones en torno a cambios de estado.

Se puede escribir un contrato para un método individual de una clase software, o para una operación del sistema completa. En este punto se verá únicamente este último caso.

Un Contrato de Operación del Sistema describe cambios en el estado del sistema cuando una operación del sistema es invocada. A continuación se ve un ejemplo de Contrato:

Contrato

Nombre: insertarTarjeta (número_tarjeta: número)

Responsabilidades: Comenzar una sesión con el sistema para realizar una operación. Presentar las opciones disponibles.

Referencias

Cruzadas: Funciones del Sistema: R1.2, R1.6, R1.7

Casos de Uso: Reintegro

Notas:

Excepciones: Si la tarjeta es ilegible, indicar que ha habido un error.

Salida:

Pre-condiciones: No hay una sesión activa.

Post-condiciones: Una nueva *Sesión* se ha creado. (*creación de instancia*)

Construcción de un Contrato

Los pasos a seguir para construir un contrato son los siguientes:

1. Identificar las operaciones del sistema a partir de los Diagramas de Secuencia del Sistema.
2. Para cada operación del sistema construir un contrato.
3. Empezar escribiendo el apartado de *Responsabilidades*, describiendo informalmente el propósito de la operación. Este es el apartado más importante del contrato.
4. A continuación rellenar el apartado de *Post-condiciones*, describiendo declarativamente los cambios de estado que sufren los objetos en el Modelo Conceptual. Puede ser que este apartado quede vacío si no cambia el valor de ningún dato que los maneja el sistema (por ejemplo en una operación del sistema que tan solo se encarga de sacar por pantalla algo al usuario).
5. Para describir las post-condiciones, usar las siguientes categorías:
   - Creación y borrado de instancias.
   - Asociaciones formadas y retiradas.
   - Asociaciones formadas y retiradas.

2.6.1 CONTRATOS DE OPERACIÓN (1)

Contrato

Nombre:	*Seleccionar tipo de usuario*
Responsabilidades:	Comenzar una sesión con el sistema como usuario nuevo o antiguo.
Referencias Cruzadas:	Funciones del Sistema: R 1.1 a R 1.6
Casos de Uso:	Identificar estudiante
Notas:	
Excepciones:	Si ha elegido salir, terminar sesión.
Salida:	
Pre-condiciones:	No hay una sesión activa.

Post-condiciones:	Una sesión con el módulo Usuario se ha creado. Una sesión con el módulo Clave se ha creado La sesión Usuario se ha asociado a la interfaz de Seguridad La sesión Clave ha asociado a Usuario.

Contrato

Nombre:	Ingresar identificación
Responsabilidades:	Validar y/o registrar la clave de usuario
Referencias Cruzadas:	Funciones del Sistema: R1.1 a R1.6
Casos de Uso:	Identificar estudiante
Notas:	
Excepciones:	Si la información ingresada es incorrecta, indicar que ha existido un error.
Salida:	Enviar clave a la base de datos para registrarla.
Pre-condiciones:	Espera información de identificación No hay sesión activa con la base de datos
Post-condiciones:	Permite o niega el acceso a los módulos del P.S.E.A.M.N La sesión Base de datos se ha asociado con Clave. La sesión Clave se ha asociado con la interfaz de Seguridad

Contrato

Nombre:	Ingresar al sistema
Responsabilidades:	Presentar la interfaz de ingreso a los módulos del sistema
Referencias Cruzadas:	Funciones del Sistema: R1.1 a R1.6
Casos de Uso:	Identificar estudiante
Notas:	
Excepciones:	Si ha elegido salir, terminar sesión.
Salida:	
Pre-condiciones:	Hay sesión activa con los módulos Usuario, Clave y la Base

	de Datos
Post-condiciones:	Ha terminado la sesión con los módulos Usuario, Clave y la Base de Datos Las asociaciones con Usuario, Clave y la base de datos han sido retiradas.

2.6.2 CONTRATOS DE OPERACIÓN (2)

Contrato

Nombre:	Elegir módulo de Contenido Teórico
Responsabilidades:	Presentar la interfaz del Contenido Teórico
Referencias Cruzadas:	Funciones del Sistema: R 2.1 a R 2.8
Casos de Uso:	Desplegar tema del contenido teórico
Notas:	
Excepciones:	Si ha elegido salir, terminar sesión.
Salida:	
Pre-condiciones:	
Post-condiciones:.	

Contrato

Nombre:	Seleccionar operación
Responsabilidades:	Presentar el contenido de los diferentes Temas de los que se compone el tutorial, ya sea mediante selección del mismo o por su búsqueda. Imprimir el contenido de texto
Referencias Cruzadas:	Funciones del Sistema: R 2.1 a R 2.8
Casos de Uso:	Leer tema del contenido teórico
Notas:	

Excepciones:	Si ha elegido salir, terminar sesión con el módulo de Contenido Teórico y presentar la interfaz de los módulos que tiene el software.
Salida:	Código del tema seleccionado
Pre-condiciones:	No hay sesión activa con el módulo de Contenido Teórico. No hay sesión activa con la base de texto. No hay sesión activa con la impresora.
Post-condiciones:	Una sesión con el módulo de Contenido Teórico se ha creado Una sesión con la base de texto se ha creado Una sesión con la impresora se ha creado La base de texto se ha asociado módulo de Contenido Teórico. La interfaz del Contenido Teórico se ha asociado con la impresora

Contrato

Nombre:	Cerrar módulo
Responsabilidades:	Terminar sesión con el módulo de Contenido Teórico
Referencias Cruzadas:	Funciones del Sistema: R 2.1 a R 2.8
Casos de Uso:	Leer tema del contenido teórico
Notas:	
Excepciones:	
Salida:	
Pre-condiciones:	Hay sesión activa con el módulo de Contenido Teórico. Hay sesión activa con la base de texto
Post-condiciones:	Ha terminado la sesión con el módulo de Contenido Teórico y la base de texto. Las asociaciones con la base de datos de texto y el Contenido Teórico han sido retiradas.

Contrato

Nombre:	Elegir módulo de Algoritmos
Responsabilidades:	Presentar los diferentes Temas que contiene el módulo de algoritmos.
Referencias Cruzadas:	Funciones del Sistema: R 5.1 a R 5.14
Casos de Uso:	Resolver sistemas de ecuaciones lineales.
Notas:	
Excepciones:	Si ha elegido salir, terminar sesión.
Salida:	
Pre-condiciones:	Espera la elección del módulo.
Post-condiciones:	

Contrato

Nombre:	Elegir Sistemas de Ecuaciones Lineales
Responsabilidades:	Presentar Interfaz de Sistemas de Ecuaciones Lineales
Referencias Cruzadas:	Funciones del Sistema: R1.2, R1.6, R1.7
Casos de Uso:	Resolver sistemas de ecuaciones lineales.
Notas:	
Excepciones:	Si ha elegido salir, terminar sesión y presentar los temas que contiene el módulo de algoritmos.
Salida:	
Pre-condiciones:	
Post-condiciones:	

Contrato

Nombre:	Seleccionar algoritmo
Responsabilidades:	Presentar la interfaz del algoritmo elegido.
Referencias Cruzadas:	Funciones del Sistema: R1.2, R1.6, R1.7
Casos de Uso:	Resolver sistemas de ecuaciones lineales.
Notas:	
Excepciones:	Si ha elegido salir, terminar sesión con el módulo de Sistemas de Ecuaciones Lineales.
Salida:	
Pre-condiciones:	No hay sesión activa con el módulo de Sistemas de Ecuaciones Lineales. No hay sesión con el módulo de Graficación. No hay modificación de atributos del módulo
Post-condiciones:	Una sesión con el módulo de Sistemas de Ecuaciones Lineales se ha creado. Sesión con módulo del Graficación ha sido creada. Las sesiones se han asociado a la interfaz de Sistemas de Ecuaciones Lineales. El atributo Tipo de Algoritmo a cambiado.

Contrato

Nombre:	Ingresar parámetros de algoritmo y de graficación
Responsabilidades:	Capturar y validar los parámetros ingresados, setear los Atributos necesarios de acuerdo al tipo de algoritmo.
Referencias Cruzadas:	Funciones del Sistema: R1.2, R1.6, R1.7
Casos de Uso:	Resolver sistemas de ecuaciones lineales.
Notas:	

Excepciones: Si ha elegido salir, terminar sesión con el módulo de Sistemas de Ecuaciones Lineales.

Salida:

Pre-condiciones: No hay modificación del atributo Parámetros del Algoritmo
No hay modificación del atributo Parámetros de graficación

Post-condiciones: Cambia el atributo Parámetros del Algoritmo.
Cambia el atributo Parámetros de Graficación.

Contrato

Nombre: Seleccionar tipo de operación

Responsabilidades: Calcular, cambiar parámetros del algoritmo y de graficación, ampliar gráfico, abrir y guardar resultados e imprimir pantalla.

Referencias Cruzadas: Funciones del Sistema: R1.2, R1.6, R1.7

Casos de Uso: Resolver sistemas de ecuaciones lineales.

Notas:

Excepciones: Si ha elegido salir, terminar sesión con el módulo de sistemas de ecuaciones lineales.

Salida:

Pre-condiciones: No hay modificación del atributo Parámetros del Algoritmo,
No hay modificación del atributo Parámetros de Graficación.
No hay sesión con la impresora

Post-condiciones: Cambia el atributo Parámetros del Algoritmo.
Cambia el atributo Parámetros de Graficación.
Sesión con la impresora ha sido creada.
La sesión se han asociado a la interfaz del tipo de algoritmo seleccionado.

Contrato

Nombre:	Cerrar módulo
Responsabilidades:	Terminar sesión con el módulo de Sistemas de Ecuaciones Lineales.
Referencias Cruzadas:	Funciones del Sistema: R1.2, R1.6, R1.7
Casos de Uso:	Resolver sistemas de ecuaciones lineales
Notas:	
Excepciones:	
Salida:	
Pre-condiciones:	Hay sesión activa con el módulo Sistemas de Ecuaciones Lineales. Hay sesión activa con el módulo Graficación. No hay sesión activa con la impresora.
Post-condiciones:	A terminado la sesión con el módulo de Sistemas de Ecuaciones Lineales A terminado la sesión con el módulo Graficación Asociación con el módulo de Graficación retirada.

Contrato

Nombre:	Elegir módulo de Graficación
Responsabilidades:	Presentar interfaz del módulo de Graficación
Referencias Cruzadas:	Funciones del Sistema: R 3.1 a R 3.20
Casos de Uso:	Graficar función
Notas:	
Excepciones:	Si ha elegido salir, terminar sesión.
Salida:	
Pre-condiciones:	
Post-condiciones:	

Contrato

Nombre:	Seleccionar tipo de gráfico
Responsabilidades:	Presentar la interfaz del gráfico de funciones o el de datos.
Referencias Cruzadas:	Funciones del Sistema: R 3.1 a R 3.20
Casos de Uso:	Graficar función.
Notas:	
Excepciones:	Si ha elegido salir, presentar los módulos que tiene el software
Salida:	
Pre-condiciones:	No hay sesión activa con el módulo de Graficación. No hay sesión activa con la aplicación Excel No hay modificación de atributos del módulo
Post-condiciones:	Una sesión con el módulo de Graficación es creada. Una Sesión con la aplicación Excel es creada La sesión con el módulo de Graficación se ha asociado a la interfaz de Graficación. La sesión con la aplicación Excel se ha asociado al módulo de Graficación. El atributo Tipo de Gráfico a cambiado.

Contrato

Nombre:	Ingresar parámetros de graficación
Responsabilidades:	Capturar y validar los parámetros ingresados, setear los Atributos necesarios de acuerdo al tipo de gráfico.
Referencias Cruzadas:	Funciones del Sistema: R 3.1 a R 3.20
Casos de Uso:	Graficar función.
Notas:	
Excepciones:	Si ha elegido salir, terminar sesión con el módulo de Graficación, la aplicación Excel y presentar los módulos que tiene el software.
Salida:	
Pre-condiciones:	No hay modificación del atributo Parámetros de Graficación
Post-condiciones:	Cambia el atributo Parámetros del Graficación.

Contrato

Nombre:	Seleccionar tipo de operación
Responsabilidades:	Cambiar parámetros de graficación, ampliar gráfico, abrir, y guardar resultados e imprimir pantalla.
Referencias Cruzadas:	Funciones del Sistema: R 3.1 a R 3.20
Casos de Uso:	Graficar función
Notas:	
Excepciones:	Si ha elegido salir, terminar sesión con el módulo de Graficación, la aplicación Excel y presentar los módulos que tiene el software
Salida:	
Pre-condiciones:	No hay modificación del atributo Parámetros de Graficación. No hay sesión con la impresora

Post-condiciones:	Cambia el atributo Parámetros del Graficación Sesión con la impresora ha sido creada. La sesión se ha asociado a la interfaz de graficación.

Contrato

Nombre:	Cerrar módulo
Responsabilidades:	Terminar sesión con el módulo de Graficación
Referencias Cruzadas:	Funciones del Sistema: R 3.1 a R 3.20
Casos de Uso:	Graficar función
Notas:	
Excepciones:	
Salida:	
Pre-condiciones:	Hay sesión activa con el módulo de Graficación Hay sesión activa con la aplicación Excel
Post-condiciones:	A terminado la sesión con el módulo de Graficación. A terminado la sesión con la aplicación Excel Las asociaciones con el módulo de Graficación y la aplicación Excel han sido retiradas.

2.6.3 CONTRATOS DE OPERACIÓN (3)

Contrato

Nombre:	Elegir módulo de Evaluación
Responsabilidades:	Presentar pantalla del tipo de evaluación
Referencias Cruzadas:	Funciones del Sistema: R 4.1 a R 4.12
Casos de Uso:	Realizar evaluación
Notas:	
Excepciones:	Si ha elegido salir, terminar sesión.

Salida:

Pre-condiciones: Espera la elección del módulo

Post-condiciones:

Contrato

Nombre:	Seleccionar Tipo de Evaluación
Responsabilidades:	Presentar la interfaz de la evaluación elegida, setear tiempo.
Referencias Cruzadas:	Funciones del Sistema: R 4.1 a R 4.12
Casos de Uso:	Realizar Evaluación
Notas:	
Excepciones:	Si ha elegido salir, presentar interfaz de los módulos que tiene el software
Salida:	
Pre-condiciones:	No hay sesión activa con el módulo Evaluación No hay modificación de atributos del módulo
Post-condiciones:	Una sesión con el módulo Evaluación ha sido creada. La sesión se ha asociado a la interfaz de Evaluación El atributo Tipo de Evaluación a cambiado.

Contrato

Nombre:	Responder a las preguntas y evaluarse
Responsabilidades:	Mostrar temas, preguntas y evaluar las respuestas, verificar que todas las preguntas estén contestadas, el tiempo transcurrido en la evaluación
Referencias Cruzadas:	Funciones del Sistema: R 4.1 a R 4.12

Casos de Uso:	Realizar evaluación
Notas:	
Excepciones:	Si ha elegido salir, terminar sesión con el módulo de Evaluación y presentar la interfaz de evaluación. Alguna pregunta no ha sido contestada, indicar el error y permitir su completación. Tiempo expirado, indicarlo.
Salida:	
Pre-condiciones:	No hay sesión con el módulo Pregunta No hay sesión creada con la base de datos
Post-condiciones:	Una sesión con el módulo Pregunta Una sesión con la base de datos se ha creado La sesión Pregunta se ha asociado con el módulo de Evaluación , interfaz del tipo de evaluación seleccionado y la base de datos.

Contrato

Nombre:	Recibir informe
Responsabilidades:	Mostrar e imprimir el informe de evaluación.
Referencias Cruzadas:	Funciones del Sistema: R 4.1 a R 4.12
Casos de Uso:	Realizar evaluación
Notas:	
Excepciones:	Si ha elegido salir, terminar sesión con los módulos de Evaluación. Pregunta y la Base de datos, presentar la interfaz de evaluación
Salida:	
Pre-condiciones:	No hay sesión activa con el módulo Informe de Evaluación. No hay sesión activa con la impresora.

Post-condiciones: Una sesión con el módulo Informe de Evaluación se ha creado.
Una sesión con la impresora ha sido creada
La sesion con Informe de Evaluación se han asociado con el módulo Evaluación, interfaz del tipo de evaluación seleccionado y con la base de datos.
La sesión con impresora se ha asociado con la interfaz del tipo de evaluación seleccionada

Contrato

Nombre: Cerrar módulo
Responsabilidades: Terminar sesión con el módulo de Evaluación
Referencias Cruzadas: Funciones del Sistema: R 4.1 a R 4.12
Casos de Uso: Realizar evaluación.
Notas:
Excepciones:
Salida:

Pre-condiciones: Hay sesión activa con el módulo de Evaluación
Hay sesión activa con el módulo Pregunta.
Hay sesión activa con el módulo de Informe de Evaluación
Hay sesión activa con la base de datos
Post-condiciones: No hay sesión activa con el módulo de Evaluación
No hay sesión activa con el módulo Pregunta.
No hay sesión activa con el módulo de Informe de Evaluación
No hay sesión activa con la base de datos
Las asociaciones con el módulo Evaluación, Pregunta, Informe de evaluación y la base de datos han sido retiradas.

Contrato

Nombre: Elegir módulo de Algoritmos

Responsabilidades: Presentar los diferentes Temas que contiene el módulo de algoritmos.

Referencias Cruzadas: Funciones del Sistema: R 5.1 a R 5.14

Casos de Uso: Resolver sistemas de ecuaciones no lineales.

Notas:

Excepciones: Si ha elegido salir, terminar sesión.

Salida:

Pre-condiciones: Espera la elección del módulo.

Post-condiciones:

Contrato

Nombre: Elegir Sistemas de Ecuaciones No Lineales

Responsabilidades: Presentar interfaz de Sistemas de ecuaciones no lineales

Referencias Cruzadas: Funciones del Sistema: R 5.1 a R 5.14

Casos de Uso: Resolver sistemas de ecuaciones no lineales.

Notas:

Excepciones: Si ha elegido salir, presentar los diferentes módulos que tiene el software

Salida:

Pre-condiciones:

Post-condiciones:

Contrato

Nombre:	Seleccionar algoritmo
Responsabilidades:	Presentar la interfaz del algoritmo elegido.
Referencias Cruzadas:	Funciones del Sistema: R 5.1 a R 5.14
Casos de Uso:	Resolver sistemas de ecuaciones no lineales.
Notas:	
Excepciones:	Si ha elegido salir, presentar los diferentes temas del módulo de algoritmos
Salida:	
Pre-condiciones:	No hay sesión activa con el módulo Sistemas de Ecuaciones No Lineales. No hay sesión activa con el módulo Sistemas de Ecuaciones Lineales. No hay sesión activa con el módulo Graficación No hay sesión activa con la aplicación Excel. No hay modificación de atributos del módulo
Post-condiciones:	Una sesión con el módulo Sistemas de Ecuaciones No Lineales se ha creado. Una nueva sesión con el módulo Sistemas de Ecuaciones Lineales se ha creado. Una nueva sesión con el módulo Graficación ha sido creada. Una nueva sesión con la aplicación Excel se ha creado. Las sesiones Sistemas de Ecuaciones Lineales, la aplicación Excel se han asociado con el módulo Sistemas de Ecuaciones No Lineales. La sesión Graficación se ha asociado a la interfaz de Sistemas de ecuaciones no lineales El atributo Tipo de Algoritmo a cambiado.

Contrato

Nombre:	Ingresar parámetros de algoritmo y de graficación
Responsabilidades:	Capturar y validar los parámetros ingresados, setear los Atributos necesarios de acuerdo al tipo de algoritmo.
Referencias Cruzadas:	Funciones del Sistema: R 5.1 a R 5.14
Casos de Uso:	Resolver sistemas de ecuaciones no lineales.
Notas:	
Excepciones:	Si ha elegido salir, terminar sesión con los módulos Sistemas de Ecuaciones No Lineales, Sistemas de Ecuaciones Lineales, Graficación y la aplicación Excel.
Salida:	
Pre-condiciones:	No hay modificación del atributo Parámetros del Algoritmo No hay modificación del atributo Parámetros de Graficación
Post-condiciones:	Cambia el atributo Parámetros del Algoritmo. Cambia el atributo Parámetros de Graficación.

Contrato

Nombre:	Seleccionar tipo de operación
Responsabilidades:	Calcular, cambiar parámetros del algoritmo y de graficación, ampliar gráfico, abrir y guardar resultados e imprimir pantalla.
Referencias Cruzadas:	Funciones del Sistema: R 5.1 a R 5.14
Casos de Uso:	Resolver sistemas de ecuaciones No lineales.
Notas:	
Excepciones:	Si ha elegido salir, terminar sesión con los módulos Sistemas de Ecuaciones No Lineales, Sistemas de Ecuaciones Lineales, Graficación y la aplicación Excel.

Salida:	Funciones ha ser evaluadas
Pre-condiciones:	No hay modificación del atributo Parámetros del Algoritmo, No hay modificación del atributo Parámetros de Graficación No hay sesión con la impresora
Post-condiciones:	Cambia el atributo Parámetros del Algoritmo. Cambia el atributo Parámetros de Graficación. Sesión con la impresora ha sido creada La sesion se han asociada con la interfaz del algoritmo seleccionado.

Contrato

Nombre:	Cerrar módulo
Responsabilidades:	Terminar sesión con el módulo de Sistemas de Ecuaciones No Lineales.
Referencias Cruzadas:	Funciones del Sistema: R 5.1 a R 5.14
Casos de Uso:	Resolver sistemas de ecuaciones No lineales
Notas:	
Excepciones:	
Salida:	
Pre-condiciones:	Hay sesión activa con los módulos Sistemas de Ecuaciones Lineales, Sistemas de Ecuaciones No Lineales Graficación y aplicación Excel.
Post-condiciones:	A terminado la sesión con los módulos de Sistemas de Ecuaciones Lineales, Sistemas de Ecuaciones No Lineales, Graficación y aplicación Excel. Las asociaciones con los módulos de Sistemas de Ecuaciones Lineales, Sistemas de Ecuaciones, No Lineales, Graficación y la aplicación Excel han sido retiradas.

2.6.4 CONTRATOS DE OPERACIÓN (4)

Contrato

Nombre:	Elegir módulo de Algoritmos
Responsabilidades:	Presentar los diferentes Temas que contiene el módulo de algoritmos.
Referencias Cruzadas:	Funciones del Sistema: R 5.1 a R 5.14
Casos de Uso:	Encontrar raíces de un polinomio.
Notas:	
Excepciones:	Si ha elegido salir, terminar sesión.
Salida:	
Pre-condiciones:	Espera la elección del módulo.
Post-condiciones:	

Contrato

Nombre:	Elegir Manipulación de Polinomios
Responsabilidades:	Presentar interfaz de Manipulación de polinomios
Referencias Cruzadas:	Funciones del Sistema: R 5.1 a R 5.14
Casos de Uso:	Encontrar raíces de un polinomio.
Notas:	
Excepciones:	Si ha elegido salir, presentar los diferentes módulos que contiene el software.
Salida:	
Pre-condiciones:	
Post-condiciones:	

Contrato

Nombre:	Seleccionar algoritmo
Responsabilidades:	Presentar la interfaz del algoritmo elegido.
Referencias Cruzadas:	Funciones del Sistema: R 5.1 a R 5.14
Casos de Uso:	Encontrar la raíz de un polinomio.
Notas:	
Excepciones:	Si ha elegido salir, Presentar los diferentes temas que tiene el módulo de algoritmos
Salida:	
Pre-condiciones:	No hay sesión activa con el módulo Manipulación de Polinomios. No hay sesión activa con el módulo Graficación No hay modificación del atributo Tipo de Algoritmo
Post-condiciones:	Una sesión con el módulo Manipulación de Polinomios a sido creado. Una sesión con el módulo Graficación ha sido creada Las sesiones se han asociado a la interfaz de Manipulación de polinomios El atributo Tipo de Algoritmo a cambiado.

Contrato

Nombre:	Ingresar parámetros de algoritmo y de graficación
Responsabilidades:	Calcular, capturar y validar los parámetros ingresados, setear los atributos necesarios de acuerdo al tipo de algoritmo.
Referencias Cruzadas:	Funciones del Sistema: R 5.1 a R 5.14
Casos de Uso:	Encontrar raíces de polinomios.
Notas:	

Excepciones:	Si ha elegido salir, terminar sesión con el módulo Manipulación de Polinomios y Graficación.
Salida:	
Pre-condiciones:	No hay modificación del atributo Parámetros del Algoritmo No hay modificación del atributo Parámetros de Graficación
Post-condiciones:	Cambia el atributo Parámetros del Algoritmo. Cambia el atributo Parámetros de Graficación.

Contrato

Nombre:	Seleccionar tipo de operación
Responsabilidades:	Calcular, cambiar parámetros del algoritmo y de graficación, ampliar gráfico, abrir y guardar resultados e imprimir pantalla.
Referencias Cruzadas:	Funciones del Sistema: R 5.1 a R 5.14
Casos de Uso:	Encontrar raíces de polinomios.
Notas:	
Excepciones:	Si ha elegido salir, terminar sesión con el módulo Manipulación de Polinomios y Graficación.
Salida:	
Pre-condiciones:	No hay modificación del atributo Parámetros del Algoritmo, No hay modificación del atributo Parámetros de Graficación No hay sesión con la impresora
Post-condiciones:	Cambia el atributo Parámetros del Algoritmo. Cambia el atributo Parámetros de Graficación Sesión con la impresora ha sido creada La sesión se ha asociado con la interfaz del tipo de algoritmo seleccionado.

Contrato

Nombre:	Cerrar módulo
Responsabilidades:	Terminar sesión con el módulo Manipulación de Polinomios.
Referencias	
Cruzadas:	Funciones del Sistema: R 5.1 a R 5.14
Casos de Uso:	Encontrar raíces de polinomios
Notas:	
Excepciones:	
Salida:	
Pre-condiciones:	Hay sesión activa con los módulos Manipulación de Polinomios y Graficación.
Post-condiciones:	A terminado la sesión con los módulo Manipulación de Polinomios y Graficación. Las asociaciones con los módulos Manipulación de Polinomios y Graficación han sido retiradas.

Contrato

Nombre:	Elegir módulo de Algoritmos
Responsabilidades:	Presentar los diferentes Temas que contiene el módulo de algoritmos.
Referencias	
Cruzadas:	Funciones del Sistema: R 5.1 a R 5.14
Casos de Uso:	Calcular la derivada de una función
Notas:	
Excepciones:	Si ha elegido salir, terminar sesión.
Salida:	
Pre-condiciones:	Espera la elección del módulo.
Post-condiciones:	

Contrato

Nombre:	Elegir Derivación
Responsabilidades:	Presentar interfaz de Derivación
Referencias Cruzadas:	Funciones del Sistema: R 5.1 a R 5.14
Casos de Uso:	Calcular la derivada de una función.
Notas:	
Excepciones:	Si ha elegido salir, presentar los diferentes módulos que contiene el software
Salida:	
Pre-condiciones:	
Post-condiciones:	

Contrato

Nombre:	Seleccionar algoritmo
Responsabilidades:	Presentar la interfaz del algoritmo elegido.
Referencias Cruzadas:	Funciones del Sistema: R 5.1 a R 5.14
Casos de Uso:	Calcular la derivada de una función.
Notas:	
Excepciones:	Si ha elegido salir, presentar los diferentes temas del módulo de algoritmos
Salida:	
Pre-condiciones:	No hay sesión activa con el módulo Derivación No hay sesión activa con la aplicación Excel. No hay modificación del atributo Tipo de Algoritmo

Post-condiciones:	Una sesión con el módulo Derivación ha sido creada Una nueva sesión con la aplicación Excel se ha creado. La sesión de la aplicación Excel se ha asociado con el módulo Derivación. La sesión del módulo Derivación se ha asociado con la Interfaz del tipo de algoritmo seleccionado. El atributo Tipo de Algoritmo a cambiado.

Contrato

Nombre:	Ingresar parámetros de algoritmo y de graficación
Responsabilidades:	Calcular, capturar y validar los parámetros ingresados, setear los atributos necesarios de acuerdo al tipo de algoritmo.
Referencias Cruzadas:	Funciones del Sistema: R 5.1 a R 5.14
Casos de Uso:	Calcular la derivada de una función.
Notas:	
Excepciones:	Si ha elegido salir, terminar sesión con el módulo Derivación Graficación y la aplicación Excel.
Salida:	
Pre-condiciones:	No hay modificación del atributo Parámetros del Algoritmo No hay modificación del atributo Parámetros de Graficación
Post-condiciones:	Cambia el atributo Parámetros del Algoritmo. Cambia el atributo Parámetros de Graficación.

Contrato

Nombre:	Seleccionar tipo de operación
Responsabilidades:	Calcular, cambiar parámetros del algoritmo y de graficación, ampliar gráfico, abrir y guardar resultados e imprimir pantalla.
Referencias Cruzadas:	Funciones del Sistema: R 5.1 a R 5.14
Casos de Uso:	Calcular la derivada de una función.
Notas:	
Excepciones:	Si ha elegido salir, terminar sesión con el módulo Derivación Graficación y la aplicación Excel
Salida:	Funciones ha ser evaluadas
Pre-condiciones:	No hay modificación del atributo Parámetros del Algoritmo, No hay modificación del atributo Parámetros de Graficación No hay sesión con la impresora
Post-condiciones:	Cambia el atributo Parámetros del Algoritmo. Cambia el atributo Parámetros de Graficación. Sesión con la impresora ha sido creada La sesión se han asociada con la interfaz del algoritmo seleccionado.

Contrato

Nombre:	Cerrar módulo
Responsabilidades:	Terminar sesión con el módulo de Derivación.
Referencias Cruzadas:	Funciones del Sistema: R 5.1 a R 5.14
Casos de Uso:	Calcular la derivada de una función.
Notas:	
Excepciones:	
Salida:	

Pre-condiciones:	Hay sesión activa con los módulos Derivación, Graficación y la aplicación Excel.
Post-condiciones:	A terminado la sesión con los módulos Derivación Graficación y la aplicación Excel. Las asociaciones con los módulos Derivación, Graficación y la aplicación Excel han sido retiradas.

Contrato

Nombre:	Elegir módulo de Algoritmos
Responsabilidades:	Presentar los diferentes Temas que contiene el módulo de algoritmos.
Referencias Cruzadas:	Funciones del Sistema: R 5.1 a R 5.14
Casos de Uso:	Resolver un sistemas de ecuaciones diferenciales ordinarias.
Notas:	
Excepciones:	Si ha elegido salir, terminar sesión.
Salida:	
Pre-condiciones:	Espera la elección del módulo.
Post-condiciones:	

Contrato

Nombre:	Elegir Sistemas de ecuaciones diferenciales ordinarias
Responsabilidades:	Presentar interfaz de Sistemas de ecuaciones diferenciales ordinarias
Referencias Cruzadas:	Funciones del Sistema: R 5.1 a R 5.14
Casos de Uso:	Resolver un sistemas de ecuaciones diferenciales ordinarias.
Notas:	
Excepciones:	Si ha elegido salir, presentar los diferentes módulos que contiene el software

Salida:

Pre-condiciones:

Post-condiciones:

Contrato

Nombre:	Seleccionar algoritmo
Responsabilidades:	Presentar la interfaz del algoritmo elegido.
Referencias Cruzadas:	Funciones del Sistema: R 5.1 a R 5.14
Casos de Uso:	Resolver un sistemas de ecuaciones diferenciales ordinarias.
Notas:	
Excepciones:	Si ha elegido salir, Presentar los diferentes temas del módulo de algoritmos
Salida:	
Pre-condiciones:	No hay sesión activa con el módulo Sistemas de Ecuaciones Diferenciales Ordinarias. No hay sesión activa con el módulo Sistemas de Ecuaciones Lineales. No hay sesión activa con el módulo Graficación No hay sesión activa con la aplicación Excel. No hay modificación de atributos del módulo
Post-condiciones:	Una sesión con el módulo Sistemas de Ecuaciones Diferenciales Ordinarias se ha creado. Una nueva sesión con el módulo Sistemas de Ecuaciones Lineales se ha creado. Una nueva sesión con el módulo Graficación ha sido creada. Una nueva sesión con la aplicación Excel se ha creado. Las sesiones Sistemas de Ecuaciones Lineales y la aplicación Excel se han asociado con el módulo Sistemas de Ecuaciones Diferenciales Ordinarias.

La sesión Graficación se ha asociado a la interfaz de
Sistemas de ecuaciones diferenciales ordinarias
El atributo Tipo de Algoritmo a cambiado.

Contrato

Nombre:	Ingresar parámetros de algoritmo y de graficación
Responsabilidades:	Capturar y validar los parámetros ingresados, setear los Atributos necesarios de acuerdo al tipo de algoritmo.
Referencias Cruzadas:	Funciones del Sistema: R 5.1 a R 5.14
Casos de Uso:	Resolver un sistemas de ecuaciones diferenciales ordinarias.
Notas:	
Excepciones:	Si ha elegido salir, terminar sesión con los módulos Sistemas de Ecuaciones Diferenciales Ordinarias, Sistemas de Ecuaciones Lineales, Graficación y la aplicación Excel.
Salida:	
Pre-condiciones:	No hay modificación del atributo Parámetros del Algoritmo No hay modificación del atributo Parámetros de Graficación
Post-condiciones:	Cambia el atributo Parámetros del Algoritmo. Cambia el atributo Parámetros de Graficación.

Contrato

Nombre:	Seleccionar tipo de operación
Responsabilidades:	Calcular, cambiar parámetros del algoritmo y de graficación, ampliar gráfico, abrir y guardar resultados e imprimir pantalla.
Referencias Cruzadas:	Funciones del Sistema: R 5.1 a R 5.14
Casos de Uso:	Resolver un sistemas de ecuaciones diferenciales ordinarias.
Notas:	

Excepciones:	Si ha elegido salir, terminar sesión con los módulos Sistemas de Ecuaciones Diferenciales Ordinarias, Sistemas de Ecuaciones Lineales, Graficación y la aplicación Excel.
Salida:	Funciones ha ser evaluadas
Pre-condiciones:	No hay modificación del atributo Parámetros del Algoritmo, No hay modificación del atributo Parámetros de Graficación No hay sesión con la impresora
Post-condiciones:	Cambia el atributo Parámetros del Algoritmo. Cambia el atributo Parámetros de Graficación. Sesión con la impresora ha sido creada La sesión se ha asociado con la interfaz del algoritmo seleccionado.

Contrato

Nombre:	Cerrar módulo
Responsabilidades:	Terminar sesión con el módulo de Sistemas de Ecuaciones Diferenciales Ordinarias.
Referencias Cruzadas:	Funciones del Sistema: R 5.1 a R 5.14
Casos de Uso:	Resolver un sistemas de ecuaciones diferenciales ordinarias.
Notas:	
Excepciones:	
Salida:	
Pre-condiciones:	Hay sesión activa con los módulos Sistemas de Ecuaciones Lineales, Sistemas de Ecuaciones Diferenciales Ordinarias Graficación y aplicación Excel.
Post-condiciones:	A terminado la sesión con los módulos de Sistemas de Ecuaciones Lineales, Sistemas de Ecuaciones Diferenciales, Ordinarias, Graficación y aplicación Excel. Las asociaciones con los módulos de Sistemas de

Ecuaciones Diferenciales Ordinarias, Sistemas de Ecuaciones Lineales, Graficación y la aplicación Excel han sido retiradas.

2.6.5 CONTRATOS DE OPERACIÓN (5)

Contrato

Nombre:	Elegir módulo de Algoritmos
Responsabilidades:	Presentar los diferentes Temas que contiene el módulo de algoritmos.
Referencias Cruzadas:	Funciones del Sistema: R 5.1 a R 5.14
Casos de Uso:	Realizar una interpolación polinomial
Notas:	
Excepciones:	Si ha elegido salir, terminar sesión.
Salida:	
Pre-condiciones:	Espera la elección del módulo.
Post-condiciones:	

Contrato

Nombre:	Elegir Interpolación
Responsabilidades:	Presentar interfaz de Interpolación polinomial
Referencias Cruzadas:	Funciones del Sistema: R 5.1 a R 5.14
Casos de Uso:	Realizar una interpolación polinomial
Notas:	
Excepciones:	Si ha elegido salir, presentar los diferentes módulos que contiene el software

Salida:

Pre-condiciones:

Post-condiciones:

Contrato

Nombre:	Seleccionar algoritmo
Responsabilidades:	Presentar la interfaz del algoritmo elegido.
Referencias Cruzadas:	Funciones del Sistema: R 5.1 a R 5.14
Casos de Uso:	Realizar una interpolación polinomial
Notas:	
Excepciones:	Si ha elegido salir, presentar los diferentes temas del módulo de algoritmos
Salida:	
Pre-condiciones:	No hay sesión activa con el módulo Interpolación Polinomial No hay sesión activa con la aplicación Excel. No hay modificación del atributo Tipo de Algoritmo
Post-condiciones:	Una sesión con el módulo Interpolación Polinomial ha sido creada Una nueva sesión con la aplicación Excel se ha creado. La sesión de la aplicación Excel se ha asociado con el módulo Interpolación Polinomial. La sesión del módulo Interpolación Polinomial se ha asociado con la Interfaz del tipo de algoritmo seleccionado. El atributo Tipo de Algoritmo a cambiado.

Contrato

Nombre:	Ingresar parámetros de algoritmo y de graficación
Responsabilidades:	Calcular, capturar y validar los parámetros ingresados, setear los atributos necesarios de acuerdo al tipo de algoritmo.
Referencias Cruzadas:	Funciones del Sistema: R 5.1 a R 5.14
Casos de Uso:	Realizar una interpolación polinomial
Notas:	
Excepciones:	Si ha elegido salir, terminar sesión con el módulo Interpolación Polinomial, Graficación y la aplicación Excel.
Salida:	
Pre-condiciones:	No hay modificación del atributo Parámetros del Algoritmo No hay modificación del atributo Parámetros de Graficación
Post-condiciones:	Cambia el atributo Parámetros del Algoritmo. Cambia el atributo Parámetros de Graficación.

Contrato

Nombre:	Seleccionar tipo de operación
Responsabilidades:	Calcular, cambiar parámetros del algoritmo y de graficación, ampliar gráfico, abrir y guardar resultados e imprimir pantalla.
Referencias Cruzadas:	Funciones del Sistema: R 5.1 a R 5.14
Casos de Uso:	Realizar una interpolación polinomial.
Notas:	
Excepciones:	Si ha elegido salir, terminar sesión con el módulo Intepolación Polinomial, Graficación y la aplicación Excel
Salida:	Funciones ha ser evaluadas
Pre-condiciones:	No hay modificación del atributo Parámetros del Algoritmo, No hay modificación del atributo Parámetros de Graficación

	No hay sesión con la impresora
Post-condiciones:	Cambia el atributo Parámetros del Algoritmo. Cambia el atributo Parámetros de Graficación. Sesión con la impresora ha sido creada La sesión se han asociada con la interfaz del algoritmo seleccionado.

Contrato

Nombre:	Cerrar módulo
Responsabilidades:	Terminar sesión con el módulo de Interpolación Polinomial
Referencias Cruzadas:	Funciones del Sistema: R 5.1 a R 5.14
Casos de Uso:	Realizar una interpolación polinomial.
Notas:	
Excepciones:	
Salida:	
Pre-condiciones:	Hay sesión activa con los módulos Interpolación Polinomial, Graficación y la aplicación Excel.
Post-condiciones:	A terminado la sesión con los módulos Interpolación Polinomial, Graficación y la aplicación Excel. Las asociaciones con los módulos Interpolación Polinomial, Graficación y la aplicación Excel han sido retiradas.

Contrato

Nombre:	Elegir módulo de Algoritmos
Responsabilidades:	Presentar los diferentes Temas que contiene el módulo de algoritmos.
Referencias Cruzadas:	Funciones del Sistema: R 5.1 a R 5.14
Casos de Uso:	Resolver ecuaciones diferenciales ordinarias.

Notas:

Excepciones: Si ha elegido salir, terminar sesión.

Salida:

Pre-condiciones: Espera la elección del módulo.

Post-condiciones:

Contrato

Nombre: Elegir ecuaciones diferenciales ordinarias

Responsabilidades: Presentar interfaz de ecuaciones diferenciales ordinarias

Referencias

Cruzadas: Funciones del Sistema: R 5.1 a R 5.14

Casos de Uso: Resolver ecuaciones diferenciales ordinarias.

Notas:

Excepciones: Si ha elegido salir, presentar los diferentes módulos que contiene el software

Salida:

Pre-condiciones:

Post-condiciones:

Contrato

Nombre: Seleccionar algoritmo

Responsabilidades: Presentar la interfaz del algoritmo elegido.

Referencias

Cruzadas: Funciones del Sistema: R 5.1 a R 5.14

Casos de Uso: Resolver ecuaciones diferenciales ordinarias.

Notas:

Excepciones:	Si ha elegido salir, Presentar los diferentes temas del módulo de algoritmos
Salida:	
Pre-condiciones:	No hay sesión activa con el módulo Ecuaciones Diferenciales Ordinarias. No hay sesión activa con el módulo Graficación No hay sesión activa con la aplicación Excel. No hay modificación de atributos del módulo
Post-condiciones:	Una sesión con el módulo Ecuaciones Diferenciales Ordinarias se ha creado. Una nueva sesión con el módulo Graficación ha sido creada. Una nueva sesión con la aplicación Excel se ha creado. La sesión aplicación Excel se han asociado con el módulo Ecuaciones Diferenciales Ordinarias. La sesión Graficación se ha asociado a la interfaz de ecuaciones diferenciales ordinarias El atributo Tipo de Algoritmo a cambiado.

Contrato

Nombre:	Ingresar parámetros de algoritmo y de graficación
Responsabilidades:	Capturar y validar los parámetros ingresados, setear los Atributos necesarios de acuerdo al tipo de algoritmo.
Referencias Cruzadas:	Funciones del Sistema: R 5.1 a R 5.14
Casos de Uso:	Resolver ecuaciones diferenciales ordinarias.
Notas:	
Excepciones:	Si ha elegido salir, terminar sesión con los módulos Ecuaciones Diferenciales Ordinarias, Graficación y la aplicación Excel.
Salida:	

Pre-condiciones:	No hay modificación del atributo Parámetros del Algoritmo
	No hay modificación del atributo Parámetros de Graficación
Post-condiciones:	Cambia el atributo Parámetros del Algoritmo.
	Cambia el atributo Parámetros de Graficación.

Contrato

Nombre:	Seleccionar tipo de operación
Responsabilidades:	Calcular, cambiar parámetros del algoritmo y de graficación, ampliar gráfico, abrir y guardar resultados e imprimir pantalla.
Referencias	
Cruzadas:	Funciones del Sistema: R 5.1 a R 5.14
Casos de Uso:	Resolver ecuaciones diferenciales ordinarias.
Notas:	
Excepciones:	Si ha elegido salir, terminar sesión con los módulos Ecuaciones Diferenciales Ordinarias, Graficación y la aplicación Excel.

Salida:	Funciones ha ser evaluadas
Pre-condiciones:	No hay modificación del atributo Parámetros del Algoritmo,
	No hay modificación del atributo Parámetros de Graficación
	No hay sesión con la impresora
Post-condiciones:	Cambia el atributo Parámetros del Algoritmo.
	Cambia el atributo Parámetros de Graficación.
	Sesión con la impresora ha sido creada
	La sesión se han asociada con la interfaz del algoritmo seleccionado.

Contrato

Nombre:	Cerrar módulo
Responsabilidades:	Terminar sesión con el módulo Ecuaciones Diferenciales Ordinarias.
Referencias Cruzadas:	Funciones del Sistema: R 5.1 a R 5.14
Casos de Uso:	Resolver ecuaciones diferenciales ordinarias.
Notas:	
Excepciones:	
Salida:	
Pre-condiciones:	Hay sesión activa con los módulos Ecuaciones Diferenciales Ordinarias, Graficación y aplicación Excel.
Post-condiciones:	A terminado la sesión con los módulos Ecuaciones Diferenciales Ordinarias, Graficación y aplicación Excel. Las asociaciones con los módulos Ecuaciones Diferenciales Ordinarias, Graficación y la aplicación Excel han sido retiradas.

Contrato

Nombre:	Elegir módulo de Algoritmos
Responsabilidades:	Presentar los diferentes Temas que contiene el módulo de algoritmos.
Referencias Cruzadas:	Funciones del Sistema: R 5.1 a R 5.14
Casos de Uso:	Resolver ecuaciones diferenciales parciales.
Notas:	
Excepciones:	Si ha elegido salir, terminar sesión.
Salida:	

Pre-condiciones: Espera la elección del módulo.

Post-condiciones:

Contrato

Nombre: Elegir ecuaciones diferenciales parciales

Responsabilidades: Presentar interfaz de ecuaciones diferenciales parciales

Referencias Cruzadas: Funciones del Sistema: R 5.1 a R 5.14

Casos de Uso: Resolver ecuaciones diferenciales parciales.

Notas:

Excepciones: Si ha elegido salir, presentar los diferentes módulos que contiene el software

Salida:

Pre-condiciones:

Post-condiciones:

Contrato

Nombre: Seleccionar algoritmo

Responsabilidades: Presentar la interfaz del algoritmo elegido.

Referencias Cruzadas: Funciones del Sistema: R 5.1 a R 5.14

Casos de Uso: Resolver ecuaciones diferenciales parciales.

Notas:

Excepciones: Si ha elegido salir, Presentar los diferentes temas del módulo de algoritmos

Salida:

Pre-condiciones: No hay sesión activa con el módulo Ecuaciones Diferenciales Parciales.
No hay sesión activa con el módulo Sistemas de Ecuaciones Lineales.
No hay sesión activa con el módulo Graficación
No hay sesión activa con la aplicación Excel.
No hay modificación de atributos del módulo

Post-condiciones: Una sesión con el módulo Ecuaciones Diferenciales Parciales se ha creado.
Una nueva sesión con el módulo Sistemas de Ecuaciones Lineales se ha creado.
Una nueva sesión con el módulo Graficación ha sido creada.
Una nueva sesión con la aplicación Excel se ha creado.
Las sesiones Sistemas de Ecuaciones Lineales y la aplicación Excel se han asociado con el módulo Ecuaciones Diferenciales Parciales.
La sesión Graficación se ha asociado a la interfaz de Sistemas de ecuaciones diferenciales ordinarias
El atributo Tipo de Algoritmo a cambiado.

Contrato

Nombre: Ingresar parámetros de algoritmo y de graficación

Responsabilidades: Capturar y validar los parámetros ingresados, setear los Atributos necesarios de acuerdo al tipo de algoritmo.

Referencias Cruzadas: Funciones del Sistema: R 5.1 a R 5.14

Casos de Uso: Resolver ecuaciones diferenciales parciales.

Notas:

Excepciones: Si ha elegido salir, terminar sesión con los módulos Sistemas de Ecuaciones Diferenciales Parciales, Sistemas de Ecuaciones Lineales, Graficación y la aplicación Excel.

Salida:

Pre-condiciones: No hay modificación del atributo Parámetros del Algoritmo
No hay modificación del atributo Parámetros de Graficación

Post-condiciones: Cambia el atributo Parámetros del Algoritmo.
Cambia el atributo Parámetros de Graficación.

Contrato

Nombre: Seleccionar tipo de operación

Responsabilidades: Calcular, cambiar parámetros del algoritmo y de graficación, ampliar gráfico, abrir y guardar resultados e imprimir pantalla.

Referencias Cruzadas: Funciones del Sistema: R 5.1 a R 5.14

Casos de Uso: Resolver ecuaciones diferenciales parciales.

Notas:

Excepciones: Si ha elegido salir, terminar sesión con los módulos Sistemas de Ecuaciones Diferenciales Parciales, Sistemas de Ecuaciones Lineales, Graficación y la aplicación Excel.

.

Salida: Funciones a ser evaluadas

Pre-condiciones: No hay modificación del atributo Parámetros del Algoritmo,
No hay modificación del atributo Parámetros de Graficación
No hay sesión con la impresora

Post-condiciones: Cambia el atributo Parámetros del Algoritmo.
Cambia el atributo Parámetros de Graficación.
Sesión con la impresora ha sido creada
La sesión se han asociada con la interfaz del algoritmo seleccionado.

Contrato

Nombre:	Cerrar módulo
Responsabilidades:	Terminar sesión con el módulo de Ecuaciones Diferenciales Parciales.
Referencias Cruzadas:	Funciones del Sistema: R 5.1 a R 5.14
Casos de Uso:	Resolver ecuaciones diferenciales parciales.
Notas:	
Excepciones:	
Salida:	
Pre-condiciones:	Hay sesión activa con los módulos Sistemas de Ecuaciones Lineales, Sistemas de Ecuaciones Diferenciales Parciales Graficación y aplicación Excel.
Post-condiciones:	A terminado la sesión con los módulos de Sistemas de Ecuaciones Lineales, Sistemas de Ecuaciones Diferenciales, Parciales, Graficación y aplicación Excel. Las asociaciones con los módulos de Sistemas de Ecuaciones Diferenciales Parciales, Sistemas de Ecuaciones Lineales, Graficación y la aplicación Excel han sido retiradas.

2.6.6 CONTRATOS DE OPERACIÓN (6)

Contrato

Nombre:	Elegir módulo de Algoritmos
Responsabilidades:	Presentar los diferentes Temas que contiene el módulo de algoritmos.
Referencias Cruzadas:	Funciones del Sistema: R 5.1 a R 5.14
Casos de Uso:	Encontrar raíces de funciones no lineales.
Notas:	
Excepciones:	Si ha elegido salir, terminar sesión.
Salida:	
Pre-condiciones:	Espera la elección del módulo.
Post-condiciones:	

Contrato

Nombre:	Elegir Raíces de Funciones No Lineales
Responsabilidades:	Presentar interfaz de Raíces de funciones no lineales
Referencias Cruzadas:	Funciones del Sistema: R 5.1 a R 5.14
Casos de Uso:	Encontrar raíces de funciones no lineales.
Notas:	
Excepciones:	Si ha elegido salir, presentar los diferentes módulos que contiene el software
Salida:	
Pre-condiciones:	
Post-condiciones:	

Contrato

Nombre:	Seleccionar algoritmo
Responsabilidades:	Presentar la interfaz del algoritmo elegido.
Referencias Cruzadas:	Funciones del Sistema: R 5.1 a R 5.14
Casos de Uso:	Encontrar raíces de funciones no lineales.
Notas:	
Excepciones:	Si ha elegido salir, Presentar los diferentes temas del módulo de algoritmos
Salida:	
Pre-condiciones:	No hay sesión activa con el módulo Raíces de Funciones No Lineales. No hay sesión activa con el módulo Graficación No hay sesión activa con la aplicación Excel. No hay modificación de atributos del módulo
Post-condiciones:	Una sesión con el módulo Raíces de Funciones No Lineales se ha creado. Una nueva sesión con el módulo Graficación ha sido creada. Una nueva sesión con la aplicación Excel se ha creado. La sesión aplicación Excel se han asociado con el módulo Raíces de Funciones No Lineales. La sesión Graficación se ha asociado a la interfaz de Raíces de funciones no lineales El atributo Tipo de Algoritmo ha cambiado.

Contrato

Nombre:	Ingresar parámetros de algoritmo y de graficación
Responsabilidades:	Capturar y validar los parámetros ingresados, setear los Atributos necesarios de acuerdo al tipo de algoritmo.
Referencias Cruzadas:	Funciones del Sistema: R 5.1 a R 5.14
Casos de Uso:	Encontrar raíces de funciones no lineales.
Notas:	
Excepciones:	Si ha elegido salir, terminar sesión con los módulos Raíces de Funciones No Lineales, Graficación y la aplicación Excel.
Salida:	
Pre-condiciones:	No hay modificación del atributo Parámetros del Algoritmo No hay modificación del atributo Parámetros de Graficación
Post-condiciones:	Cambia el atributo Parámetros del Algoritmo. Cambia el atributo Parámetros de Graficación.

Contrato

Nombre:	Seleccionar tipo de operación
Responsabilidades:	Calcular, cambiar parámetros del algoritmo y de graficación, ampliar gráfico, abrir y guardar resultados e imprimir pantalla.
Referencias Cruzadas:	Funciones del Sistema: R 5.1 a R 5.14
Casos de Uso:	Encontrar raíces de funciones no lineales.
Notas:	
Excepciones:	Si ha elegido salir, terminar sesión con los módulos Raíces de Funciones No Lineales, Graficación y la aplicación Excel.

Salida:	Funciones ha ser evaluadas
Pre-condiciones:	No hay modificación del atributo Parámetros del Algoritmo, No hay modificación del atributo Parámetros de Graficación No hay sesión con la impresora
Post-condiciones:	Cambia el atributo Parámetros del Algoritmo. Cambia el atributo Parámetros de Graficación. Sesión con la impresora ha sido creada La sesión se ha asociado con la interfaz del algoritmo seleccionado.

Contrato

Nombre:	Cerrar módulo
Responsabilidades:	Terminar sesión con el módulo Raíces de Funciones No Lineales
Referencias Cruzadas:	Funciones del Sistema: R 5.1 a R 5.14
Casos de Uso:	Encontrar raíces de funciones no lineales.
Notas:	
Excepciones:	
Salida:	
Pre-condiciones:	Hay sesión activa con los módulos Raíces de Funciones No Lineales, Graficación y aplicación Excel.
Post-condiciones:	A terminado la sesión con los módulos Raíces de Funciones No Lineales, Graficación y aplicación Excel. Las asociaciones con los módulos Raíces de Funciones No lineales, Graficación y la aplicación Excel han sido retiradas.

Contrato

Nombre: Elegir módulo de Algoritmos

Responsabilidades: Presentar los diferentes Temas que contiene el módulo de algoritmos.

Referencias Cruzadas: Funciones del Sistema: R 5.1 a R 5.14

Casos de Uso: Calcular la Integral de una función

Notas:

Excepciones: Si ha elegido salir, terminar sesión.

Salida:

Pre-condiciones: Espera la elección del módulo.

Post-condiciones:

Contrato

Nombre: Elegir Integración

Responsabilidades: Presentar interfaz de Derivación

Referencias Cruzadas: Funciones del Sistema: R 5.1 a R 5.14

Casos de Uso: Calcular la integral de una función.

Notas:

Excepciones: Si ha elegido salir, presentar los diferentes módulos que contiene el software

Salida:

Pre-condiciones:

Post-condiciones:

Contrato

Nombre:	Seleccionar algoritmo
Responsabilidades:	Presentar la interfaz del algoritmo elegido.
Referencias	
Cruzadas:	Funciones del Sistema: R 5.1 a R 5.14
Casos de Uso:	Calcular la integral de una función.
Notas:	
Excepciones:	Si ha elegido salir, presentar los diferentes temas del módulo de algoritmos
Salida:	
Pre-condiciones:	No hay sesión activa con el módulo Integración No hay sesión activa con la aplicación Excel. No hay modificación del atributo Tipo de Algoritmo
Post-condiciones:	Una sesión con el módulo Integración ha sido creada Una nueva sesión con la aplicación Excel se ha creado. La sesión de la aplicación Excel se ha asociado con el módulo Integración. La sesión del módulo Integración se ha asociado con la Interfaz del tipo de algoritmo seleccionado. El atributo Tipo de Algoritmo a cambiado.

Contrato

Nombre:	Ingresar parámetros de algoritmo y de graficación
Responsabilidades:	Calcular, capturar y validar los parámetros ingresados, setear los atributos necesarios de acuerdo al tipo de algoritmo.
Referencias	
Cruzadas:	Funciones del Sistema: R 5.1 a R 5.14
Casos de Uso:	Calcular la integral de una función.
Notas:	
Excepciones:	Si ha elegido salir, terminar sesión con el módulo Integración Graficación y la aplicación Excel.

Salida:	
Pre-condiciones:	No hay modificación del atributo Parámetros del Algoritmo
	No hay modificación del atributo Parámetros de Graficación
Post-condiciones:	Cambia el atributo Parámetros del Algoritmo.
	Cambia el atributo Parámetros de Graficación.

Contrato

Nombre:	Seleccionar tipo de operación
Responsabilidades:	Calcular, cambiar parámetros del algoritmo y de graficación, ampliar gráfico, abrir y guardar resultados e imprimir pantalla.
Referencias	
Cruzadas:	Funciones del Sistema: R 5.1 a R 5.14
Casos de Uso:	Calcular la integral de una función.
Notas:	
Excepciones:	Si ha elegido salir, terminar sesión con el módulo Integración
	Graficación y la aplicación Excel
Salida:	Funciones ha ser evaluadas
Pre-condiciones:	No hay modificación del atributo Parámetros del Algoritmo,
	No hay modificación del atributo Parámetros de Graficación
	No hay sesión con la impresora
Post-condiciones:	Cambia el atributo Parámetros del Algoritmo.
	Cambia el atributo Parámetros de Graficación.
	Sesión con la impresora ha sido creada
	La sesión se han asociada con la interfaz del algoritmo seleccionado.

Contrato

Nombre:	Cerrar módulo
Responsabilidades:	Terminar sesión con el módulo de Integración.
Referencias Cruzadas:	Funciones del Sistema: R 5.1 a R 5.14
Casos de Uso:	Calcular la integral de una función.
Notas:	
Excepciones:	
Salida:	
Pre-condiciones:	Hay sesión activa con los módulos Integración, Graficación y la aplicación Excel.
Post-condiciones:	A terminado la sesión con los módulos Integración Graficación y la aplicación Excel. Las asociaciones con los módulos Integración, Graficación y la aplicación Excel han sido retiradas

Contrato

Nombre:	Elegir módulo de Algoritmos
Responsabilidades:	Presentar los diferentes Temas que contiene el módulo de algoritmos.
Referencias Cruzadas:	Funciones del Sistema: R 5.1 a R 5.14
Casos de Uso:	Realizar una regresión polinomial
Notas:	
Excepciones:	Si ha elegido salir, terminar sesión.
Salida:	
Pre-condiciones:	Espera la elección del módulo.
Post-condiciones:	

Contrato

Nombre:	Elegir Regresión Polinomial
Responsabilidades:	Presentar interfaz de Regresión Polinomial
Referencias Cruzadas:	Funciones del Sistema: R 5.1 a R 5.14
Casos de Uso:	Realizar una Regresión polinomial
Notas:	
Excepciones:	Si ha elegido salir, presentar los diferentes módulos que contiene el software
Salida:	
Pre-condiciones:	
Post-condiciones:	

Contrato

Nombre:	Seleccionar algoritmo
Responsabilidades:	Presentar la interfaz del algoritmo elegido.
Referencias Cruzadas:	Funciones del Sistema: R 5.1 a R 5.14
Casos de Uso:	Realizar una regresión polinomial
Notas:	
Excepciones:	Si ha elegido salir, presentar los diferentes temas del módulo de algoritmos
Salida:	
Pre-condiciones:	No hay sesión activa con el módulo Regresión Polinomial No hay sesión activa con la aplicación Excel. No hay modificación del atributo Tipo de Algoritmo

Post-condiciones:	Una sesión con el módulo Regresión Polinomial ha sido creada
	Una nueva sesión con la aplicación Excel se ha creado.
	La sesión de la aplicación Excel se ha asociado con el módulo Interpolación.
	La sesión del módulo Regresión polinomial se ha asociado con la Interfaz del tipo de algoritmo seleccionado.
	El atributo Tipo de Algoritmo a cambiado.

Contrato

Nombre:	Ingresar parámetros de algoritmo y de graficación
Responsabilidades:	Calcular, capturar y validar los parámetros ingresados, setear los atributos necesarios de acuerdo al tipo de algoritmo.
Referencias Cruzadas:	Funciones del Sistema: R 5.1 a R 5.14
Casos de Uso:	Realizar una regresión polinomial
Notas:	
Excepciones:	Si ha elegido salir, terminar sesión con el módulo Regresión Polinomial, Graficación y la aplicación Excel.
Salida:	
Pre-condiciones:	No hay modificación del atributo Parámetros del Algoritmo
	No hay modificación del atributo Parámetros de Graficación
Post-condiciones:	Cambia el atributo Parámetros del Algoritmo.
	Cambia el atributo Parámetros de Graficación.

Contrato

Nombre:	Seleccionar tipo de operación
Responsabilidades:	Calcular, cambiar parámetros del algoritmo y de graficación, ampliar gráfico, abrir y guardar resultados e imprimir pantalla.
Referencias Cruzadas:	Funciones del Sistema: R 5.1 a R 5.14
Casos de Uso:	Realizar una regresión polinomial.
Notas:	
Excepciones:	Si ha elegido salir, terminar sesión con el módulo Regresión Polinomial, Graficación y la aplicación Excel
Salida:	Funciones ha ser evaluadas
Pre-condiciones:	No hay modificación del atributo Parámetros del Algoritmo, No hay modificación del atributo Parámetros de Graficación No hay sesión con la impresora
Post-condiciones:	Cambia el atributo Parámetros del Algoritmo. Cambia el atributo Parámetros de Graficación. Sesión con la impresora ha sido creada La sesión se han asociada con la interfaz del algoritmo seleccionado.

Contrato

Nombre:	Cerrar módulo
Responsabilidades:	Terminar sesión con el módulo de Regresión Polinomial
Referencias Cruzadas:	Funciones del Sistema: R 5.1 a R 5.14
Casos de Uso:	Realizar una regresión polinomial.
Notas:	
Excepciones:	
Salida:	

Pre-condiciones:	Hay sesión activa con los módulos Regresión Polinomial, Graficación y la aplicación Excel.
Post-condiciones:	A terminado la sesión con los módulos Regresión Polinomial Graficación y la aplicación Excel. Las asociaciones con los módulos Regresión Polinomial, Graficación y la aplicación Excel han sido retiradas.

CAPÍTULO 3

DISEÑO

3.1 ACTIVIDADES DE LA FASE DE DISEÑO

El capítulo anterior permitió tener una idea mejor de los objetos que componen el Paquete de Software Didáctico para la Enseñanza-Aprendizaje de Métodos Numéricos para lo cual se crearon los diagramas del modelo conceptual, los diagramas de secuencia y los contratos de operación. Este conjunto de actividades nos permitirán crear objetos con un enfoque más cercano a la implementación del mismo. Ahora diseñaremos los CLUSTERS que es un conjunto de objetos relacionados La Fig.3.1 permite observar en que fase del desarrollo del software nos encontramos.

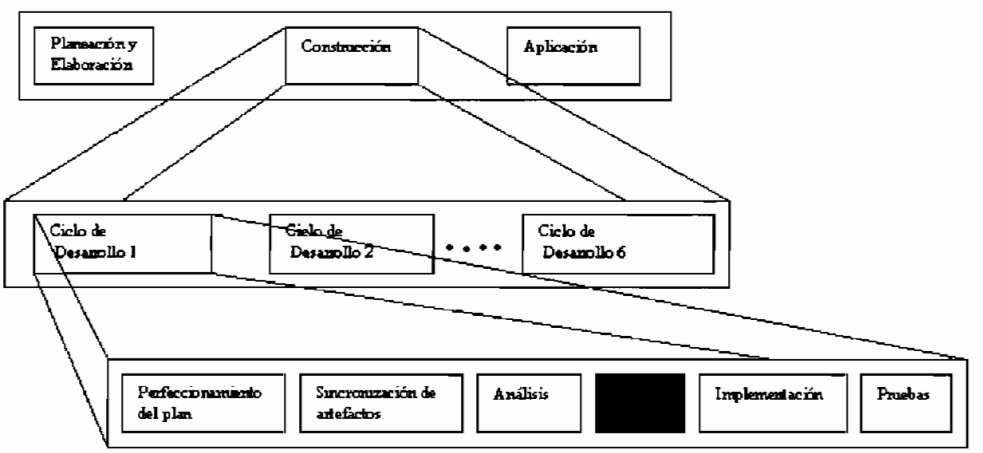


Fig. 3.1 Visión global del desarrollo de software iterativo

En la fase de Diseño se crea una solución a nivel lógico para satisfacer los requisitos, basándose en el conocimiento reunido en la fase de Análisis.

Las actividades que se realizan en la etapa de Diseño son las siguientes:

1. Definir los Casos de Uso Reales.
2. Definir Informes e Interfaz de Usuario.
3. Refinar la Arquitectura del Sistema.
4. Definir los Diagramas de Interacción.
5. Definir el Diagrama de Clases de Diseño. (*en paralelo con los Diagramas de Interacción*)
6. Definir el Esquema de Base de Datos.

3.2 CASOS DE USO REALES

Un Caso de Uso Real describe el diseño real del caso de uso según una tecnología concreta de entrada y de salida y su implementación. Si el caso de uso implica una interfaz de usuario, el caso de uso real incluirá bocetos de las ventanas y detalles de la interacción a bajo nivel con los *widgets* (botón, lista seleccionable, campo editable, etc.) de la ventana.

Como alternativa a la creación de los Casos de Uso Reales, el desarrollador puede crear bocetos de la interfaz en papel, y dejar los detalles para la fase de implementación.

En vista de que los casos de uso realizados en formato expandido tanto en la Fase de Planeación y Elaboración como en la Fase de Análisis no fueron enfocados hacia una tecnología concreta de entrada y salida, sin embargo si describen un uso real de los mismos en el sistema por lo que la realización de los casos de uso reales seria redundante. Por lo tanto se tomará la alternativa de la creación de bocetos de las interfaces, los mismos que se realizarán para cada ciclo de desarrollo.

3.2.1 BOCETOS DE LAS INTERFACES PARA EL CLUSTER (1)

En este ciclo de desarrollo se realiza el caso de uso de inicialización denominado Identificar estudiante, para el cual se ha planteado el siguiente boceto de interfaz.

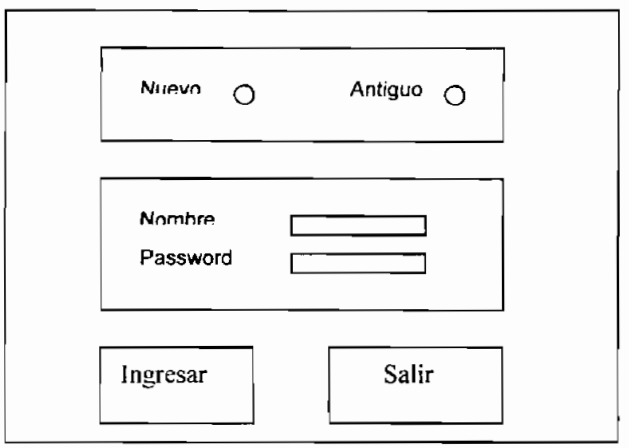


Fig. 3.2 Boceto de la interfaz de seguridad

La Fig.3.2 muestra el boceto de la interfaz de seguridad, en ella se puede observar las opciones del tipo de usuario, el ingreso del nombre y el password del usuario, al igual que los botones de ingreso y salida del software. La Fig. 3.3 muestra la interfaz de seguridad implementada

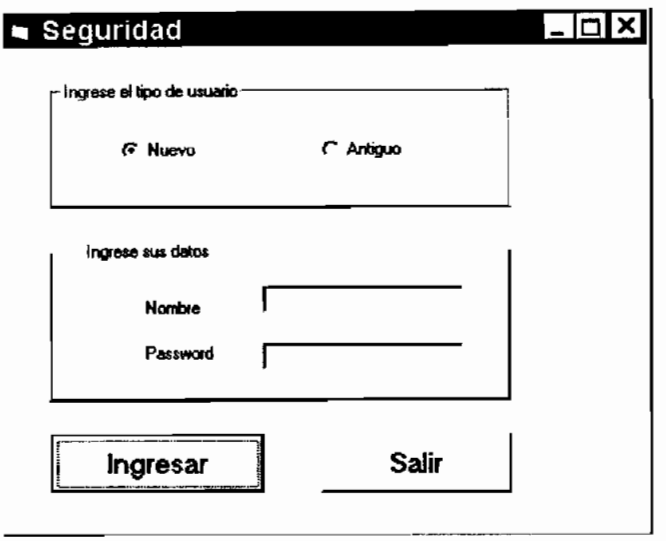


Fig. 3.3 Interfaz de seguridad implementada

En la Fig.3.4 se muestra el mensaje de error que se presenta cuando se introduce incorrectamente el nombre o el password.

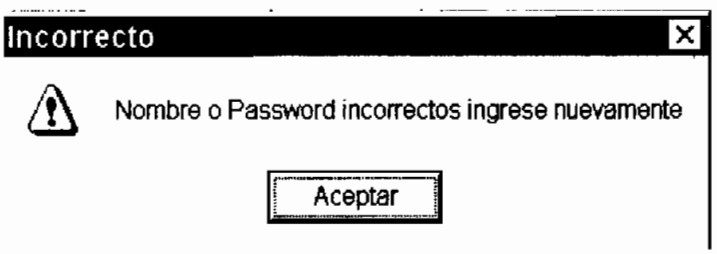


Fig. 3.4 Mensaje de error para la interfaz de seguridad

La Fig. 3.5 muestra el boceto de la interfaz de acceso a los módulos del Paquete de Software para la Enseñanza- Aprendizaje de Métodos Numéricos. La Fig.3.6 muestra la interfaz implementada

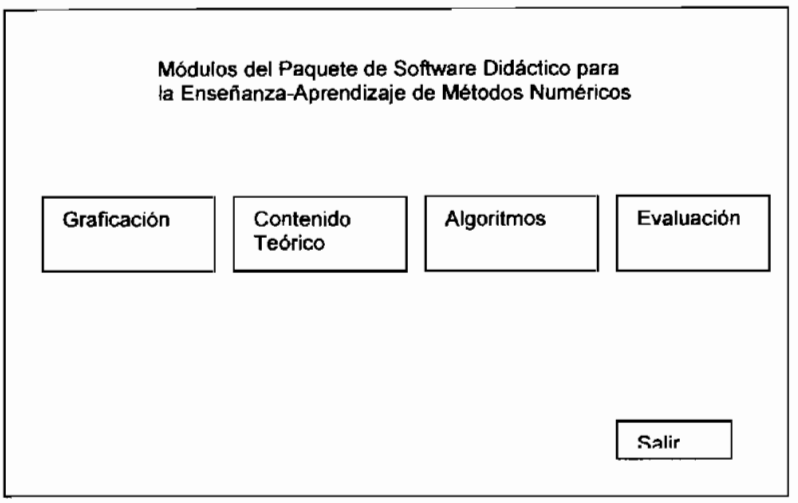
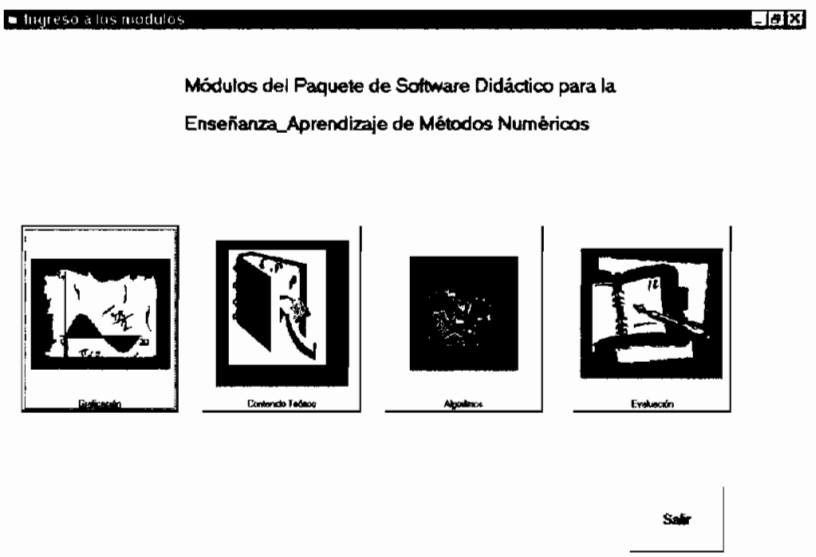


Fig. 3.5 Boceto de la interfaz de acceso a los módulos del Paquete de Software para la Enseñanza- Aprendizaje de Métodos Numéricos



La Fig. 3.6 Interfaz de acceso a los módulos del Paquete de Software para la Enseñanza- Aprendizaje de Métodos Numéricos implementada

3.2.2 BOCETOS DE LAS INTERFACES PARA EL CLUSTTER (2)

En esta etapa se analizan los tres casos de uso:

- Desplegar un tema del contenido teórico
- Resolver sistemas de ecuaciones lineales
- *Graficar una función*

3.2.2.1 Desplegar un tema del contenido teórico

La Fig.3.7 muestra el boceto de la interfaz para este caso de uso la misma que permite seleccionar un tema del contenido teórico y mostrarlo en la misma pantalla. También cuenta con la opción de búsqueda del tema o tópico deseado. Cuenta con una barra de menú para las opciones adelante, atrás e imprimir. La Fig. 3.8 muestra la interfaz implementada.

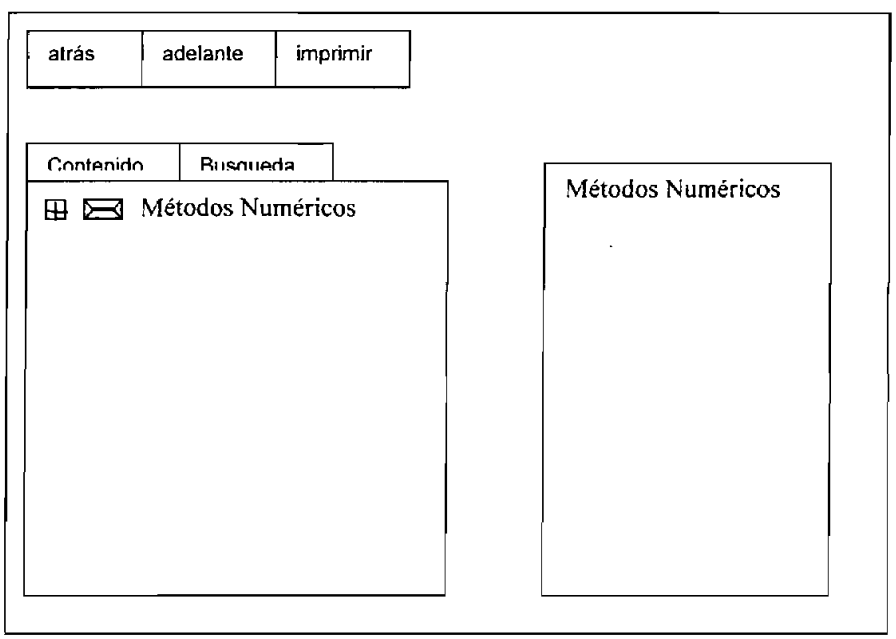


Fig. 3.7 Boceto de la interfaz del contenido teórico

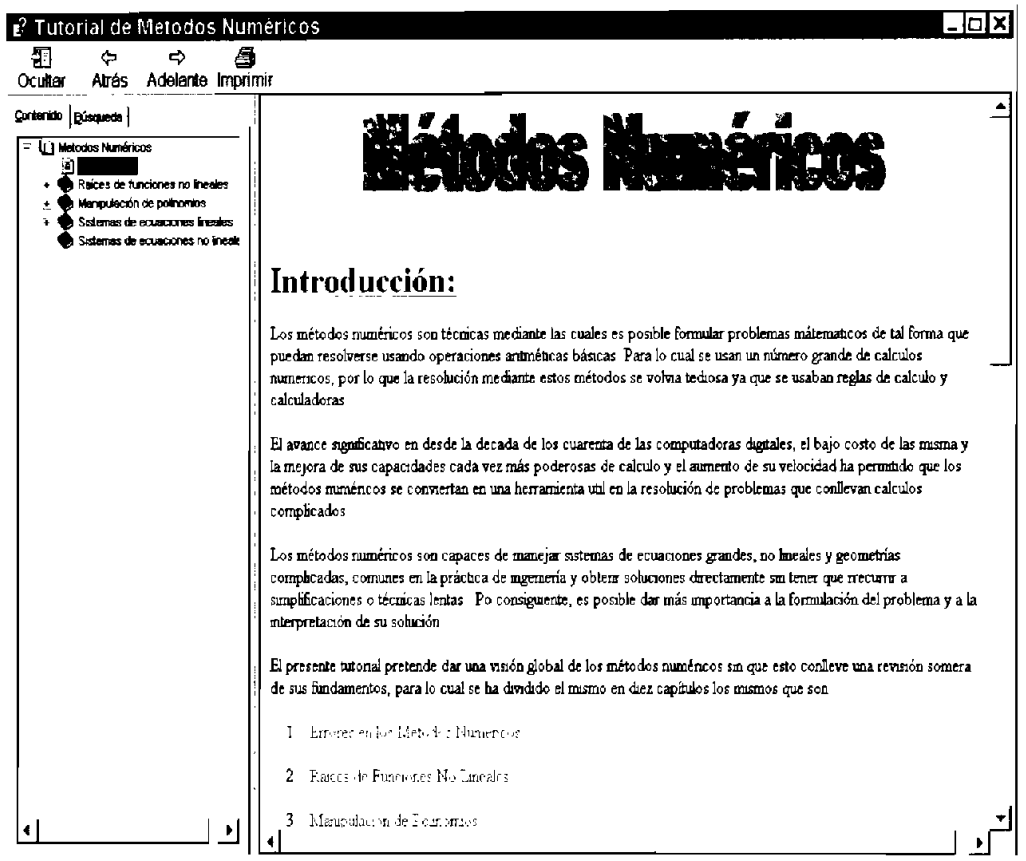


Fig.3.8 Interfaz de contenido teórico implementada

3.2.2.2 Resolver sistemas de ecuaciones lineales

La Fig.3.9 muestra el boceto de la interfaz para este caso de uso la misma que permite seleccionar entre el ingreso de parámetros para el algoritmo elegido o los parámetros de graficación si lo que desea es graficar la función y visualizar el problema antes de escoger el algoritmo a usarse, esto permite dar mayor flexibilidad al usuario ya que ambas están en la misma pantalla. Cuenta con una barra de menú para las opciones Archivo, Algoritmos, Opciones, Ayuda.

En la opción Archivo se encontraran varias subopciones al igual que en cualquier software comercial estas son: Abrir, Guardar, Imprimir, Salir. En la Opción Algoritmos se encontraran los algoritmos para este caso de uso y en Opciones algunos parámetros que pueden ser seteados para el gráfico. Se presentan los botones calcular, graficar e iteraciones que permite ver si se desea las iteraciones que produjo el algoritmo, el botón graficar muestra el gráfico de la función o de los datos ingresados.

La Fig. 3.10a muestra la interfaz implementada con la opción parámetros del algoritmo seleccionada para el algoritmo de Jacobi y la Fig. 310b muestra la interfaz con la opción de parámetros de graficación seleccionada.

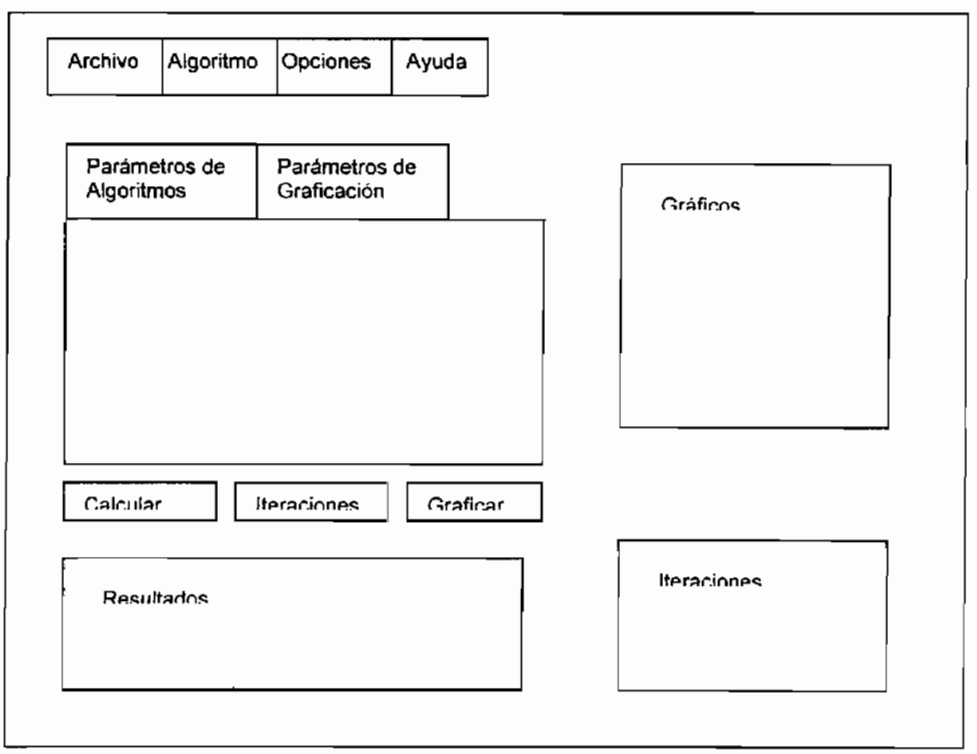


Fig.3.9a Boceto de la interfaz Resolver sistemas de ecuaciones lineales

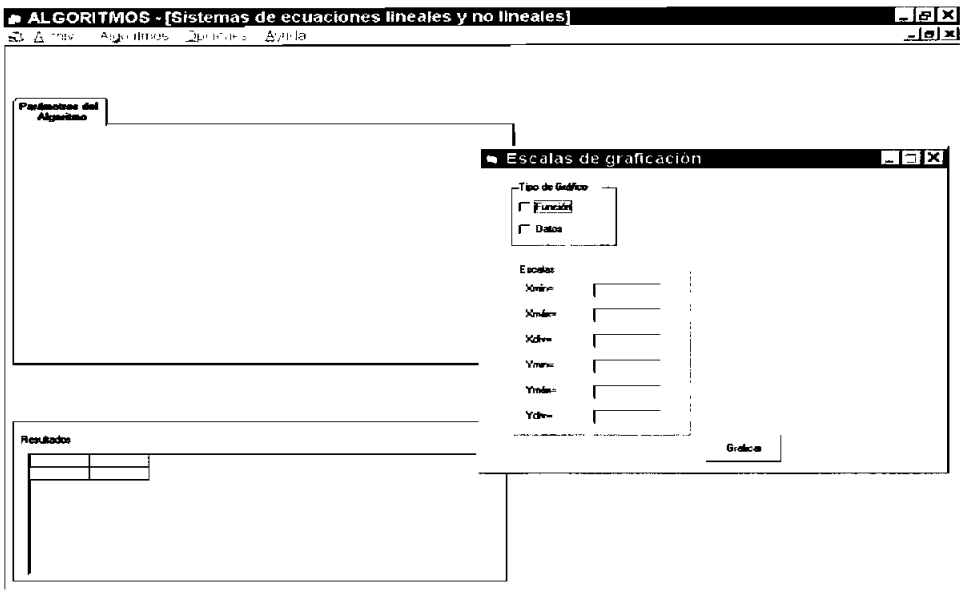


Fig. 3.10 a Interfaz de Sistemas de ecuaciones lineales implementada seleccionada la opción Parámetros de Graficación

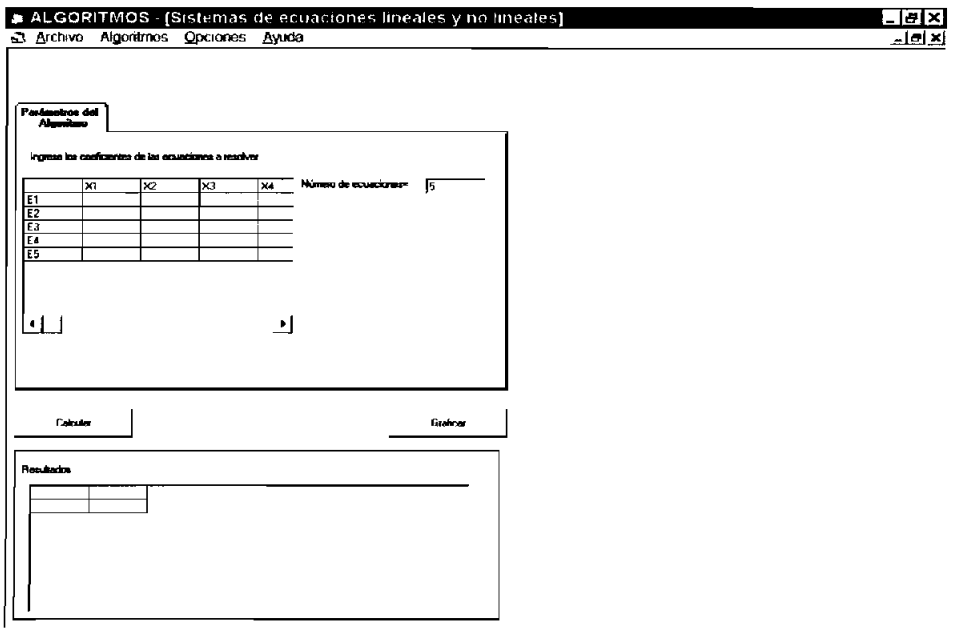


Fig. 3.10 b Interfaz de Sistemas de ecuaciones lineales implementada seleccionada la opción Parámetros del Algoritmo.

Debido a la gran cantidad de acontecimientos que pueden ocurrir al hacer uso del software, ya sea por el ingreso incorrecto de parámetros, o por la generación de errores al no encontrar un archivo necesario para el funcionamiento del mismo, se hará uso de mensajes para indicar tales acontecimientos en lo que fuere posible prever o en caso contrario se hará el manejo de errores de manera que no ejecute ninguna acción hasta que el problema sea superado.

Las Fig. 3.11 a, Fig. 3.11 b, Fig. 3.11 c, Fig. 311 d, Fig. 311 e presentan los mensajes más usuales que se verán dependiendo de lo que acontezca en el uso del software, dejando para la fase de implementación el desarrollo de mensajes referentes a un problemas especifico del algoritmo a tratar.

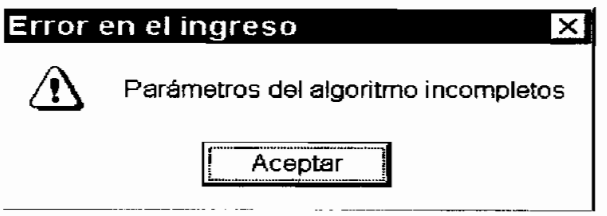


Fig. 3.11 a Mensaje de ingreso de parámetros del algoritmo incompletos

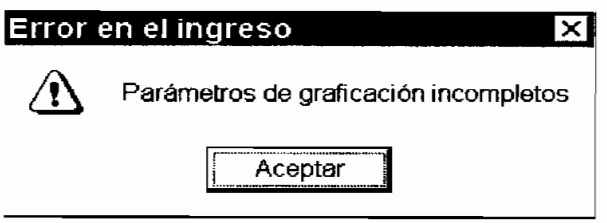


Fig. 3.11 b Mensaje de ingreso de parámetros de graficación incompletos

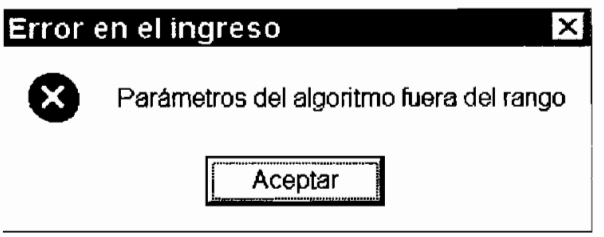


Fig. 3.11 c Mensaje de ingreso de parámetros del algoritmo fuera del rango

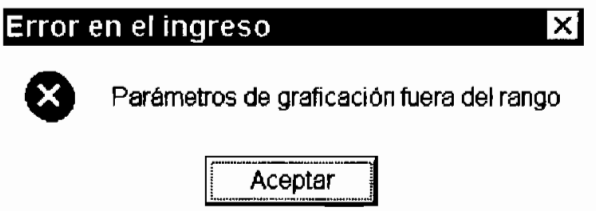


Fig. 3.11 d Mensaje de ingreso de parámetros de graficación fuera de rango

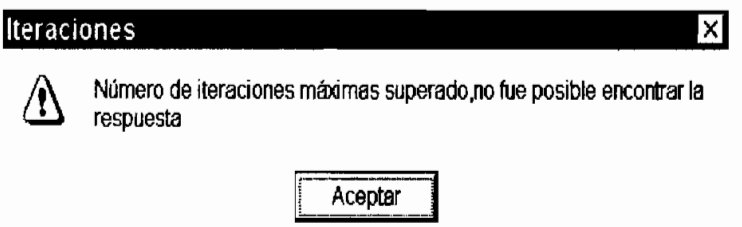


Fig. 3.11 e Mensaje de iteraciones superadas

La parte referente a graficación de la pantalla es igual a la que se implementa en el módulo de graficación e implementa todos los métodos que de este módulo, además el boceto y los mensajes son igual es para todos los casos de uso que implementen algoritmos, solamente variando en los controles que se presentan en el área de parámetros del algoritmo dependiendo de lo que necesitara el algoritmo elegido para su funcionamiento.

3.2.2.3 Interfaces para el caso de uso Graficar una función

La Fig.3.12 muestra el boceto de la interfaz para este caso de uso la misma que permite seleccionar entre graficar una función o datos, el ingreso de las funciones, los pares de datos, los parámetros de graficación (escalas). Muestra el plano donde va a ser graficada la función, el lugar donde se observaran las coordenadas del mismo así como los botones de graficar y de deshacer.

Cuenta además con una barra de menú con al opción Archivo y Ayuda; archivo tiene las siguientes subopciones: abrir, guardar, imprimir y salir. La Fig. 3.13 muestra la interfaz implementada.

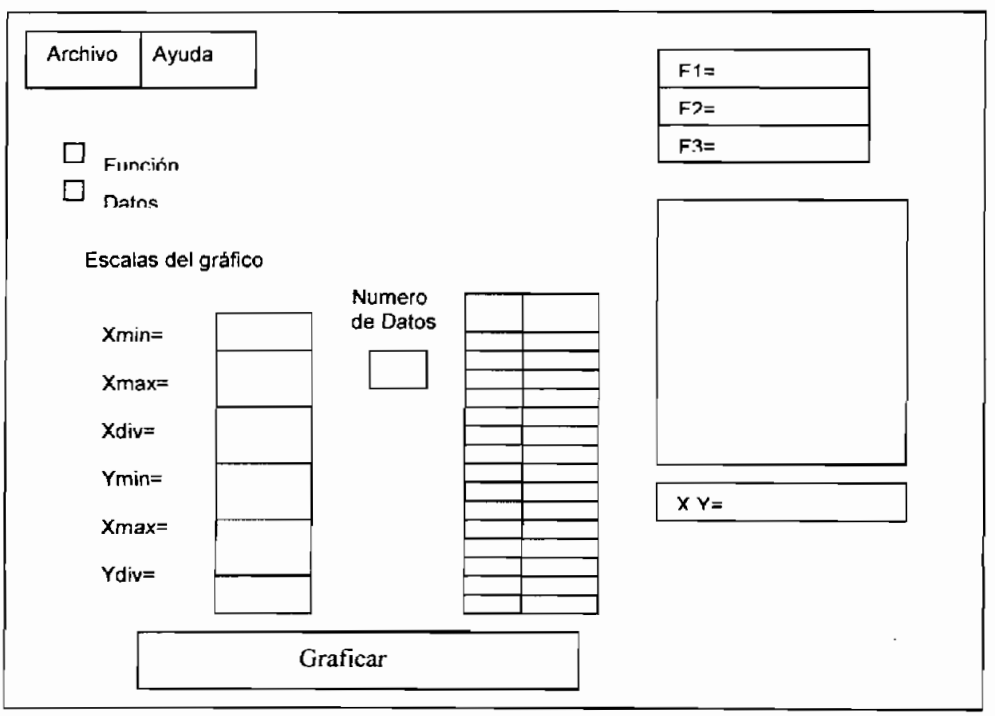


Fig. 3.12 Boceto de la interfaz de graficación

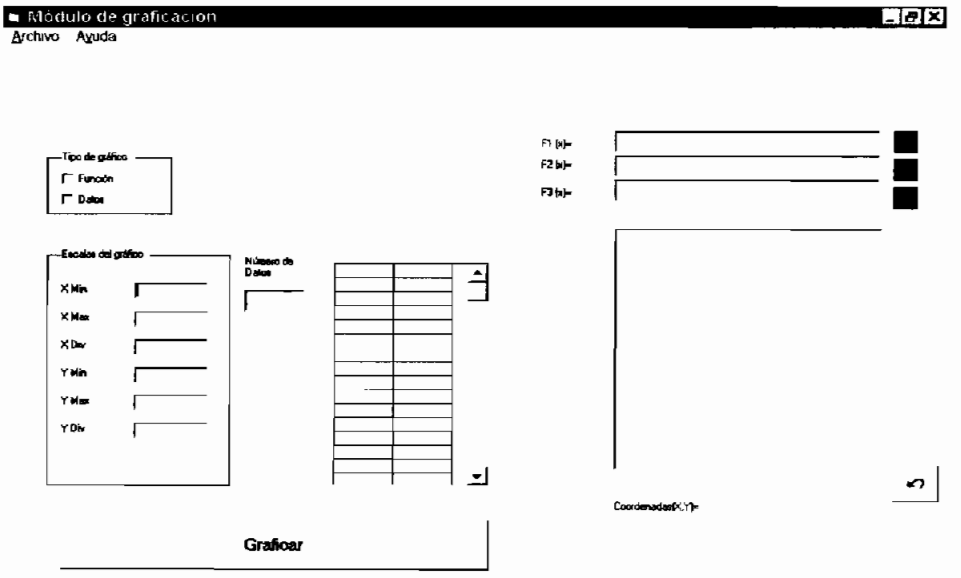


Fig.3.13 Interfaz de graficación implementada

3.2.3 BOCETOS DE LAS INTERFACES PARA EL CLUSTER (3)

En esta etapa se analizan los dos casos de uso:

- Realizar Evaluación
- Resolver sistemas de ecuaciones no lineales

3.2.3.1 Realizar Evaluación

La Fig. 3.14 muestra el boceto de la interfaz de evaluación la cual cuenta con una barra de menú que permite escoger el tipo de evaluación a realizar al igual que el tema de evaluación. También se puede elegir el tipo de pregunta a contestar ya sea verdadero falso, selección múltiple o selección múltiple con tabla, y el número de la misma.

Tiene un área donde se presenta el enunciado de la pregunta y sus posibles respuesta, además de la tabla de datos para las preguntas de selección múltiple que la requieran, también se implementa un reloj para que el usuario pueda estar pendiente del tiempo al realizar la evaluación. La Fig. 3.15 muestra la interfaz implementada

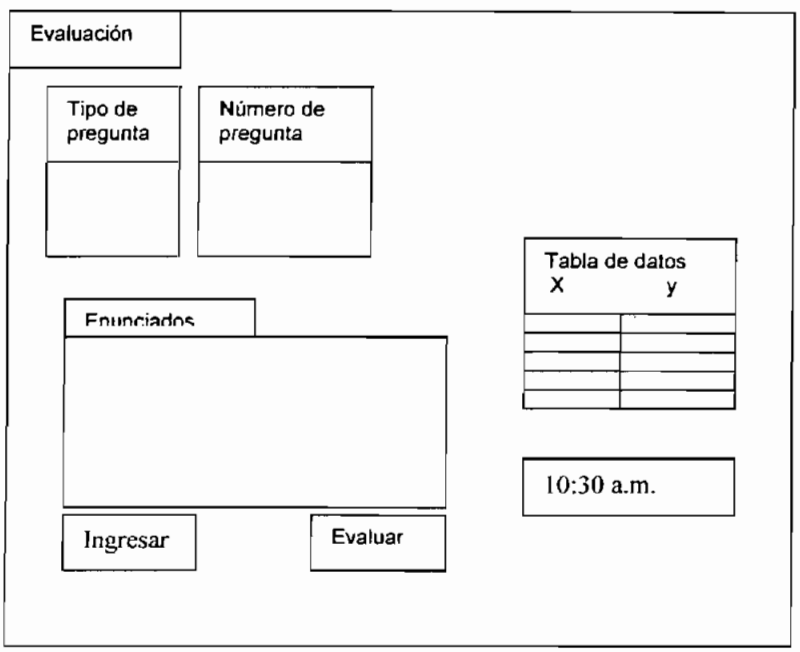


Fig. 3.14 Boceto de la interfaz de Evaluación

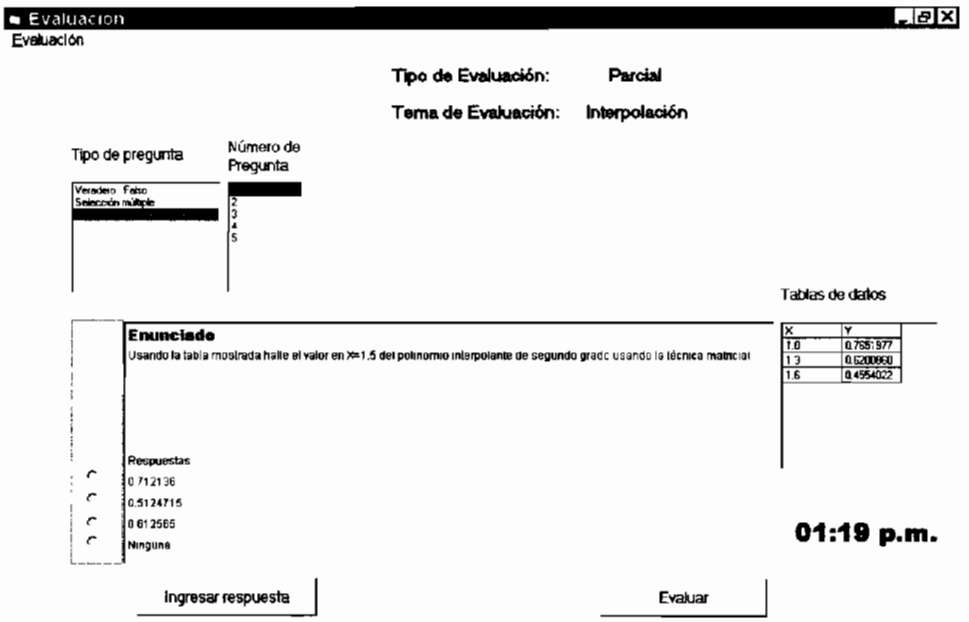


Fig. 3.15 Interfaz de evaluación implementada.

También se presenta el Boceto del informe de evaluación que es entregado al final de la evaluación realizada por el usuario en la Fig. 3.16.

INFORME DE EVALUACIÓN

Tipo de evaluación:
Tema de evaluación:
Nombre usuario:

Preguntas correctas:
Porcentaje alcanzado:
Aprobación:

Recomendadiones:

Fig. 3.16 Boceto del informe de evaluación

La Fig. 3.17 muestra la interfaz implementada.

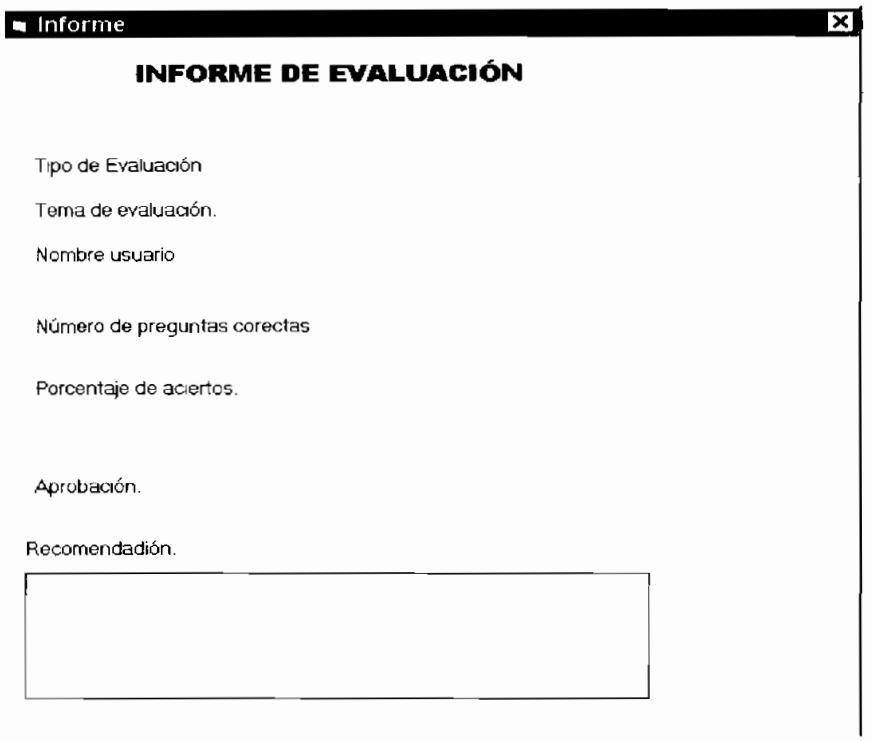


Fig. 3.17 Interfaz del informe de evaluación implementada

Los mensajes para el módulo de evaluación son los que muestran las Fig. 3.18 y Fig. 3.19 para los casos en los cuales no se han completado las preguntas y esté dentro del tiempo y para el caso en el cual el tiempo de la evaluación ha concluido y no se han contestado todas las preguntas.

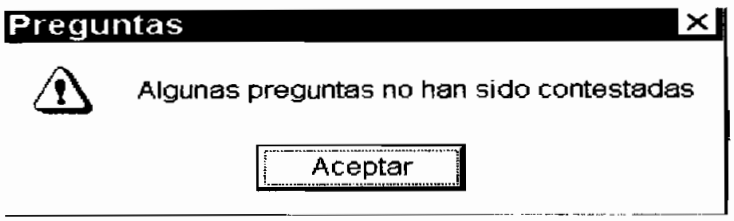


Fig. 3.18 Mensaje de preguntas no contestadas dentro del tiempo de la evaluación

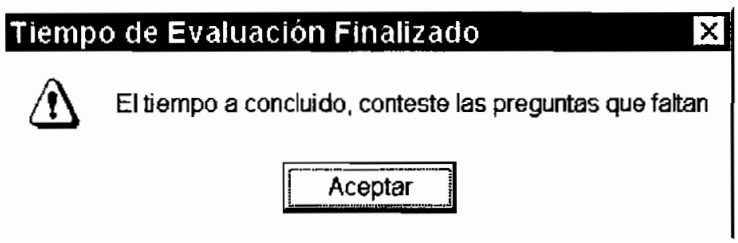


Fig. 3.19 Mensaje de las preguntas no contestadas fuera del tiempo de evaluación

3.2.3.2 Resolver sistemas de ecuaciones no lineales

El boceto de la interfaz para este caso de uso es el mismo que el del caso de uso Resolver sistemas de ecuaciones lineales de la Fig. 3.9 al igual que los mensajes que se presentan y su interfaz implementada ya que este caso de uso se implementa como parte del módulo de Sistemas de Ecuaciones Lineales y por ende la interfaz de este es la misma para este caso de uso. La Fig.3.20 muestra esta interfaz para el algoritmo de Newton.

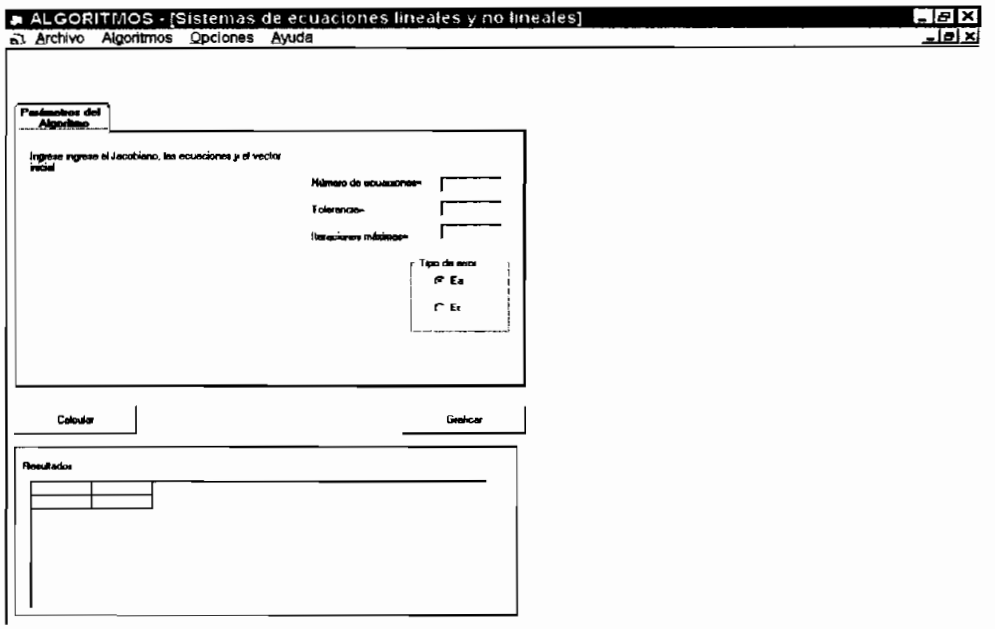


Fig. 3.20 Interfaz de Sistemas de ecuaciones lineales implementada en la cual se selecciona la opción Parámetros del Algoritmo para el algoritmo de Newton.

3.2.4 BOCETOS DE LAS INTERFACES PARA EL CLUSTER (4)

En esta etapa se analizan los dos casos de uso:

- Encontrar raíces de un polinomio
- Calcular la derivada de una función

3.2.4.1 Encontrar raíces de un polinomio

El boceto de la interfaz para este caso de uso es el mismo que el del caso de uso Resolver sistemas de ecuaciones lineales de la Fig. 3.9 al igual que los mensajes que se presentan y la parte de graficación. La Fig. 3.21 muestra la interfaz implementada para este caso de uso para el algoritmo de Newton-Bairstow.

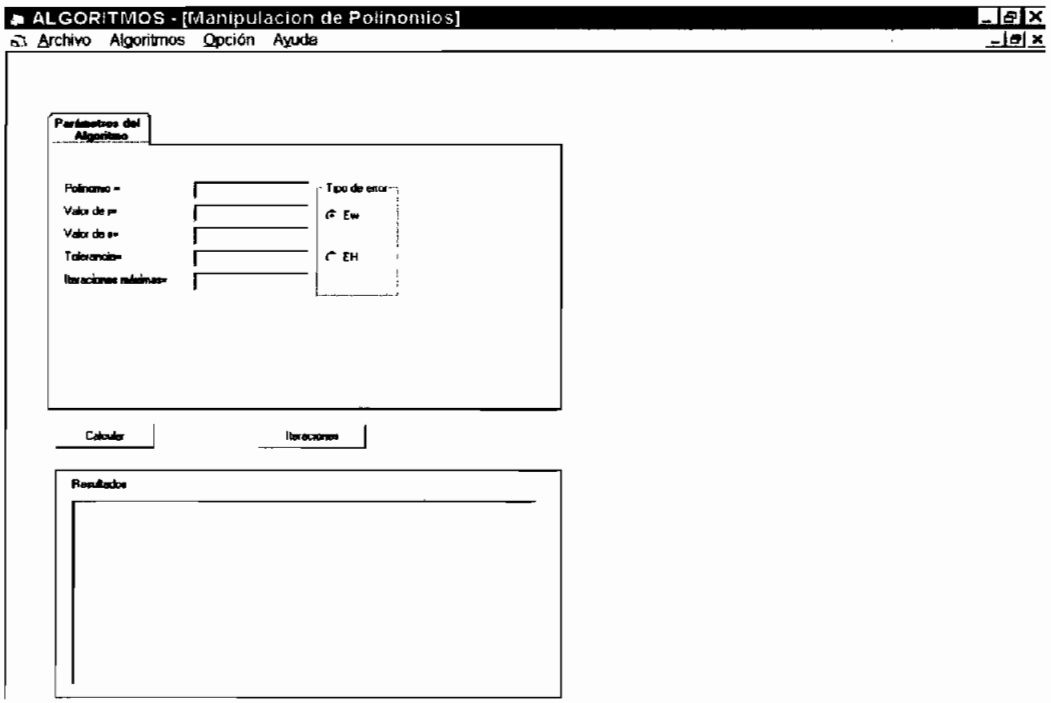


Fig. 3.21 Interfaz de Manipulación de Polinomios implementada en la cual se selecciona la opción Parámetros del Algoritmo para el algoritmo de Newton-Baistow.

3.2.4.2 Calcular la derivada de una función

El boceto de la interfaz para este caso de uso es similar que el del caso de uso Resolver sistemas de ecuaciones lineales de la Fig. 3.9 al igual que los mensajes que se presentan y la parte de graficación, salvo que este no dispone de botón de iteraciones ya que los métodos implementados no son iterativos. La Fig. 3.22 muestra la interfaz implementada para este caso de uso para el algoritmo diferencias centradas con las opciones parámetros del algoritmo y función elegidas.

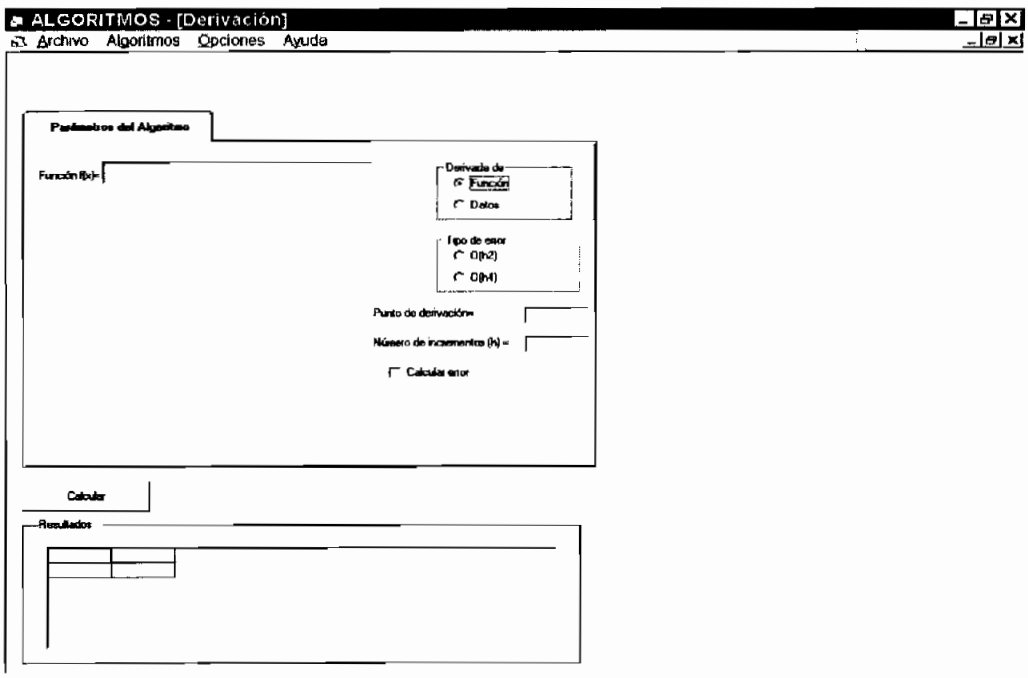


Fig. 3.22 Interfaz de Derivación implementada en la cual se selecciona la opción Parámetros del Algoritmo para el algoritmo Diferencias centradas.

3.2.5 BOCETOS DE LAS INTERFACES PARA EL CLUSTER (5)

En esta etapa se analizan los tres casos de uso:

- Resolver un sistema de ecuaciones diferenciales ordinarias
- Realizar una interpolación polinomial
- Resolver ecuaciones diferenciales ordinarias

3.2.5.1 Resolver un sistema de ecuaciones diferenciales ordinarias

El boceto de la interfaz para este caso de uso es similar que el del caso de uso Resolver sistemas de ecuaciones lineales de la Fig. 3.9 al igual que los mensajes que se presentan y la parte de graficación, salvo que éste no dispone de botón de iteraciones.

La Fig. 3.23 muestra la interfaz implementada para este caso de uso para el método de Runge-Kutta de cuarto orden, con la opción parámetro del algoritmo elegida, debe tomarse en cuenta que esta interfaz es la misma que la de ecuaciones diferenciales parciales, ya que el algoritmo de sistemas de ecuaciones lineales se implementa como un método del módulo de Ecuaciones Diferenciales Ordinarias

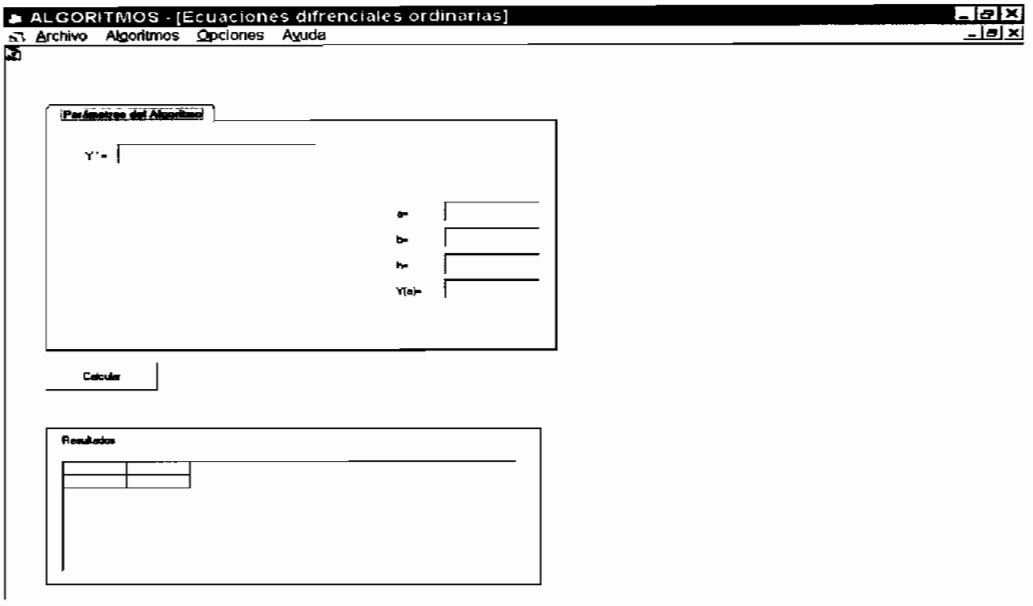


Fig. 3.23 Interfaz de Sistemas de Ecuaciones Diferenciales Ordinarias implementada en la cual se selecciona la opción Parámetros del Algoritmo para el método de Runge-Kutta de cuarto orden.

3.2.5.2 Realizar una interpolación polinomial

El boceto de la interfaz para este caso de uso es el mismo que el del caso de uso Resolver sistemas de ecuaciones lineales de la Fig. 3.9 al igual que los mensajes que se presentan, salvo por que en este no existe el botón iteraciones ya que los algoritmos implementados no son iterativos. La Fig. 3.24 muestra la interfaz implementada para este caso de uso para el algoritmo de Lagrange con la opción parámetros del algoritmo.

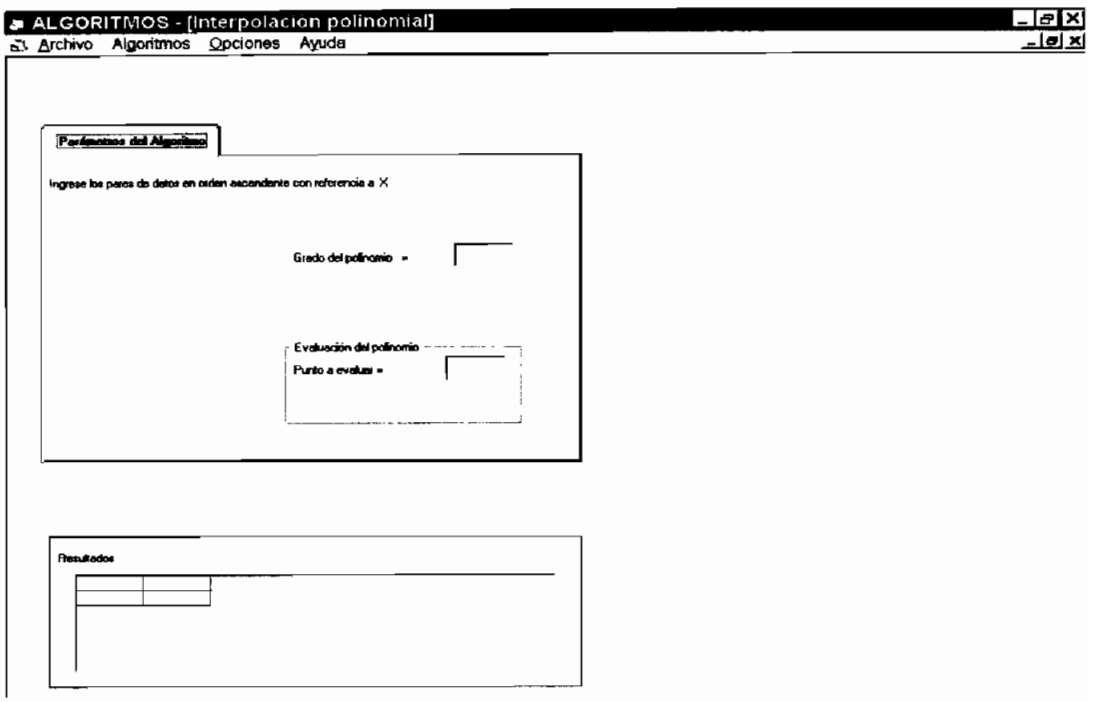


Fig. 3.24 Interfaz de Interpolación Polinomial implementada en la cual se selecciona la opción Parámetros del Algoritmo para el algoritmo de Lagrange.

3.2.5.3 Resolver ecuaciones diferenciales ordinarias

El boceto de la interfaz para este caso de uso es similar que el del caso de uso Resolver sistemas de ecuaciones lineales de la Fig. 3.9 al igual que los mensajes que se presentan y la parte de graficación, salvo que este no dispone de botón de iteraciones. La Fig. 3.25 muestra la interfaz implementada para este caso de uso para el método de Euler, con la opción parámetros del algoritmo elegida.

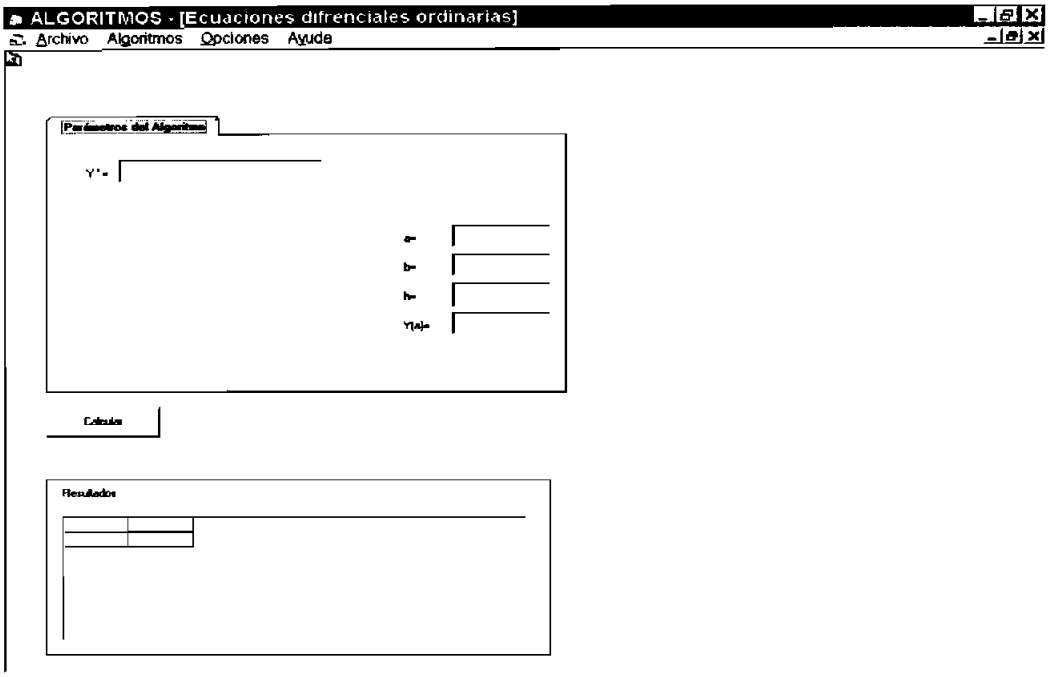


Fig. 25 Interfaz de Ecuaciones Diferenciales Ordinarias implementada en la cual se selecciona la opción Parámetros del Algoritmo para el método de Euler

3.2.6 BOCETOS DE LAS INTERFACES PARA EL CLUSTER (6)

En esta etapa se analizan los cuatro casos de uso:

- Resolver ecuaciones diferenciales parciales
- Encontrar raíces de funciones no lineales
- Calcular la integral de una función
- Realizar una regresión polinomial

3.2.6.1 Resolver ecuaciones diferenciales parciales

El boceto de la interfaz para este caso de uso es similar que el del caso de uso Resolver sistemas de ecuaciones lineales de la Fig. 3.9 al igual que los mensajes que se presentan para la parte del algoritmo ya que la parte de graficación no se implementa, debido a que la gráfica de los datos de este tipo de solución requeriría el manejo de gráficos en tres dimensiones cosa que esta fuera de las Especificaciones de Requerimientos del Paquete de Software Didáctico para la Enseñanza-Aprendizaje de Métodos Numéricos. La Fig. 3.26 muestra la interfaz implementada para este caso de uso para la ecuación del calor.

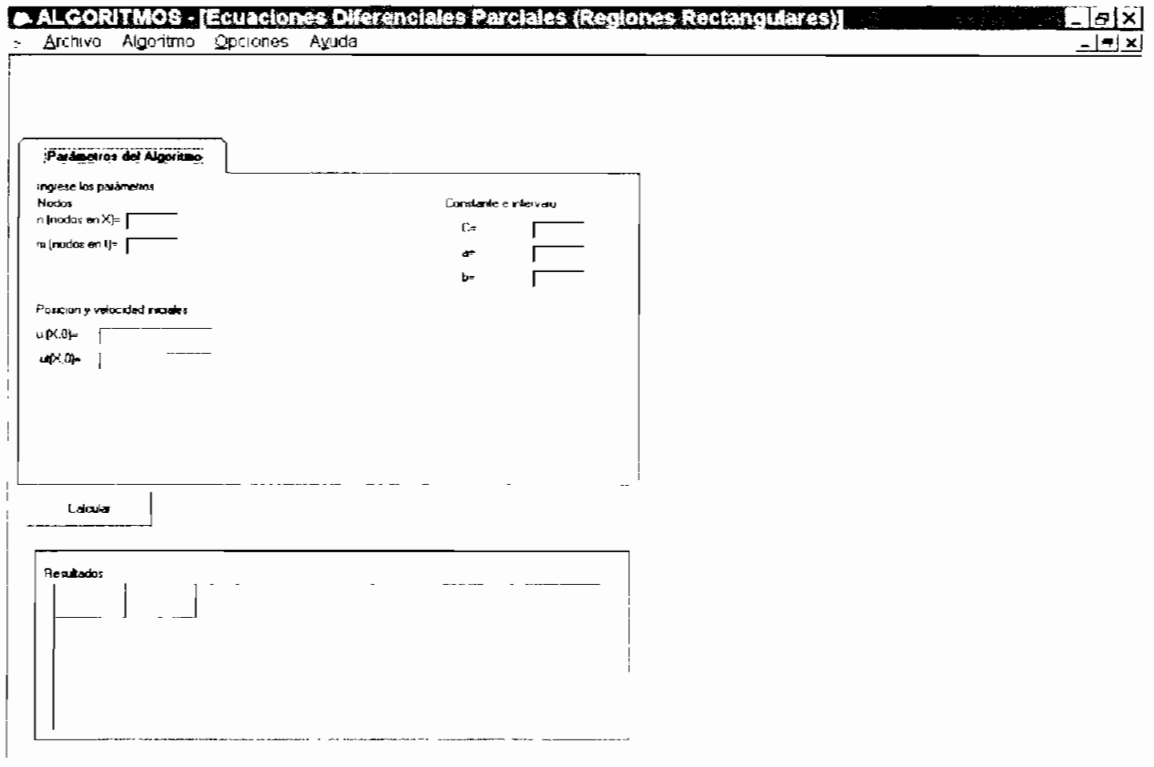


Fig. 3.26 Interfaz de Ecuaciones Diferenciales Parciales en la cual se selecciona la opción Parámetros del Algoritmo para la Ecuación de ondas

3.2.6.2 Encontrar raíces de funciones no lineales

El boceto de la interfaz para este caso de uso es similar que el del caso de uso Resolver sistemas de ecuaciones lineales de la Fig. 3.9 al igual que los mensajes que se presentan para la parte del algoritmo y la parte de graficación. La Fig. 3.27 muestra la interfaz implementada para este caso de uso para el algoritmo de la secante.

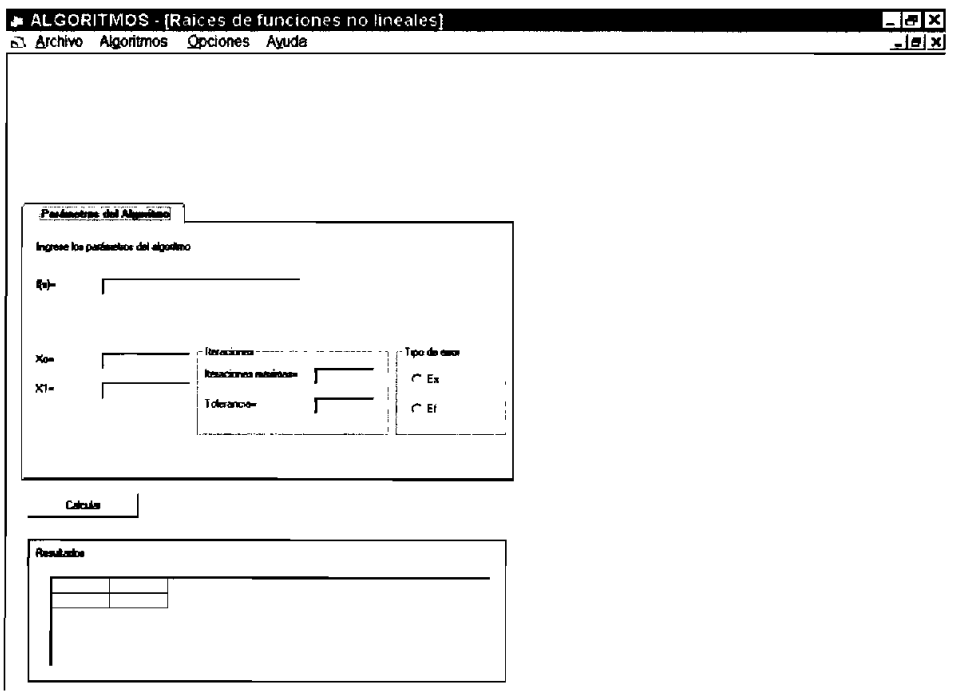


Fig. 3.27 Interfaz de Raíces de Funciones No Lineales para la cual se a elegido la opción parámetros del algoritmo para el algoritmo de la Secante

3.2.6.3 Calcular la integral de una función

El boceto de la interfaz para este caso de uso es similar que el del caso de uso Resolver sistemas de ecuaciones lineales de la Fig. 3.9 al igual que los mensajes que se presentan para la parte del algoritmo y la parte de graficación. La Fig. 3.28 muestra la interfaz implementada para este caso de uso para el algoritmo de Newton-Cotes cerrada.

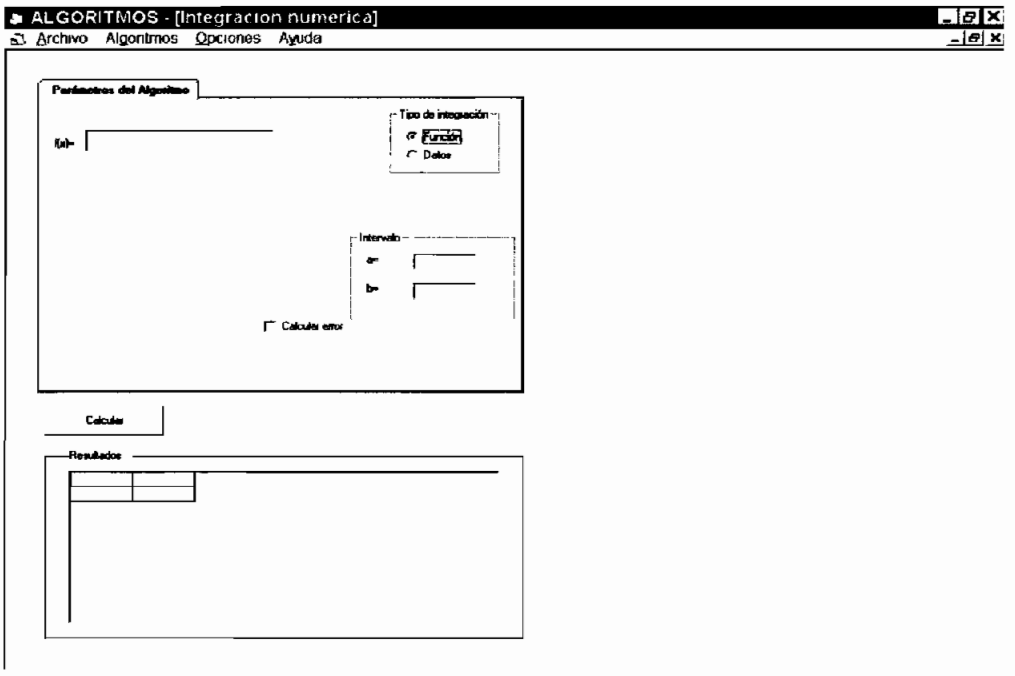


Fig. 3.28 Interfaz para Integración para la cual se a elegido la opción parámetros del algoritmo para el de Newton-Cotes cerrado.

3.2.6.4 Realizar una regresión polinomial

El boceto de la interfaz para este caso de uso es el mismo que el del caso de uso Resolver sistemas de ecuaciones lineales de la Fig. 3.9 al igual que los mensajes que se presentan, salvo por el cambio del nombre iteraciones por el de ajuste. Este caso de uso se implementa como parte del módulo Interpolación polinomial y por ende la interfaz de este es la misma para este caso de uso. La fig.3.24 muestra esta interfaz para el algoritmo Regresión Polinomial.

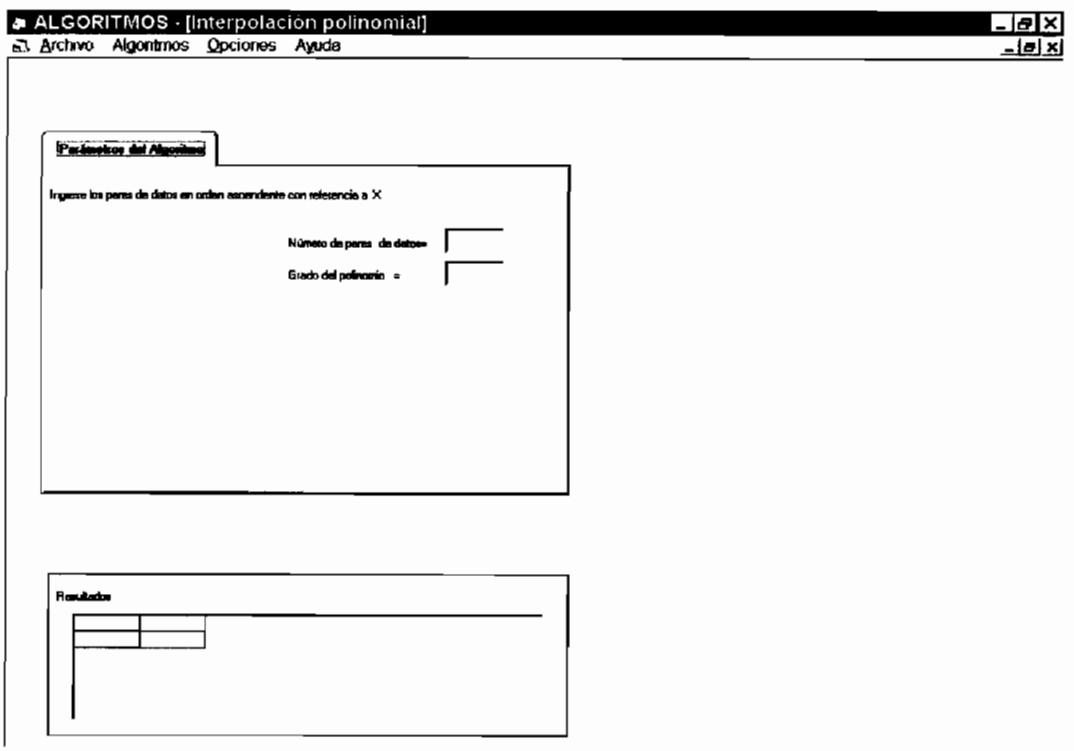


Fig. 3.29 Interfaz de Regresión Polinomial implementada en la cual se selecciona la opción Parámetros del Algoritmo para el algoritmo Regresión Polinomial.

3.3 ARQUITECTURA DEL SISTEMA

La arquitectura del sistema corresponde a la forma como se disponen los elementos que formarán el sistema ya sean estos interfaces, módulos de clase, bases de datos. Básicamente existen dos tipos de arquitectura:

- De dos capas
- De tres capas

En la Primera se coloca la lógica de las aplicaciones dentro de las declaraciones de la interfaz, que lee y escriben directamente en una base de datos; no hay una capa intermedia que separe la lógica. En la segunda las interfaces envían peticiones de trabajo a la capa intermedia y esta se comunica con la capa de almacenamiento del extremo posterior la que se comunicará con la base de datos.
Como se aprecia en la arquitectura de tres capas la interfaz realiza poco procesamiento, con lo cual se tienen las siguientes ventajas:

- Aislamiento de la lógica de aplicaciones en componentes susceptibles de reutilizarse después en otros sistemas.
- Distribución de las capas en varios nodos físicos de computo y en varios procesos.
- Asignación de los diseñadores que construyan varias capas.

El uso de la arquitectura de tres capas permite que el sistema sea más flexible por lo que se opta por implementar una arquitectura de estas características.

La Fig. 3.30 muestra una arquitectura de tres capas.

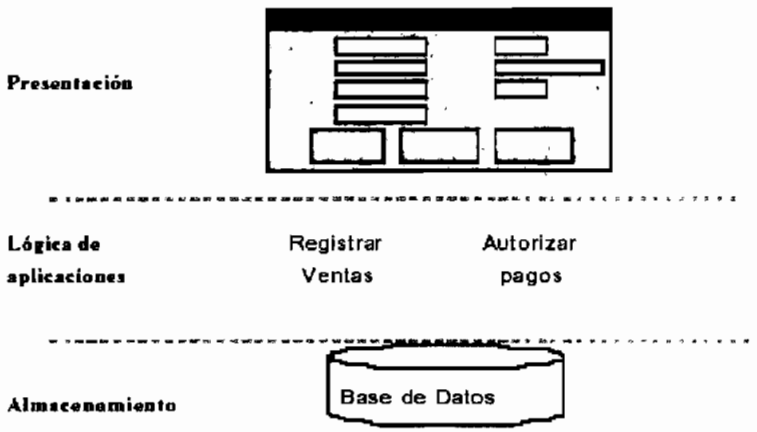


Fig. 3.30 Arquitectura de tres capas

En un diseño orientado a objetos la capa de la lógica de aplicaciones se divide en otra menos densas como son:

- Objetos de dominio: clases que representan los conceptos del dominio.
- Servicios: los objetos servicio de funciones como la interacción de bases de datos.

En la Fig. 3.31 se observa la división de la capa de la lógica de aplicaciones en los objetos de dominio venta, productos, pago. La capa servicios formada por la interfaz de la base de datos, servicios.

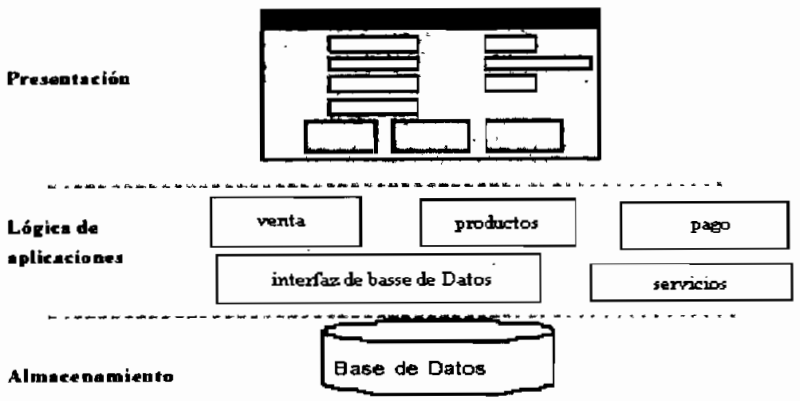


Fig. 3.31 Arquitectura de tres capas con división de la capa lógica de aplicaciones.

Como estamos trabajando con la notación UML, ésta también ofrece el mecanismo *paquete* que permite describir los grupos de elementos o subsistemas. Un paquete se muestra gráficamente como una carpeta con etiquetas. Los paquetes subordinados se incluyen en su interior y el nombre del paquete se encuentra en la etiqueta si este describe sus elementos; en caso contrario estará en el centro de la misma.

La Fig. 3.32 muestra este concepto, mientras que la Fig. 3.33 aplica ésta a la arquitectura de tres capas.

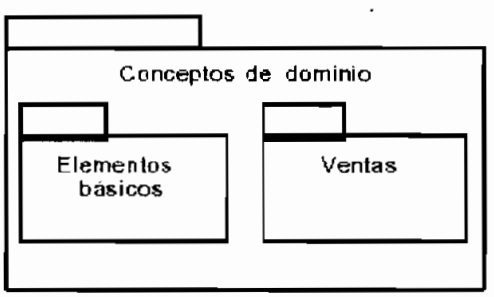


Fig. 3.32 Ejemplo de paquetes

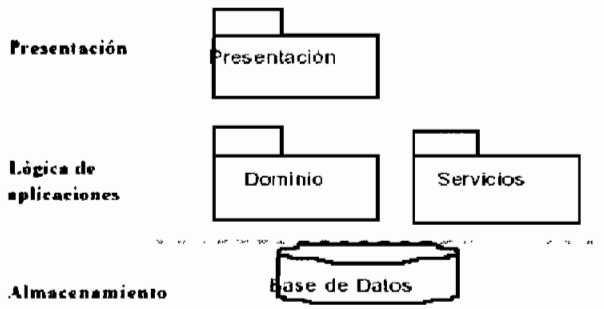


Fig. 3.33 Arquitectura de tres capas usando paquetes

A continuación se muestra la arquitectura preliminar para el Paquete de Software para la Enseñanza-Aprendizaje de Métodos Numéricos, La Fig.3.34 muestra la arquitectura usando el concepto de paquetes

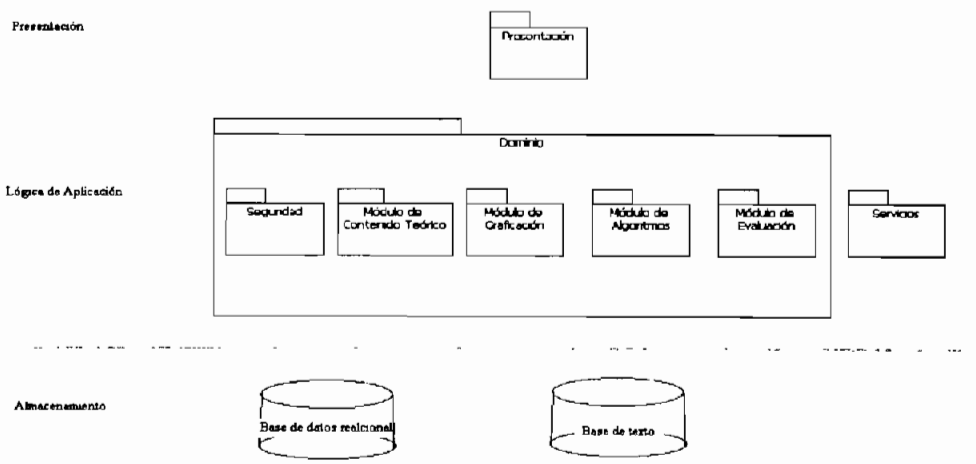


Fig. 3.34 Arquitectura del Paquete de Software Didáctico para la Enseñaza-Aprendizaje de Métodos Numéricos usando Paquetes

En cada ciclo de desarrollo debe irse mejorando la arquitectura del sistema lo que implica mostrar diagramas de arquitectura para cada ciclo de desarrollo, pero debido a que solamente en tres ciclos de desarrollo este sufre cambios el presentar los diagramas seria redundantes, por lo que se ha optado por dar una presentación final de la arquitectura del sistema la cual muestra los cambios que se han producido.

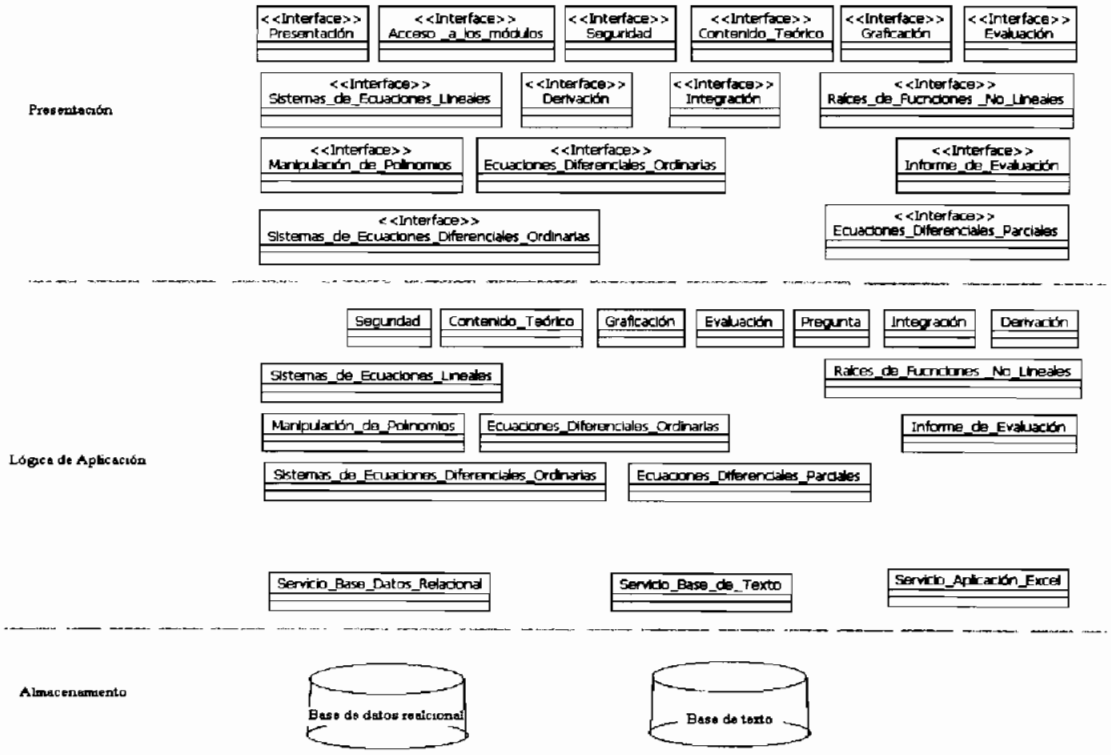


Fig.3.35 Arquitectura final del Paquete de Software Didáctico para la Enseñaza-Aprendizaje de Métodos Numéricos

Como se observa en la Fig.3.35 el módulo y la interfaz de Sistemas de Ecuaciones No Lineales no aparece en la misma debido a que en el ciclo de desarrollo 3 en la etapa de implementación se ha optado por incluirlo en el módulo Sistemas de Ecuaciones Lineales, ya que éste hace uso de los procedimientos de este módulo.

De la misma manera el ciclo de desarrollo 5 se implementa el módulo de Sistemas de Ecuaciones Diferenciales Ordinarias como un procedimiento del módulo de Ecuaciones Diferenciales Ordinarias, por lo que este módulo y su interfaz no aparecen. También el Módulo de Regresión Polinomial se ha incluido en el módulo Manipulación de Polinomios debido a que se implementa solamente un algoritmo y la creación de un módulo para éste significará el aumento innecesario de recurso.

Finalmente el módulo usuario no se implementa debido a que es un concepto del mundo real y no un componente del software, además sus funciones se incluyeron en el módulo de Clave.

3.4 DIAGRAMAS DE COLABORACIÓN

Los Diagramas de Interacción muestran el intercambio de mensajes entre instancias del modelo de clases para cumplir las post-condiciones establecidas en un contrato. Hay dos clases de Diagramas de Interacción:

1. Diagramas de Colaboración.
2. Diagramas de Secuencia.

De entre ambos tipos se prefieren los Diagramas de Colaboración por su expresividad y por su economía espacial (una interacción compleja puede ser muy larga en un Diagrama de Secuencia).

La creación de los Diagramas de Colaboración de un sistema es una de las actividades más importantes en el desarrollo orientado a objetos, pues al construirlos se toman decisiones claves acerca del funcionamiento del futuro sistema.

La creación de estos diagramas, por tanto, debería ocupar un porcentaje significativo en el esfuerzo dedicado al proyecto entero.

Creación de Diagramas de Colaboración

Para crear los Diagramas de Colaboración se pueden seguir los siguientes consejos:

1. Crear un diagrama separado para cada operación del sistema en desarrollo en el ciclo de desarrollo actual. Para cada evento del sistema, hacer un diagrama con él como mensaje inicial.
2. Si el diagrama se complica, dividirlo en diagramas más pequeños.
3. Usando los apartados de responsabilidades y de post-condiciones del contrato de operación, y la descripción del caso de uso como punto de partida, diseñar un sistema de objetos que interaccionan para llevar a cabo las tareas requeridas.

Las responsabilidades están ligadas a las obligaciones de un objeto en cuanto a su comportamiento. Básicamente, estas responsabilidades son de los dos siguientes tipos:

1. Conocer:
- Conocer datos privados encapsulados.
- Conocer los objetos relacionados.
- Conocer las cosas que puede calcular o derivar.

2. Hacer:

- . Hacer algo él mismo.
- . Iniciar una acción en otros objetos.
- . Controlar y coordinar actividades en otros objetos.

Por ejemplo, puedo decir que "un *Recibo* es responsable de imprimirse" (tipo hacer), o que "una *Transacción* es responsable de saber su fecha" (tipo conocer). Las responsabilidades de tipo"conocer" se pueden inferir normalmente del Modelo Conceptual.

Una responsabilidad no es lo mismo que un método, pero los métodos se implementan para satisfacer responsabilidades.

3.4.1 DIAGRAMAS DE COLABORACIÓN PARA EL CLUSTER (1)

Diagramas de colaboración para el caso de uso Identificar estudiante

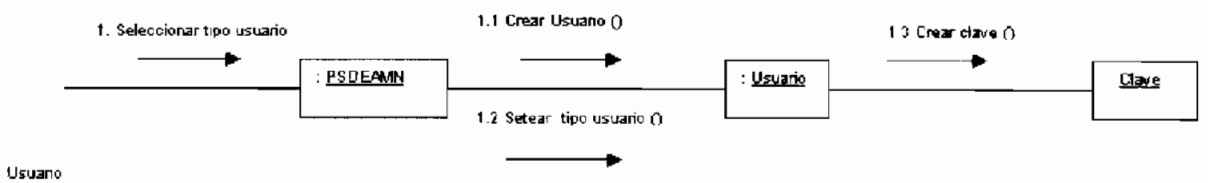


Fig. 3.36 Operación seleccionar tipo de usuario

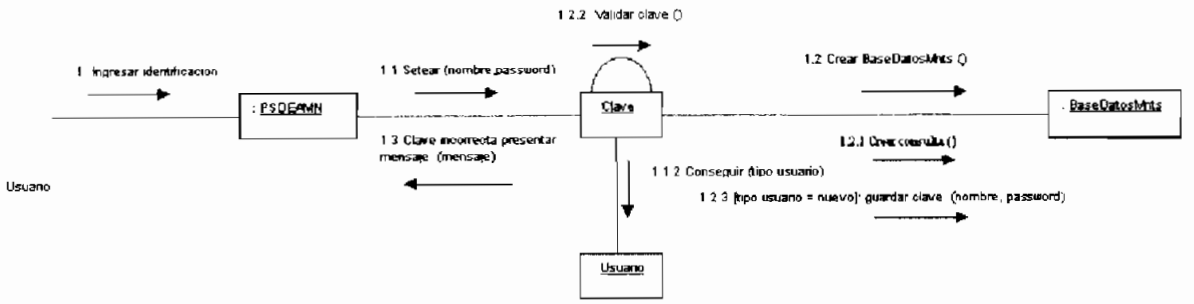


Fig. 3.37 Operación ingresar identificación

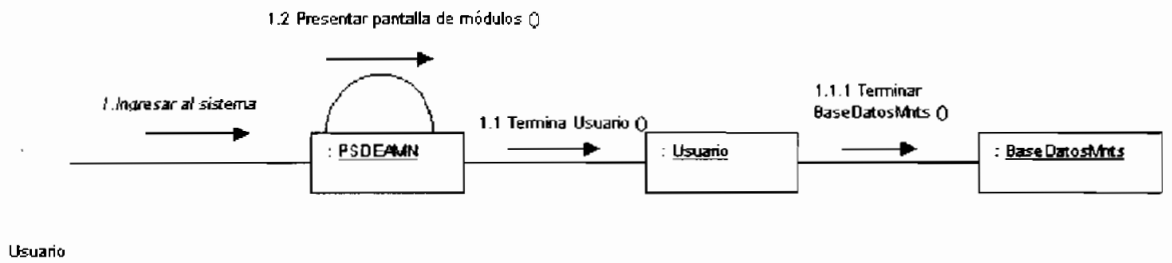


Fig. 3.38 Operación ingresar al sistema

3.4.2 DIAGRAMAS DE COLABORACIÓN PARA EL CLUSTER (2)

Diagramas de colaboración para el caso de uso Leer un tema del contenido teórico

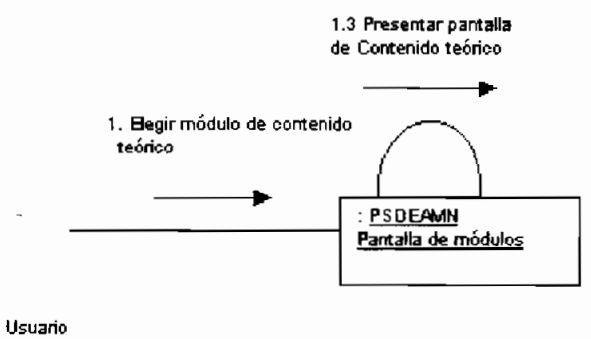


Fig. 3.39 Operación Elegir módulo de contenido teórico

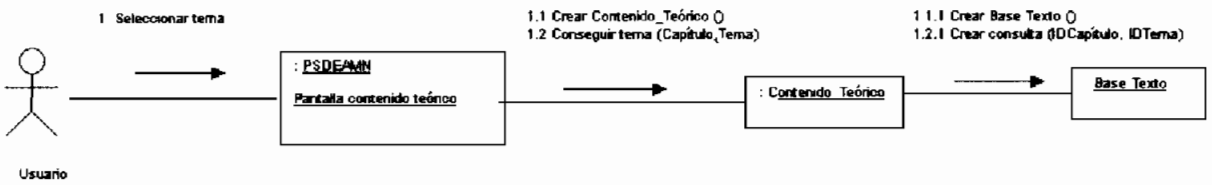


Fig. 3.40 Operación seleccionar tema

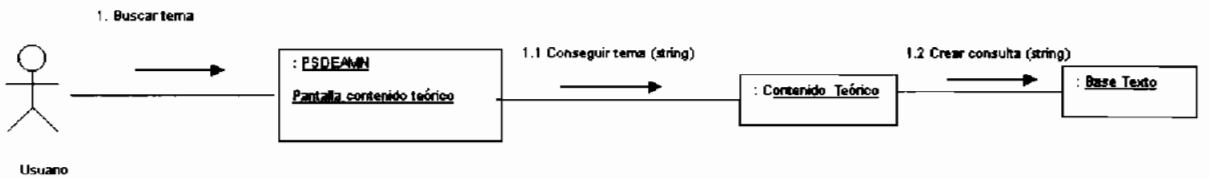


Fig. 3.41 Operación buscar tema

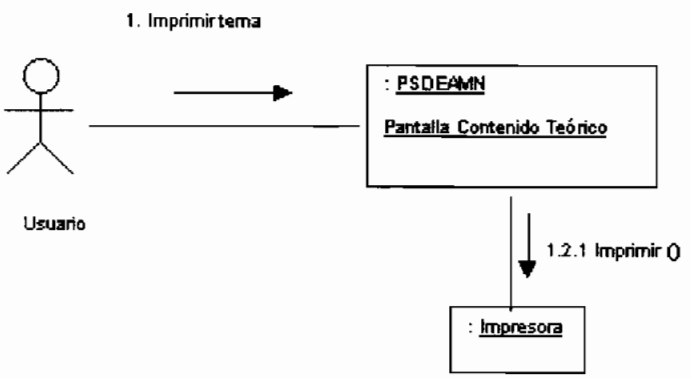


Fig. 3.42 Operación imprimir tema

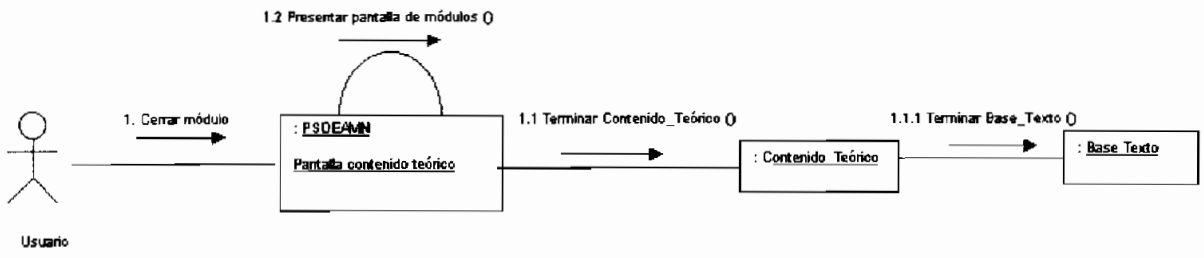


Fig. 3.43 Operación cerrar módulo

Diagramas de colaboración para el caso de uso Resolver sistemas de ecuaciones lineales

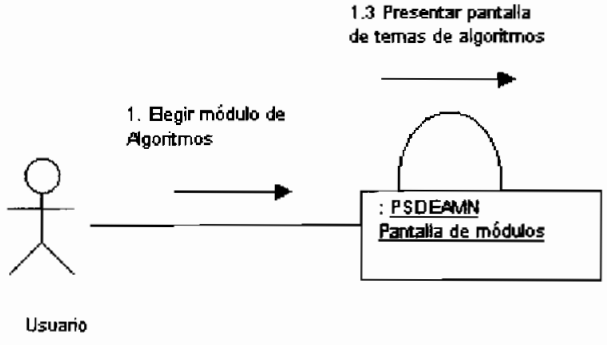


Fig. 3.44 Operación Elegir módulos de algoritmos

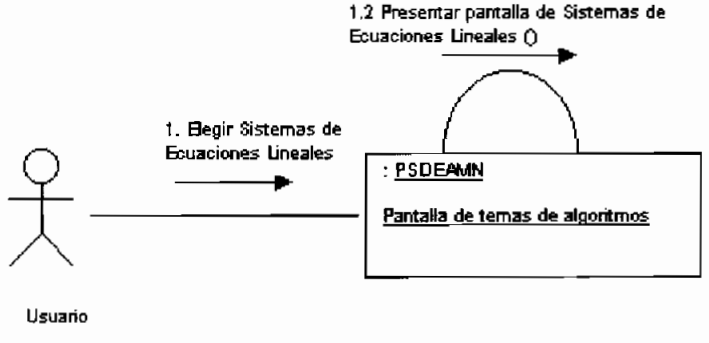


Fig. 3.45 Operación Elegir sistemas de ecuaciones lineales

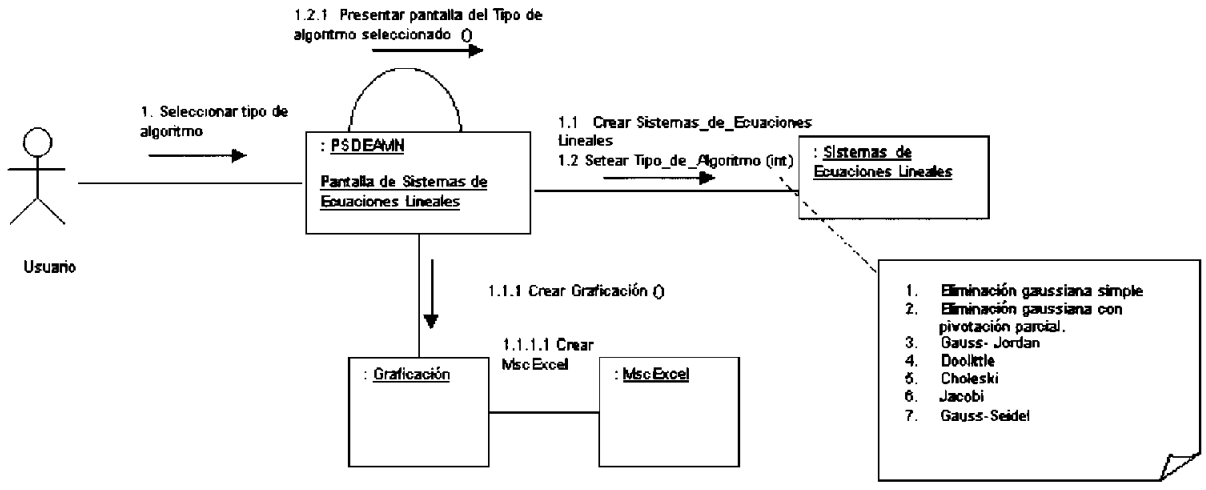


Fig. 3.46 Operación seleccionar tipo de algoritmo

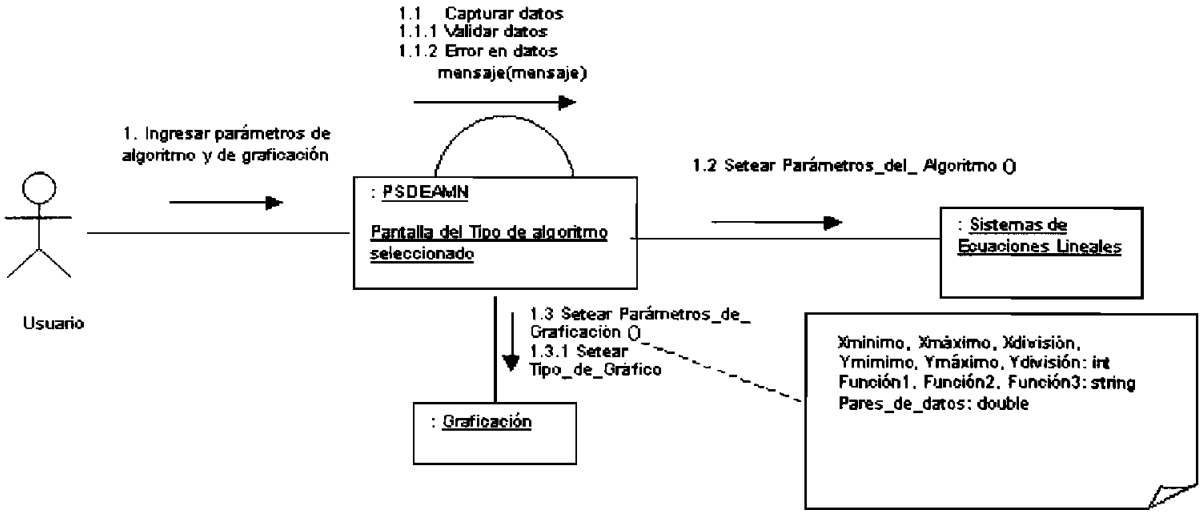


Fig. 3.47 Operación seleccionar tipo de algoritmo

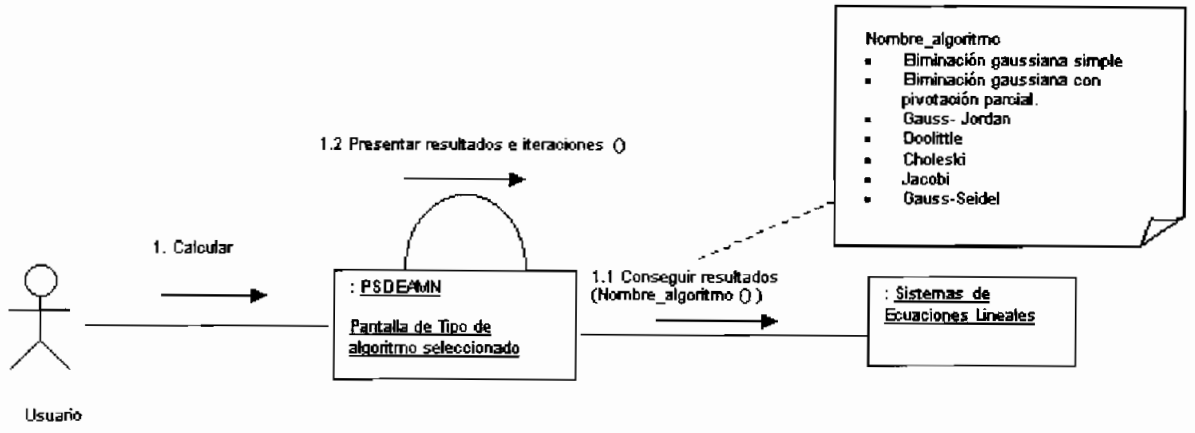


Fig. 3.48 Operación calcular

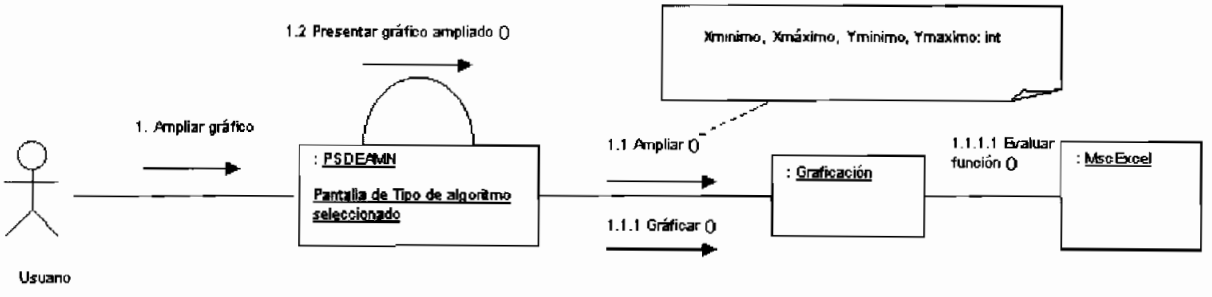


Fig. 3.49 Operación ampliar gráfico

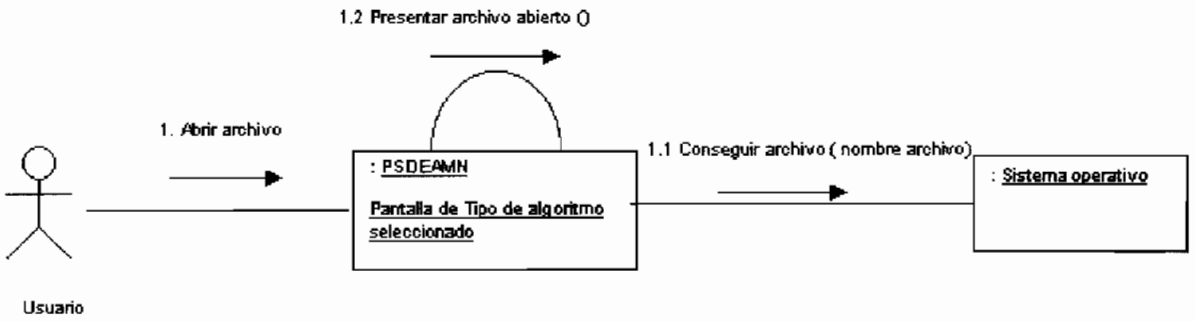


Fig. 3.50 Operación abrir archivo

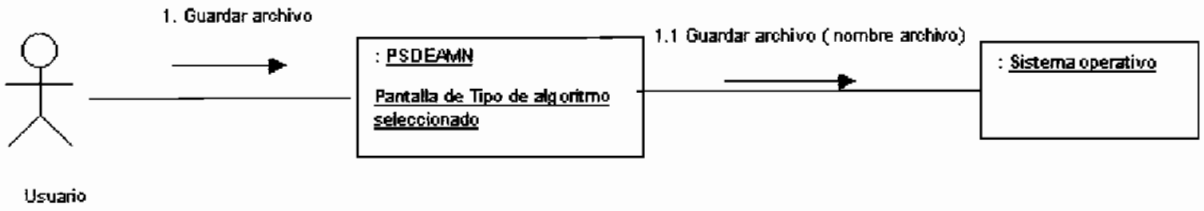


Fig. 3.51 Operación guardar archivo

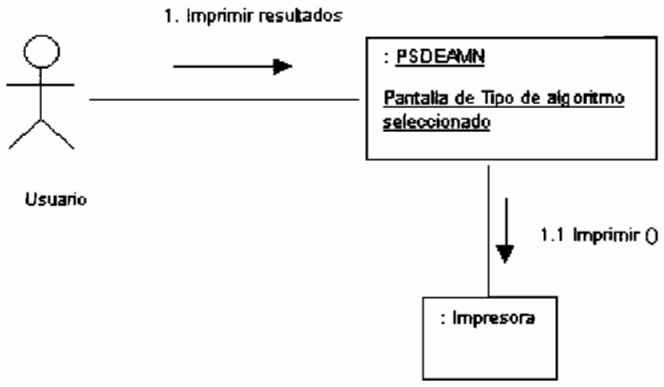


Fig. 3.52 Operación imprimir resultados

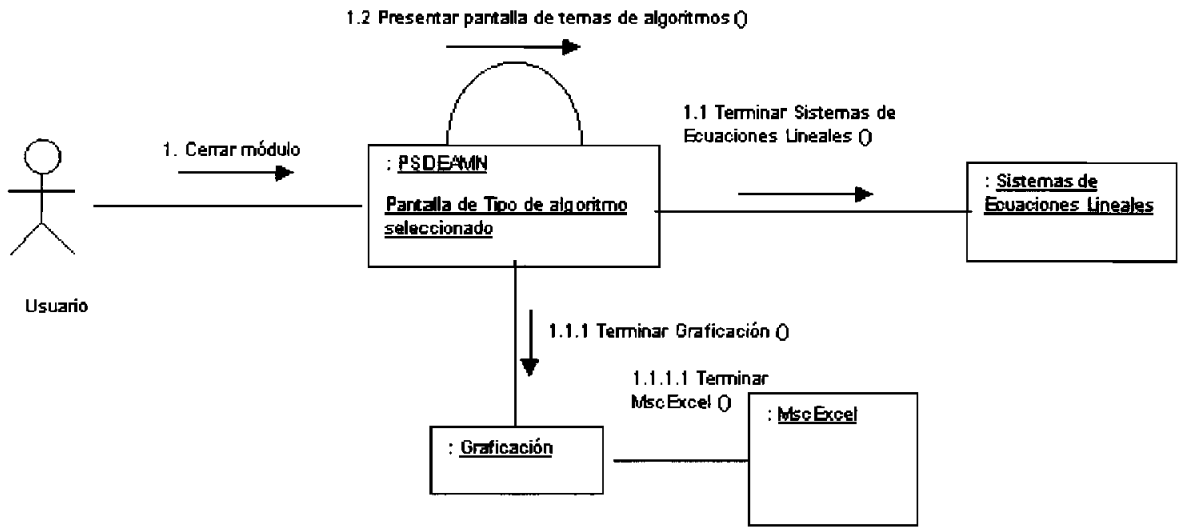


Fig. 3.53 Operación cerrar módulo

Diagramas de colaboración para el caso de uso Graficar una función

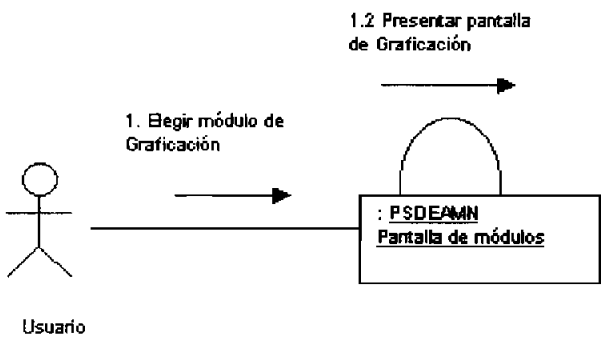


Fig. 3.54 Operación elegir módulo de graficación

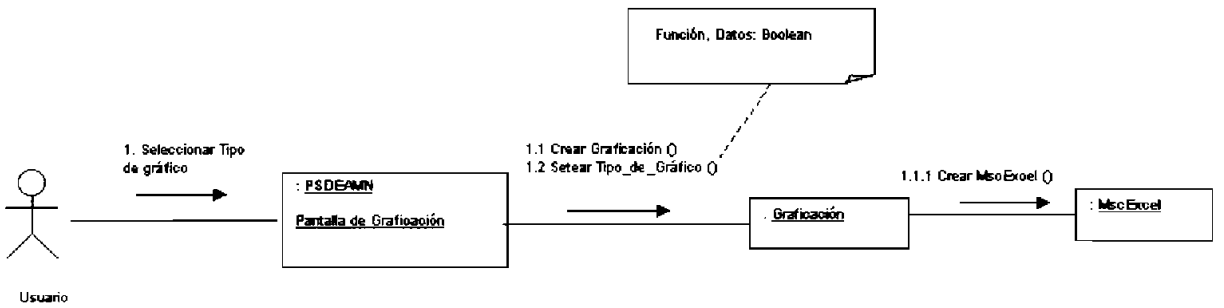


Fig. 3.55 Operación seleccionar tipo de algoritmo

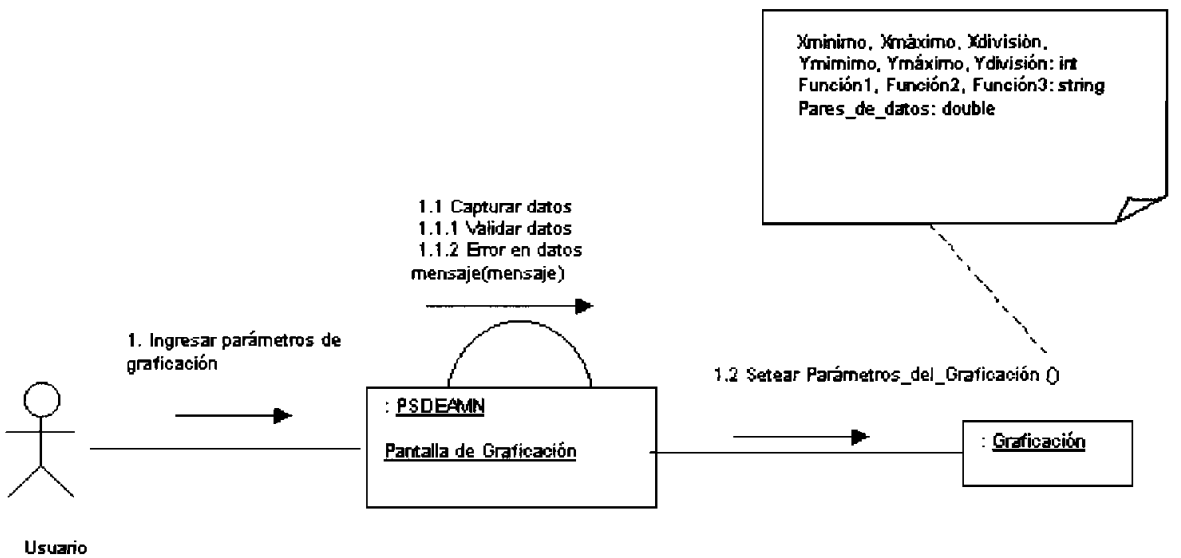


Fig. 3.56 Operación ingresar parámetros de graficación

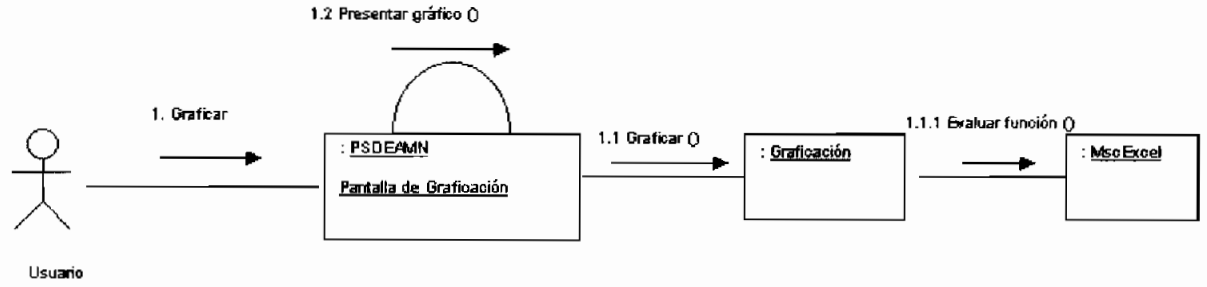


Fig. 3.57 Operación graficar

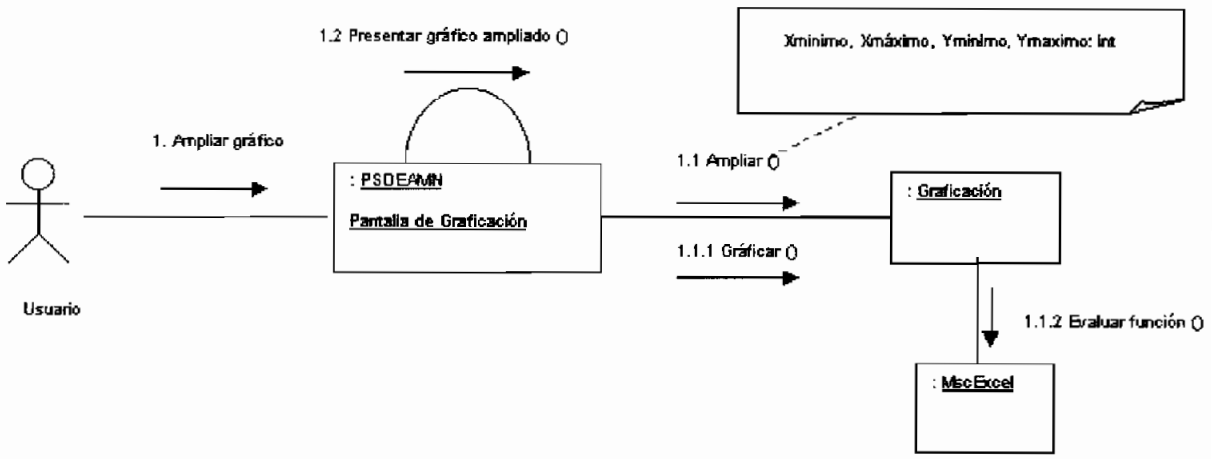


Fig. 3.58 Operación ampliar gráfico

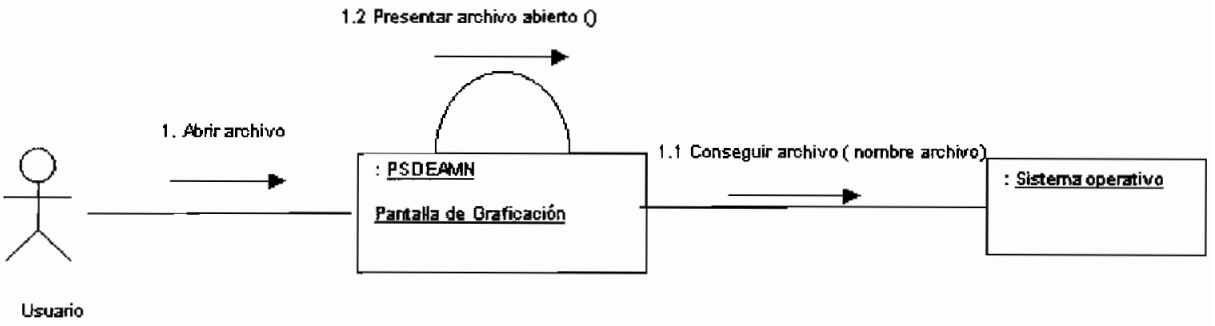


Fig. 3.59 Operación abrir archivo

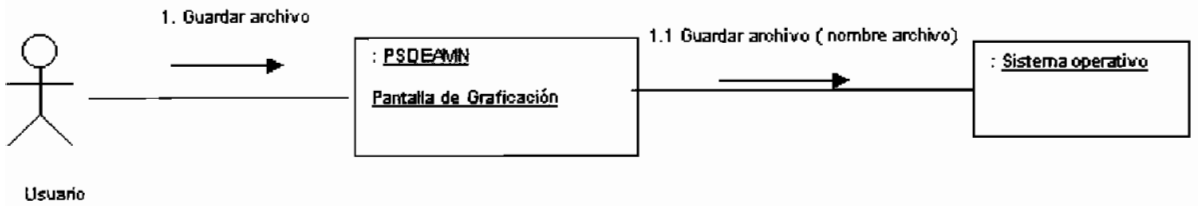


Fig. 3.60 Operación abrir archivo

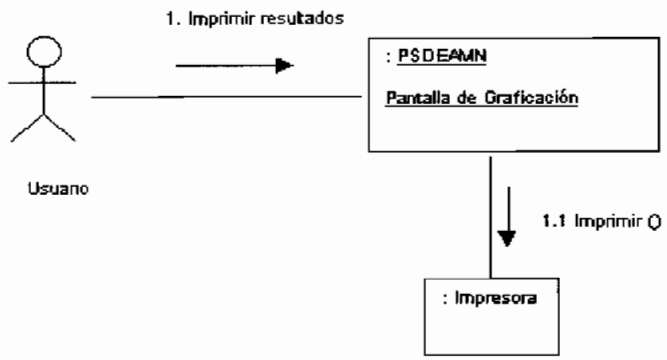


Fig. 3.61 Operación abrir archivo

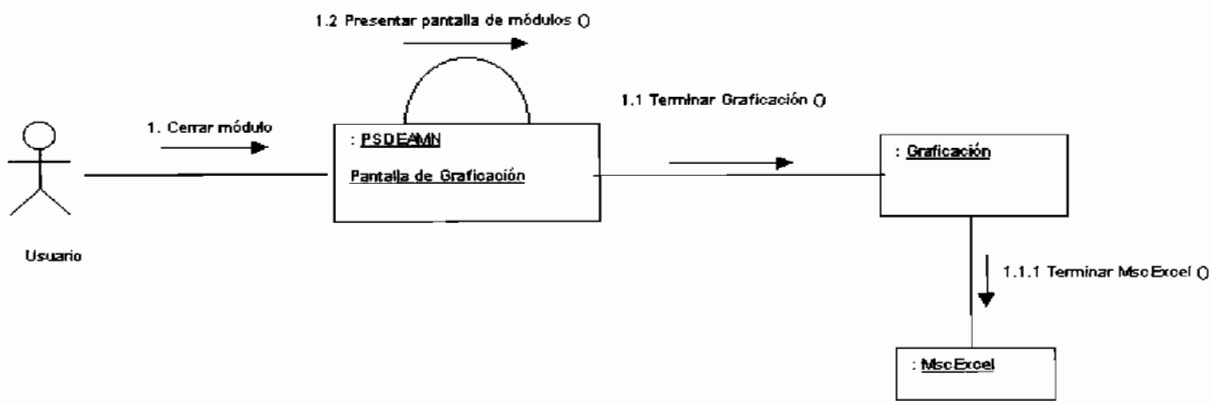


Fig. 3.62 Operación cerrar módulo

3.4.3 DIAGRAMAS DE COLABORACIÓN PARA EL CLUSTER (3)

Realizar Evaluación

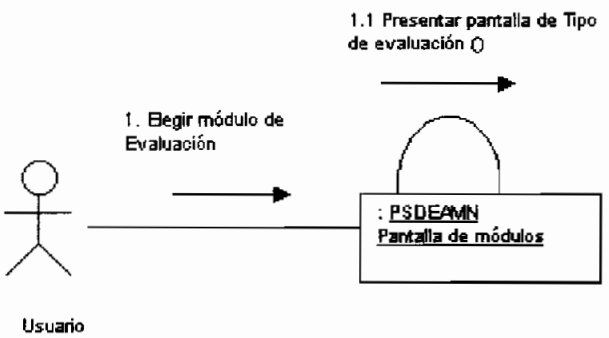


Fig. 3.63 Operación elegir módulo de evaluación

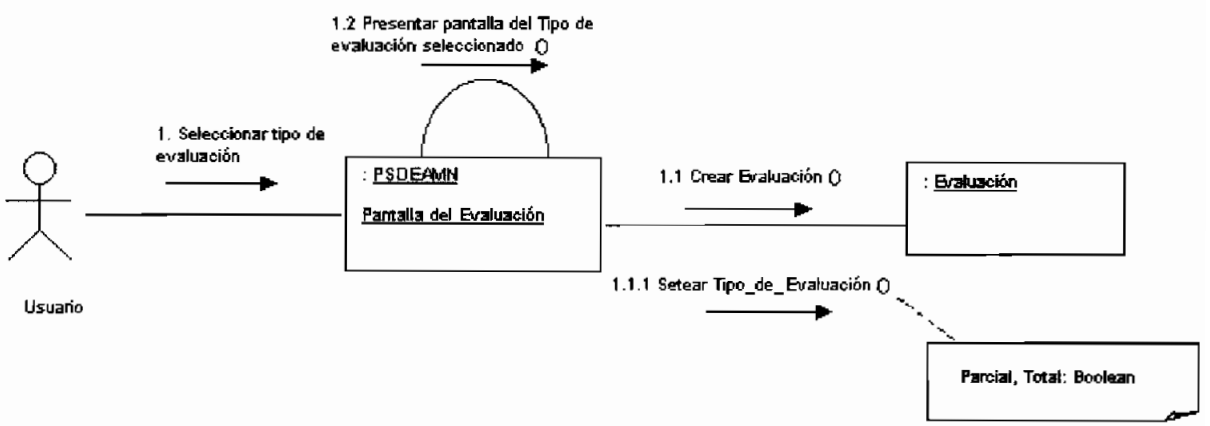


Fig. 3.64 Operación seleccionar tipo de evaluación

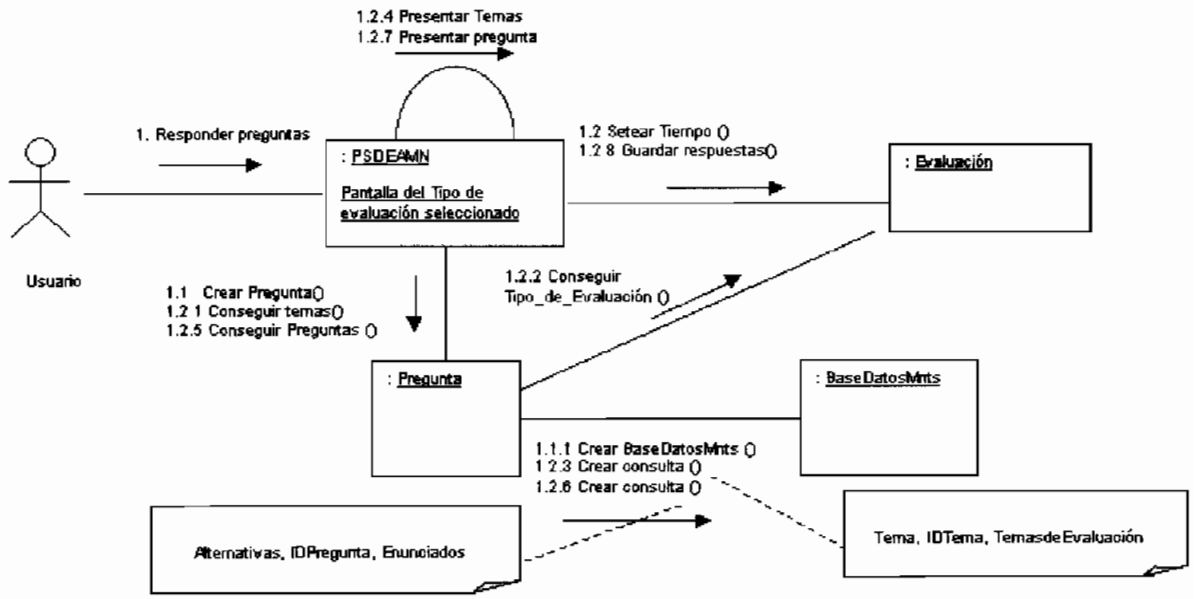


Fig. 3.65 Operación responder preguntas

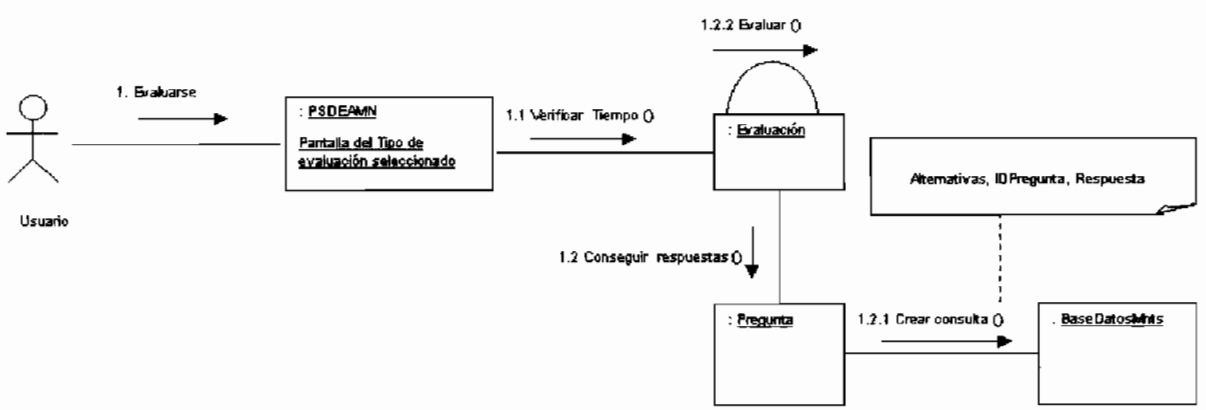


Fig. 3.66 Operación evaluarse

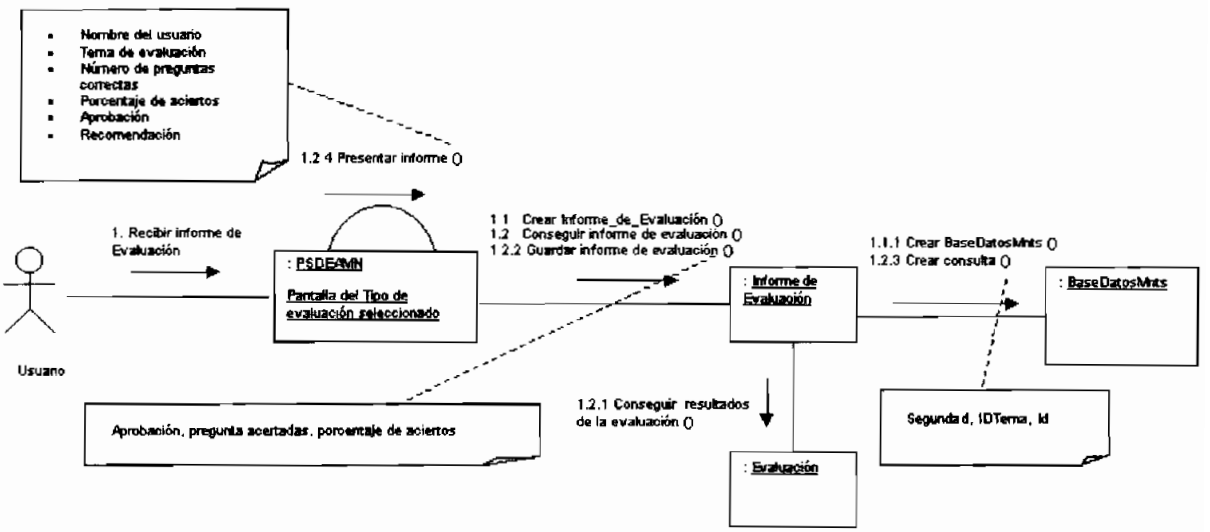


Fig. 3.67 Operación recibir informe de evaluación

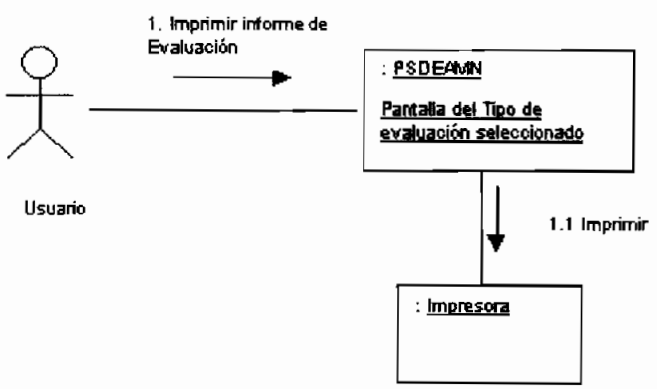


Fig. 3.68 Operación imprimir informe de evaluación

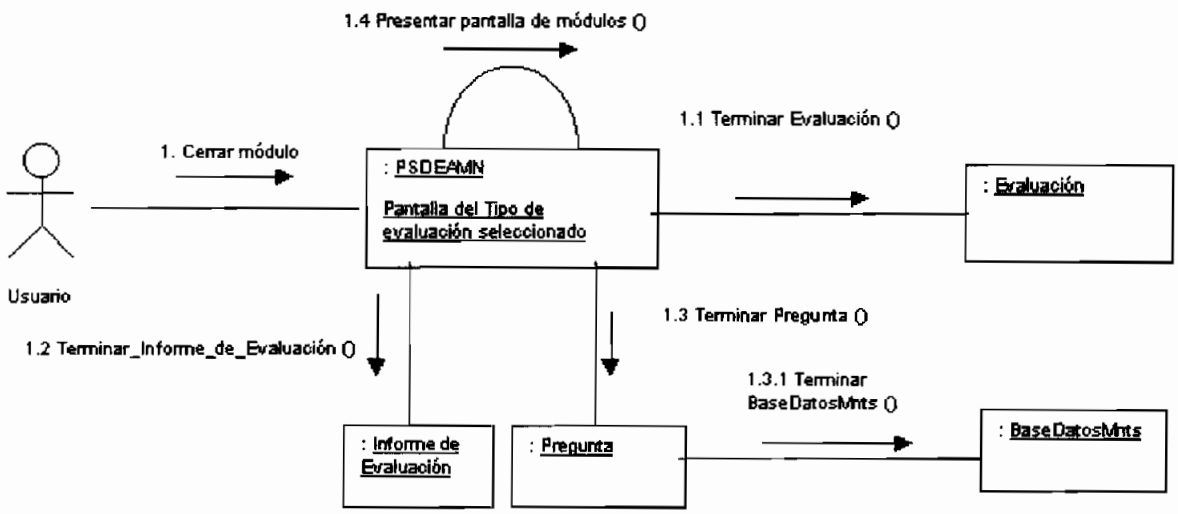


Fig. 3.69 Operación cerrar módulo

Resolver sistemas de ecuaciones no lineales

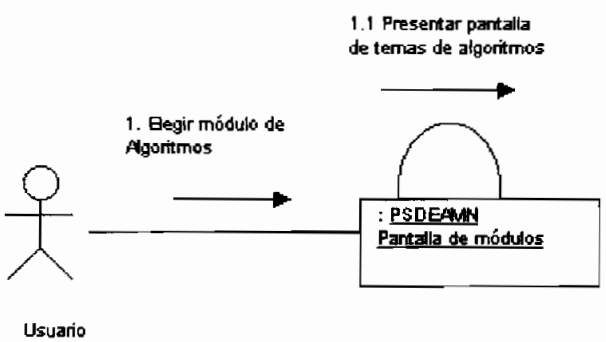


Fig. 3.70 Operación elegir módulo de algoritmos

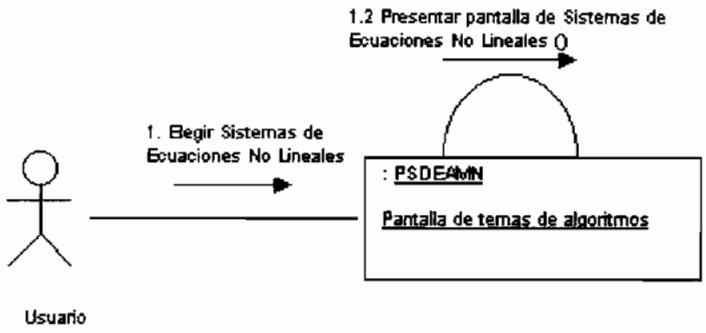


Fig. 3.71 Operación elegir sistemas de ecuaciones no lineales

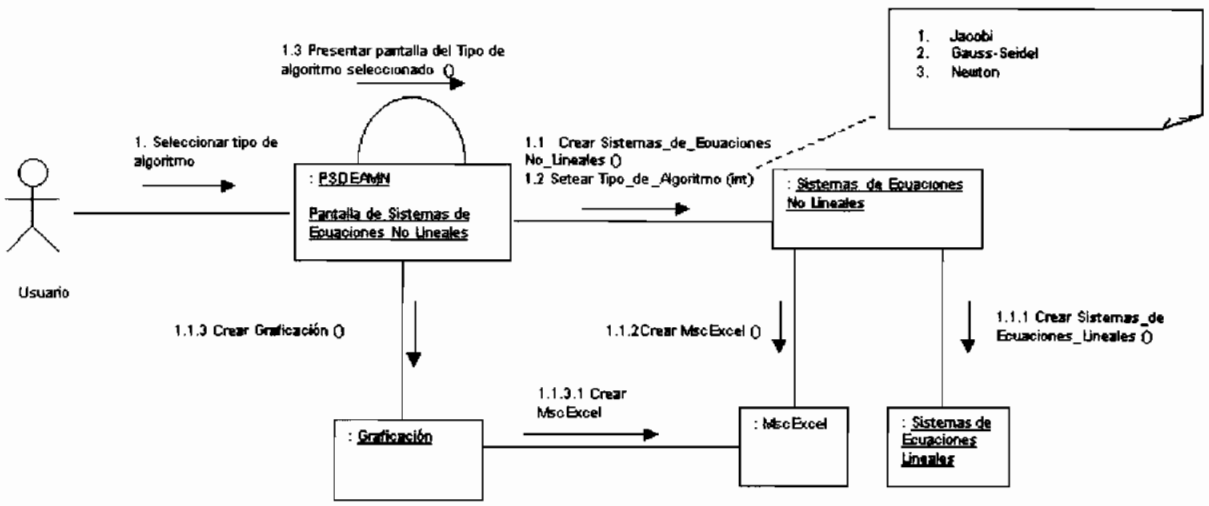


Fig. 3.72 Operación seleccionar tipo de algoritmo

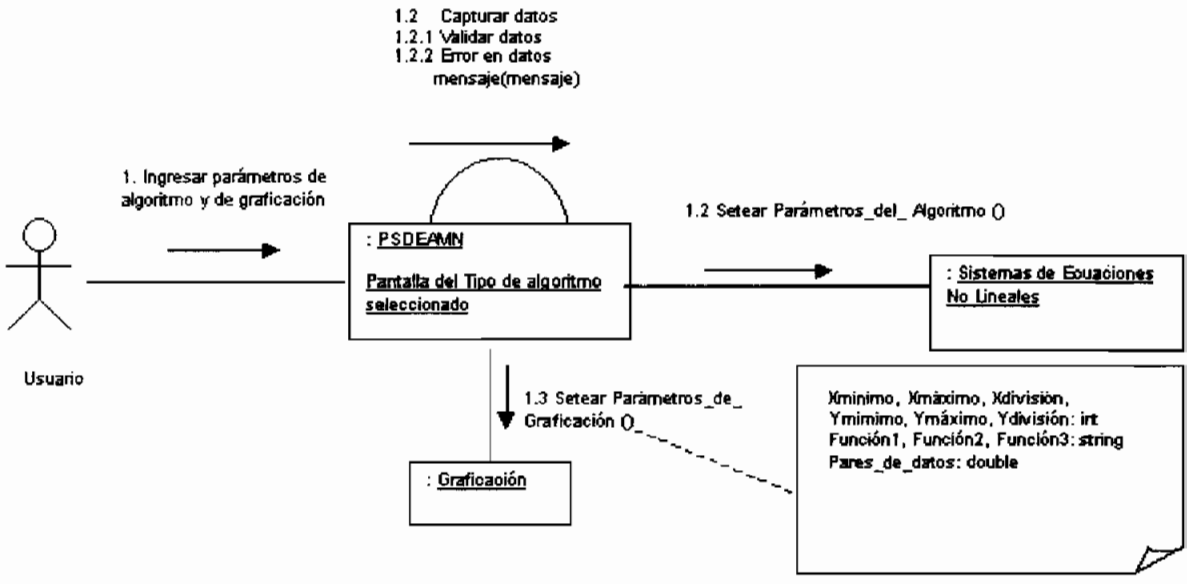


Fig. 3.73 Operación elegir ingresar parámetros del algoritmo y de graficación

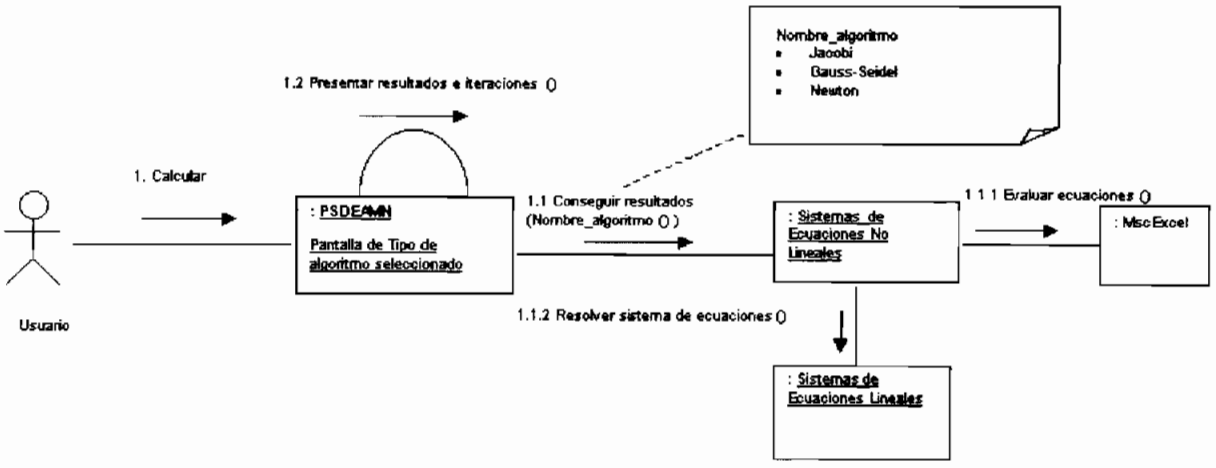


Fig. 3.74 Operación elegir calcular

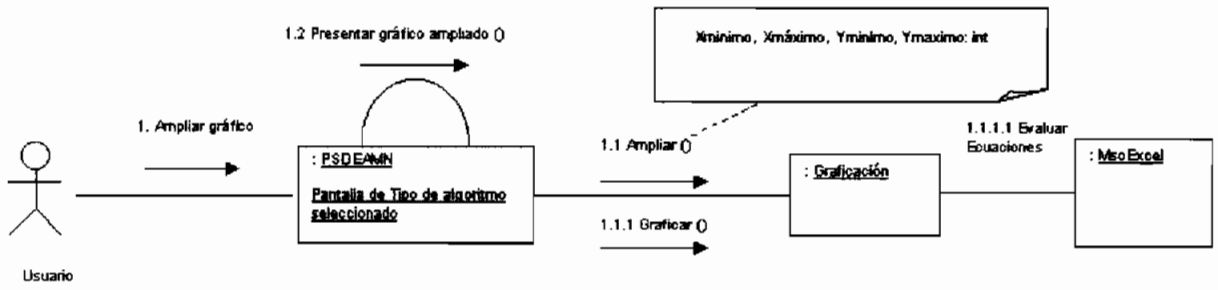


Fig. 3.75 Operación ampliar gráfico

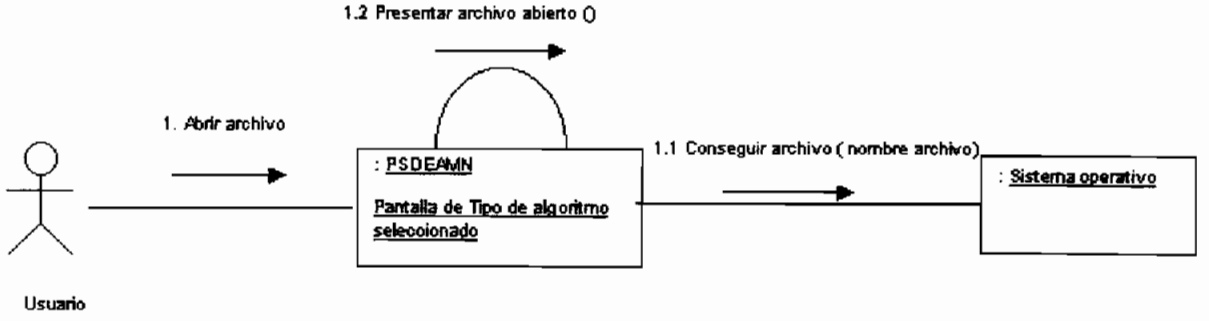


Fig. 3.76 Operación abrir archivo

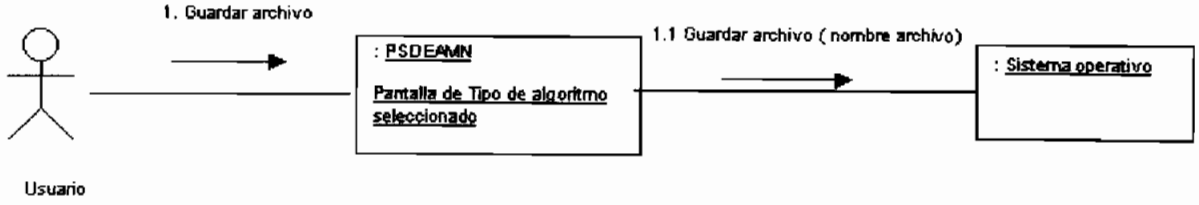


Fig. 3.77 Operación guardar archivo

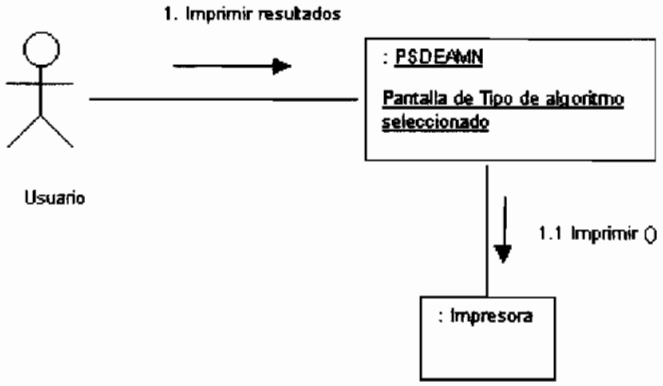


Fig. 3.78 Operación imprimir resultados

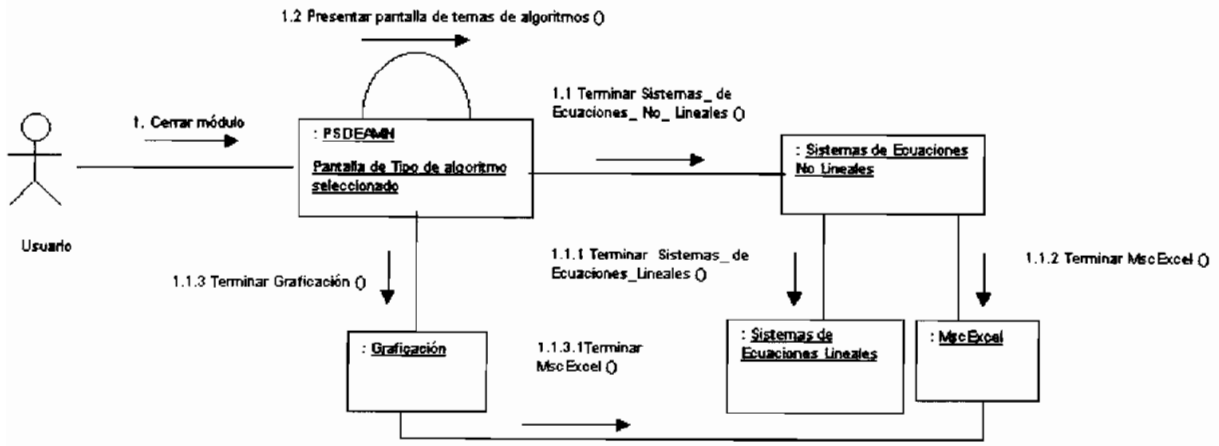


Fig. 3.79 Operación cerrar módulo

3.4.4 DIAGRAMAS DE COLABORACIÓN PARA EL CLUSTER (4)

Encontrar raíces de un polinomio

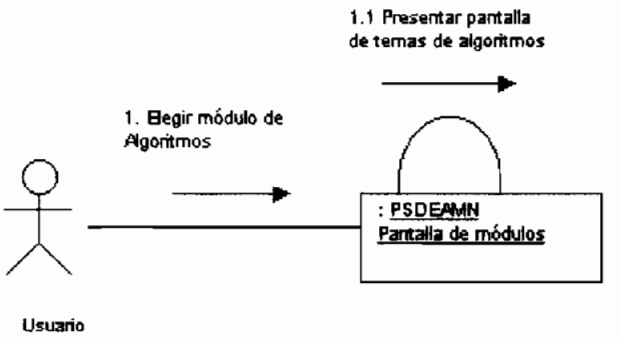


Fig. 3.80 Operación elegir módulo de algoritmos

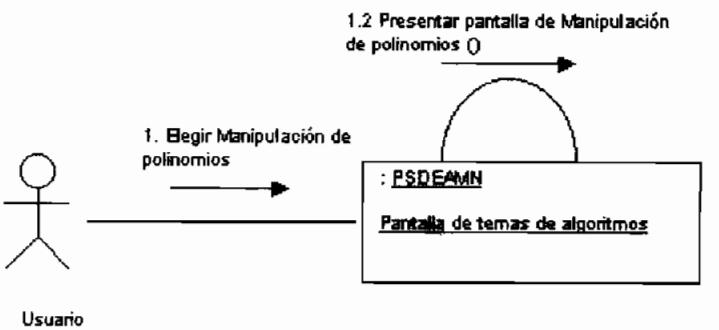


Fig. 3.81 Operación elegir manipulación de polinomios

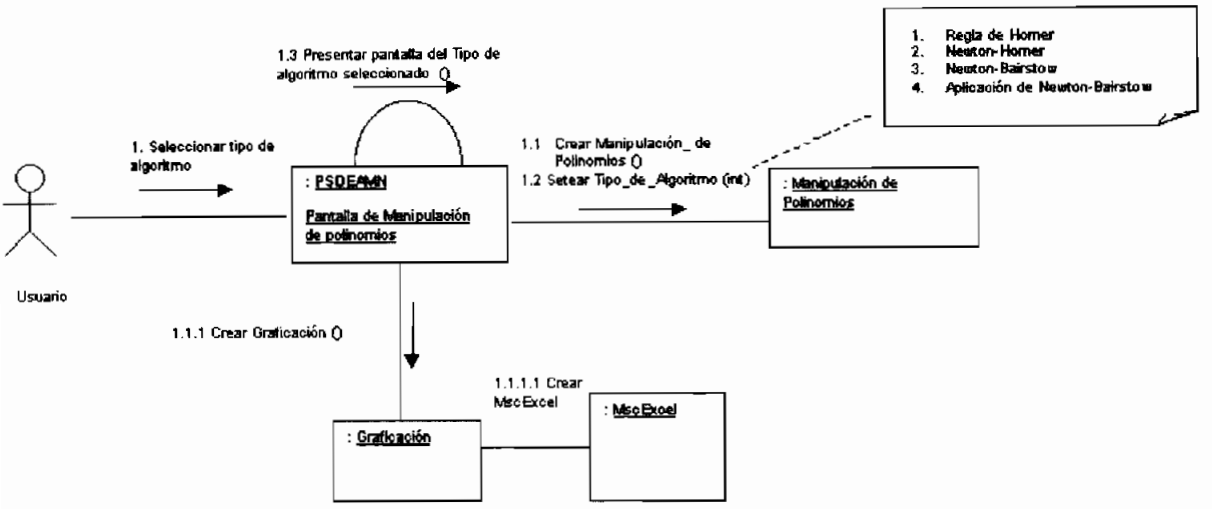


Fig. 3.82 Operación elegir seleccionar tipo de algoritmo

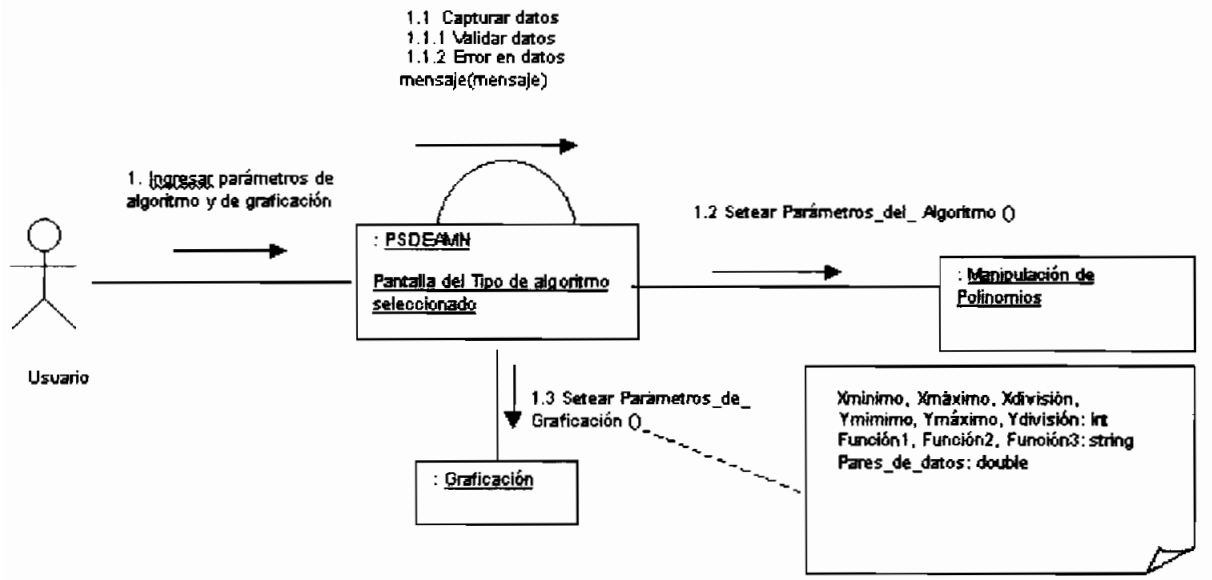


Fig. 3.83 Operación ingresar parámetros del algoritmo y de graficación

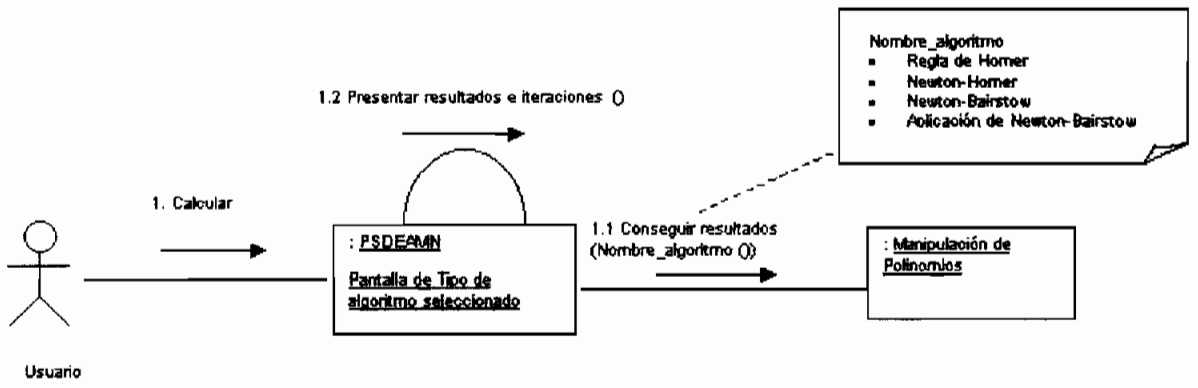


Fig. 3.84 Operación calcular

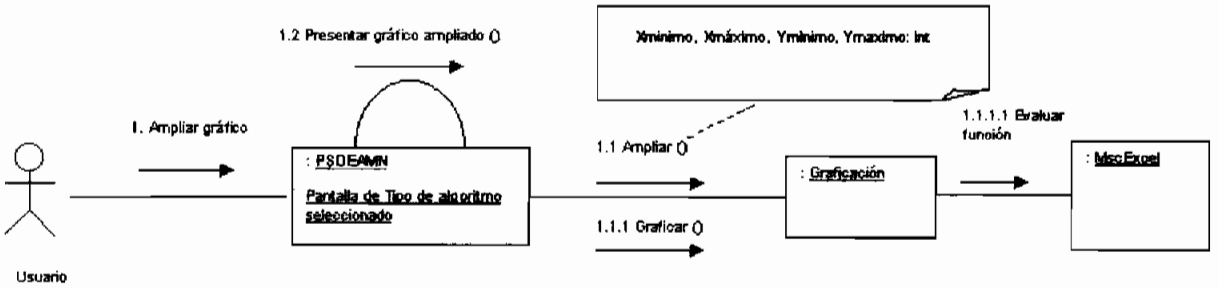


Fig. 3.85 Operación elegir ampliar gráfico

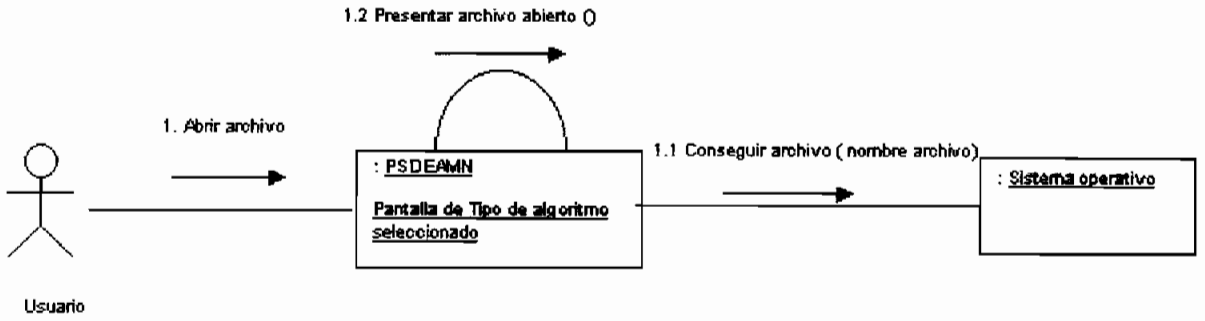


Fig. 3.86 Operación elegir módulo de algoritmos

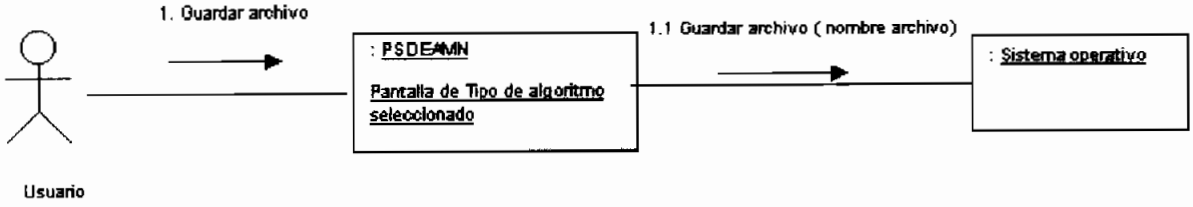


Fig. 3.87 Operación guardar archivo

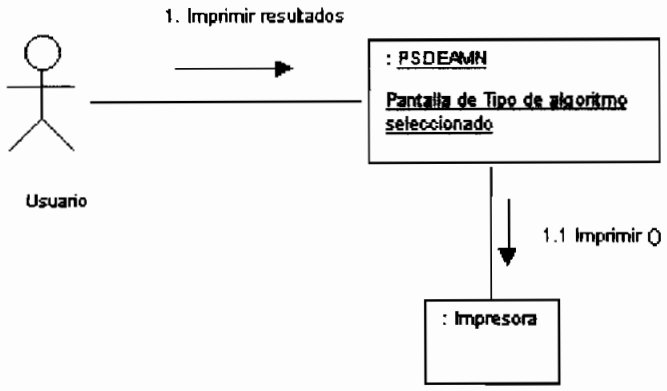


Fig. 3.88 Operación imprimir resultados

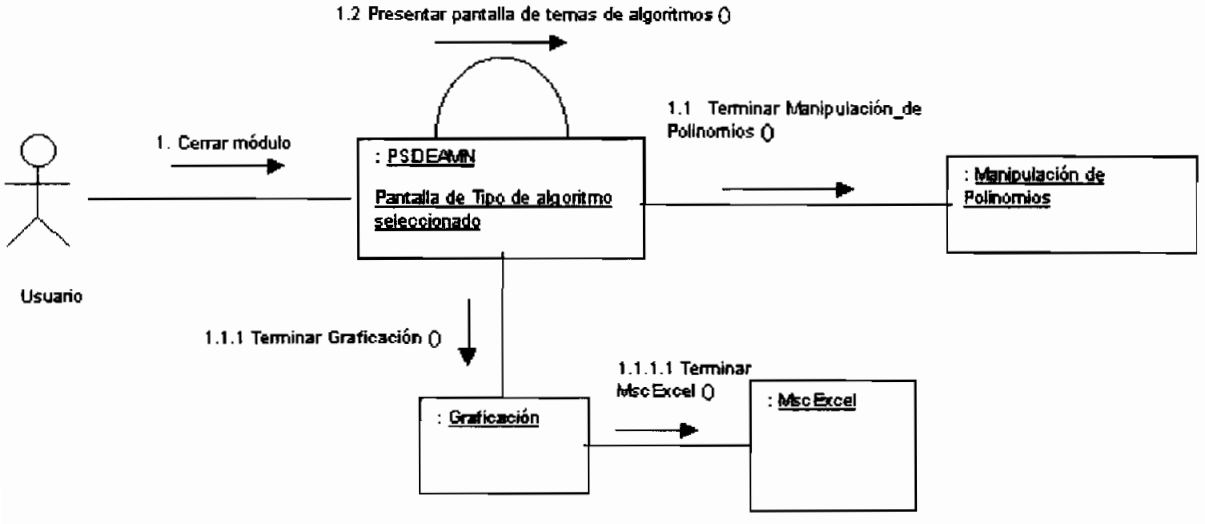


Fig. 3.89 Operación cerrar módulo

Calcular la derivada de una función

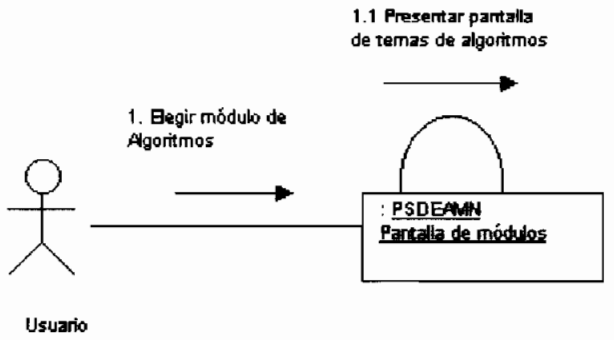


Fig. 3.90 Operación elegir módulo de algoritmos

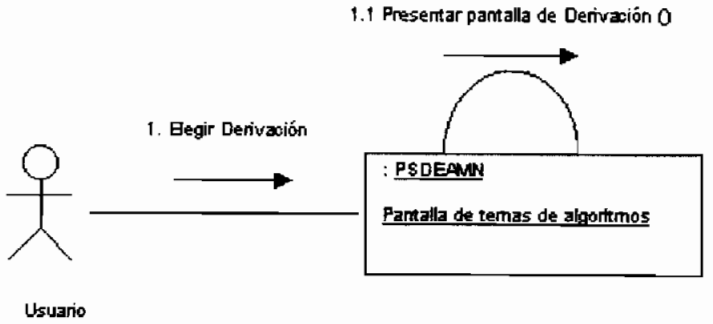


Fig. 3.91 Operación Elegir derivación

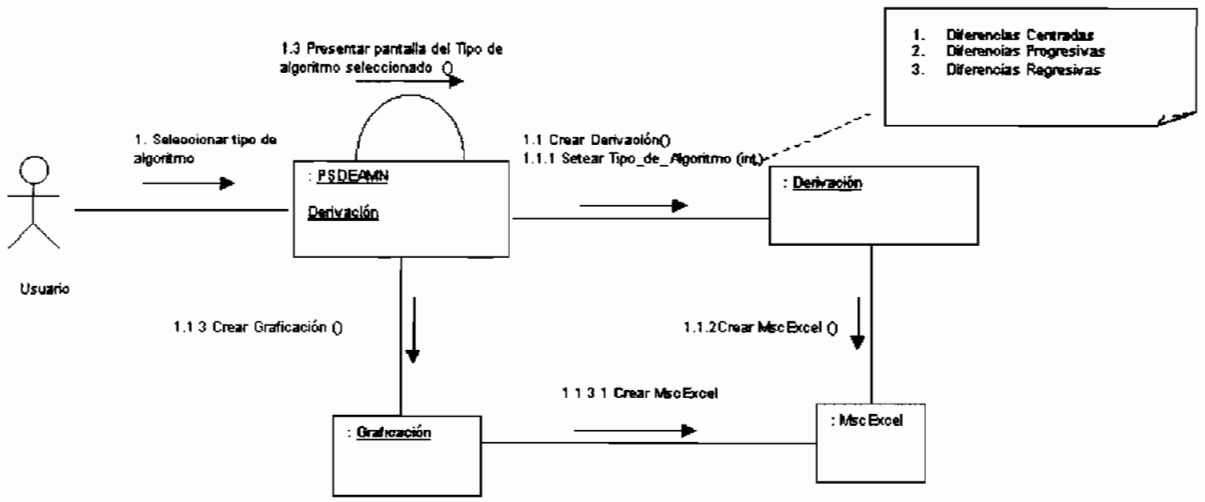


Fig. 3.92 Operación seleccionar tipo de algoritmo

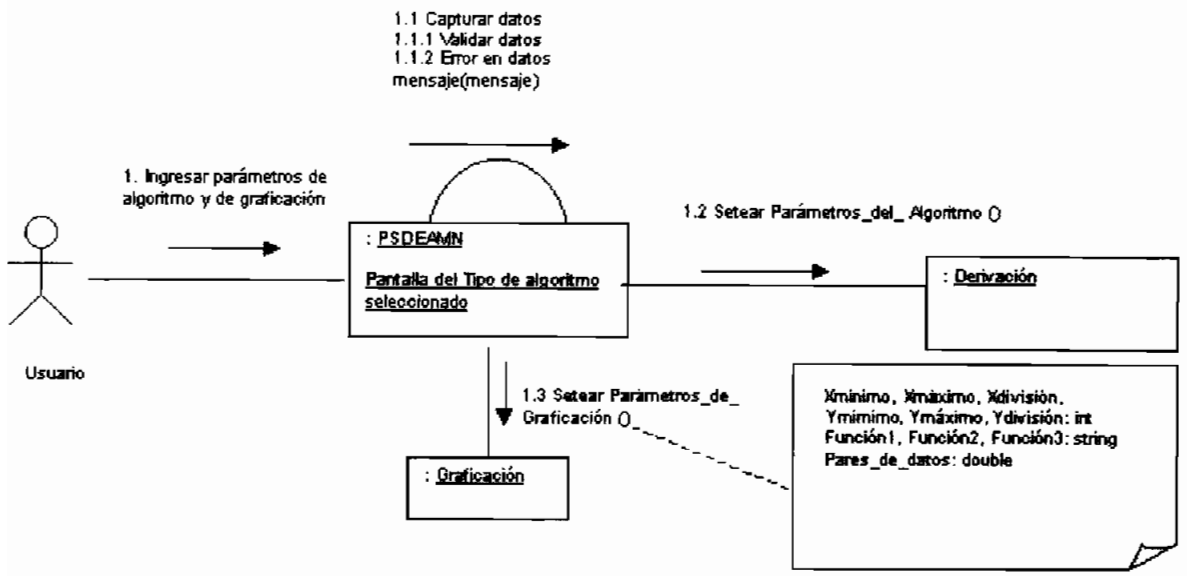


Fig. 3.93 Operación ingresar parámetros del algoritmo y de graficación

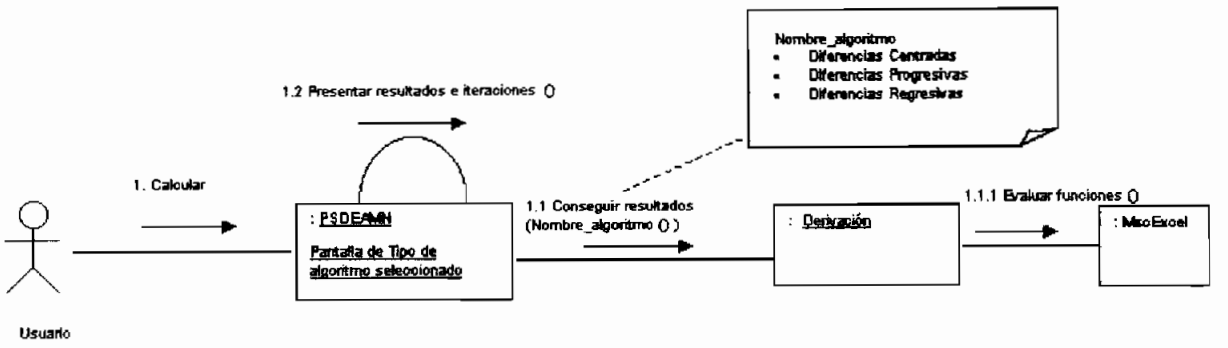


Fig. 3.94 Operación calcular

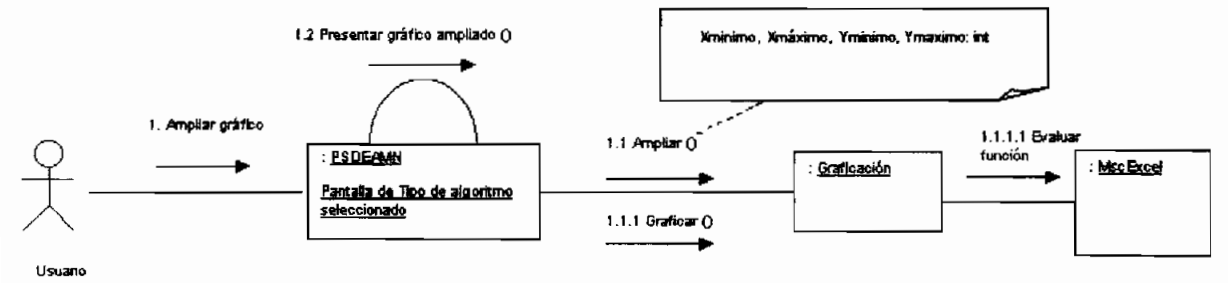


Fig. 3.95 Operación ampliar el gráfico

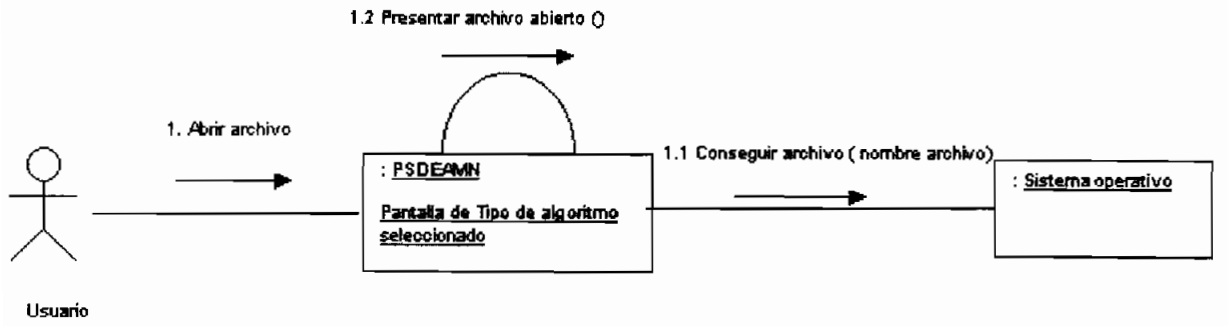


Fig. 3.96 Operación abrir archivo

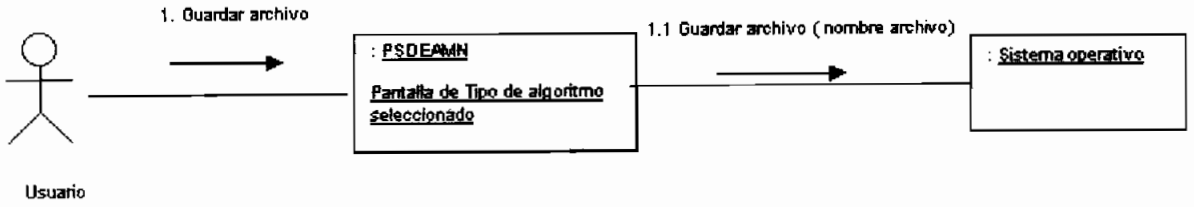


Fig. 3.97 Operación guardar archivo

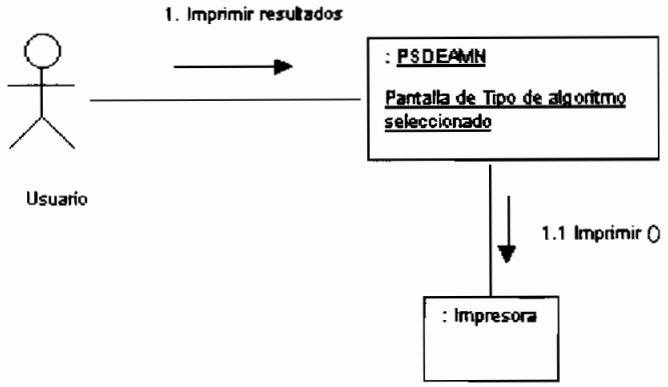


Fig. 3.98 Operación imprimir resultados

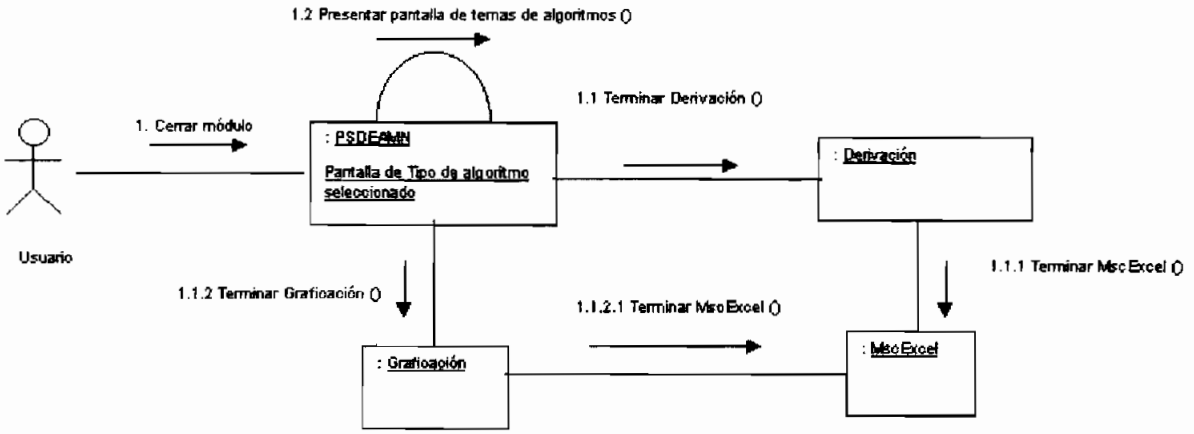


Fig. 3.99 Operación cerrar módulo

3.4.5 DIAGRAMAS DE COLABORACIÓN PARA EL CLUSTER (5)

Resolver un sistema de ecuaciones diferenciales ordinarias

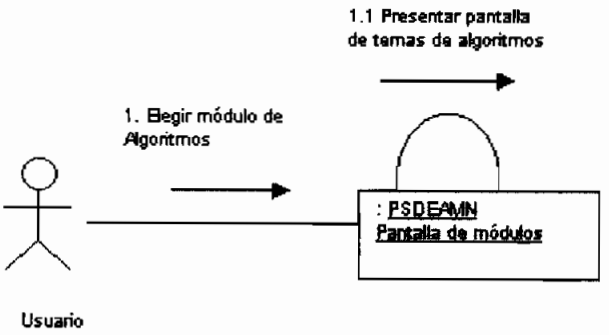


Fig. 3.100 Operación elegir módulo de algoritmos

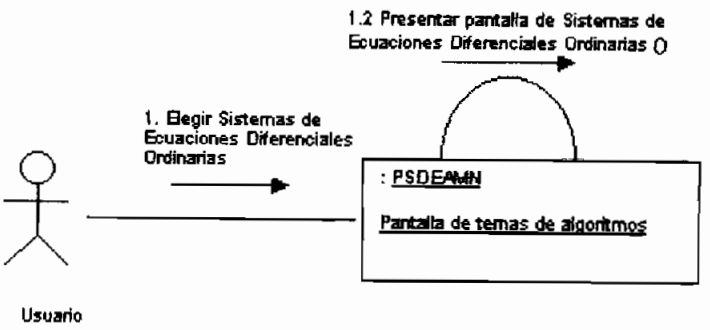


Fig. 3.101 Operación elegir sistemas de ecuaciones diferenciales ordinarias

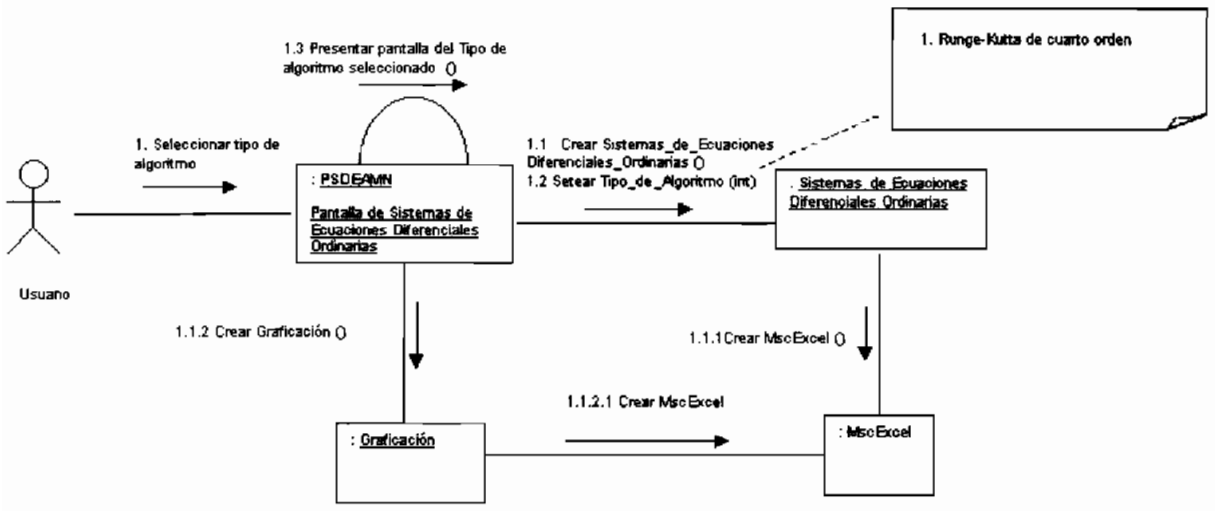


Fig. 3.102 Operación seleccionar tipo de algoritmo

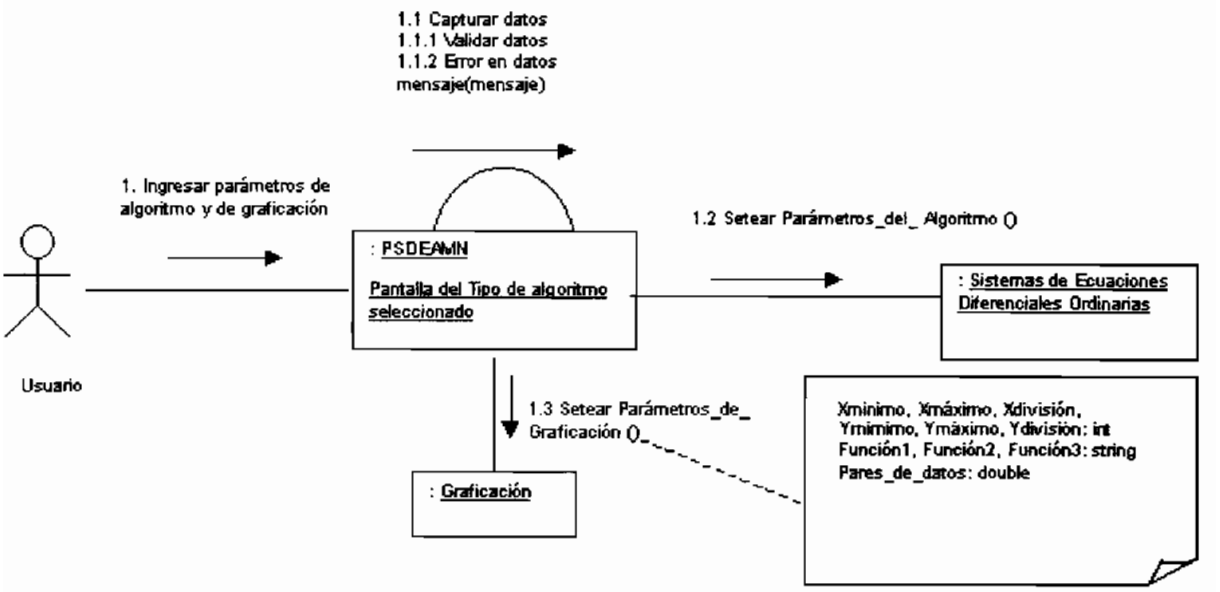


Fig. 3.103 Operación ingresar parámetros del algoritmo y de graficación

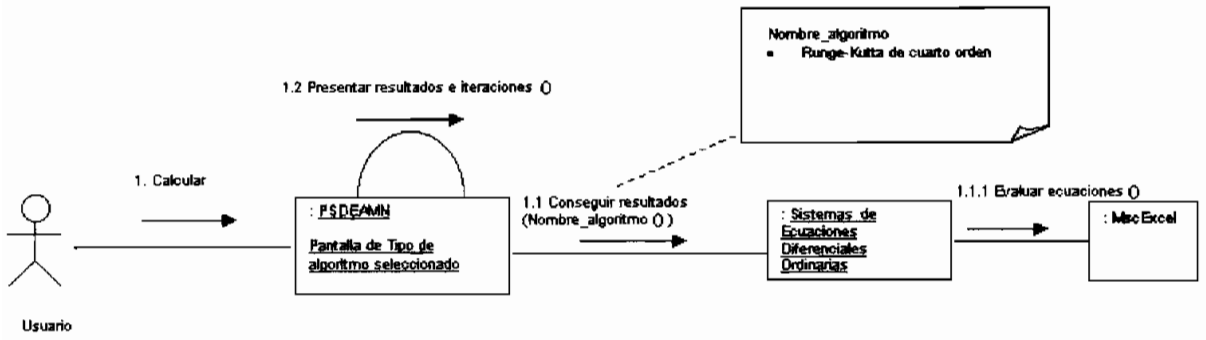


Fig. 3.104 Operación calcular

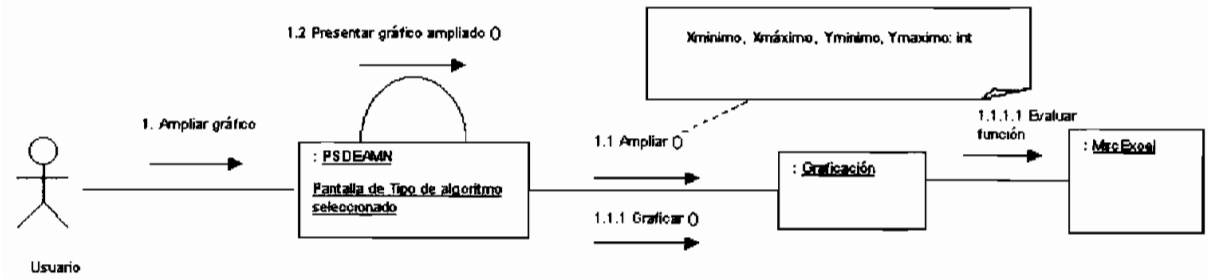


Fig. 3.105 Operación ampliar gráfico

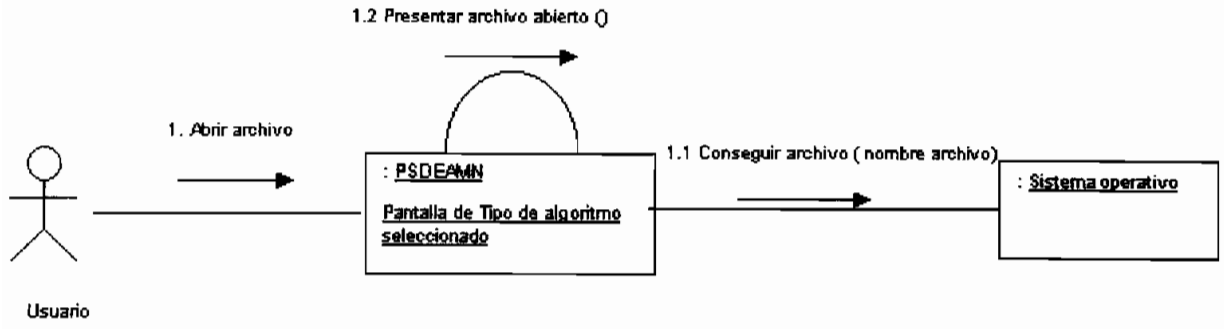


Fig. 3.106 Operación abrir archivo

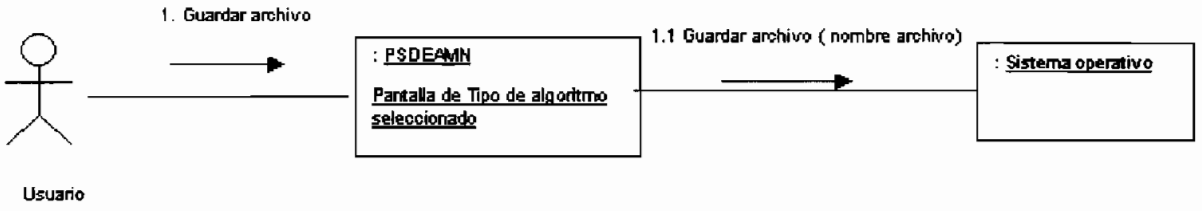


Fig. 3.107 Operación guardar archivo

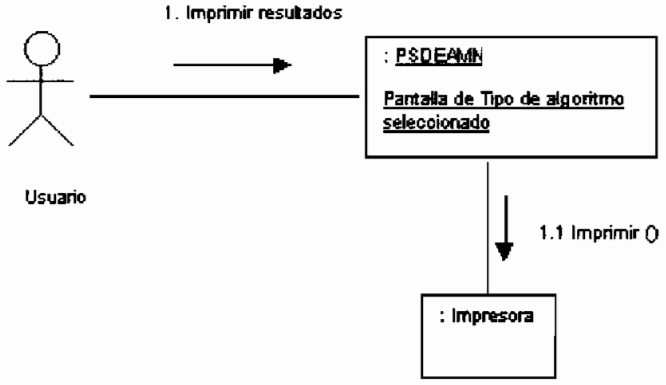


Fig. 3.108 Operación imprimir resultados

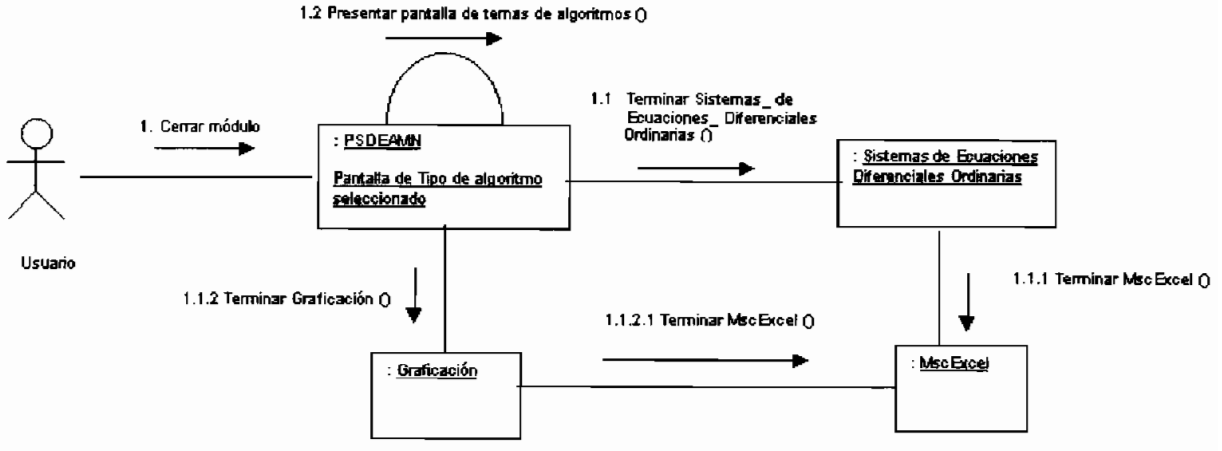


Fig. 3.109 Operación cerrar módulo

Realizar una interpolación polinomial

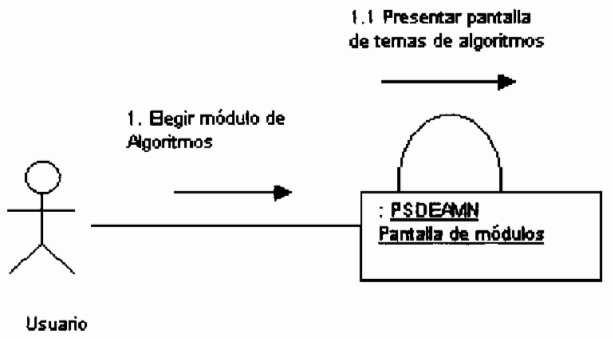


Fig. 3.110 Operación elegir módulo de algoritmos

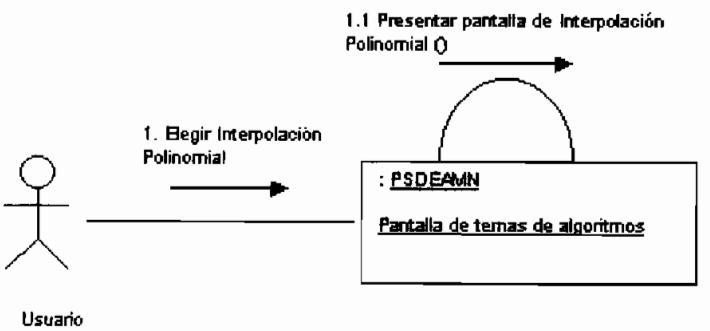


Fig. 3.111 Operación elegir interpolación polinomial

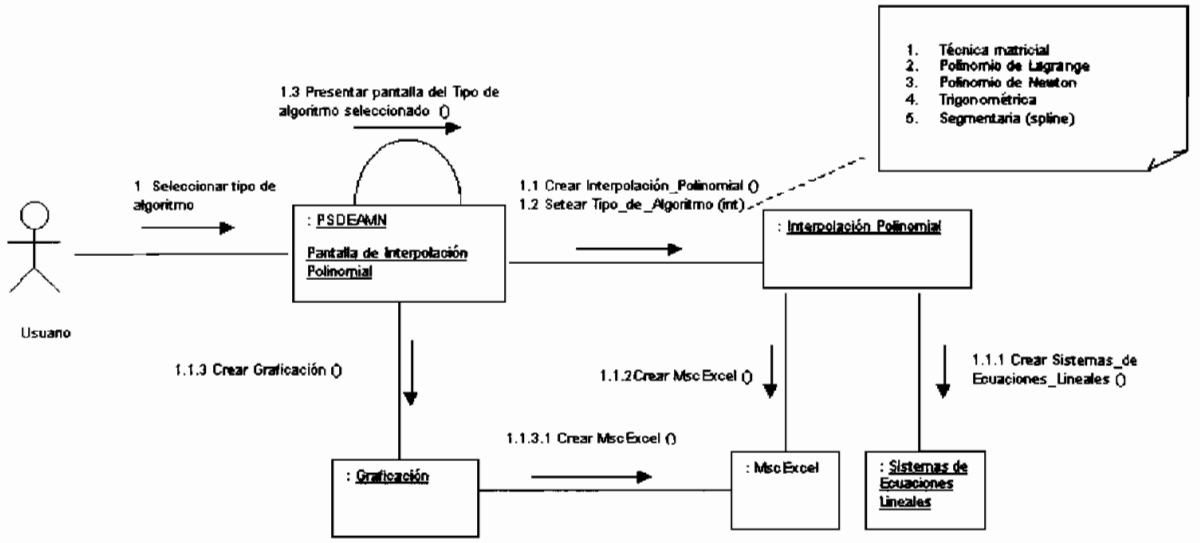


Fig. 3.112 Operación seleccionar tipo de algoritmo

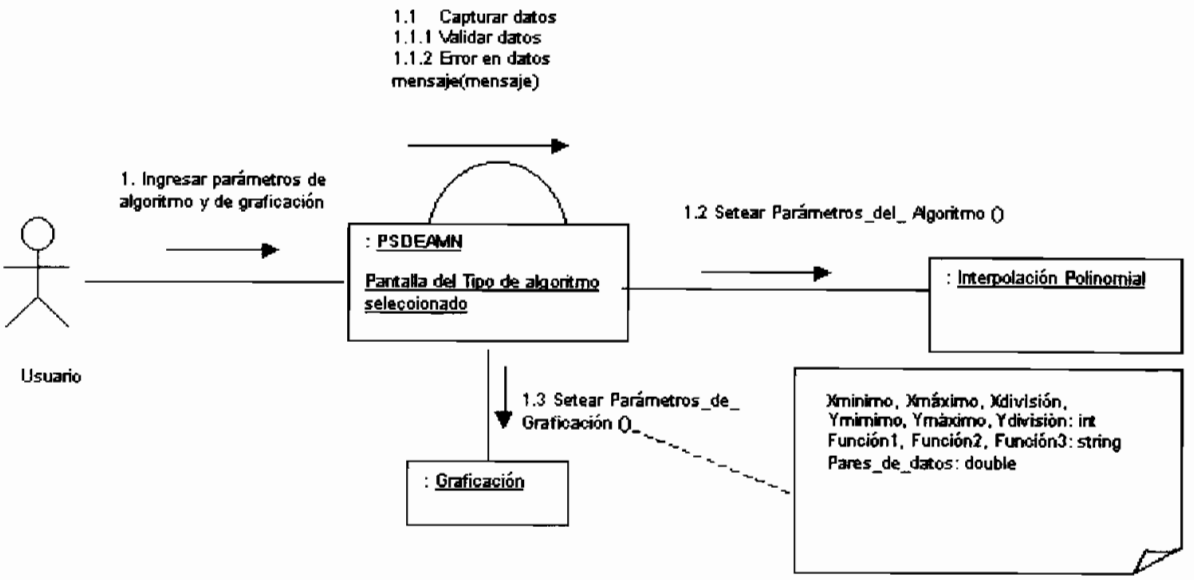


Fig. 3.113 Operación ingresar parámetros del algoritmo y de graficación

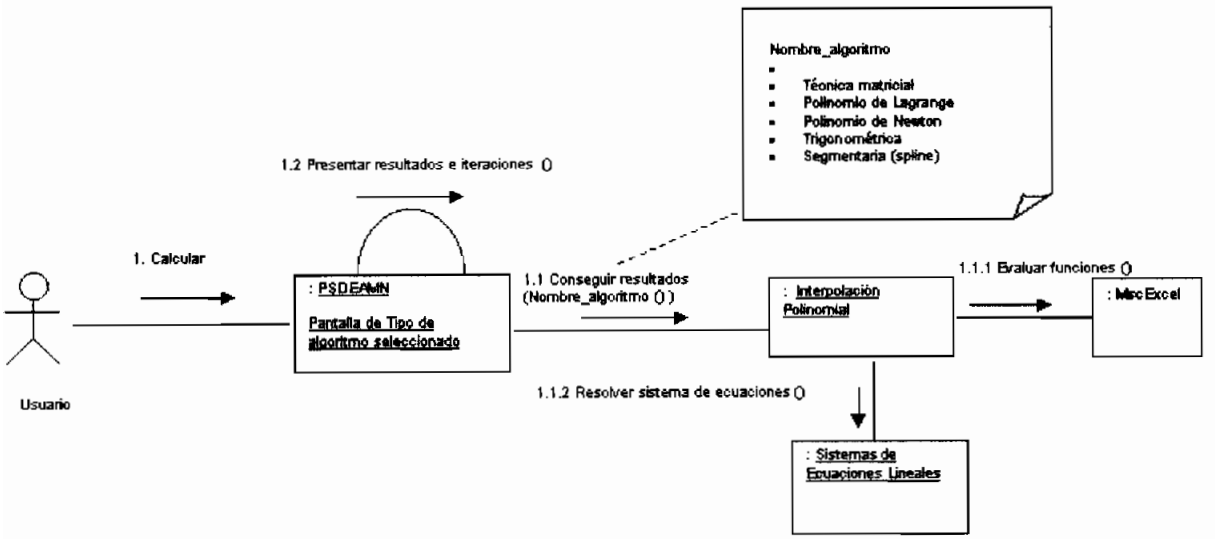


Fig. 3.114 Operación calcular

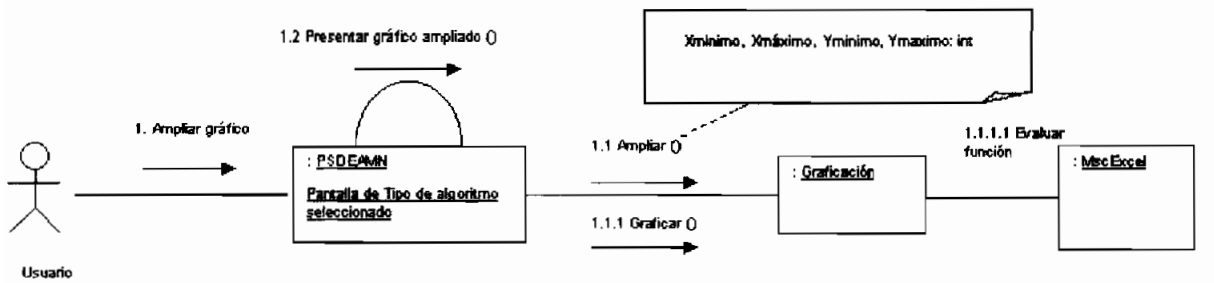


Fig. 3.115 Operación ampliar gráfico

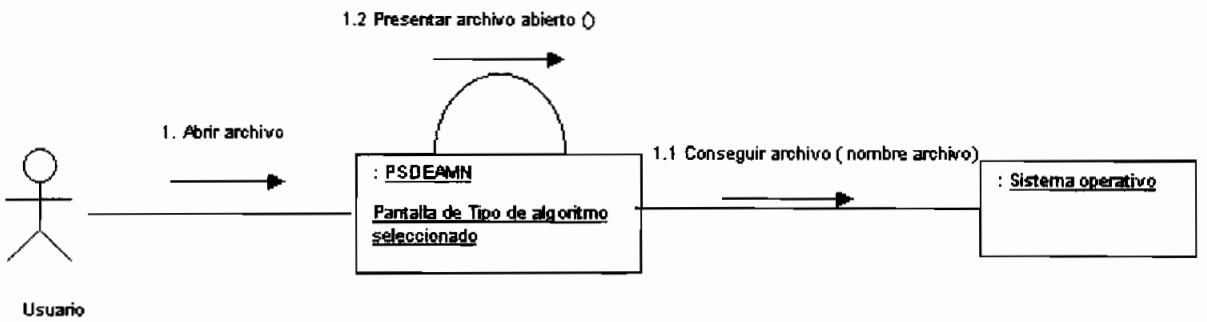


Fig. 3.116 Operación abrir archivo

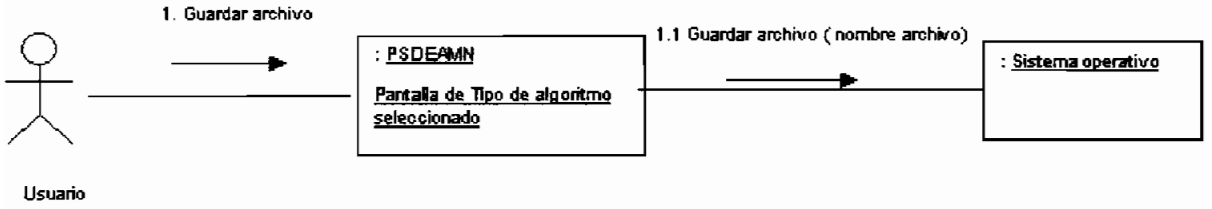


Fig. 3.117 Operación guardar archivo

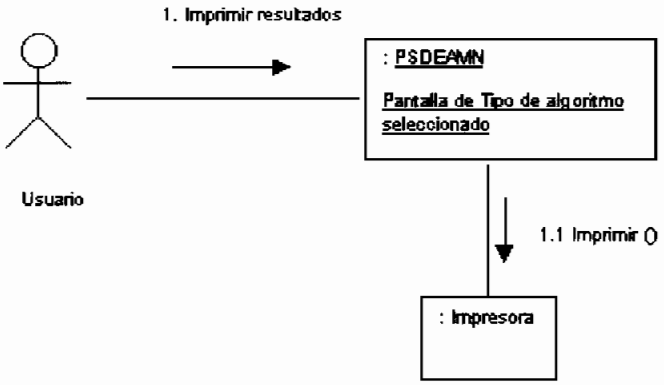


Fig. 3.118 Operación imprimir resultados

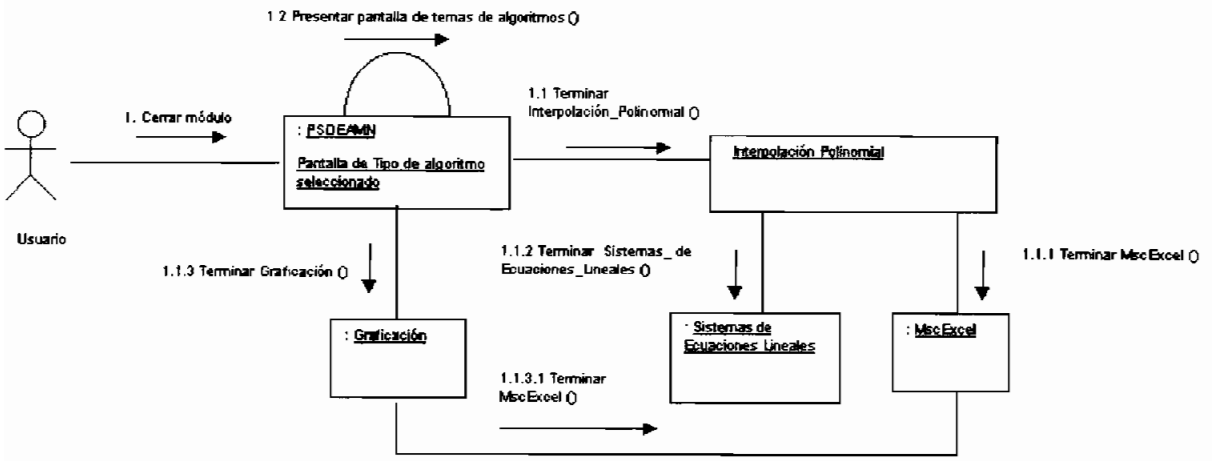


Fig. 3.119 Operación cerrar módulo

Resolver ecuaciones diferenciales ordinarias

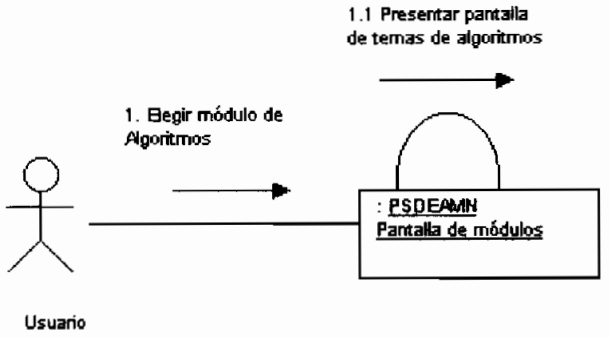


Fig. 3.120 Operación elegir módulo de algoritmos

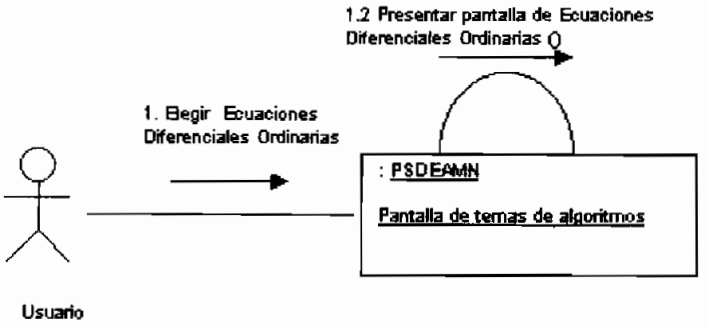


Fig. 3.121 Operación elegir ecuaciones diferenciales ordinarias

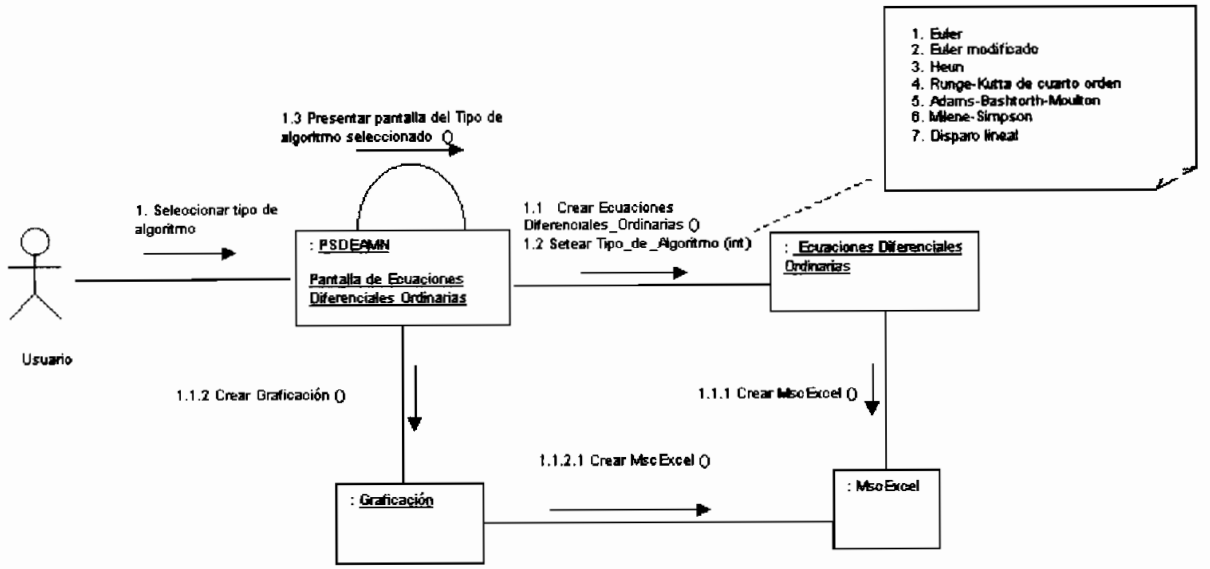


Fig. 3.122 Operación seleccionar tipo de algoritmo

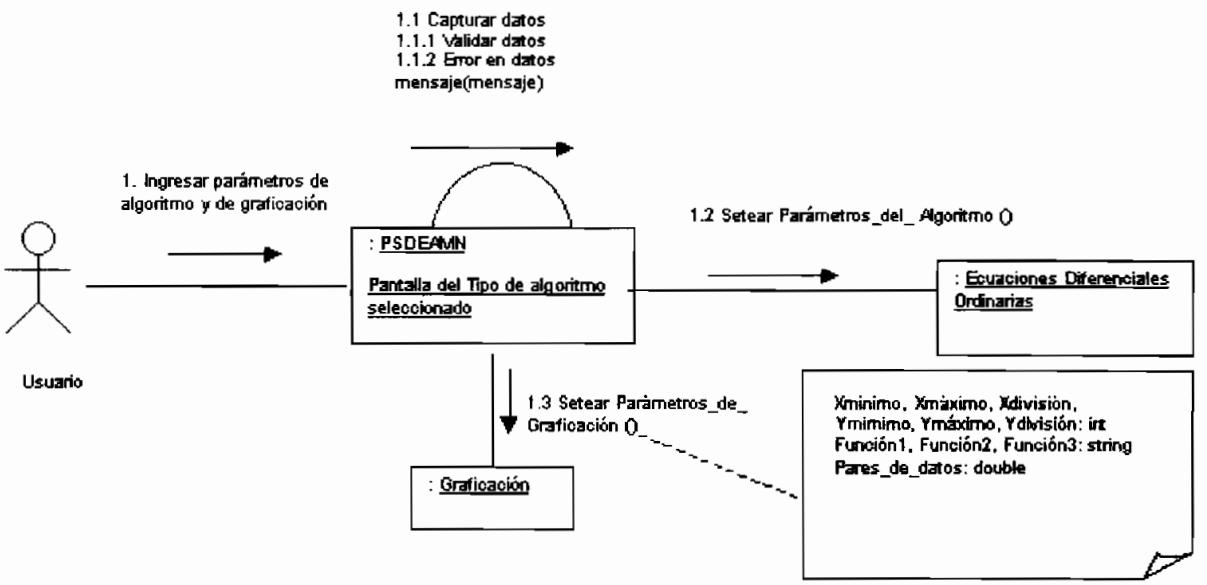


Fig. 3.123 Operación ingresar parámetros del algoritmo y de graficación

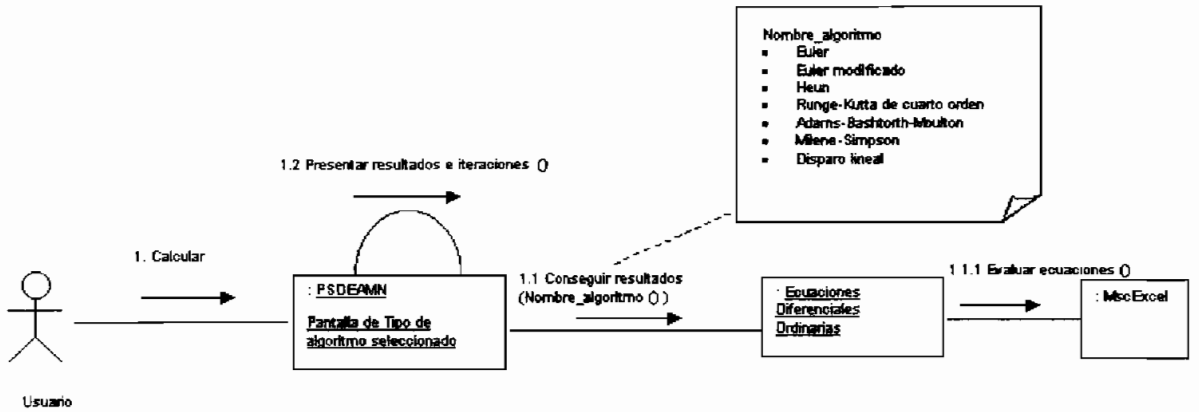


Fig. 3.124 Operación calcular

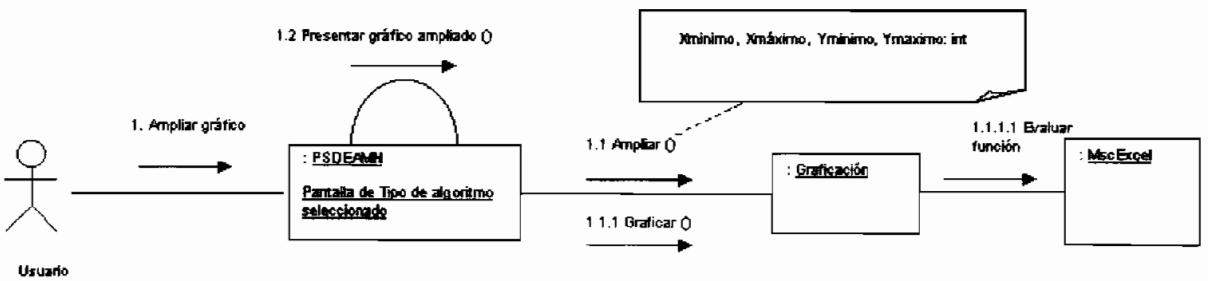


Fig. 3.125 Operación ampliar gráfico

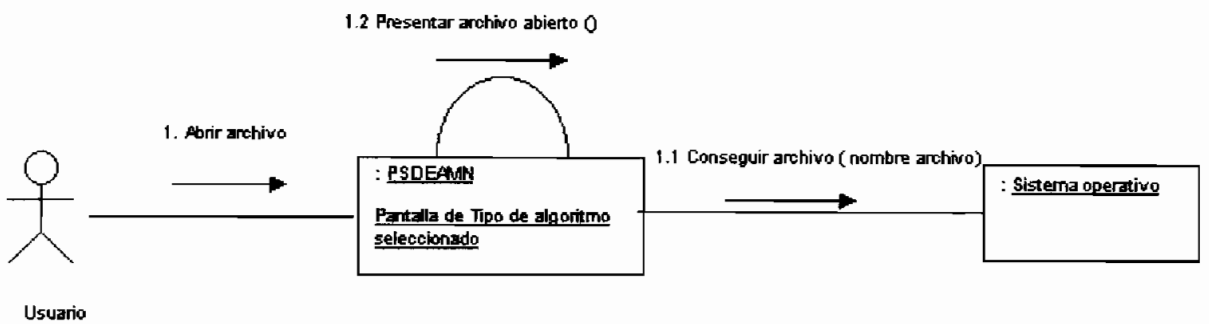


Fig. 3.126 Operación abrir archivo

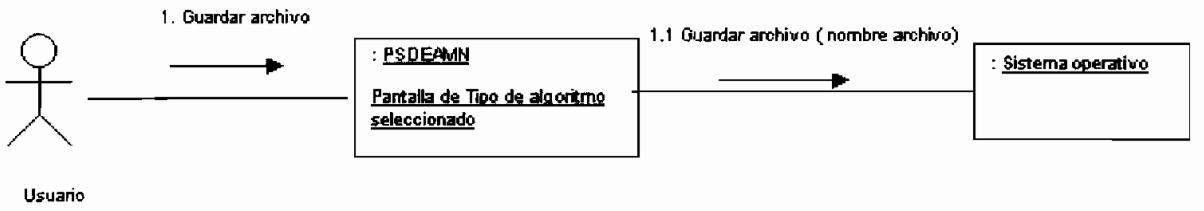


Fig. 3.127 Operación guardar archivo

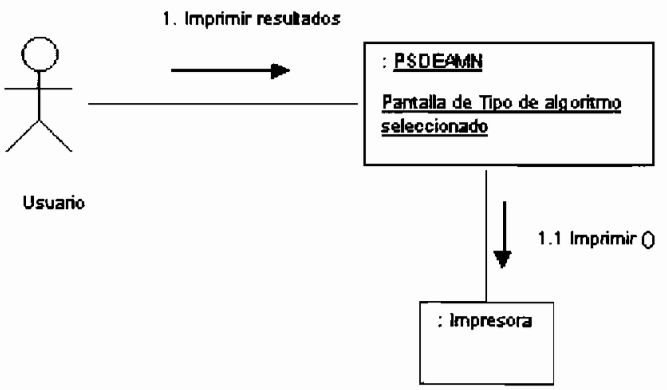


Fig. 3.128 Operación imprimir resultados

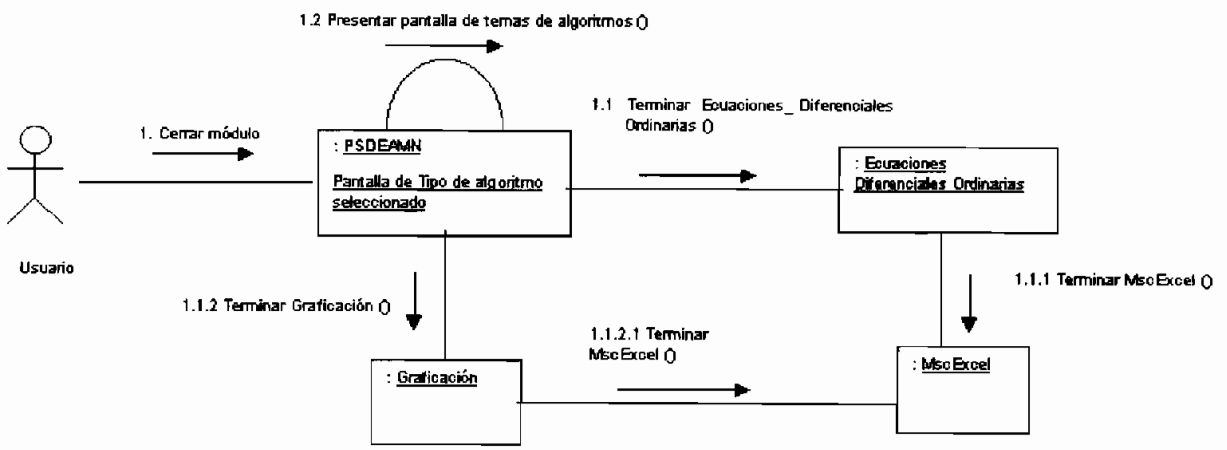


Fig. 3.129 Operación cerrar módulo

3.4.6 DIAGRAMAS DE COLABORACIÓN PARA EL CLUSTER (6)

Resolver ecuaciones diferenciales parciales

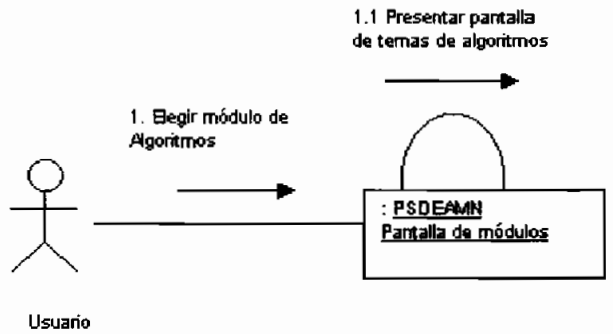


Fig. 3.130 Operación elegir módulo de algoritmos

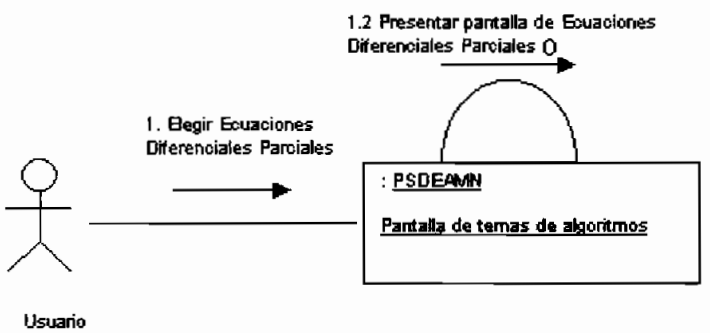


Fig. 3.131 Operación elegir ecuaciones diferenciales parciales

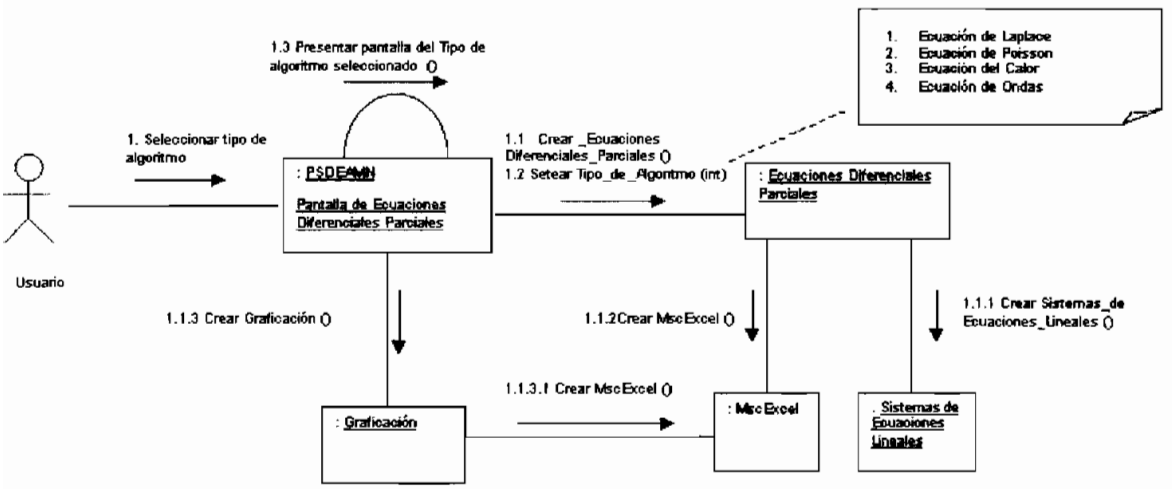


Fig. 3.132 Operación seleccionar tipo de algoritmo

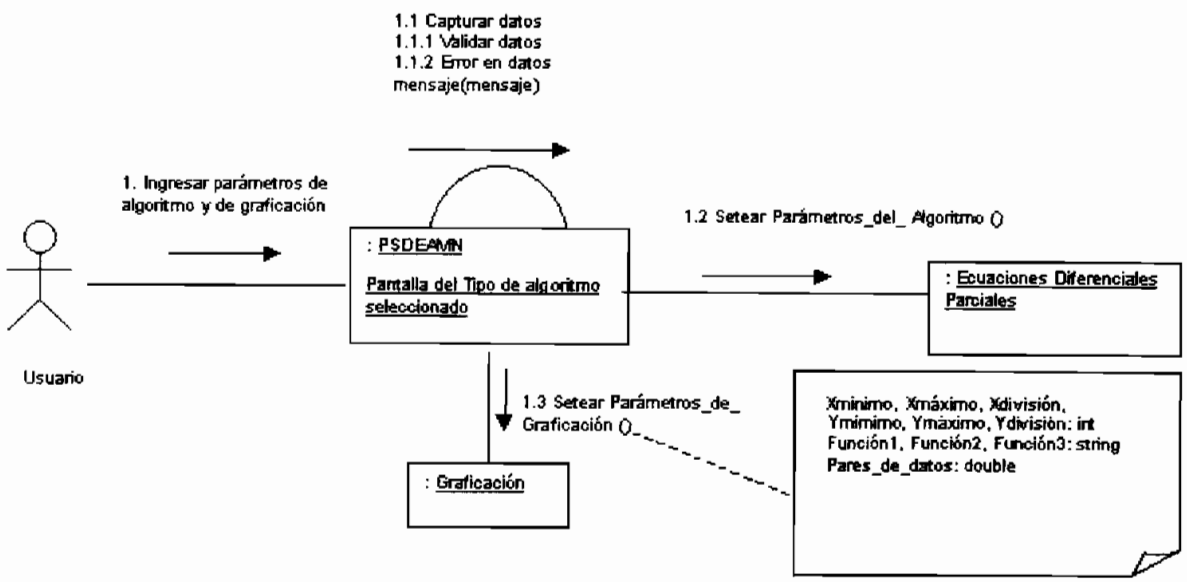


Fig. 3.133 Operación ingresar parámetros del algoritmo y de graficación

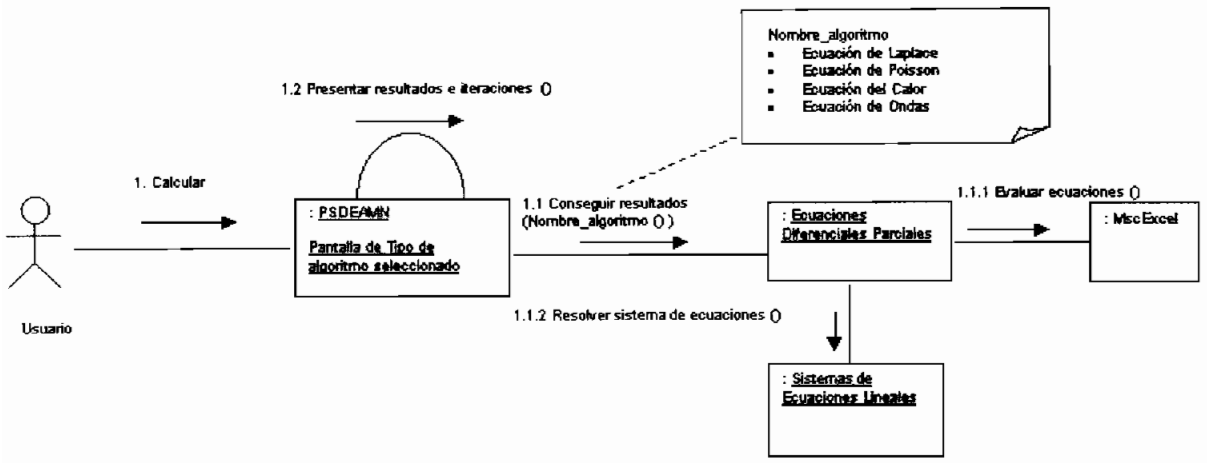


Fig. 3.134 Operación calcular

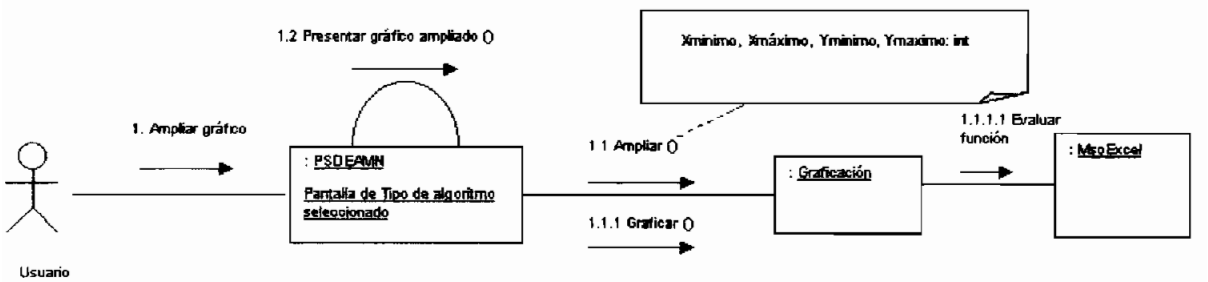


Fig. 3.135 Operación ampliar gráfico

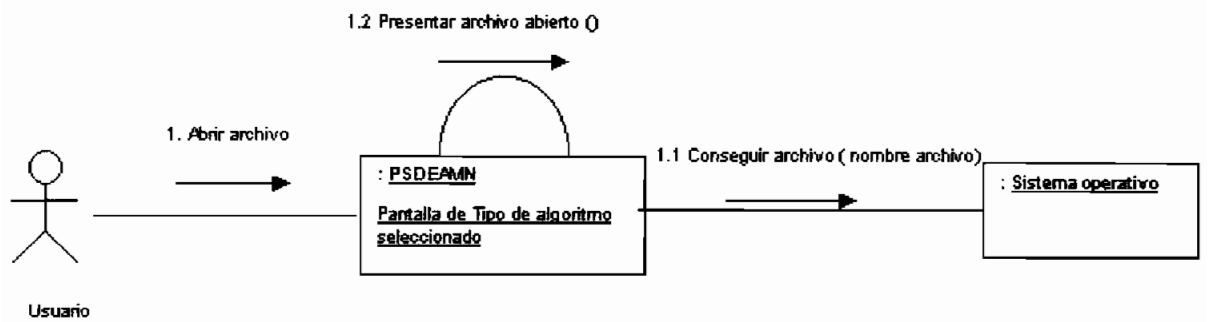


Fig. 3.136 Operación abrir archivo

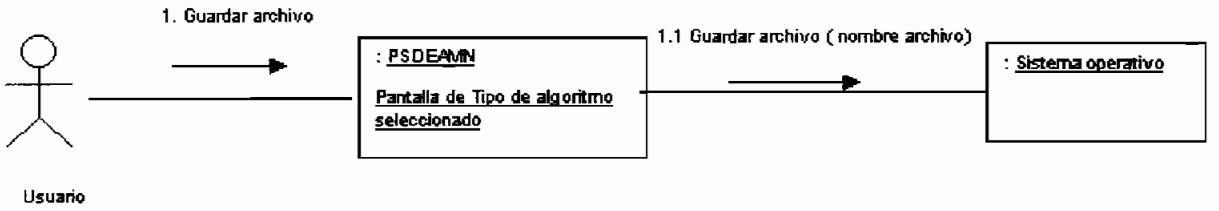


Fig. 3.137 Operación guardar archivo

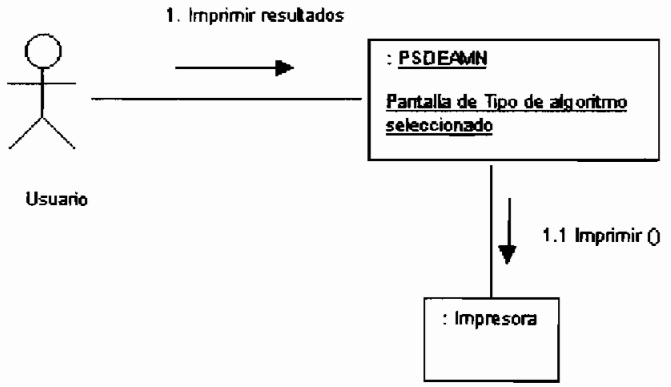


Fig. 3.138 Operación imprimir resultados

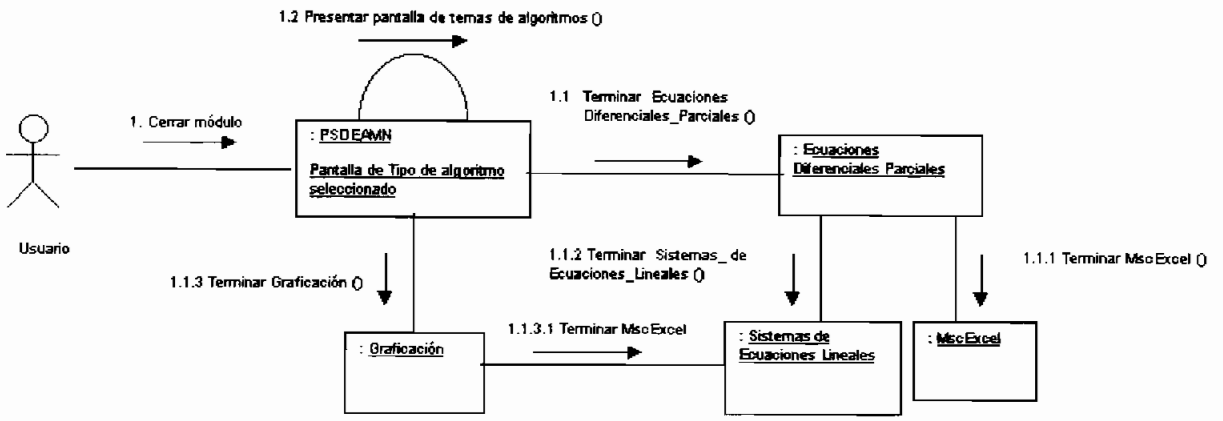


Fig. 3.139 Operación cerrar módulo

Encontrar raíces de funciones no lineales

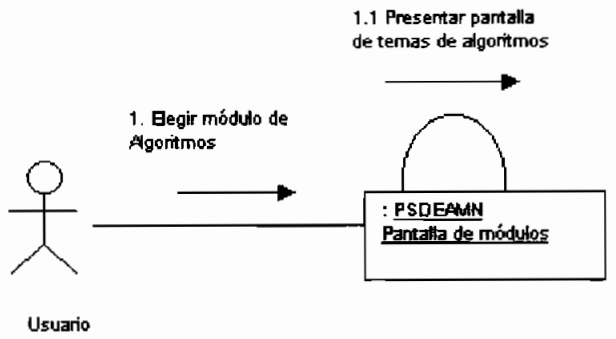


Fig. 3.140 Operación elegir módulo de algoritmos

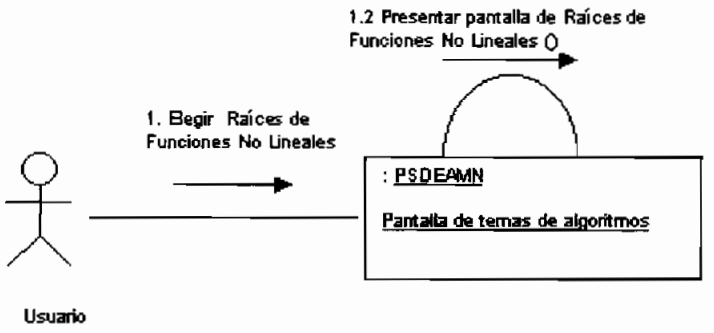


Fig. 3.141 Operación elegir raíces de funciones no lineales

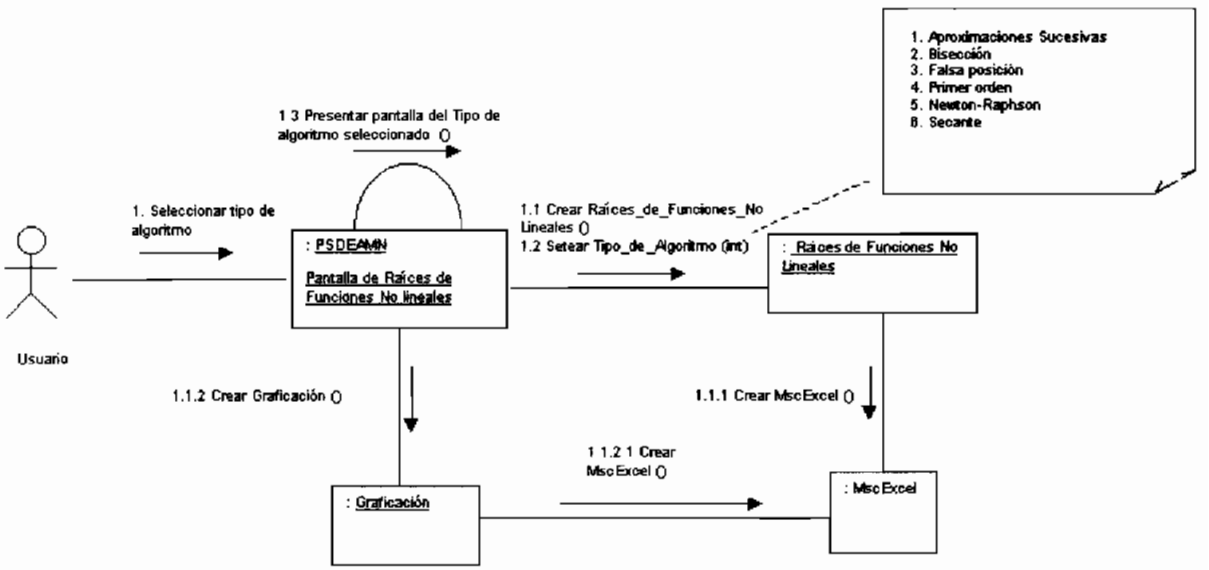


Fig. 3.142 Operación seleccionar tipo de algoritmo

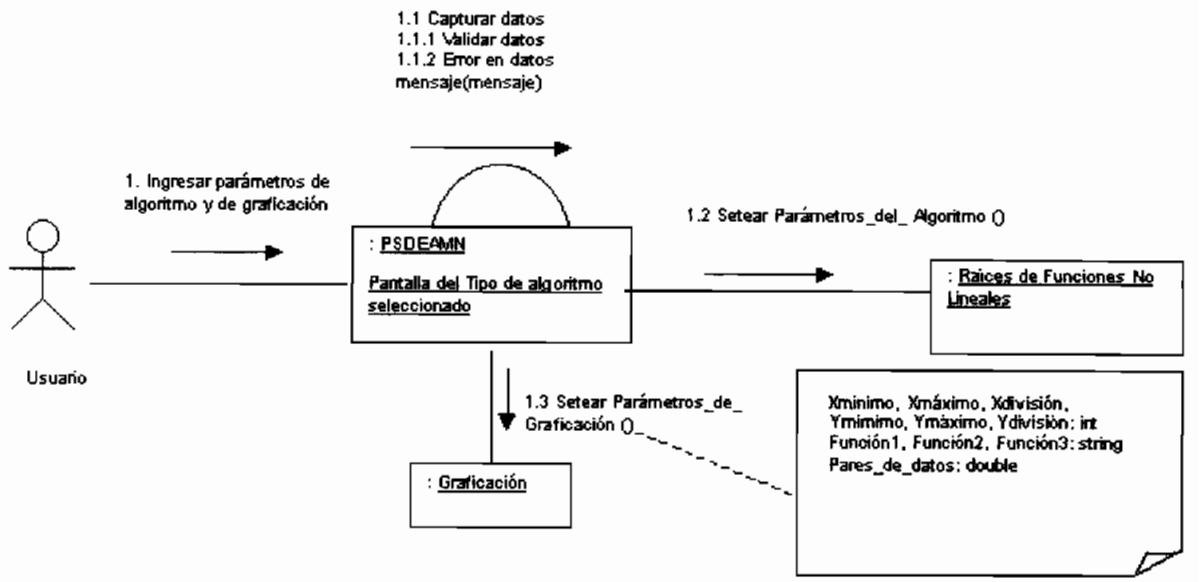


Fig. 3.143 Operación ingresar parámetros del algoritmo y de graficación

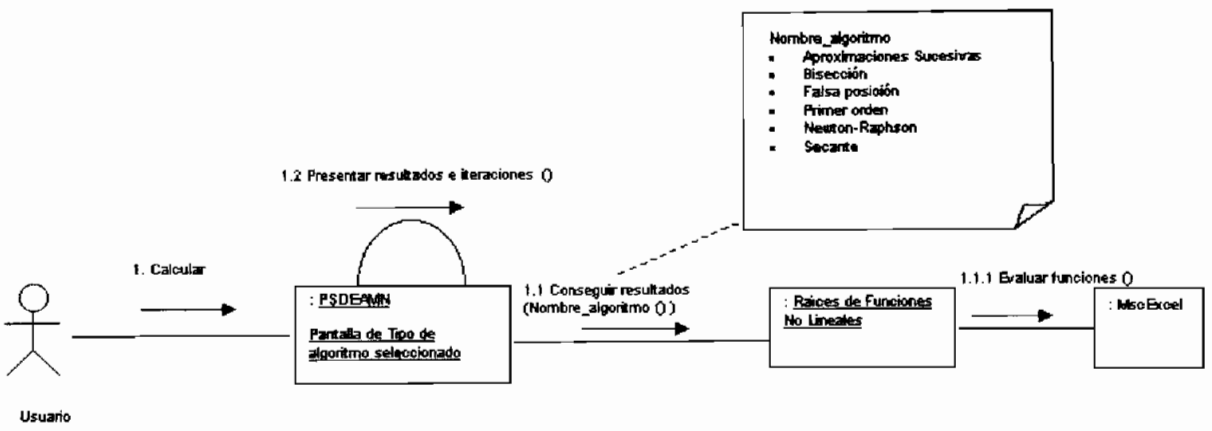


Fig. 3.144 Operación calcular

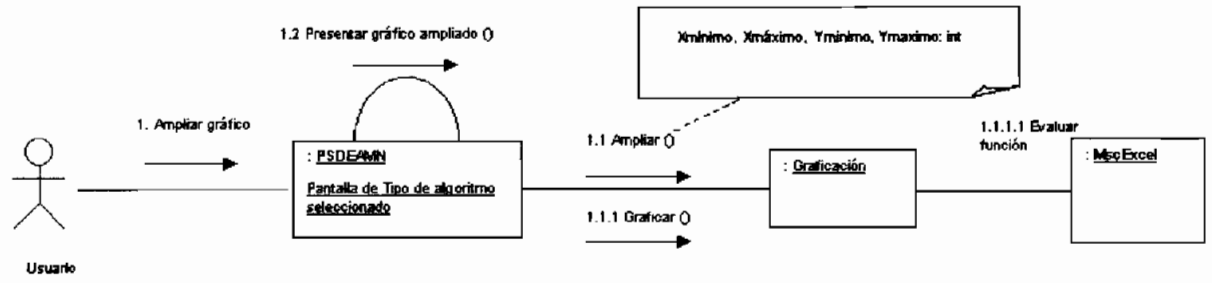


Fig. 3.145 Operación ampliar gráfico

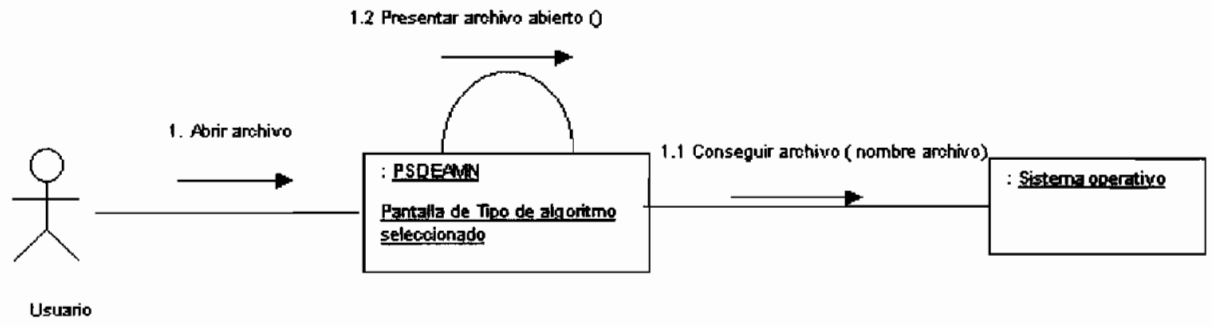


Fig. 3.146 Operación abrir archivo

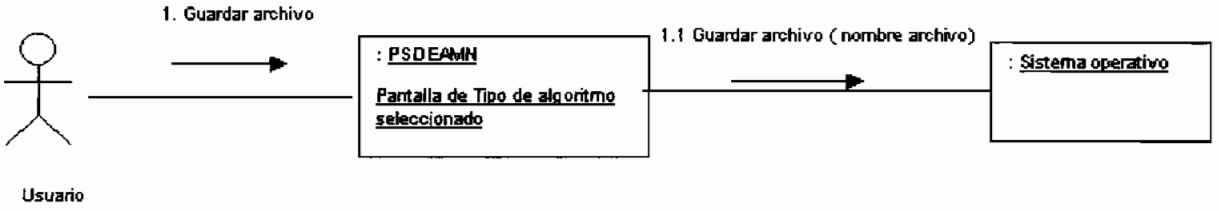


Fig. 3.147 Operación guardar archivo

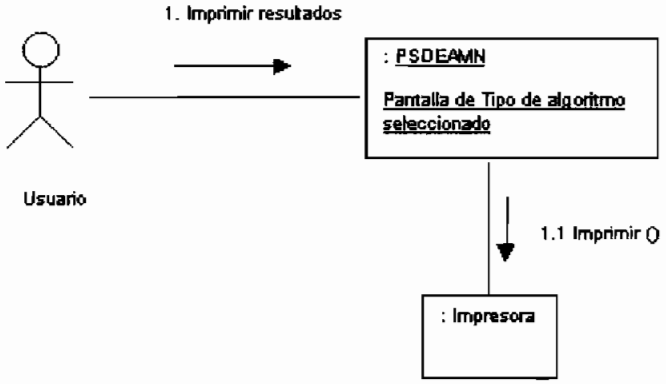


Fig. 3.148 Operación imprimir resultados

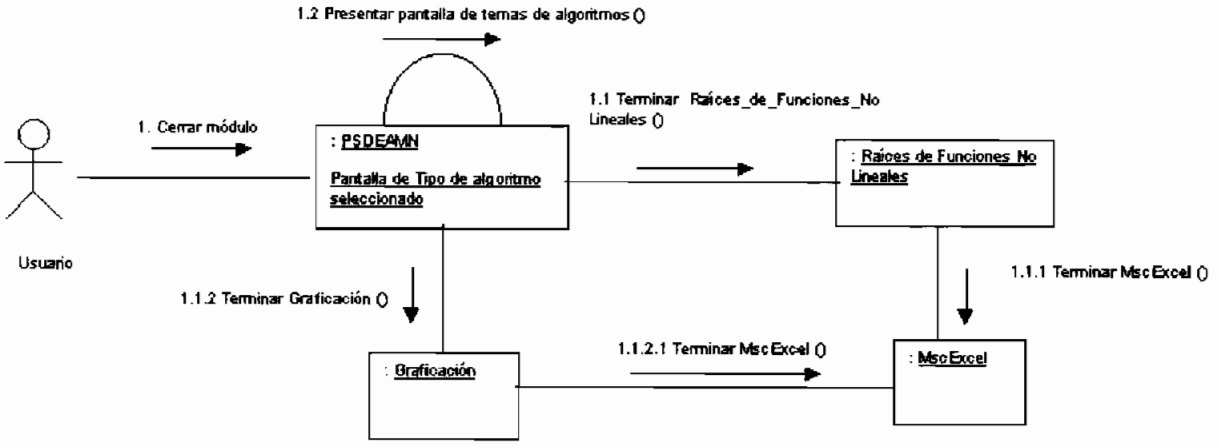


Fig. 3.149 Operación cerrar módulo

Calcular la integral de una función

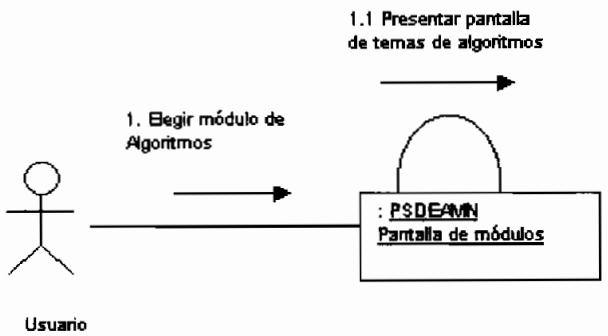


Fig. 3.150 Operación elegir módulo de algoritmos

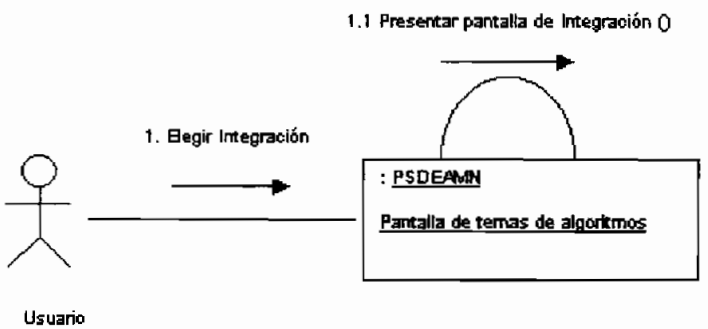


Fig. 3.151 Operación elegir integración

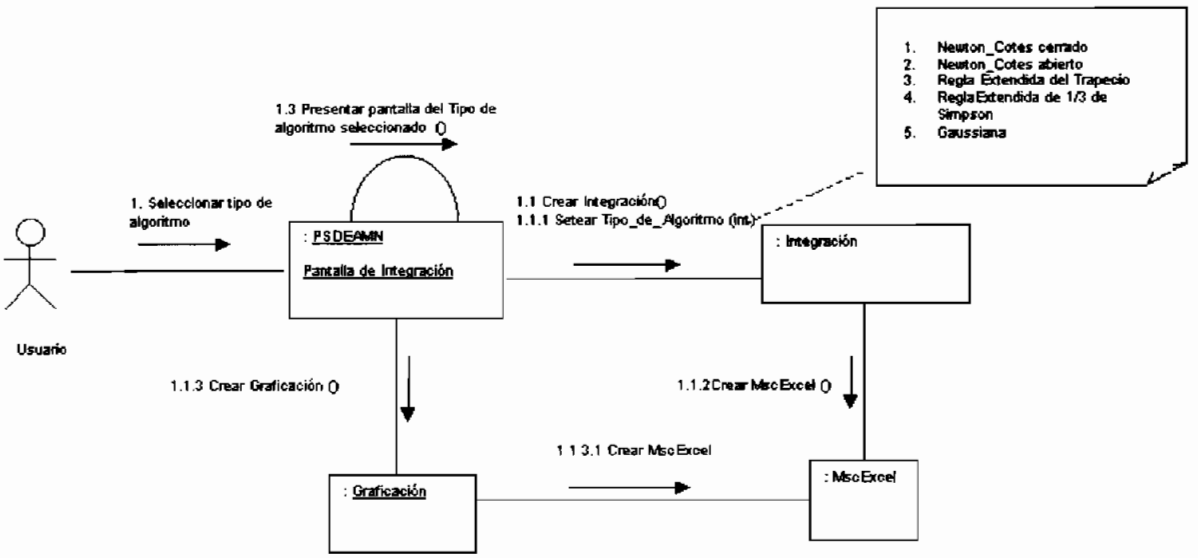


Fig. 3.152 Operación seleccionar tipo de algoritmo

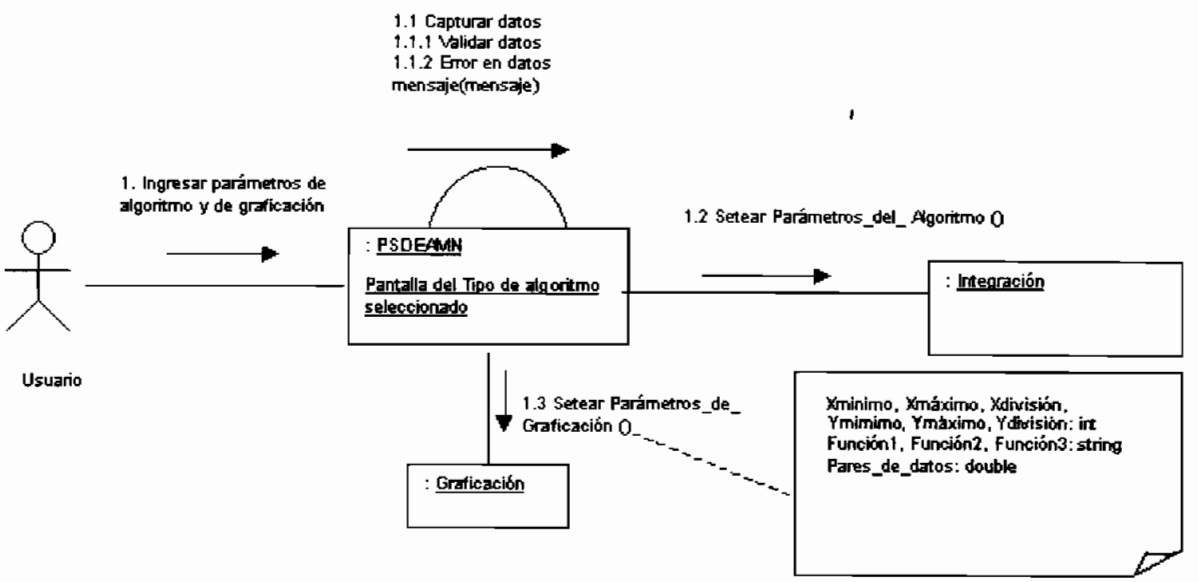


Fig. 3.153 Operación ingresar parámetros del algoritmo y de graficación

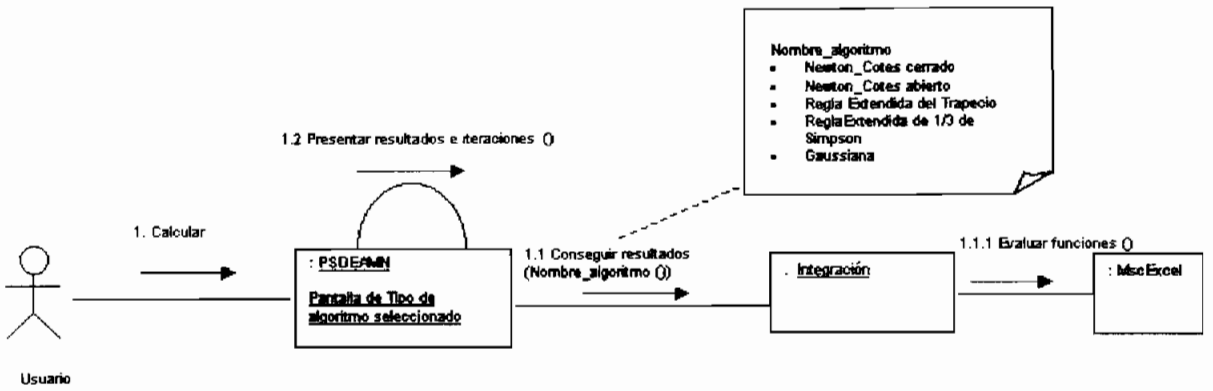


Fig. 3.154 Operación calcular

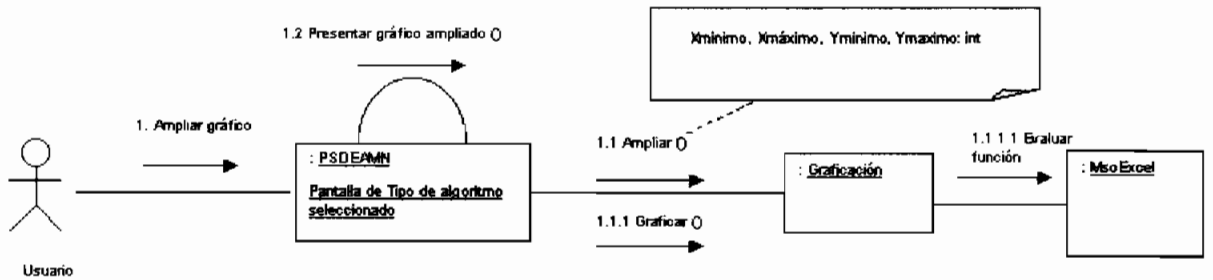


Fig. 3.155 Operación ampliar gráfico

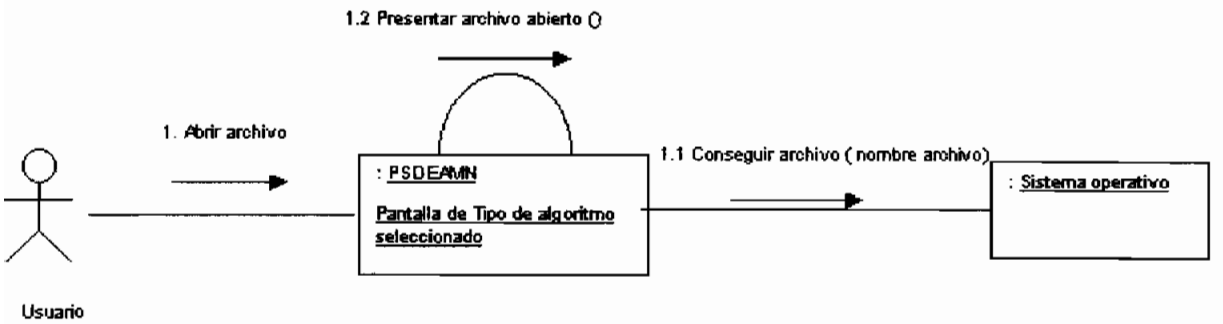


Fig. 3.156 Operación abrir archivo

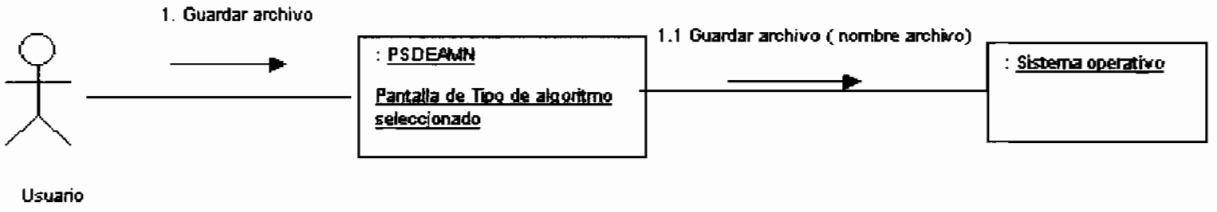


Fig. 3.157 Operación guardar archivo

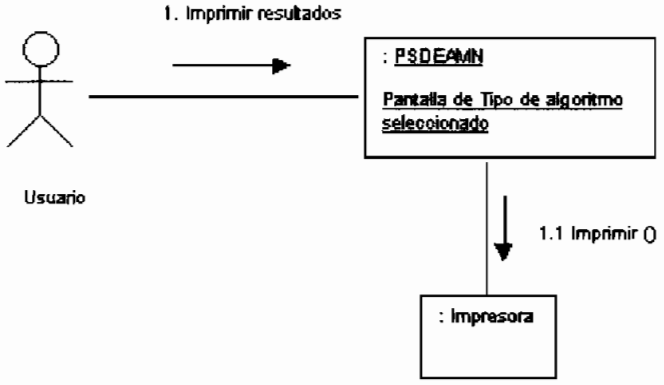


Fig. 3.158 Operación imprimir resultados

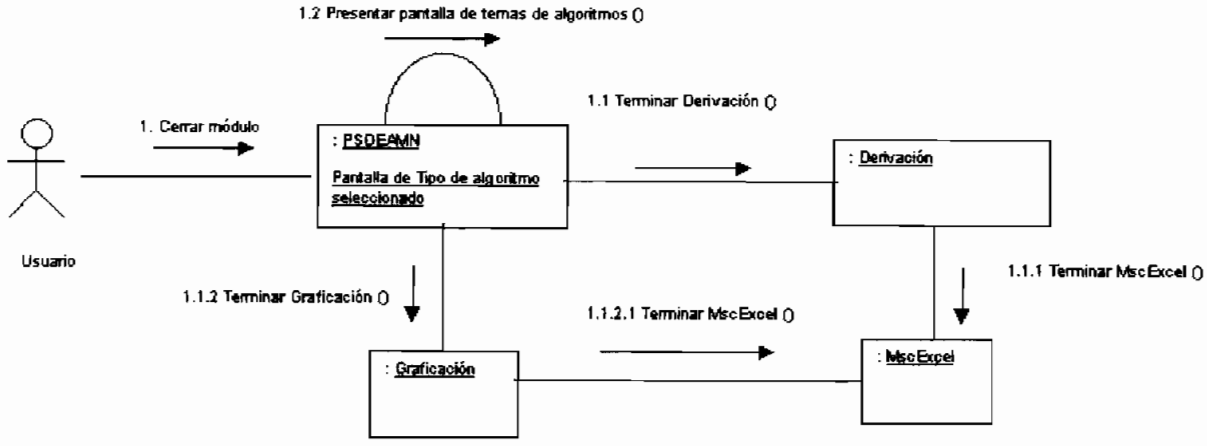


Fig. 3.159 Operación cerrar módulo

Realizar una regresión polinomial

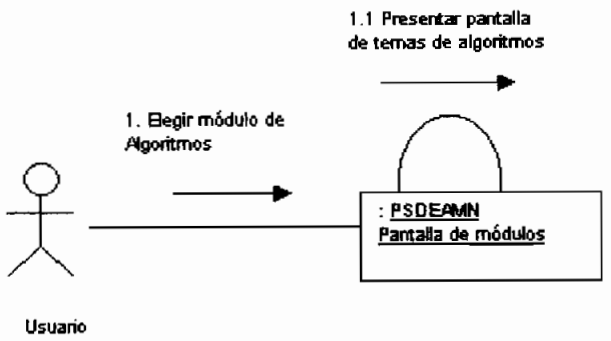


Fig. 3.160 Operación elegir módulo de algoritmos

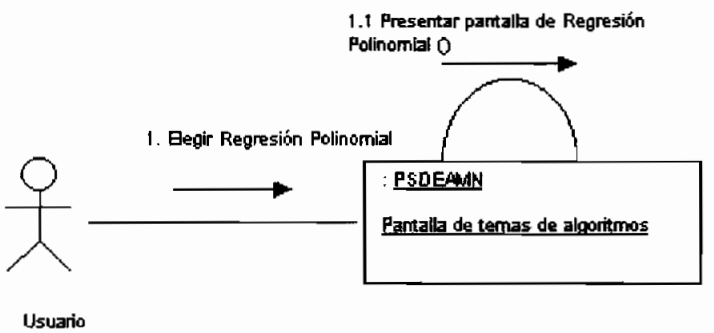


Fig. 3.161 Operación regresión polinomial

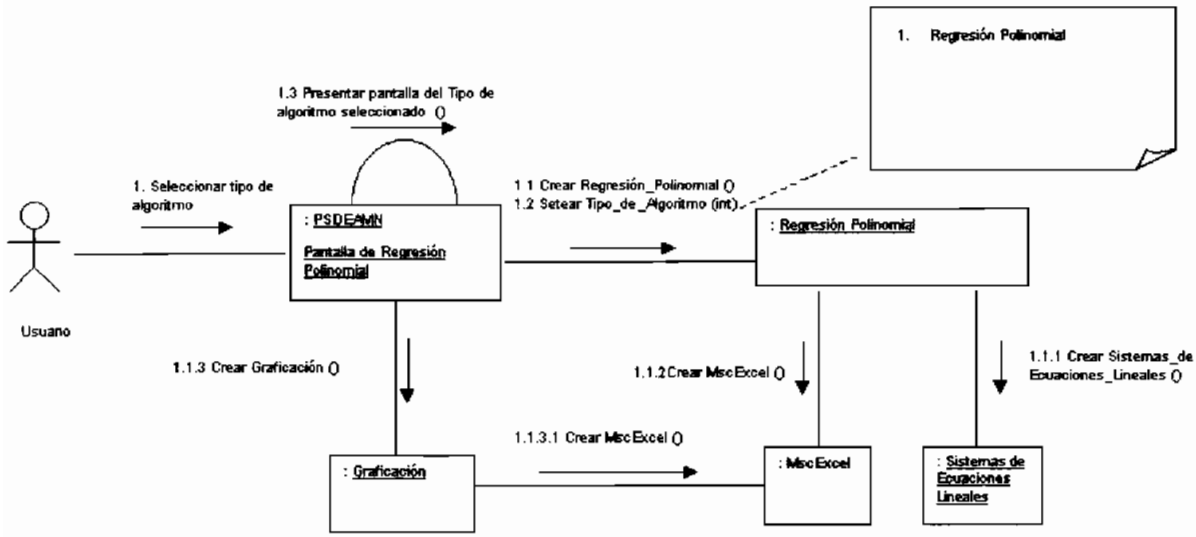


Fig. 3.162 Operación seleccionar tipo de algoritmo

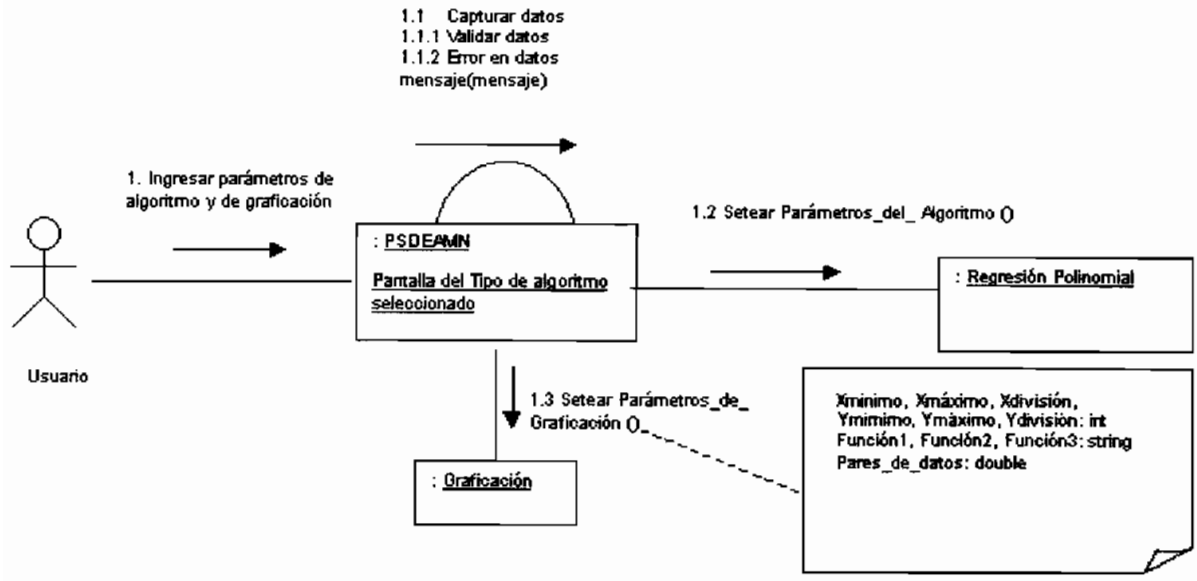


Fig. 3.163 Operación ingresar parámetros del algoritmo y de graficación

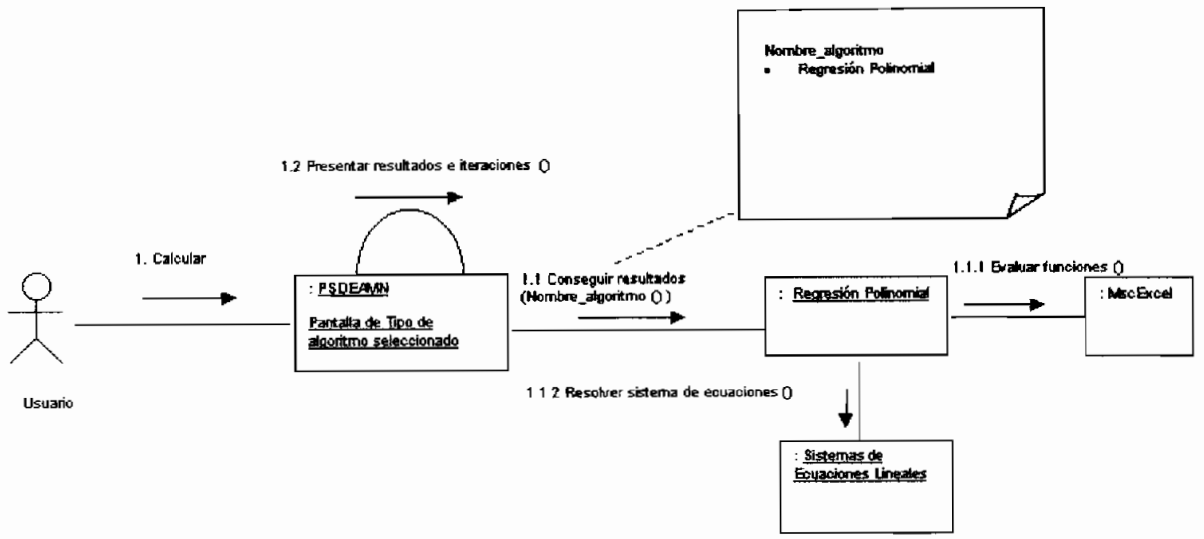


Fig. 3.164 Operación calcular

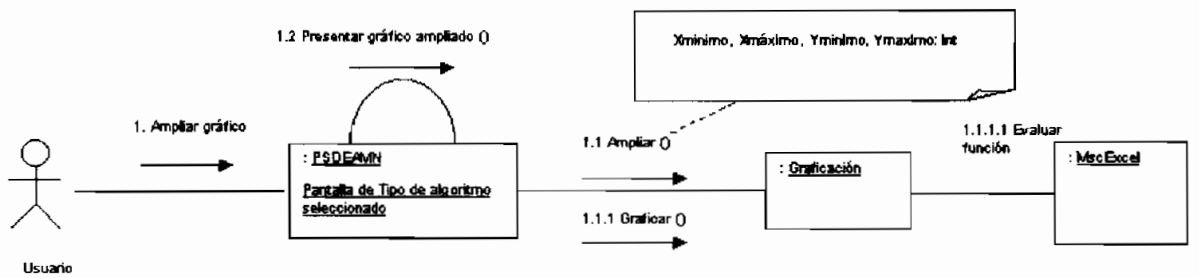


Fig. 3.165 Operación ampliar gráfico

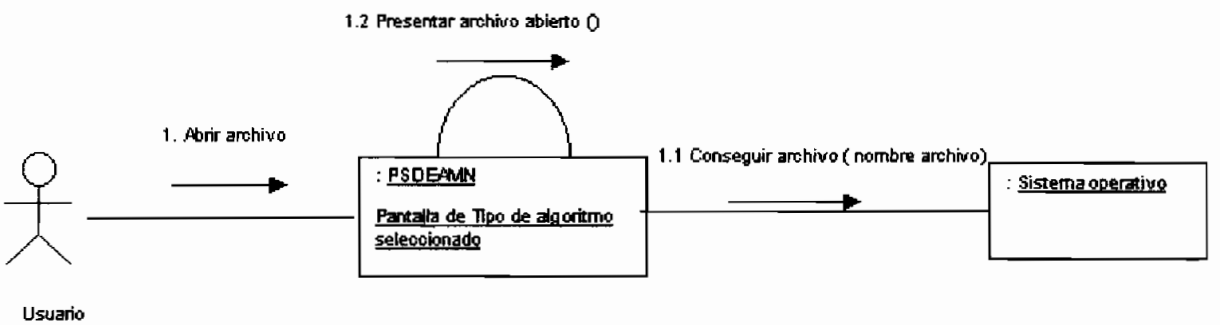


Fig. 3.166 Operación abrir archivo

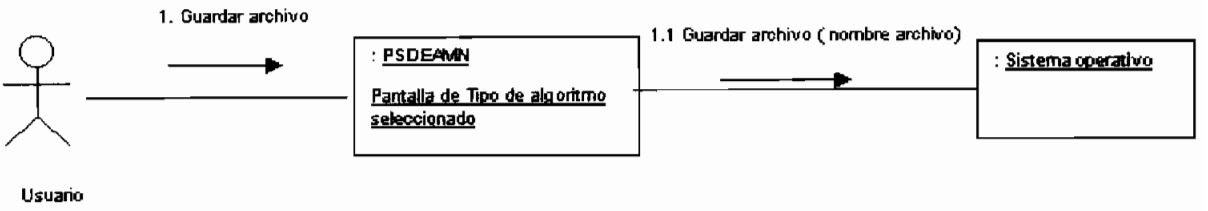


Fig. 3.167 Operación guardar archivo

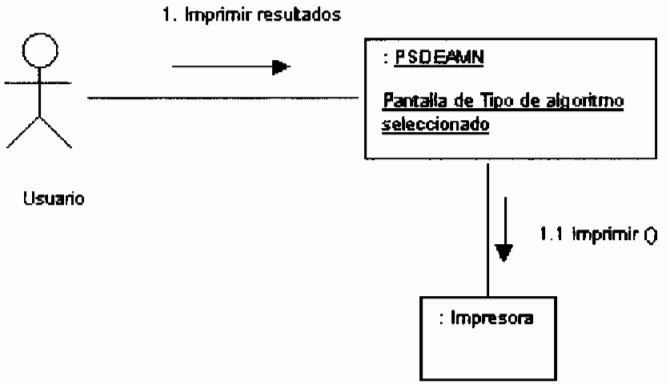


Fig. 3.1688 Operación imprimir resultados

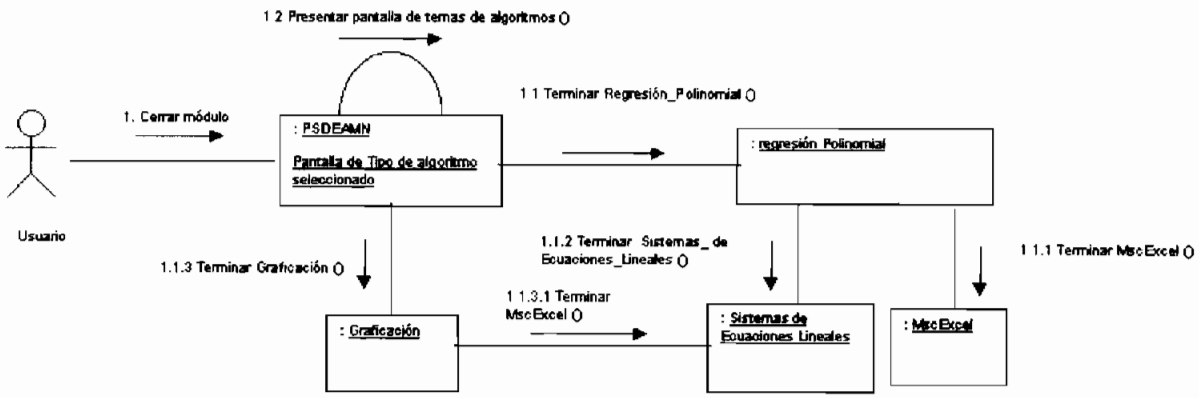


Fig. 3.169 Operación cerrar módulo

3.5 DIAGRAMAS DE CLASE DE DISEÑO

Al construir los Diagramas de Colaboración se van usando clases procedentes del Modelo Conceptual, junto con otras creadas para encargarse de responsabilidades específicas. El conjunto de todas las clases usadas, junto con sus relaciones, forma el Diagrama de Clases de Diseño.

Un Diagrama de Clases de Diseño muestra la especificación para las clases software de una aplicación. Incluye la siguiente información:

- Clases, asociaciones y atributos.
- . Interfaces, con sus operaciones y constantes.
- . Métodos.
- . Navegabilidad.
- . Dependencias.

A diferencia del Modelo Conceptual, un Diagrama de Clases de Diseño muestra definiciones de entidades software más que conceptos del mundo real. En la Fig. 3.170 se muestra un ejemplo de Diagrama de Clases de Diseño sencillo.

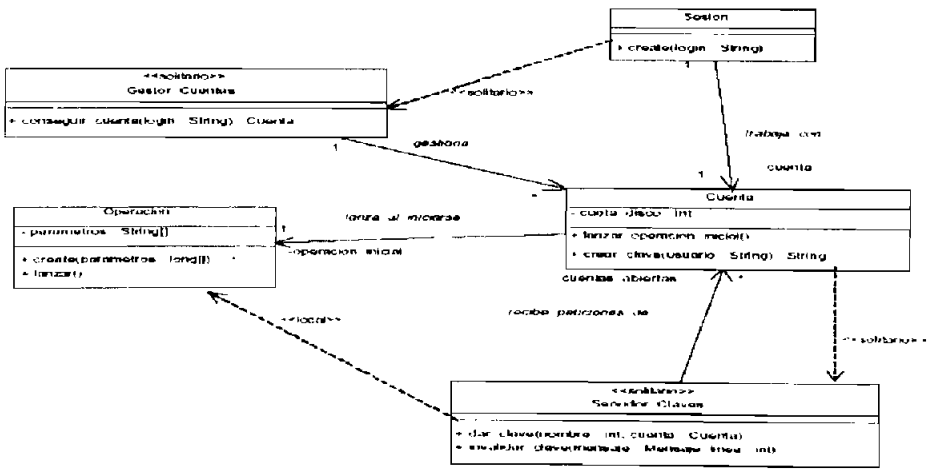


Fig. 3.170 Ejemplo de diagrama de clases de diseño

Construcción de un Diagrama de Clases de Diseño

Para crear un Diagrama de Clases de Diseño se puede seguir la siguiente estrategia:

1. Identificar todas las clases participantes en la solución software. Esto se lleva a cabo analizando los Diagramas de Interacción.
2. Representarlas en un diagrama de clases.
3. Añadir los métodos, según aparecen en los Diagramas de Interacción.
4. Añadir información de tipo a los atributos y métodos.
5. Añadir las asociaciones necesarias para soportar la visibilidad de atributos requerida.
6. Añadir flechas de navegabilidad a las asociaciones para indicar la dirección de visibilidad de los atributos.
7. Añadir relaciones de dependencia para indicar visibilidad no correspondiente a atributos.

No todas las clases que aparecían en el Modelo Conceptual tienen por qué aparecer en el Diagrama de Clases de Diseño. De hecho, tan solo se incluirán aquellas clases que tengan interés en cuanto a que se les ha asignado algún tipo de responsabilidad en el diseño de la interacción del sistema. No hay, por tanto, una transición directa entre el Modelo Conceptual y el Diagrama de Clases de Diseño, debido a que ambos se basan en enfoques completamente distintos: el primero en comprensión de un dominio, y el segundo en una solución software.

En la fase de Diseño se añaden los detalles referentes al lenguaje de programación que se vaya a usar. Por ejemplo, los tipos de los atributos y parámetros se expresarán según la sintaxis del lenguaje de implementación escogido.

A continuación se muestran los diagramas de clases de diseño para cluster usando la Herramienta Case Visual Modeler que es parte del Software Visual Studio.

3.5.1 DISEÑO DEL CLUSTER (1)

3.5.1.1 Identificar estudiante

De acuerdo a los diagramas de colaboración para este caso de uso el diagrama de clases de diseño sería el de la Fig. pero debido a que la clase usuario tiene dos atributos y solamente Tipo_usuario interviene en las relaciones para implementar este caso de uso, se ve necesario eliminarlo y que este atributo sea parte del módulo de clase Clave; el método crear clave se lo implementara en la interfaz Seguridad ya que es más conveniente que se cree una instancia del módulo clave desde la interfaz que implementar un módulo para este cometido. La Fig. muestra el diagrama de clase de diseño que se implementara

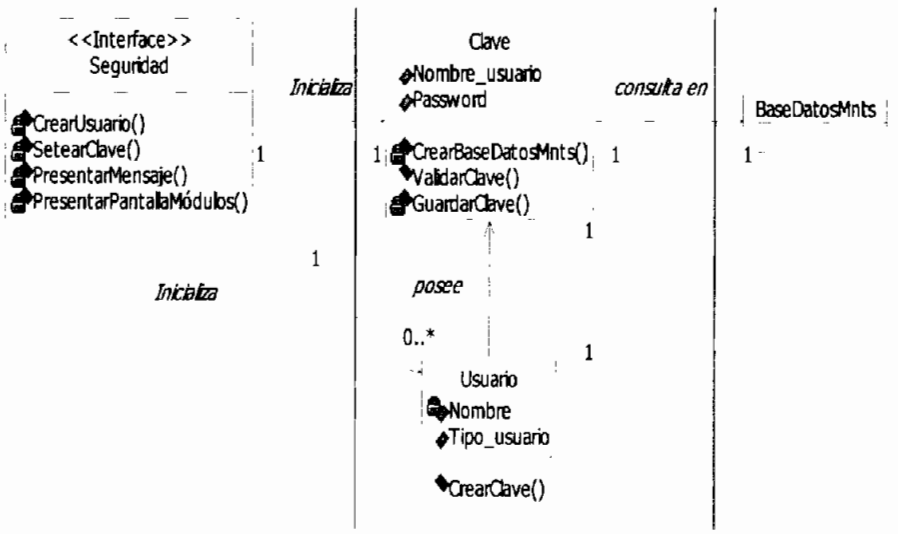


Fig. 3.171 Diagrama de clases de diseño para el caso de uso Identificar estudiante

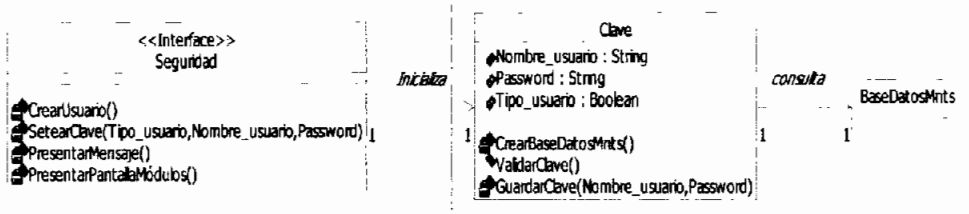


Fig. 3.172 Diagrama de clases de diseño para el caso de uso Identificar estudiante mejorado

3.5.2 DISEÑO DEL CLUSTER (2)

3.5.2.1 Desplegar tema del contenido teórico

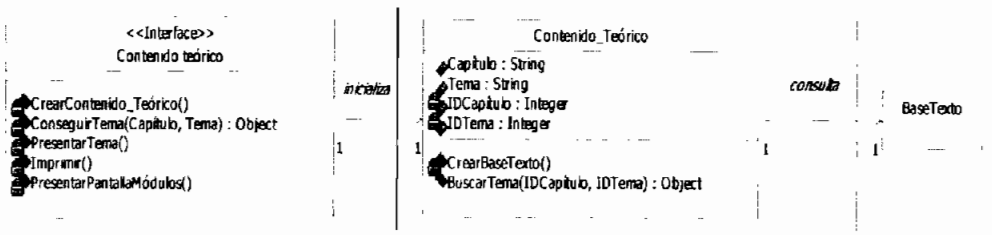


Fig. 3.173 Diagrama de clases de diseño para el caso de uso Desplegar tema del contenido teórico

3.5.2.2 Resolver sistemas de ecuaciones lineales

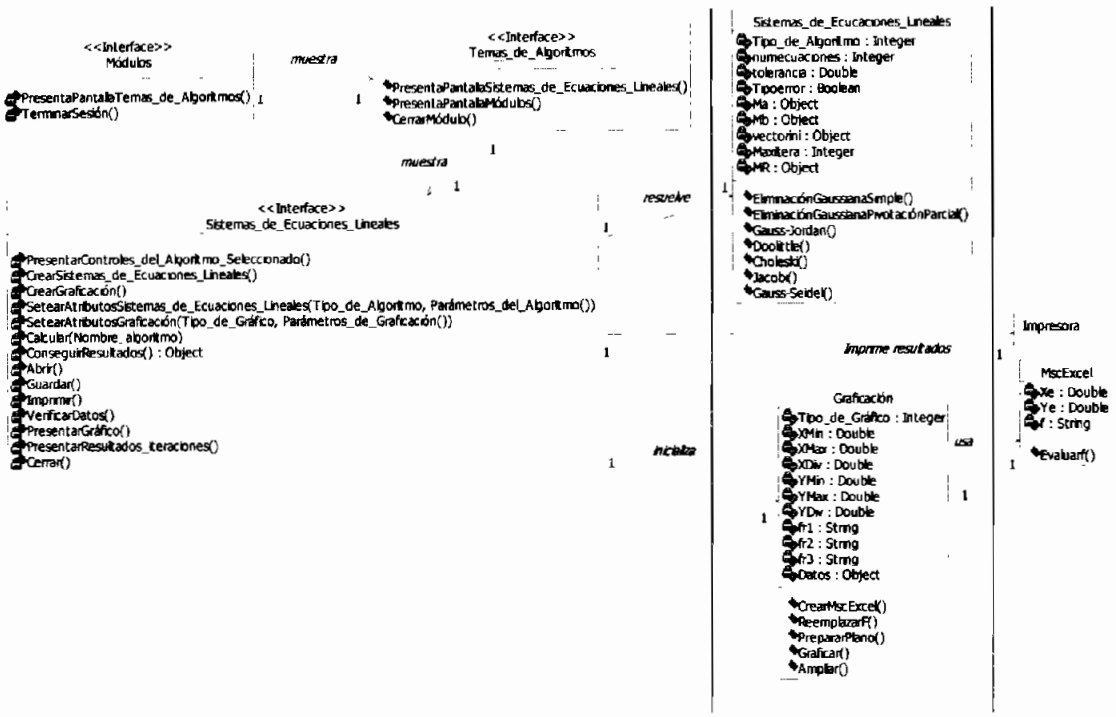


Fig. 3.174 Diagrama de clases de diseño para el caso de uso Resolver sistemas de ecuaciones lineales

3.5.2.3 Graficar una función

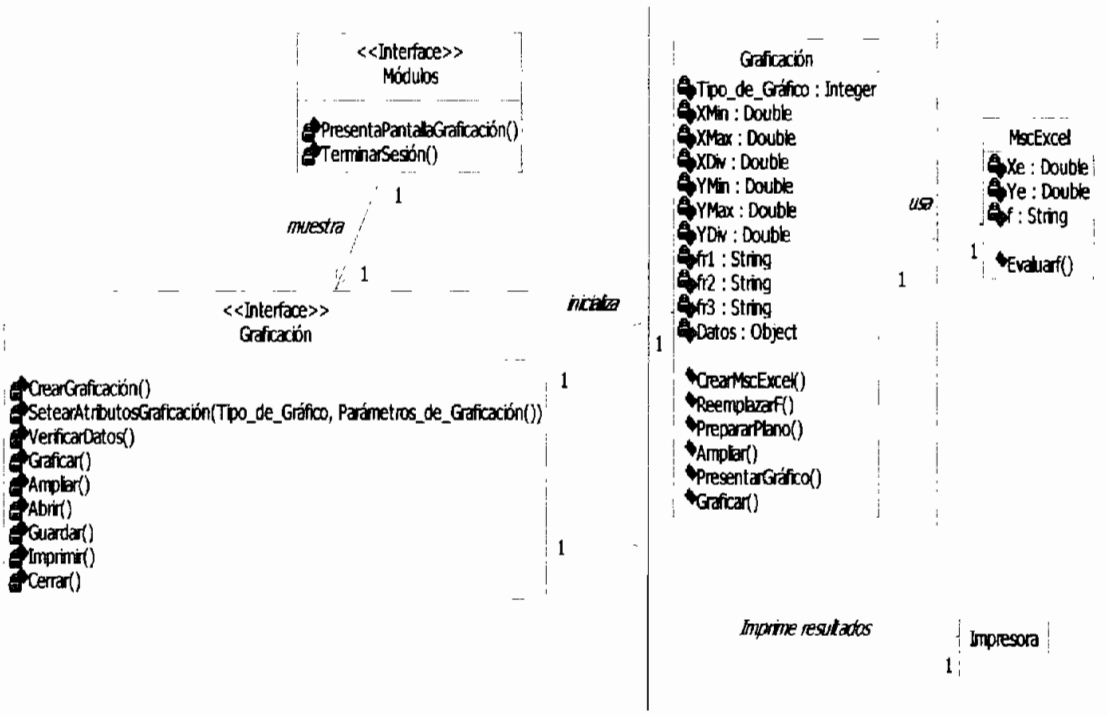


Fig. 3.175 Diagrama de clases de diseño para el caso de uso Graficar una función

3.5.3 DISEÑO DEL CLUSTER (3)

3.5.3.1 Realizar Evaluación

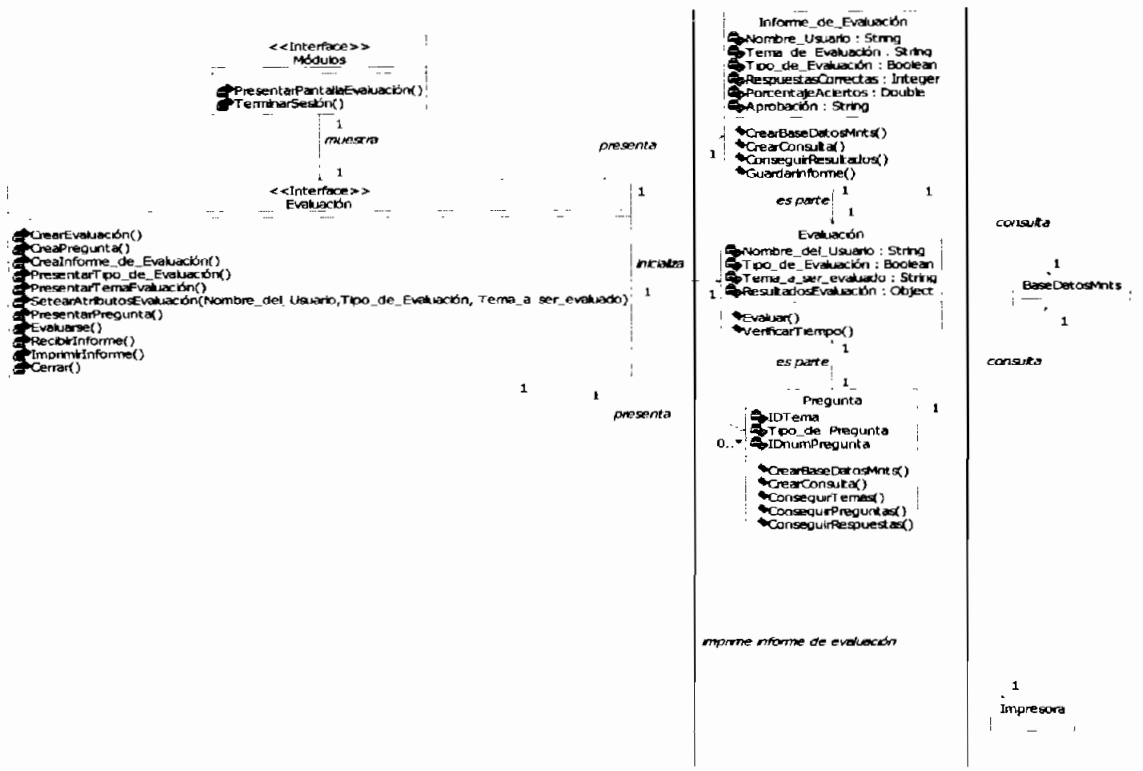


Fig. 3.176 Diagrama de clases de diseño para el caso de uso Realizar evaluación

3.5.3.2 Resolver Sistemas de ecuaciones no lineales

Debido a que los algoritmos del módulo de Sistemas de Ecuaciones No Lineales hacen uso de los algoritmos del módulo Sistemas de Ecuaciones Lineales y de sus relaciones con los módulos que forman parte del diagrama de clases de diseño, se ve la necesidad de implementar los algoritmos de Sistemas de Ecuaciones No Lineales como métodos del módulo de Sistemas de Ecuaciones Lineales lo que permitirá disminuir el número de interfaces, módulos y la escritura de instrucciones

para ello. La Fig. 3.177 muestra el diagrama de clase de diseño para este caso de uso.

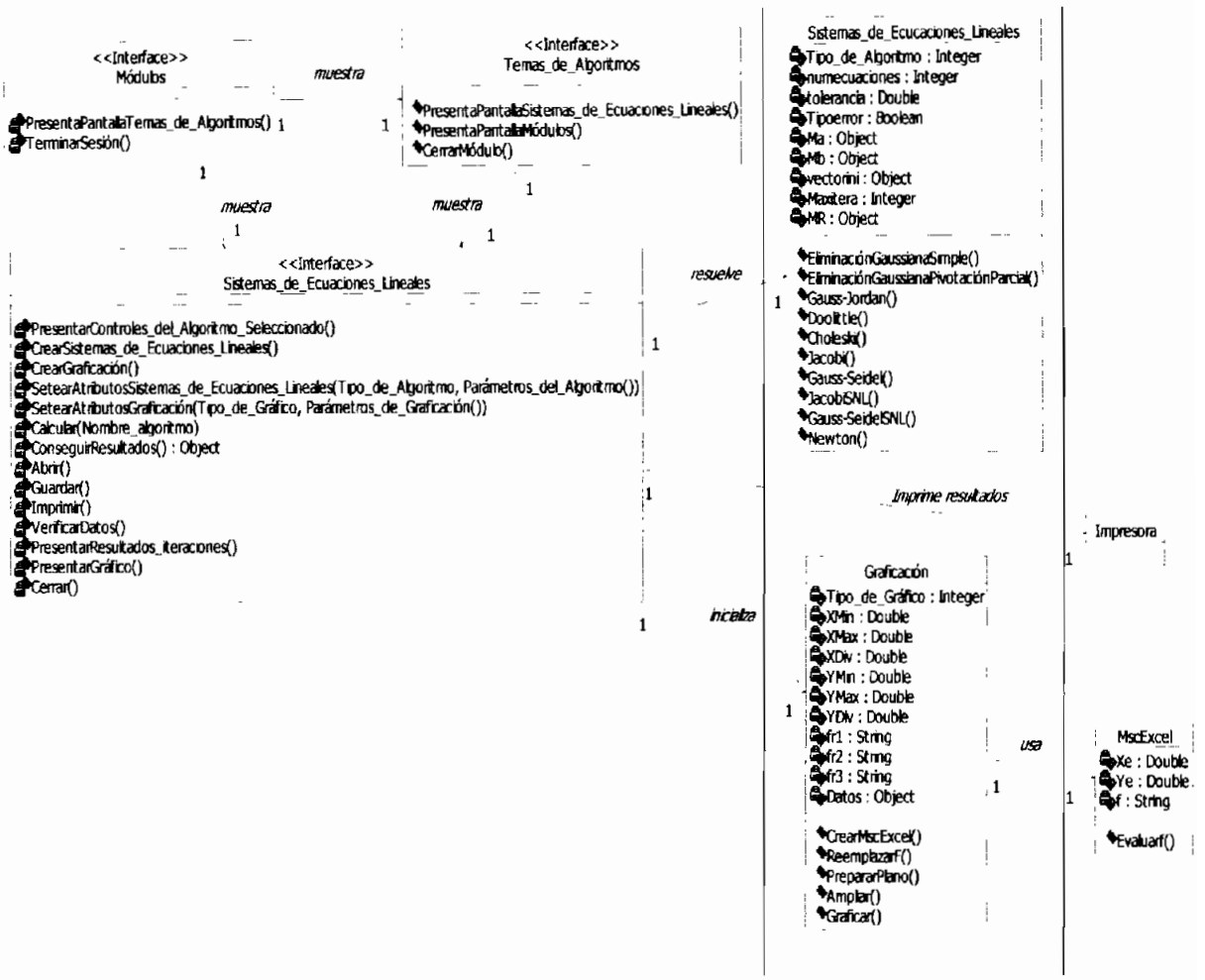


Fig. 3.177 Diagrama de clases de diseño para el caso de uso Resolver sistemas de ecuaciones no lineales.

3.5.4 DISEÑO DEL CLUSTER (4)

3.5.4.1 Encontrar raíces de un polinomio

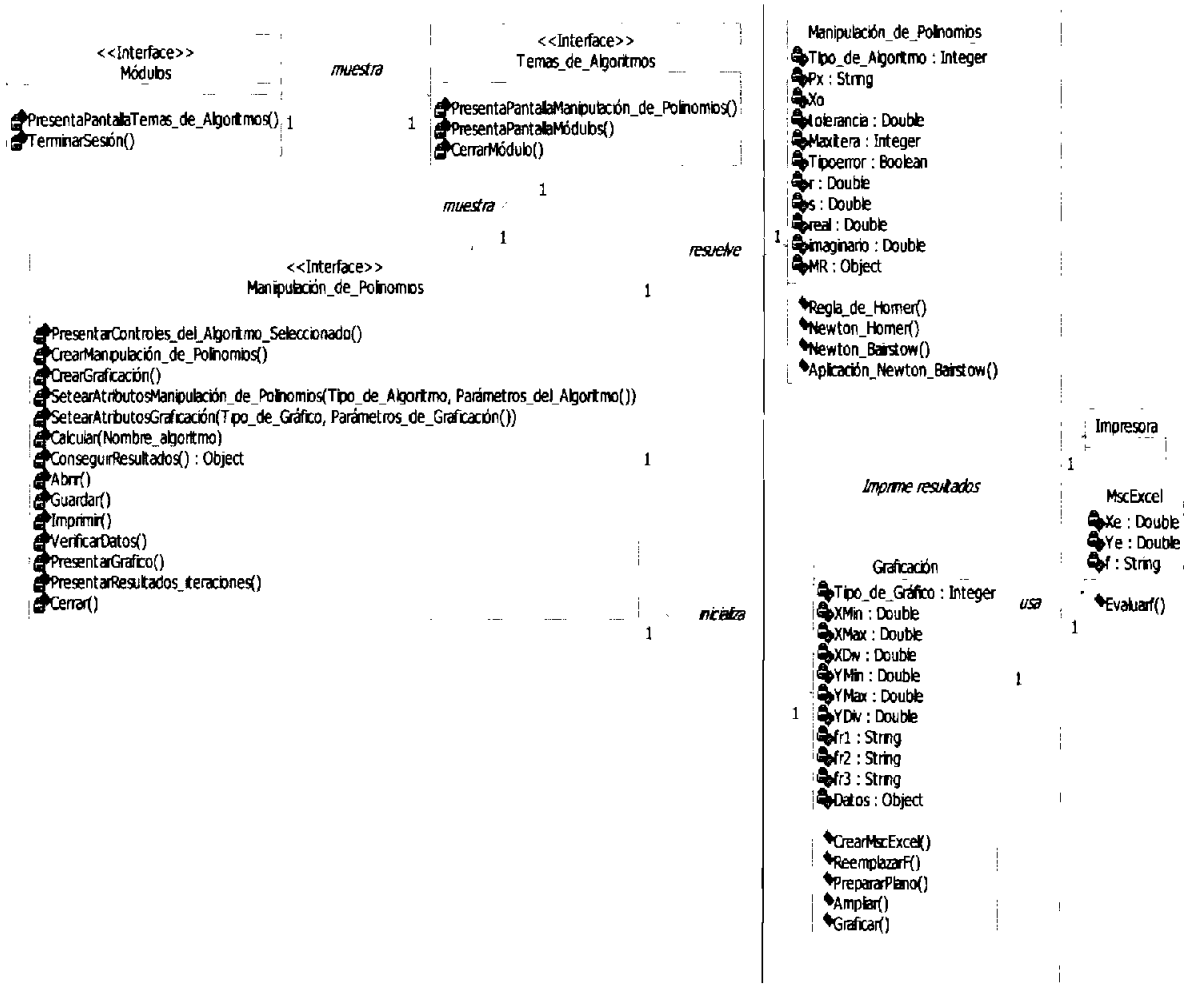


Fig. 3.178 Diagrama de clases de diseño para el caso de uso Encontrar raíces de un polinomio

3.5.4.2 Calcular la derivada de una función

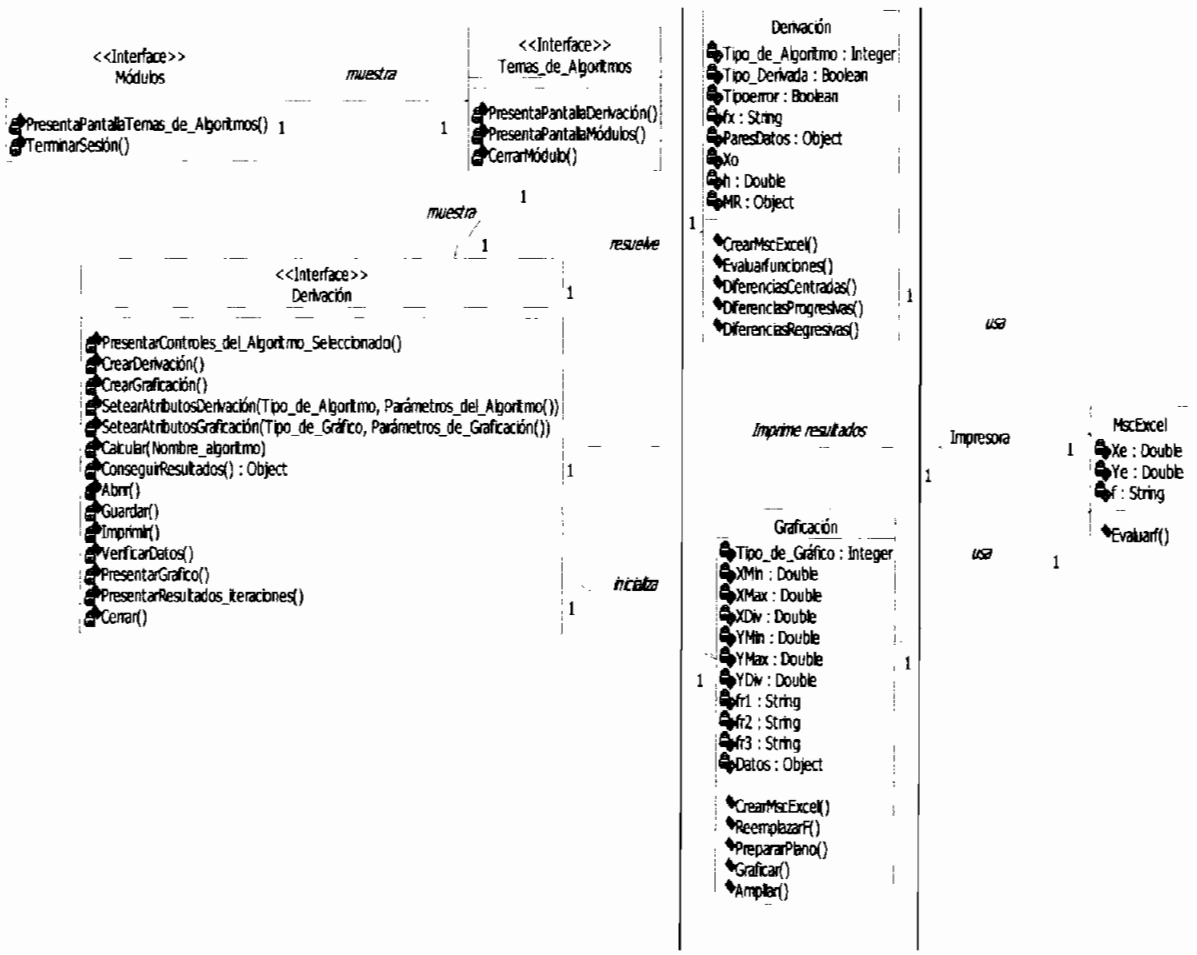


Fig. 3.179 Diagrama de clases de diseño para el caso de uso Calcular la derivada de una función

3.5.5 DISEÑO DEL CLUSTER (5)

3.5.5.1 Resolver un sistema de ecuaciones diferenciales ordinarias

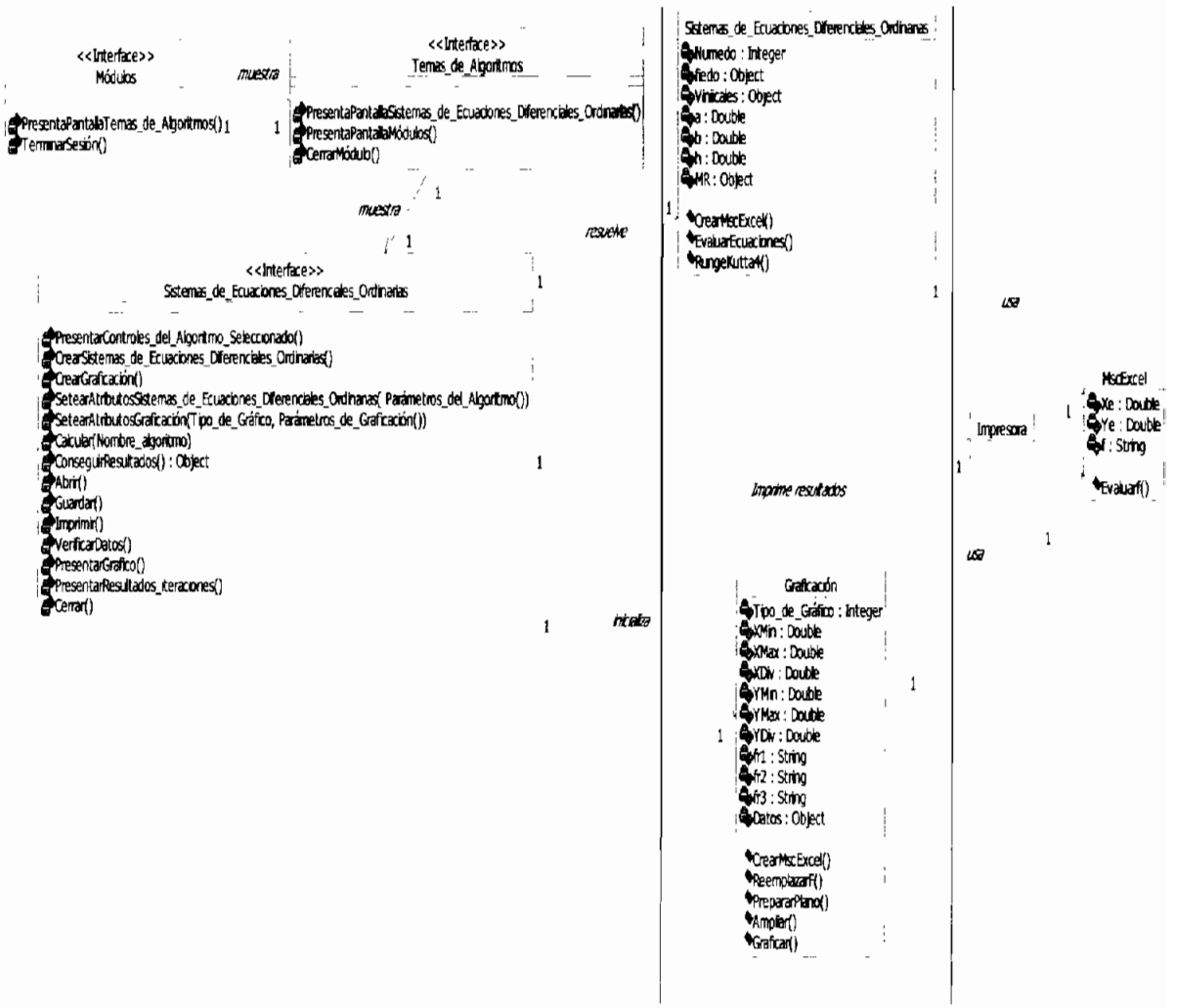


Fig. 3.180 Diagrama de clases de diseño para el caso de uso Resolver sistemas de ecuaciones diferenciales ordinarias

3.5.5.2 Realizar una interpolación polinomial

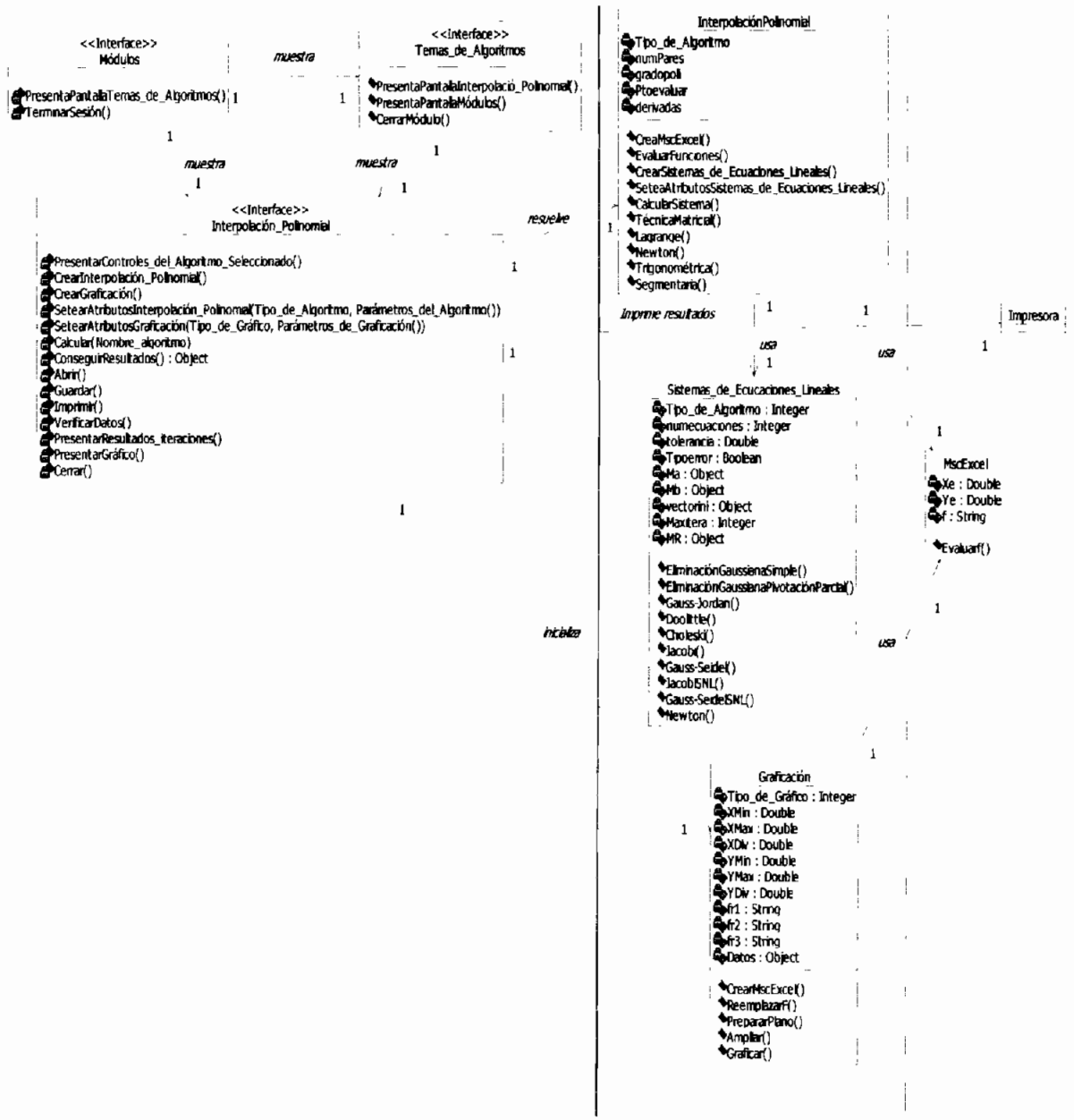


Fig. 3.181 Diagrama de clases de diseño para el caso de uso Realizar una interpolación polinomial

3.5.5.3 Resolver ecuaciones diferenciales ordinarias

Debido a que los algoritmos del módulo de Sistemas de Ecuaciones Diferenciales Ordinarias hacen uso de los algoritmos del módulo Ecuaciones Diferenciales Ordinarias y de sus relaciones con los módulos que forman parte del diagrama de clases de diseño, se ve la necesidad de implementar los algoritmos de Sistemas de Ecuaciones Diferenciales Ordinarias como métodos del módulo de Ecuaciones Diferenciales ordinarias lo que permitirá disminuir el número de interfaces, módulos y la escritura de instrucciones para ello. La Fig. 3.182 muestra el diagrama de clase de diseño para este caso de uso.

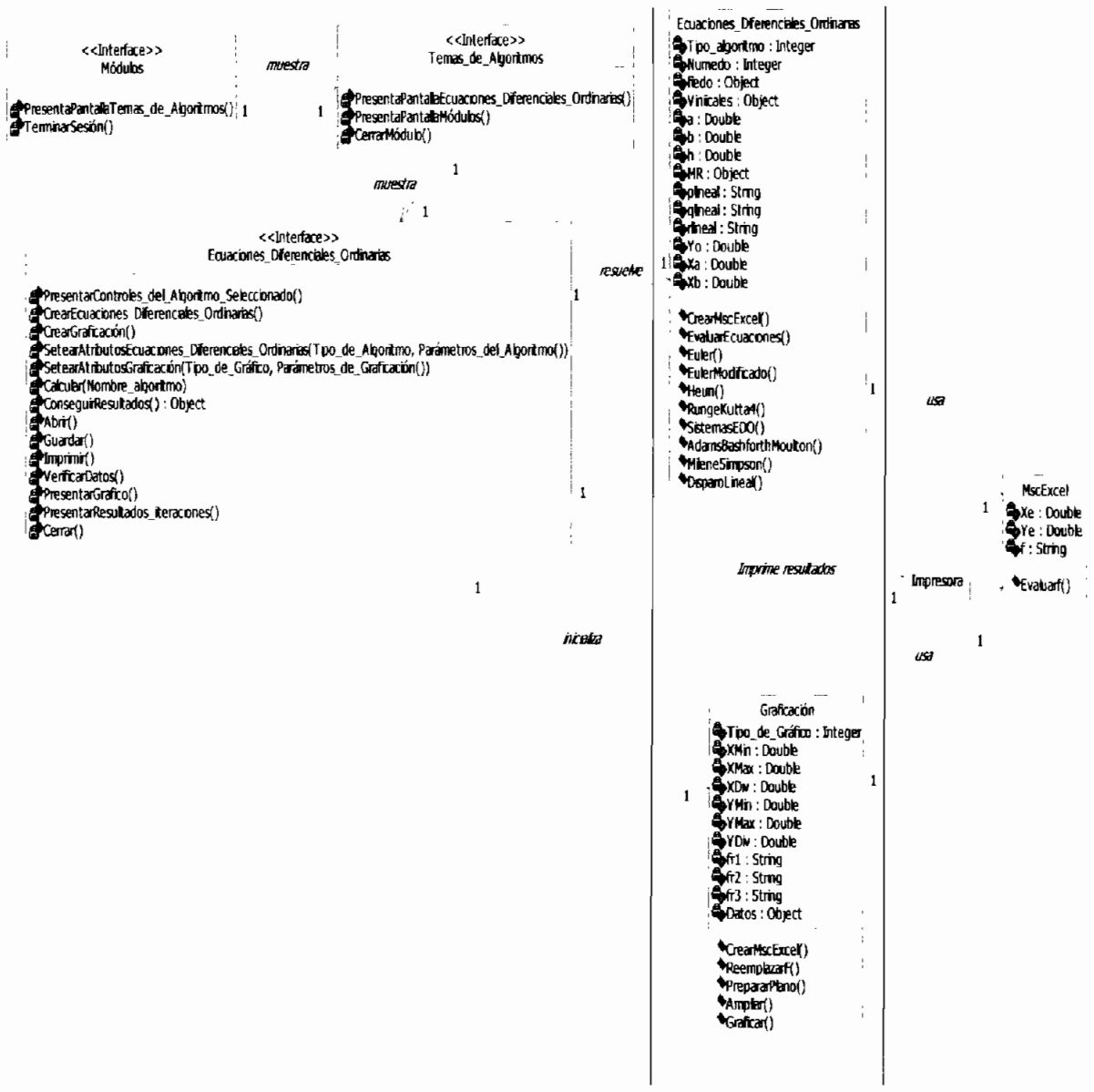


Fig. 3.182 Diagrama de clases de diseño para el caso de uso Resolver ecuaciones diferenciales ordinarias

3.5.6 DISEÑO DEL CLUSTER (6)

3.5.6.1 Resolver ecuaciones diferenciales parciales

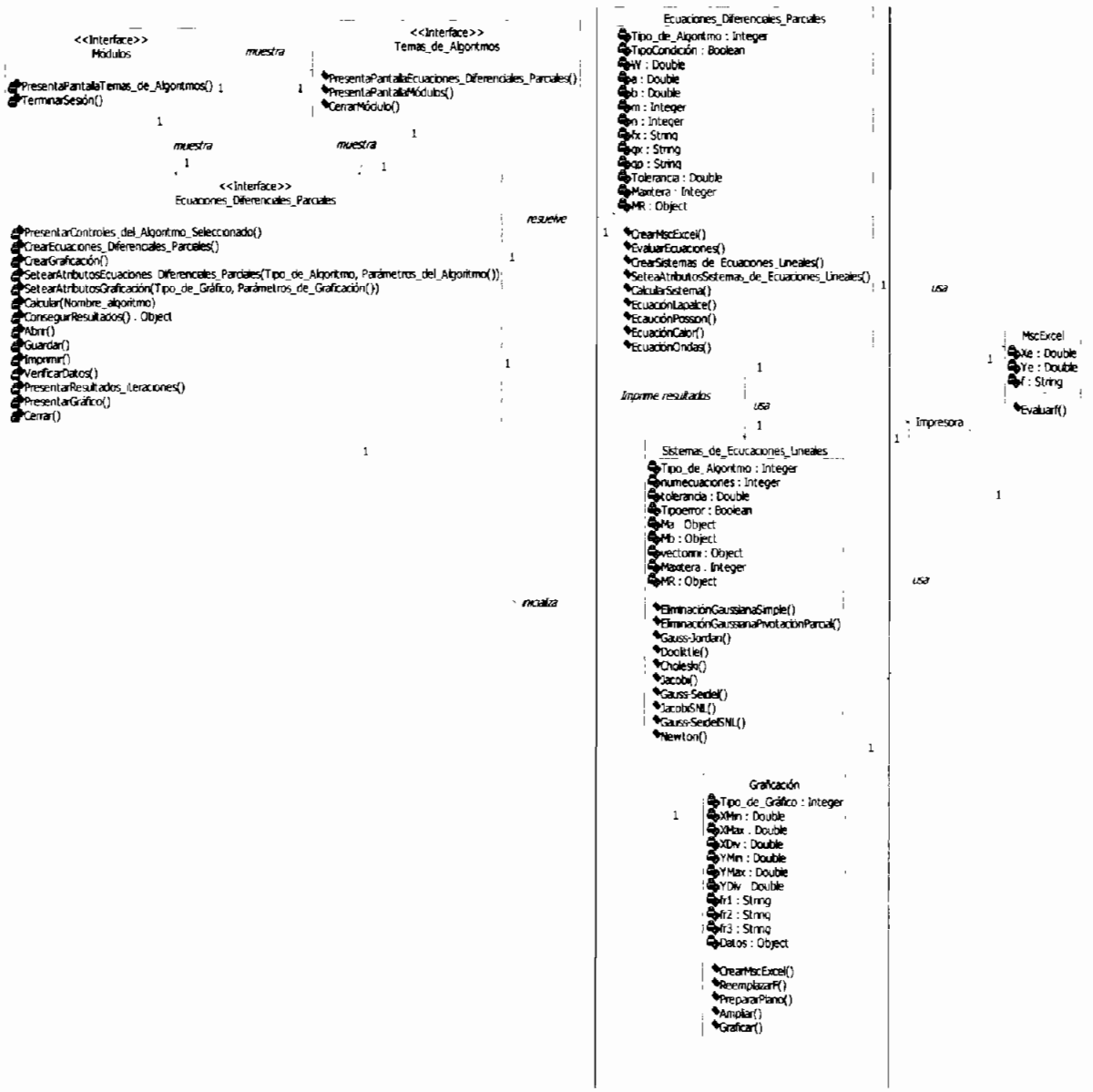


Fig. 3.183 Diagrama de clases de diseño para el caso de uso Resolver ecuaciones diferenciales parciales

3.5.6.2 Encontrar raíces de funciones no lineales

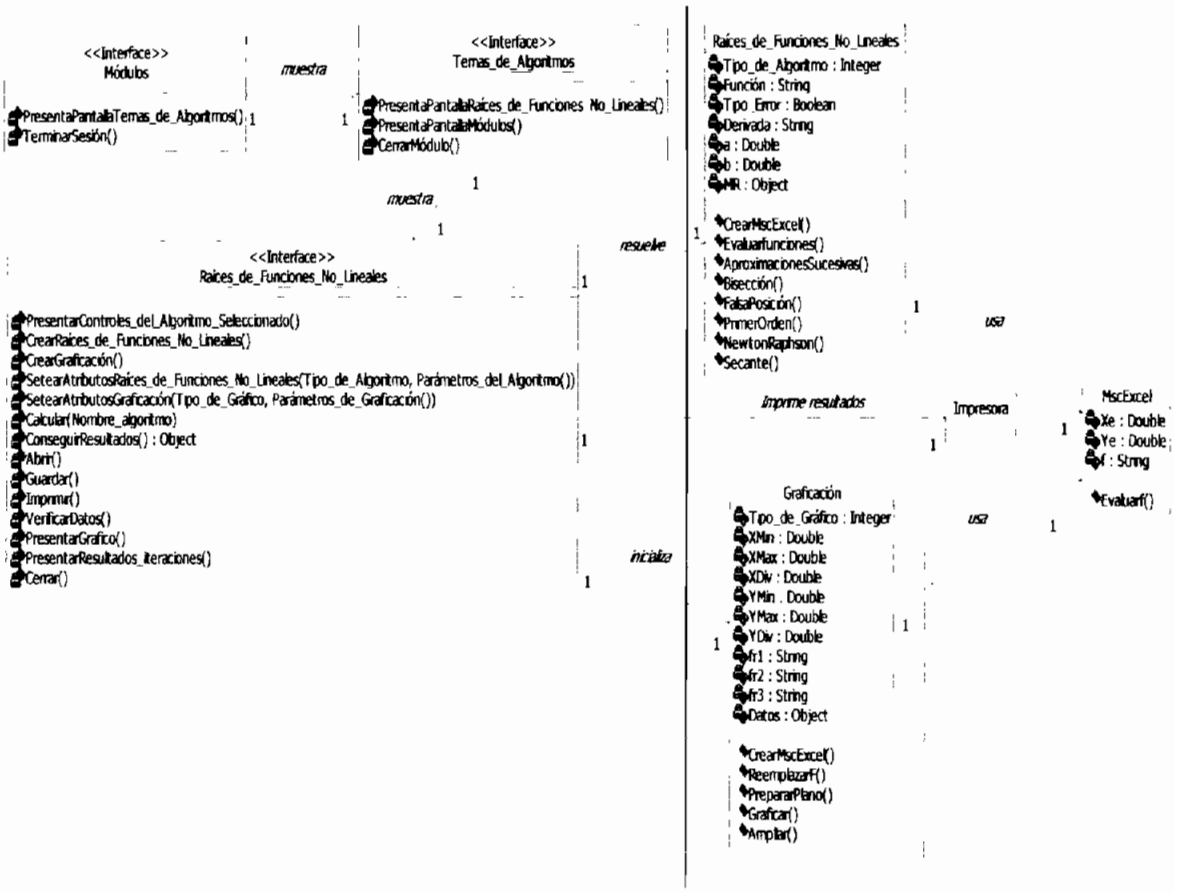


Fig. 3.184 Diagrama de clases de diseño para el caso de uso Encontrar raíces de funciones no lineales

3.5.6.3 Calcular la integral de una función

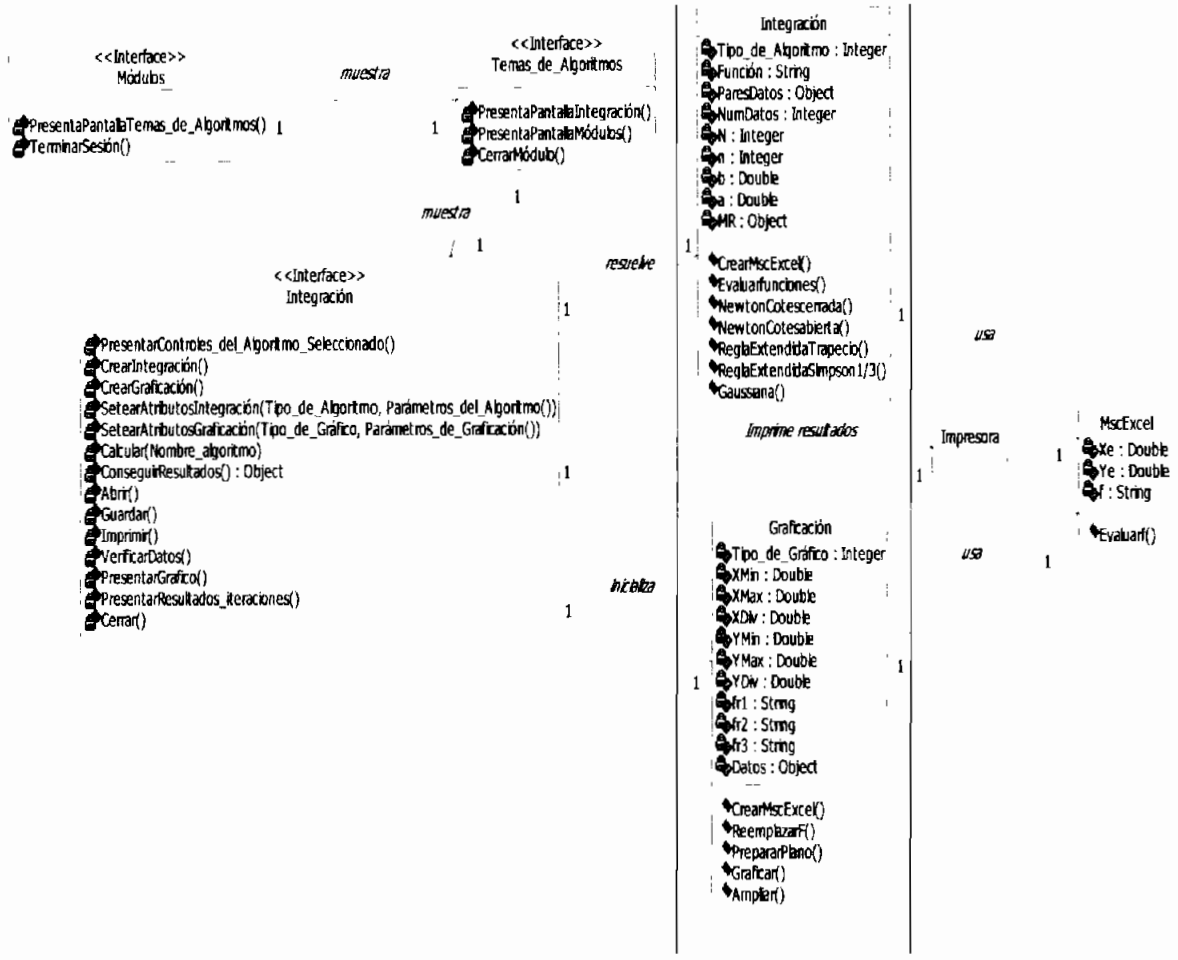


Fig. 3.185 Diagrama de clases de diseño para el caso de uso Calcular la integral de una función

3.5.6.4 Realizar una regresión polinomial

Debido a que los algoritmos del módulo Regresión Polinomial hacen uso de los algoritmos del módulo Sistemas de Ecuaciones Lineales e implementa solo un algoritmo, se ve la necesidad de implementar este algoritmo como un método del módulo Interpolación Polinomial lo que permitirá disminuir el número de interfaces, módulos y la escritura de instrucciones para ello. La Fig. 3.186 muestra el diagrama de clase de diseño para este caso de uso.

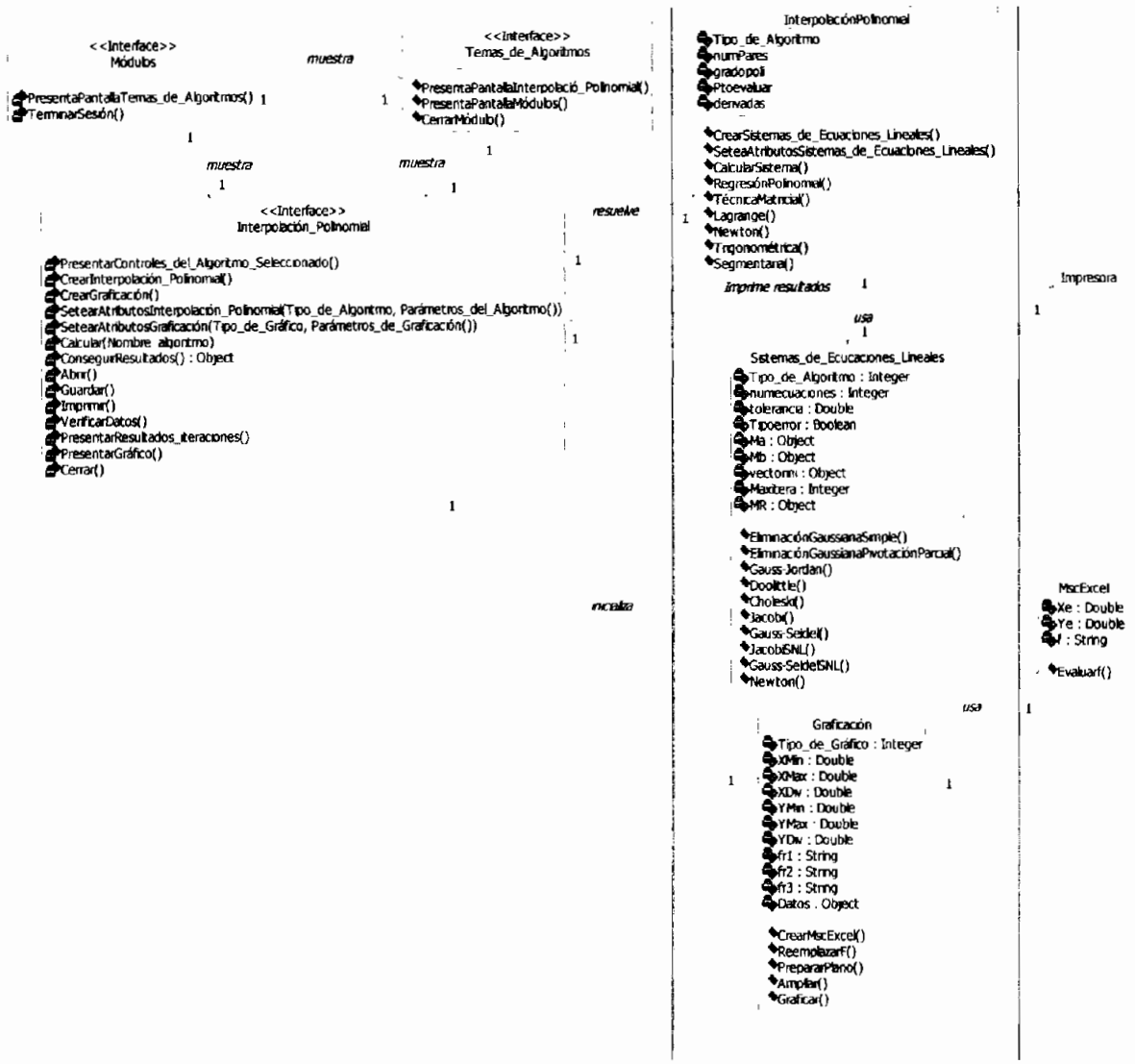


Fig. 3.186 Diagrama de clases de diseño para el caso de uso realizar una regresión polinomial

3.6 ESQUEMA DE BASE DE DATOS

3.6.1 ANÁLISIS PRELIMINAR

En los sistemas de software actuales es necesario guardar información en medios de almacenamiento persistente (Base de Datos) y el Paquete de Software Didáctico para la Enseñanza-Aprendizaje de Métodos Numéricos no es la excepción; debido a que los sistemas de software tienen objetos (objetos persistentes) que necesitan guardar información en nuestro caso especifico los objetos serán: clave, pregunta e informe de evaluación, por lo tanto es menester realizar un diseño del esquema de base de datos que soporte tal información.

Los mecanismos de almacenamiento más comunes son: base de datos orientados a objetos y base de datos relacionales.

El primero tiene la ventaja de no necesitar servicios específicos de persistencia, el segundo es el más utilizado hoy, pero tiene las desventajas de no poseer métodos para el almacenamiento de objetos y de necesitar servicios específicos de persistencia.
Un esquema de persistencia es un conjunto de clases reutilizables que prestan servicios a los objetos persistentes, en general este esquema debe traducir los objetos a registros (desmaterialización) para guardarlos en una base de datos o viceversa (materialización).

Estas clases que prestan servicios o intermediarios de base de datos se los denomina broker o agentes virtuales. Los objetos clientes deben interactuar con este agente especializado en vez de hacerlo con el objeto real, para lo cual estos agentes implementan la misma interfaz que el objeto real.

Para el diseño de nuestra base de datos, se usará una base de datos relacional por ser la más utilizada y en forma especifica Microsoft Access, que es la de más fácil acceso a los usuarios en comparación con otras como S.Q.L u Oracle. Se tomará como agente virtual al módulo de clase BaseDatosMnts que se presenta tanto en los diagramas de colaboración como en los diagramas de diseño de clases.

Se debe tomar en cuenta el grado de relación (mapeo) que existe entre una clase y su almacenamiento persistente y entre los atributos de un objeto y los campos de un registro, este grado de relación se analiza haciendo uso del patrón *Representación de objetos como tablas,* éste propone definir una tabla por objeto persistente en donde sus atributos equivalgan a una columna de la tabla. También surge la pregunta ¿Cómo acceder a varios registros que tengan relación? por ejemplo si se deseara que se presenten todos los productos que vende una empresa al ingresar el nombre de la misma, se hace entonces necesario definir la relación que éstos tuvieren e identificar cada producto que esta empresa vende con una clave única que la distinga de las demás, a esta clave se la denomina patrón identificador de objetos (IDO) el que puede ser alfanumérico, el mismo que se encuentra en la tabla de la base de datos y en el objeto persistente como atributo. La Fig. 3.187 muestra un ejemplo del objeto venta y su almacenamiento en una base de datos relacional.

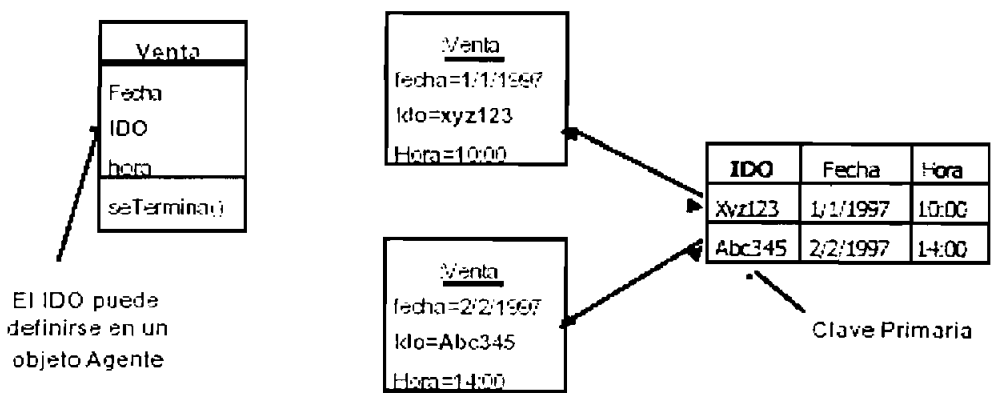


Fig. 3.187 Ejemplo del patrón identificador de objetos (IDO)

Basados en los conceptos anteriores se ha diseñado un conjunto de tablas que tienen una relación muchos a uno y por lo tanto necesitan el identificador de objetos IDO, a continuación se detalla el diseño de estas tablas y como se relacionan entre sí.

3.6.2 DISEÑO DE LA BASE DE DATOS

La implementación del software con un módulo de seguridad implica el almacenamiento de los datos de los usuarios que ingresan al sistema, al igual que los resultados de la evaluación que éste realice y de las preguntas que este contesta en la evaluación.

Por ende es necesario definir un conjunto de tablas que permitan acceder al agente virtual (BaseDatosMnts) ya sea para guardar o recuperar datos de una manera segura, cuando un objeto persistente (cliente) lo requiera.

Para el objeto Clave se crea una tabla denominada seguridad la cual tienen los siguientes campos: id, Nombre, clave (password). El IDO es el campo id que identifica de manera única a los registros de la tabla, es decir coincide con la clave principal y ésta es numérica. La Fig. muestra la tabla.

Fig. 3.188 Tabla Seguridad

Para comprender el funcionamiento de las relaciones de las tablas que se han creado para el objeto pregunta, se ha tomado como ejemplo las siguientes preguntas en las cuales la respuesta correcta esta subrayada:

Para la evaluación parcial

¿El método de Bisección encuentra una raíz aún cuando la función no sea analítica?

- V
- F

¿La raíz de la ecuación $e^x - 2 = 0$ en el intervalo [0,2] usando el algoritmo de Bisección con una tolerancia de 0.01 es?

- 0.7451
- 0.6953
- 0.7014
- Ninguna

¿La aproximación de los puntos de la tabla a una recta es?

- 0.08432+0.752X
- 0.06143+0.8214X
- 0.07143+0.8393X
- Ninguna

X	Y
1	0.5
2	2.5
3	2.0
4	4.0
5	3.5
6	6.0
7	5.5

Para la evaluación total

¿El método de Newton-Bairstow permite evaluar un polinomio en un argumento complejo?

- V
- F

¿La cuarta derivada del polinomio $P(x) = 2x^4 - 3x^2 + 3x - 4$ evaluado en x=2 es?

- 48
- 49
- 56
- Ninguna

¿ El valor en X=1.5 del polinomio interpolante de segundo grado usando la técnica matricial es?

- 0.712136
- 0.5124715
- 0.612565
- Ninguna

X	Y
1.0	0.7651977
1.3	0.6200860
1.6	0.455022

Para el objeto Pregunta se crean nueve tablas cinco para las evaluaciones parciales y cuatro para la evaluación total. Para la evaluación parcial se implementan las tablas:

- TemasdeEvalución
- Tipodepregunta
- Preguntas
- Alternativas
- Paresdedatos

La tabla TemasdeEvalución tiene dos campos: IDTema y Tema de Evaluación, donde el IDO es IDTema que es la clave principal de la tabla y corresponde al atributo Tema de Evaluación del módulo Evaluación, la que consigue el módulo pregunta antes de hacer la consulta a BaseDatosMnts.

El IDTema esta formada por siglas que representan los diferentes temas a ser evaluados así:

- RFNL: Raíces de funciones no lineales
- MP: Manipulación de polinomios
- SEL: Sistemas de ecuaciones lineales
- INT: interpolación polinomial
- INTE: Integración numérica
- DERI: Derivación numérica
- EDO: Ecuaciones diferenciales ordinarias
- EDP: Ecuaciones diferenciales parciales.

La siguiente figura muestra la tabla.

TemasdeEvaluación : Tabla

IDTema	Tema de Evaluación
DERI	Derivación numérica
EDO	Ecuaciones diferenciales ordinariasy Sistemas
EDP	Ecuaciones diferenciales parciales
INT	Interpolación polinomial
INTE	Integración numénca
MP	Manipulación de polinomios
RFNL	Raíces de funciones no lineales
SEL	Sistemas de ecuaciones lineales y no lineales

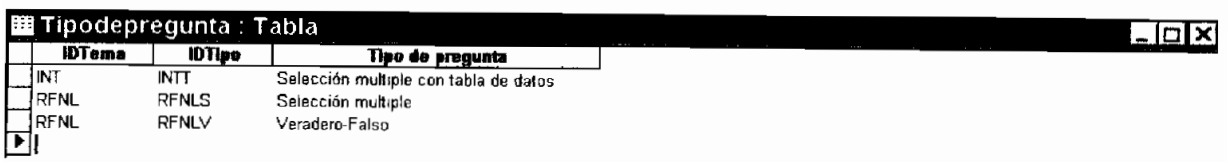
Fig. 3.189 Tabla TemasdeEvaluación

La tabla Tipodepregunta tiene tres campos: IDTema, IDTipo, Tipo de pregunta, donde el IDO es IDTema como se puede observar este campo se repite en ambas tablas, y se debe a que este debe estar en la tabla con la que se relaciona por ser una relación uno a muchos. El campo IDTipo es la clave principal de la tabla Tipodepregunta e identifica de manera unívoca a cada registro de la tabla y esta es alfanumérica; como se notara en esta tabla el IDO ya no coincide con la clave principal de la tabla ya que el IDO permite solamente relacionar información de registros en tablas diferentes.

El campo IDTipo esta formado también por siglas que representan el nuevo IDO que se usara para relacionar las tablas Tipodepregunta y Preguntas estas siglas están formadas por las mismas siglas que el campo IDTema pero con la diferencia que al final se le ha aumentado una letra más las que son V, S, T las que representan a las preguntas dicotómicas así:

- V: Verdadero-Falso
- S: Selección múltiple
- T: Selección múltiple con tabla

Selección múltiple con tabla se refiere a las preguntas que tiene que mostrar además del enunciado una tabla de valores. La figura muestra la tabla


Tipodepregunta : Tabla

IDTema	IDTipo	Tipo de pregunta
INT	INTT	Selección multiple con tabla de datos
RFNL	RFNLS	Selección multiple
RFNL	RFNLV	Veradero-Falso

Fig. 3.190 Tabla TipodePregunta

La tabla Preguntas esta formada de cuatro campos: IDTipo, IDPregunta, Número de pregunta, Respuesta. IDTipo es el IDO de la Tabla, IDPregunta es la clave principal de la tabla y es el nuevo IDO para relacionar la tabla Pregunta con las tablas Alternativas y Paresdedatos, el campo número de pregunta almacena el número de la pregunta que se mostrara el momento de la evaluación, y el campo Respuesta contiene la el número de la respuesta correcta, para esto se asume que se presentaran cuatro alternativas para las preguntas de selección múltiple a escoger así:

- Alternativa1 (numero devuelto 1)
- Alternativa2 (número devuelto 2)
- Alternativa3 (número devuelto 3)
- Ninguna (número devuelto 4)

Para las preguntas de verdadero y falso de se tiene:

- V (número devuelto 1)
- F (número devuelto 2)

Las siglas que contienen los registros del campo ID pregunta están formadas por las siglas que el campo IDTema a los cuales se les añade el número de la pregunta a ser contestada. La figura muestra la tabla

Preguntas : Tabla

IDPregunta	IDTipo	Número de Pregunta	Respuesta
INT1	INTT	1	3
RFNL2	RFNLS	2	2
RFNL3	RFNLV	3	1

Fig. 3.191Tabla Preguntas

La tabla Alternativas contiene tres campos: IDPregunta, IDAlternativa, Enunciados., El campo IDPregunta es el IDO de esta tabla, IDAlternativa es la clave principal de la tabla y el campo enunciados contiene los enunciados de la pregunta y de las posibles respuestas. IDAlternativa es numérico. IDPregunta permite devolver los enunciados que se mostrarán al usuario el momento de la evaluación. La figura. muestra la tabla.

Alternativas : Tabla

IDAlternativa	IDPregunta	Enunciados
1	INT1	¿La aproximación de los puntos de la tabla a una recta es?
2	INT1	0 08432+0 7526X
3	INT1	0 06143+0 8214X
4	INT1	0 07143+0.8393X
5	INT1	Ninguna
6	RFNL2	¿La raíz de la ecuación e^x-2=0 en [0,2] usando el algoritmo de Bisección con una tolerancia de 0.01 es?
7	RFNL2	0 7451
8	RFNL2	0 6953
9	RFNL2	0 7014
10	RFNL2	Ninguna
11	RFNL3	¿El método de Bisección encuentra la ráiz de una función aún cuando esta no sea analítica?
12	RFNL3	V
13	RFNL3	F

Fig. 3.192 Tabla Alternativas

La tabla Paresdedatos contiene cuatro campos: IDPregunta, IDTabla, X, Y, IDPregunta al igual que en la tabla anterior es el IDO y permite presentar los datos de la tabla necesaria en un enunciado de una pregunta, IDTabla es la clave principal y es numérica, X y Y son los campos que contienen los valores para la tabla del enunciado. La figura. muestra la tabla.

Paresdedatos : Tabla

IDTabla	IDPregunta	X	Y
1	INT1	1	0 5
2	INT1	2	2.5
3	INT1	3	2 0
4	INT1	4	4.0
5	INT1	5	3.5
6	INT1	6	6 0
7	INT1	7	5.5

Fig.3.193 Tabla Paresdedatos

Para la evaluación total se implementan las tablas:

- Totaltipopregunta
- Totalpreguntas
- Totalalternativas
- Totalparesdedatos

Estas tablas cumplen las mismas funciones que sus homólogos anteriores salvo que se añaden a los nombres de las tablas anteriores la palabra Total y se usan otras siglas. Las Fig. 3.194, Fig. 3.195, Fig. 3.196, y Fig.3.197 muestran las tablas.

Totaltipodepregunta : Tabla

IDTipo	Tipo de pregunta
SM	Selección múltiple
SMT	Selección múltiple con tabla
VF	Veradero - Falso

Fig. 3.194 Tabla Totaltipopregunta

Totalpreguntas : Tabla

IDPregunta	IDTipo	Número de pregunta	Respuesta
SM2	SM	2	1
SMT3	SMT	3	2
VF1	VF	1	1

Fig. 3.195 Tabla Totalpreguntas

Totalalternativas : Tabla

IDalternativa	IDPregunta	Enunciado
4	SM2	¿La cuarta derivada de P(x)=2X^4-3X^2+3X-4 evaluada en X=2 es?
5	SM2	48
6	SM2	49
7	SM2	56
8	SM2	Ninguna
9	SMT3	¿El valor en X=1.5 del polinomio interpolante de segundo grado usando la técnica matricial es?
10	SMT3	0.712136
11	SMT3	0.5124715
12	SMT3	0.612565
13	SMT3	Ninguna
1	VF1	¿El método de newton-Bairstow permite evaluar un polinomio en un argumento complejo?
2	VF1	V
3	VF1	F

Fig. 3.196 Tabla Totalalternativa

Totalparesdedatos : Tabla

IDTabla	IDPregunta	X	Y
1	SMT3	1.0	0.7651977
2	SMT3	1.3	0.6200860
3	SMT3	1.6	0.4554022

Fig.3.197 Tabla Totalparesdedatos

Con el objeto de guardar el resultados de los informes de evaluación y que éstos estén relacionados con el usuario se ha añadido a la tablas seguridad campos con siglas que identifiquen al tema de evaluación, en cuyos registros se guardarán los resultados. La Fig.3.198 muestra la tabla seguridad con este cambio.

Seguridad : Tabla

Id	NombreUsuario	Clave	RFNL	MP	SEL	Interpolación	Derivación	Integración	EDO
	Milton Tipan	DIC							
(Autonumérico)									

Fig. 3.198 Tabla Seguridad con cambio para guardar los informes de evaluación

Todo esto conlleva a la creación de un diagrama de relación de tablas la misma que se muestra en la figura siguiente.

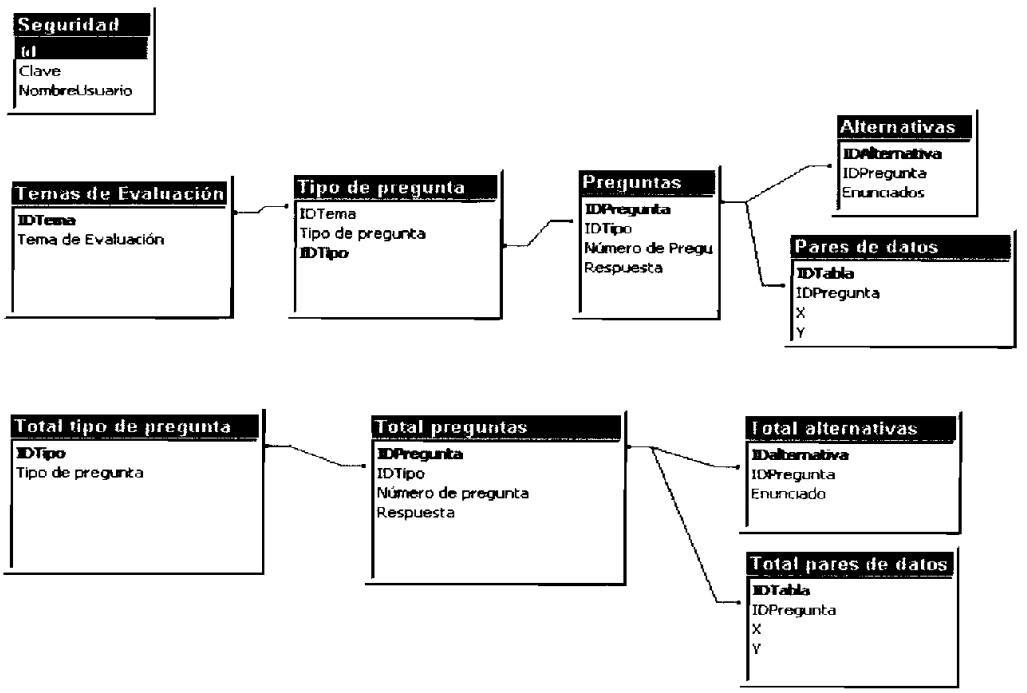


Fig. 3.199 Diagrama de relación de las Tablas de la base de datos

3.6.3 DISEÑO DE BASE DE DATOS DE TEXTO

Para el diseño de la base de datos de texto se ha optado por la creación de páginas web, ya que éstas permiten mostrar texto, gráficos, animaciones entre otros, lo que permitirá hacer más vistoso al tutorial de métodos numéricos. Además este tipo de archivos permiten una navegación mediante enlaces (Links) hacia otras páginas web dando así una gran flexibilidad al usuario en la búsqueda del tema que desee aprender.

Para la creación de estas páginas web se usará el software FrontPage Express y el enlace de las mismas mediante el programa Html Help Workshop que permitirá tener una base de texto accesible por el Paquete de Software Didáctico para la Eseñanza-Aprendizaje de Métodos Numéricos.

CAPÍTULO 4

CONSTRUCCIÓN

4.1 ACTIVIDADES DE LA FASE DE IMPLEMENTACIÓN

El capítulo anterior nos permitió obtener las clases de diseño para cada uno de los casos de uso tratados, lo que nos dan la pauta para proceder a la implementación de los componentes de diseño (la interfaz de usuario, los esquemas de base de datos, etc.) usando herramientas de desarrollo adecuado (Lenguaje de programación, gestor de base de datos, etc.)

La Fig. 4.1 nos permite observar en que parte del ciclo de desarrollo del software nos encontramos.

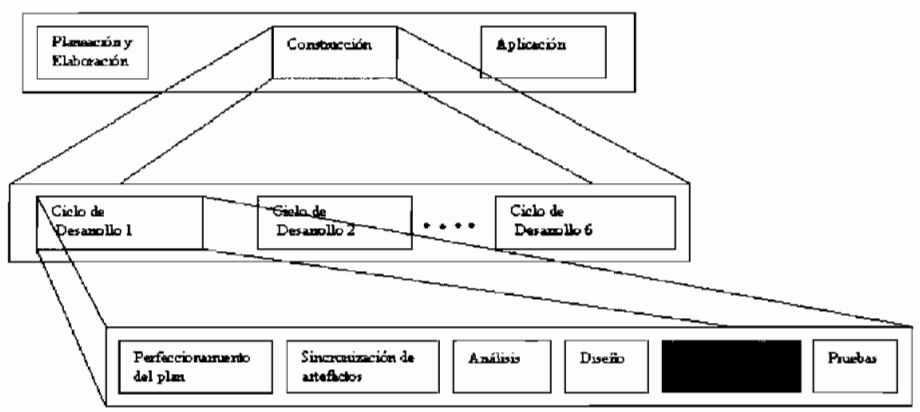


Fig. 4.1 Visión global del desarrollo de software iterativo

La implementación del software se lleva a cabo al mismo tiempo que la fase de pruebas ya que se escribe el código de la prueba de unidad antes del código a ser probado y se lo hace para todo el código de producción. Se escribe un poco de código de prueba, luego se escribe un poco de código de producción, se hace que pase las pruebas, entonces se escribe más código de prueba y así sucesivamente de manera que se cumple el desarrollo iterativo del software.

Las actividades que se realizan en la etapa de implementación son las siguientes:

1. Selección de la herramienta de desarrollo
2. Implementación de las interfaces y módulos de clases

Las bases de datos se implementarán con los módulos de clase que requieran de ellas.

4.2 SELECCIÓN DE LA HERRAMIENTA DE DESARROLLO

En el mercado existe una gran variedad de herramientas que soportan la programación orientada a objetos por lo que se analizarán las más conocidas describiendo sus características principales como se indica a continuación.

C++: Comenzando con C Bjarne Stroostrup desarrolló un efectivo lenguaje que mantiene la eficiencia de C, además añade el poder de la herencia en objetos sus características principales son:

- C++ es un lenguaje derivado de C el mayor cambio es la adición de clase y mecanismos de herencia.
- Alta facilidad para el manejo entre objetos.
- Permite la sobrecarga de operadores
- Se puede ubicar la declaración e implementación en diferentes archivos.
- Se permite clases y funciones amigas.

- Genera archivos *.exe pequeños.
- Permite la herencia simple y múltiple.
- Se puede definir clase derivadas, abstractas y paramétrizadas.
- Es posible el manejo de funciones virtuales.
- Permite el manejo de plantillas.
- Altamente transportable y con gran capacidad de aumento de librerías.
- Posee procesamiento de listas.
- Facilidad para el manejo de punteros.
- Sintaxis clara y concisa.
- Facilidad para el manejo de recursos del sistema.
- Es flexible eficiente y disponible.
- No existe un alto grado de manejo de errores.

Java: es un lenguaje de programación orientado a objetos con características especiales que lo hacen ideal para programar en el Internet, Java fue desarrollado por SunMicroSystems en 1991, sus características son:

- Modelado sobre la base de C++
- Diseñado para ser pequeño y sencillo.
- Portátil e independiente de la plataforma y sistema operativo, tanto a nivel de código fuente como de ejecutable.
- Permite unir la programación orientada a objetos con Internet.
- Robusto, elimina los problemas de memoria que existen en C y C++.
- No tiene acceso a la memoria con punteros, solo utiliza referencias.
- Realiza chequeos en tiempo de ejecución.
- Permite el manejo de Applets, Un Applets es un programa especial dinámico , iteractivo que puede ejecutarse dentro de una página Web, desplegado por un visualizador con capacidad Java.

- Se puede realizar aplicaciones estándar. La aplicación Java es un programa que no corre dentro de una página Web sino dentro del ambiente del sistema operativo, como un programa normal.
- Genera código independiente de la plataforma. Se dice que el código es independiente de la plataforma si es capaz de correr o compilarse el mismo código en distintas plataformas.

Visual C++. Es el más reciente compilador de C++ de Microsoft que da la continuad a la línea de herramientas Microsoft, para el desarrollo en Windows. El paquete de visual C++ contiene no solamente un compilador sino también bibliotecas, ejemplos y documentación necesaria para la creación de aplicaciones para Windows
Las principales características de Developer Studio y el Visual C++ son:

- Se puede integrar la herramienta de desarrollo y el compilador en Visual C++ y otras herramientas de desarrollo.
- Se incluye un editor integrado que ofrece la capacidad de mostrar saltos y el resultado de la sintaxis.
- Se usa un editor de recursos para crear un mapa de bits, iconos, cuadro de diálogo y menús.
- Un depurador integrado permite que se ejecute el programa y revisar errores.
- Contiene un sistema de ayuda en línea.
- Incluye una aplicación Wizard que sirve para crear esquemas básicos de un programa de Windows.
- Un asistente que permite añadir clases a un paquete incluyendo sus propiedades y métodos.
- OleControl Wizard se usa para crear el esqueleto de un contenedor OLE .

Además de las características del Developer Studio y de C++, Visual C++ incluye:

- Un ambiente completamente integrado que hace que sea muy fácil de crear programas.
- Toda una amplia gama de clases visuales que facilitan y hacen más amigables las aplicaciones.

- Objetos gráficos que se utilizan para la creación de la salida en un programa de Windows.
- Excepciones que se usan para saber que ha ocurrido un error.

Visual Basic: la palabra "Visual" hace referencia al método que se utiliza para crear la interfaz gráfica de usuario ((GUI). En lugar de escribir numerosas líneas de código para describir la apariencia y la ubicación de los objetos simplemente puede arrastrar y colocar objetos prefabricados en su lugar dentro de la pantalla. La palabra "Basic" hace referencia al lenguaje Basic (Beginner's All purpose Symbolic Instrucción Code) un lenguaje usado por más programadores que ningún otro en la historia de la informática. Visual Basic ha evolucionado a partir del lenguaje Basic original y ahora contiene centenares de instrucciones, funciones, palabras claves muchas de las cuales están relacionadas directamente con la interfaz gráfica de Windows.

Las características de Visual Basic son:

- Proporciona controles y clase visuales.
- La característica de acceso a datos permite la creación de aplicaciones que accedan a las bases de datos más comunes como Access o SQL
- Las capacidades de Internet facilita el acceso a documentos y aplicaciones a través de Internet desde su propia aplicación.
- Las aplicaciones terminadas son unos auténticos archivos .exe que utiliza una biblioteca de enlace dinámico DLL de tiempo de ejecución que puede distribuir con toda la aplicación.
- Se puede compilar un proyecto con código nativo para una ejecución más rápida.
- Puede combinar los controles existentes al construir uno desde cero.
- Se puede abrir múltiples proyectos desde la misma instancia de Visual Basic
- Existe la posibilidad de crear aplicaciones de documento simple , múltiple o Estilo explorador de Microsoft.
- El nuevo módulo le permite extender por programa el entorno de desarrollo y controlar proyectos, eventos, código.

- Los objetos proporcionados por sus componentes pueden desencadenar eventos que otras aplicaciones pueden tratar.
- Lista desplegable con todas sus propiedades disponibles para cada control.
- Se muestra la sintaxis de la instrucción y de funciones.
- Se puede utilizar el asistente para instalar con el fin de empaquetar sus componentes de forma específica para su despliegue en el web.
- La mayoría de los controles ahora son compatibles con operaciones de arrastre y colocar entre aplicaciones OLE.
- Proporciona funciones amigas.
- La característica implements permite que una clase admita múltiples interfaces.

4.2.1 CRITERIOS PARA LA EVALUACIÓN Y COMPARACIÓN DE LOS LENGUAJES DE PROGAMACIÓN

Existen algunos atributos que los lenguajes de programación deben cumplir para poder considerar su posterior utilización, a continuación se describen algunos de estos atributos que posteriormente servirán para la elección de la herramienta de desarrollo.

Expresividad: Es la habilidad de un lenguaje para reflejar claramente el significado deseado por el diseñador del algoritmo, en otras palabras esta característica permite que una declaración se establezca comportamiento y esfuerzo de uso así como la ayuda a programación orientada a objetos.

Ortogonalidad: Este término significa ser capaz de combinar varios rasgos de un lenguaje en todas las posibles combinaciones. por ejemplo si se posee una expresión que produce un valor y otra instrucción condicional que evalúa una expresión para obtener un valor de verdadero o falso estos dos rasgos son ortogonales , si cualquier expresión puede ser usada dentro de la instrucción condicional .

Tipo y estructura de datos: Es la habilidad de soportar una variedad de valores de datos y colecciones no elementales de ellos.

Modularidad. Esta característica tiene dos líneas; la subprogamación que quiere decir la habilidad de definir procedimientos y funciones independientes y comunicarlos vía parámetros, y la segunda el soporte a tipos definidos por el usuario.

Facilidad de verificación del programa: Un programa debería proveer herramientas que le faciliten al usuario la verificación de errores durante el desarrollo y no esperar que se compile para mostrar posibles errores.

Ambiente de programación: La estructura técnica de un lenguaje es solamente un aspecto que aumenta su uso. El poseer un apropiado ambiente de programación puede ser que aparentes deficiencias técnicas se sustituyan con eficiencia, reusabilidad y una muy buena documentación de las aplicaciones desarrolladas.

Portabilidad: Un lenguaje debe permitir transportar los programas del ambiente de desarrollo a otras máquinas que carezcan del mismo. Esto permite a los programadores "no preocuparse" por el ambiente en donde estará en ejecución la aplicación.

Eficiente: El lenguaje debe permitir una compilación y ejecución rápida sobre las máquinas en donde este se implementa.

Costos de uso: Un lenguaje debe minimizar en lo posible los costos en el desarrollo de la aplicación como: Costo de ejecución del programa(manejo eficiente de la memoria y optimización del código), costo de traslación del programa(generación del .exe, minimización del tamaño de archivos) costo de mantenimiento.

Aprendizaje: Debe ser intrínsecamente fácil de aprender y enseñar, debe contar con una documentación completa, muy conocido y usado por muchas personas.

Generalidad: Significa que el lenguaje debe ser útil en un amplio rango de aplicaciones de programación, por ejemplo no solamente poder desarrollar aplicaciones matemáticas, sino para aplicaciones en Internet, base de datos, etc.

Disponibilidad: Que tan disponible se encuentra la herramienta al usuario, no es muy aconsejable escoger una herramienta por más eficiente que sea si es muy difícil de obtenerla.

4.2.2 RESULTADO DEL PROCESO DE SELECCIÓN DE LA HERRAMIENTA DE IMPLEMENTACIÓN.

Se ha evaluado cada una de las herramientas de acuerdo a los criterios mencionados anteriormente de la siguiente manera:

0= malo

1= bueno

2= excelente

El siguiente cuadro [8] muestra los resultados del análisis :

Criterio	C++	Java	Visual C++	Visual Basic
Expresividad	2	2	2	2
Ortogonalidad	2	2	2	2
Tipo de datos	2	1	1	1
Modularidad	1	1	1	1
Verificación	1	1	1	2
Ambiente	1	1	2	2
Portabilidad	2	2	1	1
Eficiencia	1	1	1	2
Costo de uso	2	2	1	1
Aprendizaje	1	2	2	2
Generalidad	2	2	2	2
Disponibilidad	2	2	2	2
Total	19	19	18	20

Tabla 4.1 Selección de la herramienta de desarrollo

Como resultado del análisis se elige como herramienta de desarrollo a Visual Basic.

4.2.3 CONVENCIÓN DE NOMENCLATURA PARA LA IMPLEMENTACIÓN EN VISUAL BASIC

Visual Basic tiene varias versiones por lo que se opta por la más actual que es la versión 6.0, además Visual Basic contiene un sin número de instrucciones por lo tanto es necesario conocerlas y saber su sintaxis para poder desarrollar la aplicación.

El primer paso para crear una aplicación con Visual Basic es crear la interfaz, la parte visual de la aplicación con la que va a interactuar el usuario. Los formularios y controles son los elementos de desarrollo básicos que se usan para crear la interfaz; son los objetos con los que se trabaja para desarrollar la aplicación.
Los formularios son objetos que exponen las propiedades que definen su apariencia, los métodos que definen su comportamiento y los eventos que definen la forma en que interactuan con el usuario. Mediante el establecimiento de las propiedades del formulario y la escritura de código de Visual Basic para responder a sus eventos se personaliza el objeto para cubrir las necesidades de la aplicación.
Los controles son objetos que están contenidos en los objetos de formularios. Cada tipo de control tiene su propio conjunto de propiedades, métodos y eventos, que lo hacen adecuado para una finalidad determinada. Algunos de los controles que puede usar en las aplicaciones son más adecuados para escribir o mostrar texto, mientras que otros controles permiten tener acceso a otras aplicaciones y procesan los datos como si la aplicación remota formara parte del código.
Hay varios tipos de controles unos para mostrar e introducir texto, controles que muestran opciones a los usuarios, controles que muestran imágenes y gráficos, controles que se enlazan a datos, etc [19].
Los que permiten mostrar e introducir texto son:

Texto que el usuario puede modificar; por ejemplo, un campo de entrada de pedidos o un cuadro de contraseña	**TextBox** (cuadro de texto)
Texto que sólo se muestra; por ejemplo, para identificar un campo de un formulario o mostrar instrucciones al usuario	**Label** (etiqueta)

Los controles que permiten mostrar opciones a los usuarios los controles son:

Un conjunto pequeño de opciones entre las que el usuario puede elegir una o más.	**CheckBox** (casillas de verificación)
Un conjunto pequeño de opciones entre las que el usuario sólo puede elegir una.	**OptionButton** (botones de opción; use marcos si son necesarios grupos adicionales)
Una lista desplegable de opciones entre las que puede elegir el usuario.	**ListBox** (cuadro de lista)
Una lista desplegable de opciones junto con un cuadro de texto. El usuario puede elegir de la lista o escribir una opción en el cuadro de texto.	**ComboBox** (cuadro combinado)

Los que muestran imágenes y gráficos son:

Contenedor para otros controles	**Picture** (cuadro de imagen)
Métodos gráficos o de impresión.	**Picture** (cuadro de imagen)
Mostrar una imagen.	**Image** (control de imagen) o **Picture** (cuadro de imagen)
Mostrar un elemento gráfico simple	**Shape** (control de forma) o **Line** (control de línea)

Los que se enlazan a datos son:

- El control **ADO Data** (de datos) se usa para conectarse a una base de datos. Puede considerarse como una canalización entre la base de datos y los demás controles del formulario. Sus propiedades, métodos y eventos permiten explorar y manipular datos externos desde dentro de la aplicación.

- El control **DataList** es similar al control de cuadro de lista. Cuando se usa junto con un control de datos ADO, se puede llenar automáticamente con una lista de datos tomados de un campo de una base de datos externa.

- El control **DataCombo** es como una combinación del control **DataList** y un cuadro de texto. Es posible modificar el texto seleccionado en la parte de cuadro de texto; las modificaciones aparecen en la base de datos subyacente.

- El control **DataGrid** muestra datos en una cuadrícula o una tabla. Cuando se usa junto con un control de datos ADO, presenta datos completamente modificables tomados de múltiples campos de una base de datos externa.

- El control **Microsoft Hierarchical FlexGrid** es un control único para presentar múltiples vistas de datos. Se puede considerar como una combinación de una cuadrícula y un control de árbol o de esquema. En tiempo de ejecución, el usuario puede reorganizar las columnas y las filas para proporcionar diferentes vistas de los datos.

También se encuentran los botones de comando CommandButton que permiten realizar acciones al usuario al pulsar sobre estos, los formularios son las interfaces de la aplicación estos son módulos que son visibles al usuario y hay otros módulos como los de clase que no son visibles en los que se puede tener instrucciones que se compartan en la aplicación o que se acceda de otro formulario u otro módulo.

Para la implementación se toma la siguiente convención mostrada en la tabla

Control	Nomenclatura
TextBox	TxtNombre
Label	LblNombre
CheckBox	CheckNombre
OptiónButton	OptNombre
ListBox	LstNombre
ComboBox	CmbNombre
Picture	PBNombre
Shape	ShapeNombre
DataList	DLstNombre
DataCombo	DcmbNombre
DataGrid	DgNombre
Microsoft Hierarchical Flexgrid	FgNombre
CommandButton	CmdNombre
Formulario	Frm
Modulo de clase	ClasNombre/Nombre

Tabla 4.3 Nomenclatura de controles para la implementación

Así por ejemplo se tendrá para un control TextBox llamado Tolerancia el siguiente codigo: TxTTolerancia

El uso de las variables en el código es importante ya que estas representarán los atributos de los módulos de clase o variables donde se desea guardar información temporal. Los tipos de variables que se usan son:

Enteras=Integer

Presición doble=Double

Boleanas=Boolean

Variables= Variant

Las que se declaran con la instrucción: Dim Nombre As Tipo
Así por ejemplo una variable con nombre Xmin que sea de doble presición será:
Dim Xmin as Double.

Los procedimientos resultan muy útiles para condensar las tareas repetitivas o compartidas, como cálculos utilizados frecuentemente, manipulación de texto y controles, y operaciones con bases de datos.
Hay dos ventajas principales cuando se programa con procedimientos:

- Los procedimientos permiten dividir los programas en unidades lógicas discretas, cada una de las cuales se puede depurar más fácilmente que un programa entero sin procedimientos.

- Los procedimientos que se utilizan en un programa pueden actuar como bloques de construcción de otros programas, normalmente con pocas o ninguna modificación.

En Visual Basic se utilizan varios tipos de procedimientos [19]:

- Procedimientos **Sub** que no devuelven un valor.

- Procedimientos **Function** que devuelven un valor.

- Procedimientos **Property** que pueden devolver y asignar valores, así como establecer referencias a objetos.

La sintaxis de un procedimiento **Sub** es la siguiente:
[Private|Public][Static]Sub *nombreProcedimiento* (*argumentos*)
instrucciones
End Sub

La sintaxis de un procedimiento **Function** es la siguiente:

[Private|Public][Static]Function *nombreProcedimiento* (*argumentos*) [**As** *tipo*]
instrucciones
End Function

Sintaxis

[**Public** | **Private** | **Friend**] [**Static**] **Property Get** *nombre* [(*lista_argumentos*)] [**As** *tipo*]
[*instrucciones*]
[*nombre* = *expresión*]
[**Exit Property**]

Sintaxis

[**Public** | **Private** | **Friend**] [**Static**] **Property Let** *nombre* ([*lista_argumentos*,] *valor*)
[*instrucciones*]
End Property

Por ejemplo tenemos:

Procedimiento Sub

```
Public Sub RegresiónPolinomial()
Instrucciones
End sub
```

Procedimiento Function

```
Function Hipotenusa (A As Integer, B As Integer) _
As String
  Hipotenusa = Sqr(A ^ 2 + B ^ 2)
End Function
```

```
Procedimientos Property
Public Property Get Resultadointegración() As Variant
Resultadointegración = Resulintegración
End Property


Public Property Let Setearparesdatos(grado As Integer, numdatos As Integer,
Puntos() As Double)
ordendd = grado
Pares = Puntos
numpares = numdatos
End Property
```

4.3 IMPLEMENTACIÓN DE INTERFACES Y MÓDULOS DE CLASE

Para la implementación tanto de los atributos como de los procedimientos de las interfaces y módulos de clase para cada etapa de desarrollo se han tomado en consideración el diagrama de clases de diseño en lo posible ya que muchos de los procedimientos requieren una sola instrucción y no es viable la creación de un procedimiento para ello, otros se los debe dividir en varios para que desarrollen en conjunto la función del procedimiento descrito en los diagramas de clase de diseño, esto a implicado la creación de nuevos atributos e inclusive el cambio de los nombres de los procedimientos con el objeto de hacer referencia a la clase que pertenece y muchas veces la de disminuir el código por nombres largos de procedimientos o de formularios por ejemplo la clase Sistemas de Ecuaciones Lineales es un nombre muy largo para ser escrito por lo que se ha cambiado por SEL que permite una escritura más rápida y una disminución del código. Todos estos cambios se han realizado sin descuidar la función que deben cumplir cada interfaz o

módulo especificada tanto en los diagramas de colaboración como en los diagramas de clases de diseño.

4.3.1 IMPLEMENTACIÓN DEL CLUSTER (1)

Se trata el caso de uso de inicialización denominado identificar estudiante para el cual se ha planteado el boceto de interfaz de la Fig. 3.2 y el que se ha implementado en la Fig. 3.3 del capítulo 3. Esta interfaz de acuerdo al diagrama de diseño de clases debe tener los siguientes métodos: CrearUsuario(), SetearClave(), PresentarMensaje(), PresentarPantallaMódulos().

El método CrearUsuario() crea una instancia del módulo clave, luego de lo cual el método SetearClave() captura los datos de la interfaz e inicializa las variables del módulo clave; el cual crea una instancia con el módulo de servicio BaseDatosMnts para realizar la consulta con la base de datos llamada Seguridad permitiendo realizar el filtrado de datos y la validación de los mismos mediante los procedimientos CrearBasedeDatosMnts() y ValidarClave del módulo de clase Clave.

Si el resultado de la validación de la clave es correcto se procede a guardarla con el procedimiento GuardarClave() caso contrario se presenta un mensaje de clave incorrecta al usuario. Finalmente se presenta la interfaz de módulos del paquete de software llamada Frmmodulos mediante la instrucción Frmmodulos.Show

4.3.2 IMPLEMENTACIÓN DEL CLUSTER (2)

Leer un tema del contenido teórico

Para este caso de uso se optó por generar un tutorial mediante páginas web las mismas que se desarrollaron mediante el programa FrontPage Express, estas se unieron para formar una base de datos de hipertexto mediante el software HtmlHelpWorkshop el mismo que permite de una manera directa la creación de esta base de hipertexto y de los sistemas de búsqueda mediante la asignación de claves de para la identificación de las diferentes paginas y sus enlaces.

Por lo tanto las clases para este caso de uso se implementaron indirectamente, para el manejo de esta base de datos de hipertexto creada mediante el HtmlHelpWorkshop se creo en la interfaz de los módulos del paquete de software (Frmmodulos) instrucciones que permiten el manejo de la API (Interfaz de programación de aplicaciones) de Windows la que consta de funciones, mensajes, estructuras de datos, tipos de datos e instrucciones que puede usar para crear aplicaciones que se ejecuten bajo Microsoft Windows. Estas permiten el ejecutar esta base de datos desde el paquete de software.

Resolver sistemas de ecuaciones lineales

En este caso de uso se implemento el módulo de clase ClasSEL el que difiere del nombre dado en el diagrama de clase de diseño por las razones acotada anteriormente. En este se implementaron los procedimientos:

SetearSEL(): permite inicializar los atributos en el módulo para los algoritmos que manejas sistemas de ecuaciones lineales.
SetearSENL(): permite inicializar los atributos en el módulo para los algoritmos que manejas sistemas de ecuaciones no lineales.
ReturSELMR(): retorna los resultados de las iteraciones
ReturSELX(): retorna los resultados del sistema de ecuaciones
EliGaussSimple(): permite resolver sistemas de ecuaciones lineales mediante eliminación gaussiana sin pivotación.
EliGaussPivotación(): permite resolver sistemas de ecuaciones lineales mediante eliminación gaussiana con pivotación parcial escalonada de columna.

Los procedimientos siguientes permiten la solución de ecuaciones lineales y no lineales de acuerdo al nombre del procedimiento que lleva el nombre del algoritmo implementado, los procedimientos con terminación SNL son para sistemas de ecuaciones no lineales.
GaussJordan(), Dollittle(), Jacobi(), GaussSeidel(), JacobiSNL(), GaussSeidelSNL() NewtonSNL(),Cholesky().
Los siguientes procedimientos ayudan a la ejecución de los otros así:
Pivotaciónparcial(): realiza la pivotación parcial escalonada de columna
ReemplazarX(): reemplaza los valores de las diferentes variables Xi.

Para la interfaz de usuario (FrmSEL) se tiene los procedimientos siguientes:
Enviar(): permite inicializar los atributos del módulo SEL
SELpresentar(): presenta los resultados en pantalla
Titulosenfg(): pone los títulos necesarios en el control MshFlexgrid llamado Fg2

Hay procedimientos que aparecen que son de evento como el evento click y permiten realizar instrucciones de acuerdo al evento dado por ejemplo

CmdSELcalcular_Click() inicializa los atributos mediante el llamado a Enviar() y elige el algoritmo del módulo de clase SEL para el cual se desea resolver el sistema.

Para saber que algoritmo a elegido se implementa el atributo Tipo de algoritmo mediante la variable denominada tipoalg a la cual se le asigna un número entero de acuerdo al algoritmo deseado así para el algoritmo Eliminación gaussiana tipoalg =1.Para el manejo de archivos se tiene los procedimientos guardar (), abrir() e imprimir()

Graficar una función

El módulo para este caso de uso se llama Graficar el mismo que permite realizar gráficos de cualquier tipo de función y de datos este tiene los siguientes procedimientos:

Seteargrafico(): inicializa los atributos del módulo(funciones,datos,escalas)
setarpicturebox(): Asigna a la variable PB el PictureBox donde se graficará, este puede ser en el pictureBox del módulo de graficación del paquete de software o en el PictureBox que aparece como opción en el módulo de algoritmo del paquete de software
Gráfico(): Permite graficar en el PictureBox deseado mediante la llamada a los procedimientos PrepararPlano(), Graficar()
PrepararPlano(): dibuja los ejes en el PictureBox y setea las escalas para el gráfico
Graficar(): Grafica las funciones y datos usando la función ReemplazarF(X) y la llamada a la aplicación Excel para evaluar las funciones.
ReemplazarF(X): reemplaza la función en un valor X
Ampliar(): Inicializa los nuevos valores de escalas y llama a gráfico() para graficar nuevamente las funciones y datos
Desahacer(): Restablece los valores anteriores de las escalas y llama a gráfico() para graficar nuevamente las funciones y datos.

La interfaz para este caso de uso se la llamo FrmGraficando y es la mostrada en el capitulo 3 en la Fig. 3.13 la cual tiene los siguientes procedimientos:

CheckGraficandofunción_Click(): permite mostrar los controles necesarios para cuando se ha elegido graficar una función
CheckGraficandodatos_Click(): permite mostrar los controles necesarios para cuando se ha elegido graficar datos
CmdGraficandodesahacer_Click(): permite restablecer los valores anteriores de las escalas y graficar nuevamente la función y los datos
Los procedimientos PBPlano_MouseDown(), PBPlano_MouseMove(), PBPlano_MouseUp() permiten obtener los valores de las coordenadas el momento de la ampliación o mostrarlas cuando el ratón se encuentra sobre el PictureBox llamado Pbplano. Los procedimientos de manejo de archivos son similares a los de la interfaz FrmSEL.

4.3.3 IMPLEMENTACIÓN DEL CLUSTER (3)

Realizar Evaluación

En este caso de uso no se implementó el módulo pregunta ya que se usaron para acceder a la base de datos controles ADO. El control de datos usa Objetos de Datos ActiveX para crear de una manera rápida conexiones entre controles de enlace de datos y bases de datos locales o remotas. Con la ayuda de estos controles ADO se pudo enlazar de una manera rápida y sin mayor código a la base de datos Seguridad que contiene las preguntas para la evaluación y con la ayuda de los controles MshFlexgrid se muestran estas.
Esto se implemento en la interfaz llamada FrmEvaluación ya que una interfaz en Visual Basic no es más que un modulo Form el cual es visible, por lo tanto se

implemento el módulo de evaluación indirectamente. De la misma manera el módulo Informe de evaluación se implemento en la interfaz llamada FrmInformeevaluación.

Los procedimientos eventos implementados en FrmEvaluación son:
Eparcial_Click(): permite saber si se ha elegido la evaluación parcial
Etotal_Click(): permite saber si se ha elegido la evaluación Total
DlstTema_Click(): permite devolver el Id del tema elegido y presentar los tipos de preguntas para ese tema
DlstTipo_Click(): permite devolver el id del tipo de pregunta a responder y presentar los números de las preguntas a responder.
DlstNumero_Click(): permite devolver el Id del número de la pregunta a responder y presentar la pregunta
Timer1_Timer(): permite verificar el tiempo de la evaluación
Evaluar_Click(): permite evaluar las respuestas de las preguntas, guardar los resultados y presentar el informe de evaluación.

Para el informe de evaluación se tiene el procedimiento
Setearinforme(): permite inicializar los atributos (nombre del usuario, tipo de evaluación, etc.)
Imprimir_Click(): imprime el informe.

Resolver sistemas de ecuaciones no lineales

Como se vio en el capítulo tres, este fue desarrollado como parte del módulo SEL, por lo tanto la interfaz, el módulo y los procedimientos son los mismos.

4.3.4 IMPLEMENTACIÓN DEL CLUSTER (4)

Encontrar raíces de un polinomio

Para este caso de uso se implementó el módulo Manipulación de polinomios pero se lo llamo MP por las razones expuestas anteriormente el mismo que tiene los siguientes procedimientos:

AlgoritmodeHorner(): permite evaluar un polinomio y sus derivadas en un argumento real
AlgoritmodeNewtonHorner(): permite encontrar raíces reales de un polinomio de grado mayor que 3
AlgoritmodeNewtonBaristow(): permite encontrar raíces reales y/o complejas de un polinomio de grado mayor a 3
AplicacióndeNewtonBairstow(): permite evaluar un polinomio en un argumento complejo.
Pn(): permite extraer los coeficientes y exponentes de un polinomio ingresado por teclado.
Factorial(): permite encontrar el factorial de un número.

Para este caso de uso se implementó la interfaz FrmMP que dispone de los procedimientos de evento:

MPalgH_Click(): permite mostrar los controles necesarios para el algoritmo de Horner
MPalgNH_Click(): permite mostrar los controles necesarios para el algoritmo de Newton-Horner

MPalgNB_Click(): permite mostrar los controles necesarios para el algoritmo de Newton-Bairstow

MPalgANB_Click(): permite mostrar los controles necesarios para el algoritmo de Aplicación de Newton-Bairstow

CmdMPcalcular_Click(): llama al algoritmo seleccionado, para lo cual se creo la variables selección a la que se le asigna un valor numérico entero según el algoritmo elegido.

Calcular la derivada de una función

Para este caso de uso se implementó el módulo ClasDerivación el que tiene los siguientes procedimientos:

SetearDeri(): permite inicializar los atributos del módulo

Derivada(): recupera los resultados

DiferenciasCentradas(): permite seleccionar que procedimimiento de diferencias centradas se llamara de acuerdo a la derivada seleccionada

DiferenciasProgresivas(): permite seleccionar que procedimimiento de diferencias progresivas se llamara de acuerdo a la derivada seleccionada

DiferenciasRegresivas(): permite seleccionar que procedimimiento de diferencias regresivas se llamara de acuerdo a la derivada seleccionada

Centradaprimeraderi(): permite obtener la primera derivada mediante diferencias centradas

Centradasegundaderi(): permite obtener la segunda derivada mediante diferencias centradas

Progresivasprimeraderi(): permite obtener la primera derivada mediante diferencias progresivas

Progresivassegundaderi(): permite obtener la segunda derivada mediante diferencias progresivas

Regresivasprimeraderi(): permite obtener la primera derivada mediante diferencias regresivas

Regresivassegundaderi(): permite obtener la segunda derivada mediante diferencias regresivas

Feval(): permite evaluar la función mediante la llamada a al módulo de clase MscExcel.

La interfaz para este caso de uso se llama FrmDerivación y tiene los siguientes procedimientos:

SetearDerivación(): permite inicializar los atributos del móduloClasDerivación

DeriPresentar(): presenta los resultados en pantalla

TxtDeriNumincrementos_Change(): muestra el número de celdas para el ingreso de datos.

Dericentrada1_Click(),Dericentrada2_Click(),Deriprogresivas1_Click(),Deriprogresivas2_Click(),Deriregresiva1_Click(),Deriregresiva2_Click(): Presenta los controles dependiendo del algoritmo elegido usando la variables tipoalg.

OptDeriDatos_Click(): muestra los controles cuando se ha elegido derivar datos

OptDeriFunción_Click(): muestra los controles cuando se ha elegido deriva datos

CmdDeriCalcular_Click(): llama al algoritmo seleccionado

4.3.5 IMPLEMENTACIÓN DEL CLUSTER (5)

Resolver un sistema de ecuaciones diferenciales ordinarias

Para este caso de uso se implementó el módulo llamado ClasEDO en el cual también se desarrollaron los procedimientos para el caso de uso Resolver ecuaciones diferenciales ordinarias. Este módulo cuenta con los siguientes procedimientos:
SetearEdo(): permite inicializar los atributos del módulo
solución(): recupera la respuesta

Euler(), EulermodificadoyHeun(), RKF4(), AdamsBashforthMoulton(), MileneSimpson(), MileneSimpson(): permite resolver una ecuación diferencial ordinaria de primer orden mediante el algoritmo que sugiere el nombre del procedimiento
DisparoLineal(): permiten resolver una problema de valor en la frontera mediante el método de disparo lineal
SistemasEDO() : permite resolver un sistema de ecuaciones diferenciales ordinarias de primer orden usando el método de Runge-Kutta de cuarto orden-
Reemplazafedo(): Permite reemplazar los valores numéricos en la función de la ecuación diferencial ordinaria y evaluarla mediante la
'llamada al objeto MscExcel
Evaluarsistema(): Permite evaluar el sistema de EDO de n ecuaciones mediante la llamada al objeto MscExcel.

Para la interfaz de este caso de uso que se llama Frm EDO se tiene los siguientes procedimientos:
SetearEdo(): permite inicializar los atributos de ClasEDO

Edoeuler_Click(), Edomodificado_Click(), Edoheun_Click(), Edorunge_Click(), Edoadams_Click(),Edomilene_Click(),Edosistemasrunge_Click(),
Edodisparo_Click(): permiten presentar los controles para cada algoritmo de acuerdo a como lo sugiere su nombre.
CmdEdocalcular_Click(): permite llamar al procedimiento de ClasEDO elegido mediante el uso de la variable tipoalg
txtEdonum_Change(): muestra el número de celdas para el ingreso de datos.
EdoPresentar(): presenta los resultados al usuario.

Realizar una interpolación polinomial

Para este caso de uso se implementó el módulo llamado ClasInterpolación la misma que tiene los siguientes procedimientos:
Setearint(): permite inicializar los atributos del módulo
intptoevaluado(): devuelve el polinomio evaluado
intPolinomio(): devuelve el polinomio encontrado
IntpolinomioSpline(): devuelve el polinomio a trozos encontrado
Returndiferenciasdivididas(): devuelve las diferencias divididas
Setearparesdatos() inicializa la matriz de datos ingresados para la interpolación
RegresiónPolinomial(),TécnicaMatricial(),Lagrange(), Newton(), Trigonométrica(), SplineLineal(), SplineSujeta(), SplineNatural(), SplineExtrapolada(), SplineTerminaciónparabolica(), SplineCurvaturaenlosextremos(): permiten interpolar datos de acuerdo al algoritmo que sugiere su nombre
diferenciasdivididas(): permite encontrar las diferencias divididas
Calculohdu(): permite el calculo de un parámetro para la solución de la interpolación a trozos
FormadorpolinomioSpline(): forma el polinomio a trozos encontrado para presentarlo al usuario.

La interfaz de para este caso de uso se llama FrmInterpolación y tiene los siguientes procedimientos:

Intregresión_Click(),Inttécnica_Click(),Intlagrange_Click(),Intnewton_Click(),Inttrigonométrica_Click(),Intlineal_Click(),Intsujeta_Click(),Intnatural_Click(), Intextrapolada_Click(), Intparabolica_Click(), Intcurvatura_Click(): permiten mostrar los controles necesarios para cada algoritmo.
CmdIntcalcular_Click(): permite llamar al procedimiento del módulo ClasInterpolación que se ha elegido, para lo cual hace uso de la variable tipoalg
CmdIntevaluar_Click(): Permite evaluar el polinomio obtenido en el argumento elegido por el usuario
TxtIntnumpuntos_Change(): muestra el número de celdas para el ingreso de datos

4.3.6 IMPLEMENTACIÓN DEL CLUSTER (6)

Resolver ecuaciones diferenciales parciales.

Para este caso de uso se implementó el módulo ClasEDP el mismo que tiene los siguientes procedimientos:
SetearEDP(): permite inicializar los atributos del módulo
Resultados(): devuelve los resultados.
Laplace(),Poisson(),EcuaciondeOndas(),Métodoexplicito(),Métodoimplicito(), MétododeCrankNicolson(): permiten resolver una ecuación diferencial parcial mediante el algoritmo que sugiere su nombre.
Feval(): permite evaluar las condiciones de frontera en X
Fevalxy(): permite evaluar las condiciones de frontera en X e Y
Fevaly(): permite evaluar las condiciones de frontera en Y

La interfaz que se implemento se llama FrmEDP la cual dispone de los siguientes procedimientos:

Edplaplace_Click() ,Edppoisson_Click(),Edpcalor_Click(),edpondas_Click(): permiten mostrar los controles necesarios para el algoritmo que sugiere su nombre

CmdEdpcalcular_Click(): Llama al procedmiento del módulo ClasEDP que ha elegido el usuario, para lo cual usa la variable tipoalg.

EnviarEdp(): inicializa los atributos del módulo ClasEDP

Recuperarresultados(): recobra los resultados del módulo ClasEDP

Presentarresultados(): presenta los resultados.

Encontrar raíces de funciones no lineales

Para este caso de uso se ha implementado el módulo llamado RFNL el que tiene los siguientes procedimientos:

SeteraRfnl(): permite inicializar los atributos del módulo

regresarraíz(): permite obtener el resultado de la raíz

regresariteraciones(): permite obtener los resultados de las iteraciones

AproximacionesSucesivas(),Bisección(),FalsePosición(),Primerorden(), NewtonRaphson(),Secante(): permite encontrar la raíz de una función no lineal mediante el algoritmo que sugiere su nombre

Fevalx(): evalúa la función en X

Gevalx(): evalúa la primera derivada en X

La interfaz para este caso de uso se llama FrmRfnl la que dispone de los siguientes algoritmos:

Enviar(): inicializa los atributos del módulo RFNL

Rfnlaproxinmaciones_Click(),Rfnlbisección_Click()Rfnlfalsaposición_Click()Rfnlprimerorde_Click()Rfnlraphson_Click()Rfnlsecante_Click(): permite mostrar los controles necesarios para el algoritmo que sugiere su nombre

CmdRfnlcalcular_Click(): llama al procedimiento del módulo RFNL que ha elegido el usuario para lo cual hace uso de la variable tipoalg
CmdRfnliteraciones_Click(): muestra las iteraciones en pantalla
Presentar(): presenta las los resultados

Calcular la integral de una función

Para este caso de uso se ha implementado el módulo ClasIntegración el mismo que tiene los siguientes procedimientos:
Setearintegra(): permite inicializar los atributos del módulo
Resultadointegración(): devuelve los resultados
NewtonCotes(): permite elegir entre la integración de Newton-Cotes para función o datos
NewtonCotesdatos(): permite calcular la integral de datos mediante las fórmulas de Newton-Cotes.
NewtonCotesfunción(): permite calcular la integral de una función mediante las fórmulas de Newton-Cotes.
IntegraciónGaussiana(), ReglaExtendidaTrapecio(), ReglaExtendidaSimpson(): estas Permiten calcular la integral de una función mediante los algoritmos que sugiere su nombre
Feval(): permite evaluar la función para el cálculo de la integral mediante la ayuda del módulo MscExcel.

La interfaz para este caso de uso se llama Integración y tiene los procedimientos siguientes:
SetearIntegración(): inicializa los atributos del módulo ClasIntegración
integracerrada_Click(),Integratrapecio_Click(),Integrasimpson_Click(),
Integragaussiana_Click():presentan los controles necesarios para el algoritmo que sugiere su nombre
OptIntegradatos_Click(): presentan los controles necesarios la integración de datos

OptIntegrafunción_Click(): presentan los controles necesarios para la integración de funciones
CmdIntegracalcular_Click(): llama al procedimiento del modulo ClasIntegración que ha elegido el usuario con el uso de la variable tipoalg
IntegraPresentar(): presenta los resultados

Realizar una regresión polinomial

Como se indico en el capítulo 3 este caso de uso se trata en el ciclo de desarrollo 5 ya que el algoritmo de este se implementa en el módulo ClasInterpolación de modo que los procedimientos para este son los mismos que el del módulo al igual que la interfaz FrmInterpolación.

Todas las interfaces(Formularios) son de tipo MDIChild es decir están contenidas dentro de un formulario mayor MDI el cual se llama FrmPrincipalalgoritmos que presenta a todas las interfaces del módulo algoritmos de la aplicación.

Cada Interfaz del módulo de algoritmos puede acceder al módulo de graficación mediante la opción graficar que se ha implementado en el menú de cada interfaz, esta presenta al formulario llamado Frmescalas en el cual setean los parámetros de graficación y cuyo gráfico se presenta en el formulario llamado FrmGráfico.

Se implemento el módulo MscExcel el mismo que permite evaluar una función en la aplicación Excel ya sea pasando la función y el punto a evaluar o pasando la función directamente ya reemplazada. Este consta de los siguientes procedimientos:
PtX(): permite inicializar el punto en donde se evaluara la función
Fa(): permite inicializar la función a evaluar (la función a evaluar es pasada)

Tipox: Permite inicializar el tipo de evaluación si es true se evalúa directamente si e false se reemplaza la variable primero.

ReemplazarF(X): permite reemplazar el valor en la variable de la función pasada, solamente en la variable X

Fdy(): devuelve el resultado de la evaluación

CAPÍTULO 5

PRUEBAS

5.1 ACTIVIDADES DE LA FASE DE PRUEBAS

En el capítulo anterior se implementó la aplicación usando la herramienta de desarrollo Visual Basic 6.0, la misma que permitió incorporar todas las funciones que los requerimientos del Paquete de Software Didáctico para la Enseñanza-Aprendizaje estipulan. En esta fase se pretende verificar si lo implementado cumple con los requerimientos y si la aplicación tiene errores.

La Fig. 5.1 nos permite observar en que parte del ciclo de desarrollo del software nos encontramos

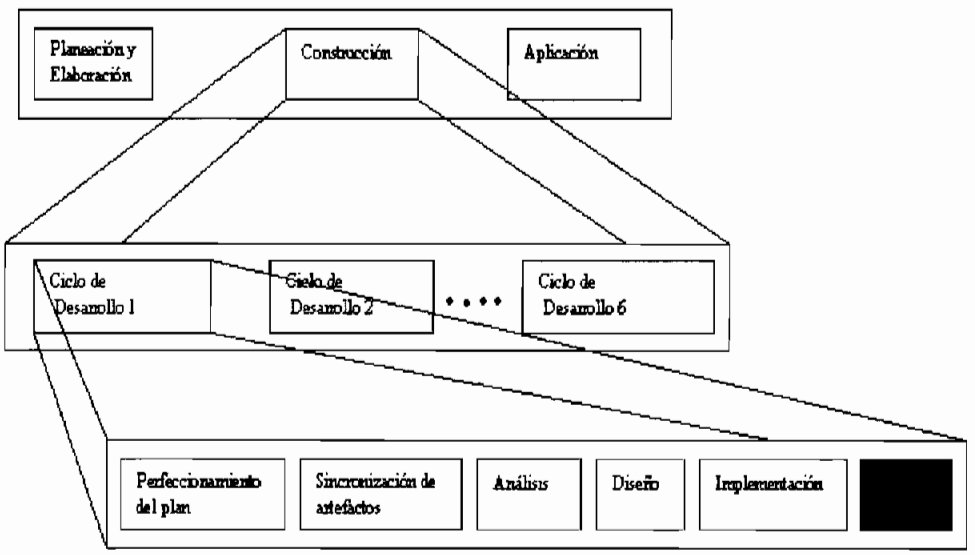


Fig. 5.1 Visión global del desarrollo de software iterativo

La prueba del software es un elemento crítico para la garantía de calidad del software, y representa un último repaso de las especificaciones, del diseño y la codificación.

Durante las fases anteriores, intenta construir el software partiendo de un concepto abstracto y llegando a una implementación tangible.

En esta etapa se crea una serie de casos de prueba que intentan "demoler" el software que ha sido construido.

La gente que desarrolla software es por naturaleza constructiva. La prueba requiere que se descarten ideas preconcebidas sobre la corrección del software que se acaba de desarrollar y superar cualquier conflicto de intereses que aparezca cuando se descubran errores.

Tradicionalmente existe una concepción incorrecta del término "probar", tal como:

- Probar es demostrar que no hay errores presentes en un programa.
- Probar es mostrar que el programa realiza correctamente las funciones esperadas.

Estas definiciones describen casi todo lo contrario de lo que se debe considerar como prueba.

La concepción tradicional ha sido considerar que un caso de prueba que NO descubre errores es exitoso. Sin embargo, dado que no descubre error alguno, la prueba constituye un desperdicio de tiempo y dinero, por lo que no parece adecuado llamarla exitosa.

Bajo este punto de vista la mejor definición de prueba del software es:

Proceso de ejecutar un programa con el fin de encontrar errores.

Esta última definición nos plantea un cambio de concepción con respecto a lo que se considera prueba exitosa y no exitosa.

No hay que comenzar la prueba del software pensando que el programa está libre de errores, conviene comenzar suponiendo que el programa tiene errores (lo que seguramente es verdad) y tratar de encontrar la mayor cantidad posible de éstos.

Los objetivos de las pruebas son:

- La prueba es el proceso de ejecución de un programa con la intención de *descubrir un error.*
- Un *buen caso de prueba* es aquel que tiene una alta probabilidad de mostrar un error no descubierto hasta entonces.
- Una prueba tiene *éxito si descubre un error no detectado hasta entonces.*

Lo primordial es diseñar pruebas que sistemáticamente detecten diferentes clases de errores, haciéndolo con la menor cantidad de tiempo y esfuerzo. Sin embargo, hay algo que no puede hacer la prueba:

La prueba no puede asegurar la ausencia de defectos; sólo puede demostrar que existen defectos en el software.

Las estrategias para llevar a cabo las pruebas son básicamente:

- Pruebas de unidad
- Pruebas de Sistema

Y poseen las siguientes características:

- La prueba comienza a nivel de un módulo, y se avanza hacia fuera hasta lograr la integración completa del sistema.
- En distintos puntos son necesarios a la vez distintas técnicas de prueba.

- La prueba y la depuración son actividades diferentes. Independientemente de la estrategia de prueba que se este utilizando, una vez terminada la misma, si se encontraron errores serán depurados y se volverá a realizar la prueba (retesting).

Cualquier producto de ingeniería puede ser probado de dos formas:

1. Conociendo la función especifica para la que fue diseñado el producto se pueden llevar a cabo pruebas que demuestren que cada función es completamente operativa (*prueba de **caja negra***).
2. Conociendo el funcionamiento del producto se pueden desarrollar pruebas que aseguren que la operación interna se ajusta a las especificaciones y que todos los componentes internos se han comprobado en forma adecuada (*prueba de **caja blanca***).

A continuación se diagrama el proceso general de prueba de software:

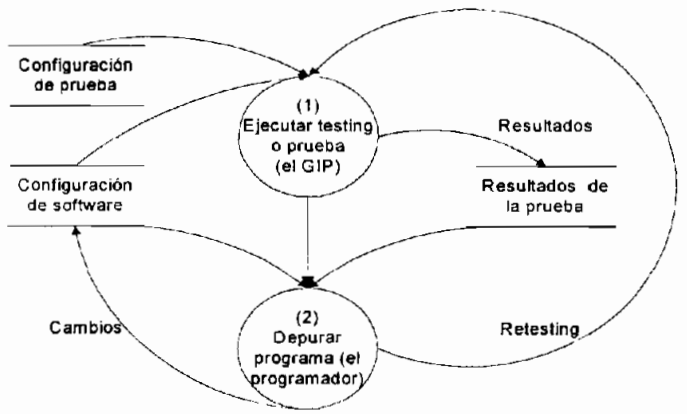


Fig. 5.2 Proceso general de prueba de software

Se lleva a cabo la prueba y se comparan los resultados de la prueba con lo esperado; si se descubren errores, estos serán solucionados durante la depuración o debugging. A medida que se van recopilando y evaluando los resultados de la prueba, comienza a vislumbrarse una medida cualitativa de la calidad y fiabilidad del software.

Si el funcionamiento parece ser correcto y los errores encontrados son fácilmente corregibles, se puede sacar una de dos conclusiones:

- La calidad y la *fiabilidad* del software son *aceptables.*
- Las *pruebas* son *inadecuadas* para encontrar errores serios.

Dado que la prueba depende de la ejecución controlada del código debemos proveer valores para que aquel se ejecute con éstos. Cada *conjunto de datos* creados para la ejecución del código, junto con el *resultado esperado* de la correcta ejecución, se denomina *caso de prueba*.

Datos + Resultado Esperado = Caso De Prueba

Para la realización de casos de prueba es necesario la creación de un documento que permita conocer que caso de prueba se ejecuta, los datos ingresados, el resultado esperado y algún comentario al respecto, por lo que para nuestro caso se tiene la siguiente planilla:

CASO DE PRUEBA:
DATOS DE PRUEBA:
RESULTADOS ESPERADOS:
RESULTADOS OBTENIDOS:
COMENTARIO:

5.2 PRUEBAS DE UNIDAD

Es el proceso de verificación de la menor unidad de software: el *módulo*. Se probarán:

- La interfaz
- Las estructuras de datos locales
- Las condiciones límite
- Los caminos independientes
- Los caminos de manejo de errores

Se aplican las técnicas de caja negra y caja blanca.

La prueba de unidad comienza cuando se han terminado de escribir y corregir las sintaxis del código fuente correspondiente al módulo. Dado que el módulo no es un programa independiente es necesario crear cierto entorno para que pueda ejecutarse.

5.2.1 PRUEBAS DEL CLUSTER (1)

CASO DE PRUEBA:	Creación de la instancia BaseDatosMnts
DATOS DE PRUEBA:	Se envia "Milton Tipán" para consulta
RESULTADOS ESPERADOS:	"Milton Tipán"
RESULTADOS OBTENIDOS:	"Milton Tipán"
COMENTARIO:	Se ha creado correctamente la clase de servicio BaseDatosMnts, ya que devuelve correctamente la Cadena "Milton Tipán"

CASO DE PRUEBA:	Escritura en la Base de Datos Seguridad
DATOS DE PRUEBA:	Sé envía "Milton Tipán" para escritura
RESULTADOS ESPERADOS:	"Milton Tipán" en la base de datos Seguridad
RESULTADOS OBTENIDOS:	"Milton Tipán" en la base de datos Seguridad
COMENTARIO:	Se ha escrito correctamente en la base de datos

CASO DE PRUEBA:	Filtrado para la Base de Datos Seguridad
DATOS DE PRUEBA:	La Base de Datos Seguridad tiene "Milton Tipán" "Milton Simbaña", " Simbaña Tipán"; se envía "Lopez Pedro"
RESULTADOS ESPERADOS:	0 Registros
RESULTADOS OBTENIDOS:	0 Registros
COMENTARIO:	El servicio de filtrado opera correctamente

CASO DE PRUEBA:	Filtrado para la Base de Datos Seguridad
DATOS DE PRUEBA:	La Base de Datos Seguridad tiene "Milton Tipán" "Milton Simbaña", " Simbaña Tipán"; se envía "Milton Tipán"
RESULTADOS ESPERADOS:	1 Registro
RESULTADOS OBTENIDOS:	1 Registro
COMENTARIO:	El servicio de filtrado opera correctamente

CASO DE PRUEBA:	Validación de Clave usuario nuevo
DATOS DE PRUEBA:	La Base de Datos Seguridad tiene "Milton Tipán" "DIC", se envía "Milton Tipán" "DIC"
RESULTADOS ESPERADOS:	No se escribe en la base de datos
RESULTADOS OBTENIDOS:	Se escribe en la base de datos
COMENTARIO:	El servicio de validación falla; se corrige para que no duplique la información

CASO DE PRUEBA:	Validación de Clave usuario nuevo
DATOS DE PRUEBA:	La Base de Datos Seguridad tiene "Milton Tipán" "DIC", se envía "Milton Tipán" "DIC"
RESULTADOS ESPERADOS:	No se escribe en la base de datos
RESULTADOS OBTENIDOS:	Se escribe en la base de datos
COMENTARIO:	El servicio de validación falla; se corrige para que no duplique la información

CASO DE PRUEBA:	Validación de Clave usuario antiguo
DATOS DE PRUEBA:	La Base de Datos Seguridad tiene "Milton Tipán" "DIC", se envía "Milton Tipán" "DIC"
RESULTADOS ESPERADOS:	bandera=1
RESULTADOS OBTENIDOS:	bandera=1
COMENTARIO:	El servicio de validación de usuario antiguo es Correcto y se implementa en lugar de bandera la instrucción para que muestre el formulario de módulos

CASO DE PRUEBA:	Cerrar conexión con la base de datos
DATOS DE PRUEBA:	La Base de Datos Seguridad tiene "Milton Tipán" "DIC", se envía "Milton Tipán" "DIC"
RESULTADOS ESPERADOS:	error, no se puede abrir la base de datos
RESULTADOS OBTENIDOS:	error, no se puede abrir la base de datos
COMENTARIO:	El servicio de cierre de conexión es correcto.

5.2.2 PRUEBAS DEL CLUSTER (2)

Desplegar tema de contenido teórico

CASO DE PRUEBA:	Móstrar tutorial
DATOS DE PRUEBA:	Se envía HelpContextID= 1000
RESULTADOS ESPERADOS:	Presentación del tutorial
RESULTADOS OBTENIDOS:	No se presenta el tutorial
COMENTARIO:	El archivo de cabecera Ayuda.h para el HtmlHelp Work Shop no indexa correctamente a los tópicos del tutorial. Se corrige el archivo Ayuda.h.

Resolver sistemas de ecuaciones lineales

CASO DE PRUEBA:	Procedimiento EliGaussSimple()
DATOS DE PRUEBA:	$\begin{matrix} 4 & 1 & 2 & 9 \\ 2 & 4 & -1 & -5 \\ 1 & 1 & -3 & -9 \end{matrix}$
RESULTADOS ESPERADOS:	2.9,-1.1,1.1
RESULTADOS OBTENIDOS:	2.9,-1.1,1.1
COMENTARIO:	El procedimiento funciona correctamente encontrando la solución del sistema.

CASO DE PRUEBA:	Procedimiento EliGaussPivotación()
DATOS DE PRUEBA:	$30 \quad 591400 \quad 591700$ $5.291 \quad -6.130 \quad 46.78$
RESULTADOS ESPERADOS:	1,9.96
RESULTADOS OBTENIDOS:	3
COMENTARIO:	El procedimiento Pivotaciónparcial no funciona correctamente, el pivoteo de columna no lo toma todas las columnas. Se soluciona el problema redimensionando las estructuras de repetición.

CASO DE PRUEBA:	Procedimiento GaussJordan()
DATOS DE PRUEBA:	$4 \quad 1 \quad 2 \quad 9$ $2 \quad 4 \quad -1 \quad -5$ $1 \quad 1 \quad -3 \quad -9$
RESULTADOS ESPERADOS:	1,-1,2.9
RESULTADOS OBTENIDOS:	1,-1,2.9
COMENTARIO:	El procedimiento funciona correctamente encontrando la solución del sistema.

CASO DE PRUEBA:	Procedimiento Doolittle()
DATOS DE PRUEBA:	$4 \quad 1 \quad 2 \quad 9$ $2 \quad 4 \quad -1 \quad -5$ $1 \quad 1 \quad -3 \quad -9$
RESULTADOS ESPERADOS:	1,-1,3
RESULTADOS OBTENIDOS:	1,-1,3
COMENTARIO:	El procedimiento funciona correctamente encontrando la solución del sistema.

CASO DE PRUEBA: Procedimiento Choleski()

DATOS DE PRUEBA:

$$\begin{matrix} 4 & 1 & 2 & 9 \\ 2 & 4 & -1 & -5 \\ 1 & 1 & -3 & -9 \end{matrix}$$

RESULTADOS ESPERADOS: 1,-1,3

RESULTADOS OBTENIDOS: 1,-1,6

COMENTARIO: El procedimiento no funciona correctamente
La matriz transpuesta fue formada correctamente se
se soluciona el problema mediante el cambio
de los valores limites de la matriz.

CASO DE PRUEBA: Procedimiento Jacobi()

DATOS DE PRUEBA:

$$\begin{matrix} 10 & -1 & 2 & 0 & 6 \\ -1 & 11 & -1 & 3 & 25 \\ 2 & -1 & 10 & -1 & 11 \\ 0 & 3 & -1 & 8 & 15 \end{matrix}$$

RESULTADOS ESPERADOS: Primera iteración:0.6, 2.2727, -1.1,1.875

RESULTADOS OBTENIDOS: Primera iteración:0.6, 2.2727, -1.1,1.875

COMENTARIO: El procedimiento funciona correctamente

CASO DE PRUEBA: Procedimiento GaussSeidel()

DATOS DE PRUEBA:

$$\begin{matrix} 10 & -1 & 2 & 0 & 6 \\ -1 & 11 & -1 & 3 & 25 \\ 2 & -1 & 10 & -1 & 11 \\ 0 & 3 & -1 & 8 & 15 \end{matrix}$$

RESULTADOS ESPERADOS: Primera iteración:0.6, 2.3272, -0.9873, 0.8789

RESULTADOS OBTENIDOS: Primera iteración:0.6, 2.3272, -0.9873, 0.8789

COMENTARIO: El procedimiento funciona correctamente

CASO DE PRUEBA: Procedimiento JacobiSNL()

DATOS DE PRUEBA:

$$\frac{1}{3}\cos(X_2 X_3) + 1/6$$

$$\frac{1}{9}\left[X_1^{\,2} + SenX_3 + 1.06\right]$$

$$-\frac{1}{20}e^{-X1-X2} - \frac{10\pi - 3}{20}$$

RESULTADOS ESPERADOS: Primera iteración:0.49999593, 0.00002557, -0.52336331

RESULTADOS OBTENIDOS: Primera iteración:0.49999593, 0.00002557, -0.52336331

COMENTARIO: El procedimiento funciona correctamente

CASO DE PRUEBA: Procedimiento GaussSeidelSNL()

DATOS DE PRUEBA:

$$\frac{1}{3}\cos(X_2 X_3) + 1/6$$

$$\frac{1}{9}\left[X_1^{\,2} + SenX_3 + 1.06\right]$$

$$-\frac{1}{20}e^{-X1-X2} - \frac{10\pi - 3}{20}$$

RESULTADOS ESPERADOS: Primera iteración:0.49998333, 0.02222979, -0.52304613

RESULTADOS OBTENIDOS: Primera iteración:0.49995593, 0.00002557, -0.52336331

COMENTARIO: El procedimiento no funciona correctamente
Función evaluada incorrectamente. Se corrige

CASO DE PRUEBA: Procedimiento NewtonSNL()

DATOS DE PRUEBA:

$$\frac{1}{3}\cos(X_2 X_3) + 1/6$$

$$\frac{1}{9}\left[X_1^{\,2} + SenX_3 + 1.06\right]$$

$$-\frac{1}{20}e^{-X1-X2} - \frac{10\pi - 3}{20}$$

RESULTADOS ESPERADOS: Primera iteración:0.49998333, 0.02222979, -0.52304613

RESULTADOS OBTENIDOS: Primera iteración:0.49995593, 0.00002557, -0.52336331

COMENTARIO: El procedimiento funciona correctamente

CASO DE PRUEBA: Inicializar atributos setearSEL()

DATOS DE PRUEBA:

n=3

T=0.001

Itera=20

Ea= true

A y b= igual al a matriz ampliada del procedimiento Jacobi pero separada.

RESULTADOS ESPERADOS: numecuaciones = n

tolerancia = T

Maxitera = itera

Tipoerror = Ea

Ma = A

Mb = b

RESULTADOS OBTENIDOS: numecuaciones = n
tolerancia = T
Maxitera = itera
Tipoerror = Ea
Ma = A
Mb = b

COMENTARIO: procedimiento Property Let correcto.

CASO DE PRUEBA: Recuperación de resultados ReturSELMR()

DATOS DE PRUEBA:

RESULTADOS ESPERADOS: Primera iteración:0.6, 2.3272, -0.9873, 0.8789

RESULTADOS OBTENIDOS: Primera iteración:0.6, 2.3272, -0.9873, 0.8789

COMENTARIO: procedimiento Property Get correcto.

Graficar una función

CASO DE PRUEBA: Procedimiento Graficar()

DATOS DE PRUEBA: $f(x) = x^2 + 4x - 1$; x=2.5

RESULTADOS ESPERADOS: 15.25

RESULTADOS OBTENIDOS: Se obtiene error

COMENTARIO: El procedimiento no funciona correctamente, el Ingreso a al aplicación MsExcel, no es valido. Se corrige.

CASO DE PRUEBA: Procedimiento PrepararPlano()

DATOS DE PRUEBA: Xmin=-5,Xmax=5;Xdiv=1;Ymin=-5;Ymax=5;Ydiv=1

RESULTADOS ESPERADOS: Se dibujan los ejes coordenados

RESULTADOS OBTENIDOS: Ejes coordenados dibujados
COMENTARIO: El procedimiento funciona correctamente,

CASO DE PRUEBA: Procedimiento Gráfico()
DATOS DE PRUEBA: Xmin=-5,Xmax=5;Xdiv=1;Ymin=-5;Ymax=5;Ydiv=1
$f(x) = \operatorname{sen} x$; puntos (2,-3);(5,4)
RESULTADOS ESPERADOS: se dibuja la función y los datos
RESULTADOS OBTENIDOS: se dibuja la función y los datos
COMENTARIO: El procedimiento funciona correctamente,

CASO DE PRUEBA: Procedimiento ampliar
DATOS DE PRUEBA: Xmin=-5,Xmax=5;Xdiv=1;Ymin=-5;Ymax=5;Ydiv=1
$f(x) = \operatorname{sen} x$; puntos (2,-3);(5,4)
RESULTADOS ESPERADOS: se dibuja la función ampliada en las coordenadas que captura el mouse
RESULTADOS OBTENIDOS: se genera error
COMENTARIO: El procedimiento no funciona correctamente
el evento mouseup no funciona correctamente,
Se corrige

CASO DE PRUEBA: Procedimiento desahacer
DATOS DE PRUEBA: Xmin=-5,Xmax=5;Xdiv=1;Ymin=-5;Ymax=5;Ydiv=1
$f(x) = \operatorname{sen} x$; puntos (2,-3);(5,4)
RESULTADOS ESPERADOS: se dibuja la función anterior a la ampliada, y los datos
RESULTADOS OBTENIDOS: se dibuja la función anterior a la ampliada y los datos
COMENTARIO: El procedimiento funciona correctamente.

5.2.3 PRUEBAS DEL CLUSTER (3)

Realizar evaluación

CASO DE PRUEBA:	Recuperación de temas de la base de datos
DATOS DE PRUEBA:	La base de datos Seguridad en la tabla Temas tiene Raíces de Funciones no lineales Manipulación de polinomios
RESULTADOS ESPERADOS:	Raíces de Funciones no lineales Manipulación de polinomios
RESULTADOS OBTENIDOS:	Raíces de Funciones no lineales Manipulación de polinomios
COMENTARIO:	procedimiento de acceso a tabla Temas correcto.

CASO DE PRUEBA:	Recuperación de Tipos de preguntas de la base de datos
DATOS DE PRUEBA:	La base de datos Seguridad en la tabla Tipopreguntas tiene: Verdadero y falso Selección múltiple Selección múltiple con tabla
RESULTADOS ESPERADOS:	Verdadero y falso Selección múltiple Selección múltiple con tabla

RESULTADOS OBTENIDOS:	Verdadero y falso Selección múltiple Selección múltiple con tabla
COMENTARIO:	procedimiento de acceso a tabla Tipopreguntas correcto.

CASO DE PRUEBA:	Recuperación del número de las preguntas de la base de datos
DATOS DE PRUEBA:	La base de datos Seguridad en la tabla Tipopreguntas tiene 20 preguntas númeradas del 1 al 20

RESULTADOS ESPERADOS:	número de la pregunta correspondiente a la clave IDTipo
RESULTADOS OBTENIDOS:	número de la pregunta correspondiente al IDTipo
COMENTARIO:	procedimiento de acceso a tabla Tipopreguntas correcto. Devolución de los registro del campo Número de preguntas correcto

CASO DE PRUEBA:	Recuperación del enunciado de las preguntas de la base de datos

DATOS DE PRUEBA:	La base de datos Seguridad en la tabla Alternativas tiene 20 enunciados uno para cada pregunta .
RESULTADOS ESPERADOS:	Enunciado de la pregunta correspondiente a la clave IDPregunta
RESULTADOS OBTENIDOS:	Enunciado de la pregunta correspondiente al IDPregunta no es el correcto
COMENTARIO:	Númeración del IDPregunta errónea se corrige

CASO DE PRUEBA:	Recuperación de la tabla de datos de las preguntas con tabla de la base de datos
DATOS DE PRUEBA:	La base de datos Seguridad en la tabla Paresdedatos tiene: (2.5, 4.8),(3.7,8.2)(9.2,12.3)
RESULTADOS ESPERADOS:	retorno de los pares de datos
RESULTADOS OBTENIDOS:	(2.5, 4.8),(3.7,8.2)(9.2,12.3)
COMENTARIO:	la recuperación de los datos a las tablas es eficiente y rápido con los controles ADO.

CASO DE PRUEBA:	Validación de datos y presentación de informe
DATOS DE PRUEBA:	La base de datos Seguridad en la tabla preguntas en el campo Respuestas tiene: Cinco 1, diez 3, cinco 2; representando las respuestas de verdadero y falso, selección múltiple y selección *múltiple* con datos.
RESULTADOS ESPERADOS:	correctas 5, porcentaje 3.5
RESULTADOS OBTENIDOS:	correctas 5, porcentaje 3.5
COMENTARIO:	El procedimiento de evaluación es correcto y formato para la presentación es acertado.

CASO DE PRUEBA:	Verificación del tiempo de la evaluación
DATOS DE PRUEBA:	Tiempo seteado de paro=2 minutos
RESULTADOS ESPERADOS:	Bandera=1 a los 2minutos
RESULTADOS OBTENIDOS:	Bandera=1 a los 2minutos
COMENTARIO:	El procedimiento de verificación del tiempo funciona correctamente. Se implementa para 1 hora para la Evaluación Parcial y 1 hora 30 minutos para la Evaluación Total

5.2.4 PRUEBAS DEL CLUSTER (4)

Encontrar raíces de un polinomio

CASO DE PRUEBA:	Procedimiento AlgoritmodeHorner()
DATOS DE PRUEBA:	$2X^4 - 3X + 3X - 4$ Xo= -2
RESULTADOS ESPERADOS:	P(-2)=10, P'(-2)= -49
RESULTADOS OBTENIDOS:	P(-2)=8, P'(-2)= 70
COMENTARIO:	El procedimiento Pn () no devuelve correctamente los coeficientes obtenido de el polinomio. Se corrige la discriminación del signo de los términos del polinomio

CASO DE PRUEBA:	Procedimiento AlgoritmodeNewtonHorner()
DATOS DE PRUEBA:	$X^4 + X^3 - 2X - 6X - 4$ Xo= -4/6
RESULTADOS ESPERADOS:	

$$(x+1.000157606)(x-2)(x^2+1.999842394x+1.999842419)$$

RESULTADOS OBTENIDOS:

$$(x+1.000157606)(x-2)(x^2+1.999842394x+1.999842419)$$

COMENTARIO: El procedimiento funciona correctamente

CASO DE PRUEBA: Procedimiento AlgoritmodeNewtonBaristow()

DATOS DE PRUEBA: $X^3 - X + X - 2$

(0.5,-1.5)

RESULTADOS ESPERADOS: Primera iteración (-0.22222,-0.75)

RESULTADOS OBTENIDOS: Primera iteración (-0.70,-0.85)

COMENTARIO: El procedimiento no funciona correctamente, lazo para el calculo de los r, s parciales no entrega el resultado adecuado. Se corrige el problema

CASO DE PRUEBA: Procedimiento AplicacióndeNewtonBairstow()

DATOS DE PRUEBA: $X^3 - 6X + 11X - 6$

(3,-4)

RESULTADOS ESPERADOS: P(3,-4)=(-48,56)

RESULTADOS OBTENIDOS: P(3,-4)=(-48,56)

COMENTARIO: El procedimiento funciona correctamente

Calcular la derivada de una función

CASO DE PRUEBA: Procedimiento DiferenciasCentradas()

DATOS DE PRUEBA: Tipoalgoritmo= True; Tipofunción=True;

$-0.1x^4 - 0.15x^3 - 0.5x^2 - 0.25x + 1.2$

x=0.5;h=0.25

RESULTADOS ESPERADOS: -0.9125
RESULTADOS OBTENIDOS: -0.9125

COMENTARIO: El procedimiento funciona correctamente

CASO DE PRUEBA: Procedimiento DiferenciasProgresivas()
DATOS DE PRUEBA: Tipoalgoritmo= True; Tipofunción=True;
$-0.1x^4 - 0.15x^3 - 0.5x^2 - 0.25x + 1.2$
x=0.5;h=0.25

RESULTADOS ESPERADOS: -0.859375
RESULTADOS OBTENIDOS: -0.524368

COMENTARIO: El procedimiento no funciona correctamente, el Procedimiento feval() no evalúa la función por estar mal llamado. Se corrige

CASO DE PRUEBA: Procedimiento DiferenciasRegresivas()
DATOS DE PRUEBA: Tipoalgoritmo= True; Tipofunción=True;
$-0.1x^4 - 0.15x^3 - 0.5x^2 - 0.25x + 1.2$
x=0.5;h=0.25

RESULTADOS ESPERADOS: -0.878125
RESULTADOS OBTENIDOS: -0.878125

COMENTARIO: El procedimiento funciona correctamente

5.2.5 PRUEBAS DEL CLUSTER (5)

Resolver un sistema de ecuaciones diferenciales ordinarias

CASO DE PRUEBA: Procedimiento Euler()

DATOS DE PRUEBA:

$$\frac{dy}{dt} = -2x^3 + 12x^2 - 20x + 8.5$$

y(0)=1;[0,4];h=0.25

RESULTADOS ESPERADOS: y(0.5)=5.25

RESULTADOS OBTENIDOS: y(0.5)=5.25

COMENTARIO: El procedimiento funciona correctamente

Resolver un sistema de ecuaciones diferenciales ordinarias

CASO DE PRUEBA: Procedimiento EulermodificadoyHeun()

DATOS DE PRUEBA: EulerHeun= True

$$\frac{dy}{dt} = 4e^{0.8t} - 0.5y \qquad y(0)=2;[0,4];h=1$$

RESULTADOS ESPERADOS: Y(2)=6.7010819

RESULTADOS OBTENIDOS: y(2)=5.2523589

COMENTARIO: El procedimiento no funciona correctamente, las constantes que identifican tanto al método de Heun

Como al de Euler modificado son incorrectas. Sé cambian las constantes

CASO DE PRUEBA: Procedimiento RKF4()

DATOS DE PRUEBA:

$\frac{dy}{dt} = (t - y)/2$ y(0)=1;[0,3];h=1

RESULTADOS ESPERADOS: Y(1)=16701860

RESULTADOS OBTENIDOS: Y(1)=16701860

COMENTARIO: El procedimiento funciona correctamente

CASO DE PRUEBA: Procedimiento AdamsBashforthMoulton()

DATOS DE PRUEBA:

$\frac{dy}{dt} = (t - y)/2$ y(0)=1;[0,3];h=0.5

RESULTADOS ESPERADOS: Y(1)=0.83640227

RESULTADOS OBTENIDOS: Y(1)=0.83640227

COMENTARIO: El procedimiento funciona correctamente

CASO DE PRUEBA: Procedimiento MileneSimpson()

DATOS DE PRUEBA:

$$\frac{dy}{dt} = (t - y)/2 \qquad y(0)=1;[0,3];h=0.5$$

RESULTADOS ESPERADOS: Y(1)=0.83640231

RESULTADOS OBTENIDOS: Y(1)=0.98364054

COMENTARIO: El procedimiento no funciona correctamente, resultados de la aproximación de los cuatro primeros puntos mediante RKF4 son erróneos. Sé corrige

CASO DE PRUEBA: Procedimiento SistemasEDO()

DATOS DE PRUEBA:

$$\frac{dy1}{dt} = y2 \qquad y1(0)= =3 \qquad h=0.1$$

$$\frac{dy2}{dt} = -5y1 - 4y2 \qquad y2(0)=-5$$

RESULTADOS ESPERADOS: Y(0.1)= 6.29354551,4.53932490

RESULTADOS OBTENIDOS: Y(0.1)= 6.98364054,5.36783525

COMENTARIO: El procedimiento no funciona correctamente, resultados el procedimiento Evaluar sistema no evalúa correctamente las variables. Se corrige

CASO DE PRUEBA: Procedimiento DisparoLineal()

DATOS DE PRUEBA:

$$x''(t) = \frac{2t}{1+t^2}x'(t) - \frac{2}{1+t^2} + 1 \qquad [0,4];h=0.2$$

x(0)=1.25 x(4)=-0.95

RESULTADOS ESPERADOS: Y(0.2)=1.317308
RESULTADOS OBTENIDOS: Y(0.2)=1.317308

COMENTARIO: El procedimiento funciona correctamente,

Realizar una interpolación polinomial

CASO DE PRUEBA: Procedimiento RegresiónPolinomial()

DATOS DE PRUEBA:

X	Y
1	0.5
2	2.5
3	2.0
4	4.0
5	3.5
6	6.0
7	5.5

RESULTADOS ESPERADOS: ao=0.8393;a1=0.07143
RESULTADOS OBTENIDOS: ao=0.8393;a1=0.07143

COMENTARIO: El procedimiento funciona correctamente,

CASO DE PRUEBA: Procedimiento TécnicaMatricial()

DATOS DE PRUEBA:

X	Y
1	0.7651977
1.3	0.620086

RESULTADOS ESPERADOS: ao=1.333716;a1=-0.548946

RESULTADOS OBTENIDOS: ao=2.8393;a1=0.99143

COMENTARIO: El procedimiento no funciona correctamente, El módulo SEL no entrega correctamente los Resultados, Se corrige el problema

CASO DE PRUEBA: Procedimiento Lagrange()

DATOS DE PRUEBA:

X	Y
1	0.7651977
1.6	0.4554022

X=1.5

RESULTADOS ESPERADOS: x=0.5102968

RESULTADOS OBTENIDOS: x=0.5102968

COMENTARIO: El procedimiento funciona correctamente

CASO DE PRUEBA: Procedimiento Newton()

DATOS DE PRUEBA:

X	Y
0.2	0.09944
0.3	0.19787
0.4	0.29479

X=0.25

RESULTADOS ESPERADOS: x=0.37086

RESULTADOS OBTENIDOS: x=2.5102

COMENTARIO: El procedimiento no funciona correctamente
El procedimiento diferenciasdivididas() no obtiene los resultados correctos. Se corrige el problema

CASO DE PRUEBA: Procedimiento Trigonométrica()

DATOS DE PRUEBA: f(x)=x/2 doce puntos

$x_k = -\pi + k\pi/6$ para k=1 a 12, grado = 5

RESULTADOS ESPERADOS:

$b1$	0.97704862
$b2$	−0.45344984
$b3$	0.261179939
$b4$	−0.15114995
$b5$	0.07014893

RESULTADOS OBTENIDOS:

$b1$	0.97704862
$b2$	−0.45344984
$b3$	0.261179939
$b4$	−0.15114995
$b5$	0.07014893

COMENTARIO: El procedimiento funciona correctamente

CASO DE PRUEBA: Procedimiento SplineLineal()

DATOS DE PRUEBA:

X	Y
3	2.5
4.5	1
7	2.5
9	0.5

RESULTADOS ESPERADOS: para 4.5 a 7 m=0.60

RESULTADOS OBTENIDOS: para 4.5 a 7 m=0.60

COMENTARIO: El procedimiento funciona correctamente

CASO DE PRUEBA: Procedimiento SplineSujeta()

DATOS DE PRUEBA:

X	Y	
0	0	
1	0.5	s'(0)=0.2;s'(3)= -1
2	2	
3	1.5	

RESULTADOS ESPERADOS: m0= -0.36;m1=2.52;m2= -3.72;m3=0.36

RESULTADOS OBTENIDOS: m0= -0.36;m1=2.52;m2= -3.72;m3=0.36

COMENTARIO: El procedimiento funciona correctamente

CASO DE PRUEBA: Procedimiento SplineNatural()

DATOS DE PRUEBA:

X	Y	
0	0	
1	0.5	s''(0)=0;s''(3)= 0
2	2	
3	1.5	

RESULTADOS ESPERADOS: m1=2.4;m2= -12

RESULTADOS OBTENIDOS: m1=2.4;m2= -12

COMENTARIO: El procedimiento funciona correctamente

CASO DE PRUEBA: Procedimiento SplineExtrapolada()

DATOS DE PRUEBA:

X	Y	
0	0	
1	0.5	s'(0)=0.2;s'(3)= -1
2	2	
3	1.5	

RESULTADOS ESPERADOS: m0= 4;m1=2.52;m2= -3.72;m3=-5.0

RESULTADOS OBTENIDOS: m0= -0.36;m1=1;m2= -2;m3=0.36

COMENTARIO: El procedimiento funciona correctamente

CASO DE PRUEBA: Procedimiento SplineCurvaturaenlosextremos()

DATOS DE PRUEBA:

X	Y	
0	0	
1	0.5	s''(0)= -0.3;s''(3)= 3.3
2	2	
3	1.5	

RESULTADOS ESPERADOS: m0= -0.3;m1=2.7;m2= -4.5;m3=3.3

RESULTADOS OBTENIDOS: m0= -0.3;m1=2.7;m2= -4.5;m3=3.3

COMENTARIO: El procedimiento funciona correctamente

5.2.6 PRUEBAS DEL CLUSTER (6)

Resolver ecuaciones diferenciales parciales

CASO DE PRUEBA: Procedimiento Laplace()

DATOS DE PRUEBA: 0<x<4;0<y<4condiciones de contorno
U(x,0)=20;U(x,4)=180 para 0<x<4
U(0,y)=80;U(4,y)=0 para 0<y<4
h=0.5

RESULTADOS ESPERADOS: Primera iteración

80
51.3931
40.5195
35.1691
31.2899
27.2335
21.99
14.1791
0

RESULTADOS OBTENIDOS: Primera iteración

80
51.3931
40.5195
35.1691
31.2899
27.2335
21.99
14.1791
0

COMENTARIO: El procedimiento funciona correctamente,

CASO DE PRUEBA: Procedimiento EcuaciondeOndas()

DATOS DE PRUEBA: utt(x,t)=4Uxx(x,t) para 0<x<1 ; 0<t<0.5 condicones De contorno:

U(0,t)=0 y u(1,t)=0 para 0<t<0.5

U(x,0)=f(x)= $\text{sen}(\pi x) + n(2\pi x)$ para 0<x<1

Ut(x,0)=g(x)=0 para 0<x<1

h=0.1

RESULTADOS ESPERADOS:	Primera iteración	0.896802 1.538842 1.760074 1.538842 1 0.363271 −0.142040 −0.363271 −0.278768
RESULTADOS OBTENIDOS:	Primera iteración	0.896802 1.538842 1.760074 1.538842 1 0.363271 −0.142040 −0.363271 −0.278768
COMENTARIO:	El procedimiento	funciona correctamente,

CASO DE PRUEBA: Procedimiento Métodoexplicito()

DATOS DE PRUEBA: Ut(x,t)=0.835Uxx(x,t) para 0<x<10;0<t<3condición inicial : u(x,0)=0 para t=0 y a<x<10

Condiciones de contorno:

U(0,t)=100 para x=0 y 0<t<b

U(10,t)=50 para x=10 y a<t<3

h=2;r=0.1

RESULTADOS ESPERADOS: Primera iteración

2.0875
0
0
1.0438

RESULTADOS OBTENIDOS: Primera iteración

2.0875
5
2
2.0458

COMENTARIO: El procedimiento no funciona correctamente, El módulo SEL no envía correctamente los resultados de las ecuaciones ,se corrige la llamada al módulo SEL

CASO DE PRUEBA: Procedimiento Métodoimplicito()

DATOS DE PRUEBA: Ut(x,t)=0.835Uxx(x,t) para 0<x<10;0<t<3condición inicial : u(x,0)=0 para t=0 y a<x<10

Condiciones de contorno:

U(0,t)=100 para x=0 y 0<t<b

U(10,t)=50 para x=10 y a<t<3

h=2;r=0.1

RESULTADOS ESPERADOS:	Primera iteración	2.0047 0.0406 0.0209 1.0023
RESULTADOS OBTENIDOS:	Primera iteración	2.0047 0.0406 0.0209 1.0023
COMENTARIO:	El procedimiento	funciona correctamente,

CASO DE PRUEBA:	Procedimiento MétododeCrankNicolson()
DATOS DE PRUEBA:	Ut(x,t)=0.835Uxx(x,t) para 0<x<10;0<t<3condición inicial : u(x,0)=0 para t=0 y a<x<10 Condiciones de contorno: U(0,t)=100 para x=0 y 0<t<b U(10,t)=50 para x=10 y a<t<3 h=2;r=0.1

RESULTADOS ESPERADOS:	Primera iteración	2.045 0.021 0.0107 1.0225
RESULTADOS OBTENIDOS:	Primera iteración	2.045 0.021 0.0107 1.0225
COMENTARIO:	El procedimiento	funciona correctamente,

Encontrara raíces de funciones no lineales

CASO DE PRUEBA: Procedimiento AproximacionesSucesivas()

DATOS DE PRUEBA: $f(x) = x^2 - 4$;a=1;b=3;tolerancia=0.01;h=0.1

RESULTADOS ESPERADOS: devuelve la ráiz x= -2

RESULTADOS OBTENIDOS: devuelve la ráiz x= -2

COMENTARIO: El procedimiento funciona correctamente

CASO DE PRUEBA: Procedimiento Bisección()

DATOS DE PRUEBA: $f(x) = x^3 + 4x^2 - 10$;a=1;b=2;tolerancia 0.0001

RESULTADOS ESPERADOS: raíz aproximada=1.365112305

RESULTADOS OBTENIDOS: raíz aproximada=1.365112305

COMENTARIO: El procedimiento funciona correctamente

CASO DE PRUEBA: Procedimiento FalsePosición()

DATOS DE PRUEBA: f(x)=CosX-X;a=0:b=1;tolerancia=0.000001

RESULTADOS ESPERADOS: Raíz aproximada=0.7390847824

RESULTADOS OBTENIDOS: Raíz aproximada=0.7390847824

COMENTARIO: El procedimiento funciona correctamente

CASO DE PRUEBA: Procedimiento Primerorden()

DATOS DE PRUEBA: $x = g(x) = \frac{1}{2}\left(10 - x^3\right)^{1/2}$

RESULTADOS ESPERADOS: Raíz aproximada=1.365230013

RESULTADOS OBTENIDOS: Raíz aproximada=1.365230013

COMENTARIO: El procedimiento funciona correctamente

CASO DE PRUEBA: Procedimiento NewtonRaphson()

DATOS DE PRUEBA: f(x)= cosx-x;f'(x)=-(senx+1);a=o;b=$\pi/2$;
Tolerancia= 0.0001

RESULTADOS ESPERADOS: Raíz aproximada=0.7390851334

RESULTADOS OBTENIDOS: Raíz aproximada=0.7390851334

COMENTARIO: El procedimiento funciona correctamente

CASO DE PRUEBA: Procedimiento Secante()

DATOS DE PRUEBA: f(x)= cosx-x;f'(x)=-(senx+1);a=o;b=$\pi/2$;
Tolerancia= 0.0001:Xo=0.5;X1=$\pi/4$

RESULTADOS ESPERADOS: Raíz aproximada=0.7390851493

RESULTADOS OBTENIDOS: Raíz aproximada=0.7390851493

COMENTARIO: El procedimiento funciona correctamente

Calcular la integral de una función

CASO DE PRUEBA: Procedimiento NewtonCotesfunción()

DATOS DE PRUEBA: $f(x) = 0.2 + 5x - 200 + 675x^3 - 900x^4 + 400x^5$

a=0;b=0.8

RESULTADOS ESPERADOS: Integral aproximada =

Trapecio	0.1728
$1/3 Simpson$	1.367467
$1/8 Simpson$	1.519170

RESULTADOS OBTENIDOS: Integral aproximada =

Trapecio	0.1728
$1/3 Simpson$	1.367467
$1/8 Simpson$	1.519170

COMENTARIO: El procedimiento funciona correctamente, se realizo las pruebas para datos tabulados para la misma función y se obtuvieron resultados similares.

CASO DE PRUEBA: Procedimiento ReglaExtendidaTrapecio()

DATOS DE PRUEBA: $f(x) = 0.2 + 5x - 200 + 675x^3 - 900x^4 + 400x^5$

a=0;b=0.8 N=2

RESULTADOS ESPERADOS: Integral aproximada = 1.0688

RESULTADOS OBTENIDOS: Integral aproximada = 1.0688

COMENTARIO: El procedimiento funciona correctamente.

CASO DE PRUEBA: Procedimiento ReglaExtendidaSimpson()

DATOS DE PRUEBA: $f(x) = 0.2 + 5x - 200 + 675x^3 - 900x^4 + 400x^5$

a=0;b=0.8 N=4

RESULTADOS ESPERADOS: Integral aproximada = 1.623467

RESULTADOS OBTENIDOS: Integral aproximada = 5.812688

COMENTARIO: El procedimiento no funciona correctamente, El laso para los intervalos impares no funciona se corrige para que tomo los intervalos impares

CASO DE PRUEBA: Procedimiento IntegraciónGaussiana()

DATOS DE PRUEBA: $f(x) = 0.2 + 5x - 200 + 675x^3 - 900x^4 + 400x^5$

a=0;b=0.8 para 2 puntos

RESULTADOS ESPERADOS: Integral aproximada = 1.822578

RESULTADOS OBTENIDOS: Integral aproximada = 3.597812

COMENTARIO: El procedimiento no funciona correctamente, El no se indexa correctamente la matriz que contiene el arreglo de coeficientes se corrige mejorando el arreglo de los coeficientes.

5.3 PRUEBAS DE SISTEMA

En esta estrategia de prueba el software es incorporado a otros elementos del sistema (hardware, datos) y se realizan pruebas de integración del sistema y de validación para verificar que se han integrado adecuadamente todos los elementos del sistema y que funcionan de acuerdo a lo esperado.
Su objetivo es la comprobación del sistema global, normalmente se realizan pruebas de cuatro tipos distintos:

5.3.1 PRUEBA DE RECUPERACIÓN DE FALLOS

Tiene por objetivo forzar al software a fallar y verificar que la recuperación se lleve a cabo adecuadamente.
Si la recuperación es *automática* hay que evaluar el correcto funcionamiento de los casos que la componen (reinicialización, recuperación de datos de arranque, etc.).
Si la recuperación *requiere de intervención humana* se deberá evaluar los tiempos medios de reparación para determinar si está dentro de los límites aceptables.

CASO DE PRUEBA:	Cierre repentino
DATOS DE PRUEBA:	Falta de energía
RESULTADOS ESPERADOS:	El software inicia sin problemas
RESULTADOS OBTENIDOS:	El software inicia sin problemas
COMENTARIO:	La falta de energía no afecta a la inicialización del software pero se han perdido los últimos datos del usuario que se encontraba en el sistema

CASO DE PRUEBA:	No disponible un software adicional
DATOS DE PRUEBA:	Falta aplicación MsExcel
RESULTADOS ESPERADOS:	error en la ejecución de los algoritmos
RESULTADOS OBTENIDOS:	error en la ejecución de los algoritmos
COMENTARIO:	Se genera un error grave al utilizar un Algoritmo. Se corrige

CASO DE PRUEBA:	No disponible un software adicional
DATOS DE PRUEBA:	Falta aplicación Access
RESULTADOS ESPERADOS:	error en la ejecución de los algoritmos
RESULTADOS OBTENIDOS:	error en la ejecución de los algoritmos
COMENTARIO:	Se genera un error grave al utilizar un Algoritmo. Se corrige

CASO DE PRUEBA:	Un archivo del programa borrado
DATOS DE PRUEBA:	Falta del archivo Proyect1
RESULTADOS ESPERADOS:	Mensaje de error del sistema operativo
RESULTADOS OBTENIDOS:	Mensaje de error del sistema operativo
COMENTARIO:	Se genera un error grave al si se borra o cambia de directorio a un archivo no pudiendo utilizarse el programa.

5.3.2 PRUEBA DE SEGURIDAD

Intenta verificar que los mecanismos de protección realmente protejan al sistema de la penetración indebida (protecciones de la información).

El encargado de esta prueba debe ubicarse en el lugar del que intenta penetrar al sistema en forma no autorizada. Todo recurso es válido para este tipo de prueba: tratar de conseguir las claves, producir errores intencionales en el sistema para entrar en la etapa de recuperación, etc.

Evidentemente si se le da a una persona el tiempo y los recursos suficientes, terminará por penetrar el sistema; por lo tanto, el objetivo es lograr que el costo de penetrar el sistema sea mayor que el valor de la información.

Para el caso del Paquete de Software Didáctico para la Enseñanza Aprendizaje de Métodos Numéricos la pruebas de seguridad no son realizadas con el enfoque anterior ya que el ser un medió de aprendizaje no se requiere de una clave estricta para el ingreso, ya que si la olvido puede ingresar como usuario nuevo con una distinta.

CASO DE PRUEBA:	Ingreso como usuario Nuevo
DATOS DE PRUEBA:	"Milton Tipan" "DIC"
RESULTADOS ESPERADOS:	mensaje de la existencia este nombre o password
RESULTADOS OBTENIDOS:	mensaje de la existencia este nombre o password
COMENTARIO:	El ingreso como usuario nuevo también requiere de validación para la no duplicación de información

CASO DE PRUEBA:	Ingreso como usuario Antiguo
DATOS DE PRUEBA:	"Milton Tipan" "NIC"
RESULTADOS ESPERADOS:	mensaje de nombre o password incorrecto
RESULTADOS OBTENIDOS:	mensaje de nombre o password incorrecto
COMENTARIO:	El filtrado de datos y la validación funcionan Correctamente

5.3.3 PRUEBA DE RESISTENCIA Y DE RENDIMIENTO

Está diseñada para enfrentar al sistema con situaciones anormales, es decir una mayor demanda de recursos, tanto en cantidad como en frecuencia. Esto tiene por objeto responder a la pregunta: ¿a que potencia se puede poner a funcionar un sistema antes de que falle?

Se lleva a cabo en cada etapa del ciclo de prueba, pero no puede asegurarse el rendimiento total del sistema hasta que no se lo tenga totalmente integrado. Muchas veces se realiza simultáneamente con las pruebas de resistencia.
Debido a la falta de problemas que para ser resueltos requieran un número grande de datos se opto por realizar esta prueba de rendimiento con la solución de un problema para la ecuación del calor con distintos intervalos de división para la malla generada para la solución; esta malla genera un número de ecuaciones simultaneas grande asi si la malla es 10X10 genera 10 diferentes sistemas de ecuaciones 10 veces los mismos que deben ser resueltos mediante el módulo SEL y el Módulo MscExcel que accede a la aplicación MsExcel para la evaluación de la función, entonces entre más fina la malla más ecuaciones genera, por lo tanto el consumo de recursos del sistema es considerable. La Fig.5.3 muestra el número de divisiones de la malla y el tiempo de la espera para la obtención de la respuesta del problema:

Ut(x,t)=Uxx(x,t) para 0<x<1 y 0<t<0.1
Con condiciones iniciales:
U(x,0)= $f(x) = \text{sen}(\pi x) + \text{sen}(3\pi x)$ t=0 y 0<x<1
Con condiciones de contorno:
U(0,t)=g1(t)=0 para x=0 y 0<t<0.1
U(0,t)=g2(t)=0 para x=0 y 0<t<0.1

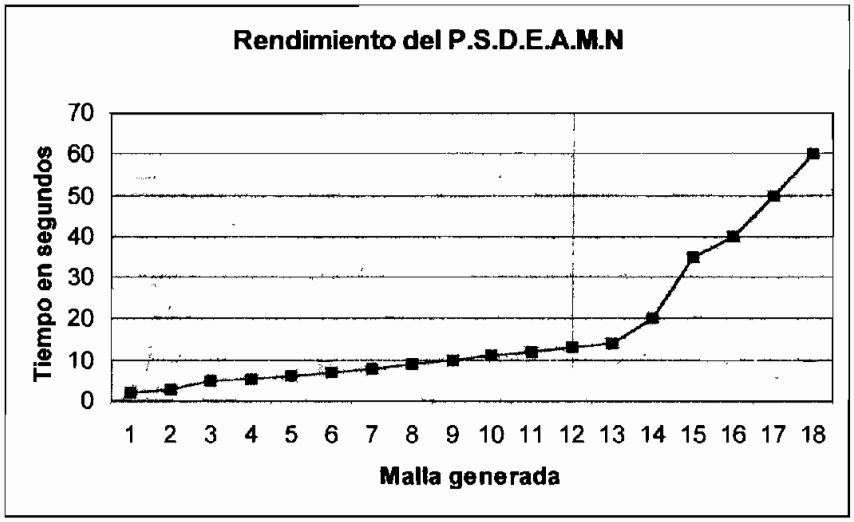


Fig. 5.3 Tiempo de Respuesta del sistema para la solución de la Ecuación del Calor

La gráfica muestra en el eje X la malla generada donde 1 es una malla 10X10, 2 es 20X20, etc.

Por lo tanto se puede observar el incremento de recursos del sistema indirectamente observando el incremento de tiempo que necesita el sistema para mostrar la solución. El incremento de recurso que usa el sistema es más considerable cuanto más fina es la malla. Estas pruebas fueron realizadas en una computadora de las siguientes características:

Disco Duro: =	80 GB
RAM=	256 MB
CPU=	1.6 Gigas Intel Pentiun 4

El incremento en el uso de los recursos del sistema aumentara considerablemente en una computadora inferior en características como una 486, por lo que es aconsejable el uso del P.S.D.E.A.M.N en una computadora que tenga las características anotadas anteriormente si lo que desea es hacer cálculos que conllevan operaciones que requieran de muchos recursos del sistema. Si no es así basta que se pueda usar una computados 486 con al menos 128 MB de memoria RAM.

CAPÍTULO 6

CONCLUSIONES Y RECOMENDACIONES

6.1 CONCLUSIONES

1. Se cumplió con el objetivo de brindar al estudiante que toma materia de Métodos Numéricos la posibilidad de contar con una herramienta que le permita afianzar sus conocimientos, y al profesor que la dicta, utilizarla para mostrar a sus alumnos los cambios que ocurren cuando los parámetros de un algoritmo son modificados.

2. El desarrollo del Paquete de Software Didáctico para la Enseñanza-Aprendizaje de Métodos Numéricos permite al usuario aprender de manera interactiva, al ritmo que él desee y más aún la facilidad que este brinda para la realización de los cálculos de los algoritmos, que de otra manera el realizarlos a mano sería tedioso y le quitaría el interés de aprender al estudiante.

3. El desarrollo de software educativo no pretende reemplazar al profesor que dicta la materia objeto del software educativo, sino ser una herramienta que permita mejorar la comprensión de la materia. Ya que un software no necesariamente puede engendrar en el estudiante las destrezas cognitivas, afectivas y Psicomotoras que este necesita en su vida profesional, sino es necesaria la presencia del profesor que con su guía formativa y ejemplo de vida, ayuda a que el estudiante logre y alcance sus metas.

4. Es muy importante el realizar el documento de Especificación de requerimientos pus en el se define todos y cada uno de los límites y funciones que debe cumplir la aplicación desarrollada.

5. La elección de una metodología para el desarrollo de software debe contar con un conjunto definido de fases y diagramas de modelamiento que indiquen claramente que se debe realizar en cada etapa, como se debe ejecutar cada una de las actividades y que diagramas deben ser usados para modelar el conjunto de objetos definidos para la aplicación.

6. La metodología de desarrollo de software orientada a objetos propuesta por Craig Larman y el uso de del lenguaje de modelamiento UML facilitó el desarrollo del Paquete de Software Didáctico para la Enseñanza-Aprendizaje de Métodos Numéricos ya que este especifica claramente las fases y las actividades que deben ser realizadas. Además el incorporar el lenguaje de modelamiento UML permite estandarizar los diagramas necesarios para el modelamiento de los objetos del software.

7. El desarrollo de los diagramas de colaboración es crítico a la hora de desarrollar la fase de diseño del software ya que es aquí donde se designan las responsabilidades y se crean objetos que interactuan entres si para cumplir con lo establecido en los contratos de operación, por lo tanto a la elaboración de estos diagramas se les debe dedicar el mayor tiempo posible en el desarrollo de la aplicación.

8. La herramienta de desarrollo escogida para la implementación no necesariamente tiene que ser la más costosa, sino la que brinde las facilidades necesarias para la implementación de la aplicación. Visual Basic en su versión 6.0 brinda muchas facilidades en las que se cuentan: facilidad para la implementación de las interfaces, acceso a bases de datos mediante controles ADO, etc. lo que lo hace adecuado para la implementación de aplicaciones que deban trabajar con objetos persistentes.

9. El uso de una arquitectura de tres capas permite implementar de mejor manera paradigma de orientación a objetos, ya que divide al sistema en tres partes: presentación, lógica de aplicaciones, almacenamiento con lo que se consigue una mejor distribución de las responsabilidades de los objetos creados.

10. El uso de paginas Web permite que el tutorial sea más interactivo con el usuario ya que facilita la búsqueda de temas y sus enlaces permiten navegar con mucha facilidad entre las mismas. Además el uso de Gifs en las mismas brinda un atractivo visual para el usuario.

11. El uso de una herramienta Case tal como Visual Modeler facilita en gran medida la implementación ya que permite pasar de manera automática de un diagrama de clases de diseño con arquitectura de tres capas, a la creación de las clases, sus atributos y procedimientos a código para Visual Basic.

6.2 RECOMENDACIONES

1. Mejorar el Paquete de Software para la Enseñanza-Aprendizaje de Métodos Numéricos incluyendo algoritmos que no han sido tomados en cuenta como Runge-Kutta, Fehlberg, valores y vectores propios, entre otros.

2. Desarrollar aplicaciones de graficación más completas, ya que aunque el módulo de graficación, grafica cualquier tipo de función de una variable, en muchas ocasiones se requiere la graficación de funciones con más variable, como en el caso de las ecuaciones diferenciales parciales que necesitan de una aplicación de graficación en tres dimensiones para observar como se comporta la solución de la misma.

3. En el desarrollo de una aplicación es necesario tomar en consideración todos los estándares que hasta ese momento estén vigentes. Ya que permiten avalizar el mismo y el disponer de toda la documentación que permita desarrollar la aplicación sin ninguna complicación.

4. Antes de elegir una metodología para el desarrollo de la aplicación se debe conocer que facilidades que brinda la misma para tal efecto y si concuerda con el paradigma de desarrollo (orientado a datos, orientado a objetos) escogido.

5. Si se usa la metodología de Craig Larman, desarrollar por lo menos dos iteraciones a los contratos de operación en especial dándole mayor importancia a los apartados de Pre-condicón y Post-condicón ya que en estos se basan los diagramas de colaboración.

6. Se debe realizar una investigación preliminar de los requerimientos del software a desarrollarse, usando para ello alguna técnica que permita ir obteniendo esta información tal: como mesas redondas o entrevistas a las personas involucradas en el mismo.

7. En los diagramas de Casos de Uso se debe tomar en consideración las relaciones extiende y usa las que permiten tener una idea de cómo colaboraran las clases en lo posterior, además no se deben graficar más de 15 casos de uso por cada diagrama.

8. Utilizar el Internet como biblioteca, ya que este es un lugar donde reside gran cantidad de información de toda índole y puede ayudarnos a solucionar dudas que surgen en el desarrollo de un proyecto de software.

9. Crear un sitio Web que permita a los estudiantes acceder a todos los tutoriales que se han ejecutado como parte de una Tesis de Grado o Proyecto de Titulación ya el objetivo de la creación del mismo, es el de que sirva para facilitar y mejorar el aprendizaje.

BIBLIOGRAFIA

LIBROS:

1. Alan M. Davis, Sotware Requirements. Objects, Functions and States, Editorial Prentice Hall, 1993.
2. Larman. Craig. UML y PATRONES. Análisis y Diseño Orientado a Objetos, Editorial Prentice Hall, 1999.
3. Boch Grandy, Análisis y Diseño Orientado a Objetos con aplicaciones, Segunda Edición, Editorial Diana, 1996.
4. Coleman DereK, Object Oriented Development. The Fusión Method, Editorial Prentice Hall 1994.
5. Rumbaugh. James, Object-Oriented Modeling ans Desing, Editorial Prentice Hall, 1994.
6. Falcon Grazón Christian Leonard, Sistema de Software para la Enseñanza de una Metodología de construcción de Software Orientada a Objetos, EPN, 2000
7. Acosta Hurtado Tania Aleyda, Estándares para el Desarrollo de Software Orientado a Objetos, EPN, 1999.
8. Del Pozo Andrade. Marcial Ernesto, Herramienta Generadora de Sistemas de Ayuda en Línea Y tutoriales de Usuario para Windows, EPN, 1996.
9. Ministerio de Educación y Cultura Proyecto EB/PRODEC, Manual de Evaluación del Aprendizaje,Quito,1998
10. Poveda Elva, Pedagogía de la Evaluación, Tercera Edición, Editorial Nuevo Día, 1997.
11. Confedec, El Ciclo Experiencial, 2003.
12. Steven C. Chapra, Raymond P. Canale, Métodos Numéricos para Ingenieros, Tercera Edición, Editorial Mc Graw-Hill, 1999
13. Burden,R.L. Fairres, J.D. ,Análisis Numérico, Editorial Iberoamericano, 1998.
14. Cerón, O.E Poligrafiado de Métodos Numéricos, EPN, 1999.
15. NaKamura, Shoichiro, Métodos Numéricos Aplicados con Software, Editorial Prentice Hall, 1992.

INTERNET:

16. http:// areaint.com Curso de Visual Basic 6.0
17. http:// ourworld.compuserve.com/homepages/hlr/evalsed.html.
18. http:// www.microsoft.com
19. http:// www.msdn.com
20. http:// www.macromedia.com
21. http:// www.surfenet.com/muniain/it/spanish/curso12.html
22. http:// www.rational.com/uml/resources/documentatión/ocl./index.html
23. http:// wwww.geocites.com/siliconValley/Network/7390/terminos.html
24. http:// www.edutool.com/architecture/Itsa-400.html
25. http://www.ucm.es/info/multidoc/multidoc/revista/num8/segnivel2.htm
26. http://www.multingles.net/articulos.htm
27. http://www.cnice.mecd.es/recursos2/orientacion/01apoyo/op03.htm
28. http: // www.inf.uttsm.d/liuba/taller/ad_001.pdf
29. http:// lawebdel programador.com
30. http://go.to/acervo
31. http://guille.costasol.net
32. http: // www.rodrigogarcia.es.fm
33. http:// www.redvisual.net/articulos/otros/hlp/indice.htm.
34. http://www.vrml.org/VRML2.0/FINAL.
35. http://www.portalvb.com

CODIGO FUENTE DEL PAQUETE DE SOFTWARE DIDÁCTICO PARA LA ENSEÑANZA-APRENDIZAJE DE METODOS NUMERICOS

CODIGO DEL CLUSTER (1)

Identificar estudiante

Interfaz frmSeguridad

```
Private VLConexion As ADODB.Connection
Private VLRegistro As ADODB.Recordset
Public Nombre As String

Private Sub CmdIngresar_Click()

Dim BaseDatos As New BasedatosMnts
Dim rstPublishers As ADODB.Recordset
Dim NumRecorset As Integer
Dim strField As String
Dim strFilter As String
Dim strFields As String
Dim strFilters As String
Dim v As String

On Error GoTo error1
'Vemos si hay datos ingresados
If Len(TxtNombre.Text) = 0 Then
v = MsgBox("Debe ingresar un Nombre", 48, "Incorrecto")
Exit Sub
End If
If Len(TxtPassword.Text) = 0 Then
v = MsgBox("Debe ingresar un Password", 48, "Incorrecto")
Exit Sub
End If
'asignamos nombre de los campos de la base de datos y el texto de ingreso
'para compararlas com las existentes en la base de datos
strField = "NombreUsuario"
strFilter = TxtNombre.Text
strFields = "Clave"
strFilters = TxtPassword.Text

BaseDatos.AccesoSeguridad  'accedemos objeto comand de la base de datos
```

```
If OptNuevo.Value = True Then  'si a seleccionado usuario nuevo
'Verificamos si existe un nombre y password igual al ingresado
BaseDatos.rsAccesoSeguridad.Filter = strField & " = '" & strFilter & "'" & " " & " AND "
& strFields & " = '" & strFilters & "'"
NumRecorset = BaseDatos.rsAccesoSeguridad.RecordCount
   If NumRecorset = 0 Then
   BaseDatos.rsAccesoSeguridad.AddNew 'añadimos un nuevo registro a la base de
datos con los datos del usuario nuevo
   BaseDatos.rsAccesoSeguridad!NombreUsuario = TxtNombre.Text
   BaseDatos.rsAccesoSeguridad!clave = TxtPassword.Text
   BaseDatos.rsAccesoSeguridad.Update
   Frmmodulos.Show
   Me.Visible = False
   Nombre = TxtNombre.Text
   Else
   v = MsgBox("Ingrese un nuevo password por que el que ingreso ya existe", 48,
"Mensaje")
   End If
Else
'si es usuario antiguo verificamos si es correcto su nombre y password
BaseDatos.rsAccesoSeguridad.Filter = strField & " = '" & strFilter & "'" & " " & " AND "
& strFields & " = '" & strFilters & "'"
NumRecorset = BaseDatos.rsAccesoSeguridad.RecordCount
   If NumRecorset = 0 Then 'si no es presentamos mensaje que son incorrectos
   v = MsgBox("Nombre o Password incorrectos ingrese nuevamente", 48,
"Incorrecto")
   Else ' si son correctos permitimos el acceso
      PLConexion
      PLIDUsuario BaseDatos.rsAccesoSeguridad.Filter
      Frmmodulos.Show
      Me.Visible = False
       Nombre = TxtNombre.Text
   End If
End If
BaseDatos.rsAccesoSeguridad.Close 'cerramos la conexión con la base de datos
Exit Sub
error1:
   If err.Number <> 0 Then
      MsgBox "Se ha provocado el siguiente error: " & err.Description & ". Consulte
con el administrador", vbCritical, "Error"
   End If
End Sub



Private Sub CmdSalirSeguridad_Click()
```

```
    Unload Me
End Sub

Private Sub PLConexion()
Dim VTCad As String
Dim VTBase As Database
Dim VTcadena As String
    VTcadena = App.Path & "/Seguridad.mdb"
    Set VLConexion = New ADODB.Connection
    VTCad = "Provider=Microsoft.Jet.OLEDB.4.0;Data Source='" & VTcadena &
"';Persist Security Info=False"
    VLConexion.Open VTCad
End Sub

Private Sub PLIDUsuario(parConsulta As String)
Dim VTReg As ADODB.Recordset
Dim VTCad As String
    VTCad = "select Id from seguridad " _
          & " where " & parConsulta
    Set VTReg = VLConexion.Execute(VTCad)
    VGIDUsuario = VTReg(0)

End Sub
```

CODIGO DEL CLUSTER (2)
Desplegar un tema del contenido teórico
Interfaz frmmodulos

```
Option Explicit
'En Microsoft TechNet puedes encontrar este artículo:
'HOWTO: Use HTML Help API in a Visual Basic 5.0 Application
'PSS ID Number: Q183434
'
'Aunque la definición de la Enumeración y la primera declaración
'es de las news
'
'Htmlhelp consts
Private Enum HH_COMMAND
    HH_DISPLAY_TOPIC = &H0
    HH_HELP_FINDER = &H0        ' WinHelp equivalent
    HH_DISPLAY_TOC = &H1        ' not currently implemented
    HH_DISPLAY_INDEX = &H2      ' not currently implemented
    HH_DISPLAY_SEARCH = &H3     ' not currently implemented
    HH_SET_WIN_TYPE = &H4
    HH_GET_WIN_TYPE = &H5
    HH_GET_WIN_HANDLE = &H6
    HH_GET_INFO_TYPES = &H7     ' not currently implemented
```

```
    HH_SET_INFO_TYPES = &H8     ' not currently implemented
    HH_SYNC = &H9
    HH_ADD_NAV_UI = &HA         ' not currently implemented
    HH_ADD_BUTTON = &HB         ' not currently implemented
    HH_GETBROWSER_APP = &HC     ' not currently implemented
    HH_KEYWORD_LOOKUP = &HD
    HH_DISPLAY_TEXT_POPUP = &HE ' display string resource id
                        ' or text in a popup window
    HH_HELP_CONTEXT = &HF       ' display mapped numeric value
                        ' in dwData
    HH_TP_HELP_CONTEXTMENU      ' Text pop-up help, similar to
                        ' WinHelp's HELP_CONTEXTMENU.
    HH_TP_HELP_WM_HELP = &H11   ' text pop-up help, similar to
                        ' WinHelp's HELP_WM_HELP.
    HH_CLOSE_ALL = &H12         ' close all windows opened directly
                        ' or indirectly by the caller
    HH_ALINK_LOOKUP = &H13      ' ALink version of HH_KEYWORD_LOOKUP
End Enum

'HtmlHelp api call
'NOTA: Si se usa esta forma, hay que indicar el último parámetro
'     con la palabra ByVal delante...
'Private Declare Function HtmlHelp Lib "hhctrl.ocx" Alias "HtmlHelpA" _
    (ByVal hwndCaller As Long, ByVal pszFile As String, _
    ByVal uCommand As HH_COMMAND, dwData As Any) As Long
'Con esta funciona perfectamente
Private Declare Function HtmlHelp Lib "hhctrl.ocx" Alias "HtmlHelpA" _
    (ByVal hwndCaller As Long, ByVal pszFile As String, _
    ByVal uCommand As HH_COMMAND, ByVal dwData As Long) As Long




Private Sub Cmd1modulos_Click()
    VGForma = "Graficacion General"
    FrmGraficando.Show
    Me.Visible = False
End Sub

Private Sub Cmd2modulos_Click()
'Así se llamaría para mostrar un tópico de la ayuda
```

```
    Dim h As Long

    'h = HtmlHelp(Me.hWnd, "hhw_vb.chm", HH_DISPLAY_TOPIC, 0&)

    h = HtmlHelp(Me.hWnd, "Tutorial.chm", HH_HELP_CONTEXT, 1040&)
End Sub

Private Sub Cmd3modulos_Click()
    'VGForma = ""
    'Unload Me
    FrmPrincipalalgoritmos.Show
    Me.Visible = False
End Sub

Private Sub Cmd4modulos_Click()
FrmEvaluación.Show
Me.Visible = False
End Sub

Private Sub Cmdsalirmodulos_Click()
End
End Sub

Private Sub Form_Unload(Cancel As Integer)
    End
End Sub
```

Resolver sistemas de ecuaciones lineales
Interfaz frmSEL

```
'En Microsoft TechNet puedes encontrar este artículo:
'HOWTO: Use HTML Help API in a Visual Basic 5.0 Application
'PSS ID Number: Q183434
'
'Aunque la definición de la Enumeración y la primera declaración
'es de las news
'
'Htmlhelp consts
Private Enum HH_COMMAND
    HH_DISPLAY_TOPIC = &H0
    HH_HELP_FINDER = &H0        ' WinHelp equivalent
    HH_DISPLAY_TOC = &H1        ' not currently implemented
    HH_DISPLAY_INDEX = &H2      ' not currently implemented
    HH_DISPLAY_SEARCH = &H3     ' not currently implemented
    HH_SET_WIN_TYPE = &H4
    HH_GET_WIN_TYPE = &H5
    HH_GET_WIN_HANDLE = &H6
```

```
    HH_GET_INFO_TYPES = &H7     ' not currently implemented
    HH_SET_INFO_TYPES = &H8     ' not currently implemented
    HH_SYNC = &H9
    HH_ADD_NAV_UI = &HA         ' not currently implemented
    HH_ADD_BUTTON = &HB         ' not currently implemented
    HH_GETBROWSER_APP = &HC     ' not currently implemented
    HH_KEYWORD_LOOKUP = &HD
    HH_DISPLAY_TEXT_POPUP = &HE ' display string resource id
                                ' or text in a popup window
    HH_HELP_CONTEXT = &HF       ' display mapped numeric value
                                ' in dwData
    HH_TP_HELP_CONTEXTMENU      ' Text pop-up help, similar to
                                ' WinHelp's HELP_CONTEXTMENU.
    HH_TP_HELP_WM_HELP = &H11   ' text pop-up help, similar to
                                ' WinHelp's HELP_WM_HELP.
    HH_CLOSE_ALL = &H12         ' close all windows opened directly
                                ' or indirectly by the caller
    HH_ALINK_LOOKUP = &H13      ' ALink version of HH_KEYWORD_LOOKUP
End Enum

'HtmlHelp api call
'NOTA: Si se usa esta forma, hay que indicar el último parámetro
'      con la palabra ByVal delante...
'Private Declare Function HtmlHelp Lib "hhctrl.ocx" Alias "HtmlHelpA" _
    (ByVal hwndCaller As Long, ByVal pszFile As String, _
    ByVal uCommand As HH_COMMAND, dwData As Any) As Long
'Con esta funciona perfectamente
Private Declare Function HtmlHelp Lib "hhctrl.ocx" Alias "HtmlHelpA" _
    (ByVal hwndCaller As Long, ByVal pszFile As String, _
    ByVal uCommand As HH_COMMAND, ByVal dwData As Long) As Long


Dim SEL As New ClasSEL
Dim A() As Double
Dim b() As Double
Dim r()
Dim RS() As String
Dim XR()
Dim Vinicial() As Double
Dim n As Integer
Dim contador1 As Integer
Dim contador As Integer
Dim tol As Double
Dim itermax As Integer
Dim err As Boolean
Dim tipoalg As Integer
```

```
Dim mensaje As String
Dim aviso As Boolean
Dim arribaImprimir
Dim arribaSstab
Dim izquierdaSstab
Dim anchoSstab
Dim altoSstab
Dim arribaResul
Dim altoResul
Dim anchoResul
Dim arribaFgresul
Dim anchoFgresul
Dim altoFgresul
Dim arribaFgitera
Dim izquierdaFgitera
Dim anchoFgitera
Dim altoFgitera
Dim colororg
Dim VLabel As Boolean
Dim Vadelante As Boolean
Dim Vatras As Boolean
Private Acepta As Boolean
Private VLConexion As ADODB.Connection
Private VLRegistro As ADODB.Recordset
Private VLNombre As String
Private VLNombreTrabajo As String
Private VLVerifica As Boolean
Private VLBDD As Boolean
Private VLJacobiano As String
Private VLJacobianoMatriz(1 To 1000) As String

Property Let Aceptar(parAceptar As Boolean)
   Acepta = parAceptar
End Property

Property Let Nombre(parNombre As String)
   VLNombre = parNombre
End Property

Property Let NombreTrabajo(parNombreTrabajo As String)
   VLNombreTrabajo = parNombreTrabajo
End Property

Function Fgit(r As Integer, c As Integer) As Integer
  Fgit = c + fgSELiteraciones.Cols * r
End Function
```

```
Function Fgi(r As Integer, c As Integer) As Integer
  Fgi = c + Fg2.Cols * r
End Function
Function Fgr(r As Integer, c As Integer) As Integer
  Fgr = c + fgSELresultados.Cols * r
End Function


Private Sub CmdSEladelante_Click()
Dim i As Integer
Dim j As Integer
Dim k As Integer
Dim limite As Integer
On Error GoTo err1
Select Case (tipoalg)
Case Is = 1

If contador1 <= n - 3 Then
k = contador1 + 1
   contador1 = contador1 + 1
   For i = 0 To n - 1
      For j = 0 To n
      fgSELiteraciones.TextMatrix(i + 1, j + 1) = r(i, j, k)
      Next j
   Next i
End If

Case Is = 2
limite = ((n - 1) * 2) - 2
If contador1 <= limite Then
k = contador1 + 1
   contador1 = contador1 + 1
   For i = 0 To n - 1
      For j = 0 To n
      fgSELiteraciones.TextMatrix(i + 1, j + 1) = r(i, j, k)
      Next j
   Next i
End If
Case Is = 3
If contador1 <= n - 2 Then
k = contador1 + 1
   contador1 = contador1 + 1
   For i = 0 To n - 1
      For j = 0 To n
      fgSELiteraciones.TextMatrix(i + 1, j + 1) = r(i, j, k)
      Next j
```

```
    Next i
End If


Case Is = 4
If contador1 <= 1 Then
k = contador1 + 1
    contador1 = contador1 + 1
    For i = 0 To n - 1
        For j = 0 To n
        fgSELiteraciones.TextMatrix(i + 1, j + 1) = r(i, j, k)
        Next j
    Next i
End If



Case Is = 5
If contador1 <= 1 Then
k = contador1 + 1
    contador1 = contador1 + 1
    For i = 0 To n - 1
        For j = 0 To n
        fgSELiteraciones.TextMatrix(i + 1, j + 1) = r(i, j, k)
        Next j
    Next i
End If

Case Is = 6
Case Is = 7
Case Is = 8
Case Else
End Select
Exit Sub
err1:
End Sub

Private Sub CmdSELatras_Click()
Dim i As Integer
Dim j As Integer
Dim k As Integer
Dim valor As Double
On Error GoTo err1
Select Case (tipoalg)
Case Is = 1
Select Case (contador1)
```

```
Case Is = 0
contador1 = contador1 - 1
For i = 0 To n - 1
  For j = 0 To n
    fgSELiteraciones.TextMatrix(i + 1, j + 1) = Fg2.TextMatrix(i + 1, j + 1)
    Next j
Next i
Case Is <= -1
Exit Sub
Case Else

k = contador1 - 1
 contador1 = contador1 - 1
    For i = 0 To n - 1
       For j = 0 To n

       valor = r(i, j, k)
       If (valor <= 0 Or valor > 0.000000000000001) Then
       fgSELiteraciones.TextMatrix(i + 1, j + 1) = valor
       Else
       fgSELiteraciones.TextMatrix(i + 1, j + 1) = 0
       End If
       'fgSELiteraciones.TextMatrix(i + 1, j + 1) = r(i, j, k)
       Next j
    Next i
End Select



Case Is = 2
Select Case (contador1)

Case Is = 0
contador1 = contador1 - 1
For i = 0 To n - 1
  For j = 0 To n
    fgSELiteraciones.TextMatrix(i + 1, j + 1) = Fg2.TextMatrix(i + 1, j + 1)
    Next j
Next i
Case Is <= -1
Exit Sub
Case Else

k = contador1 - 1
```

```
 contador1 = contador1 - 1
   For i = 0 To n - 1
      For j = 0 To n
      valor = r(i, j, k)
      If (valor <= 0 Or valor > 0.000000000000001) Then
      fgSELiteraciones.TextMatrix(i + 1, j + 1) = valor
      Else
      fgSELiteraciones.TextMatrix(i + 1, j + 1) = 0
      End If
      'fgSELiteraciones.TextMatrix(i + 1, j + 1) = r(i, j, k)
      Next j
   Next i
End Select

Case Is = 3
Select Case (contador1)

Case Is = 0
contador1 = contador1 - 1
For i = 0 To n - 1
  For j = 0 To n
   fgSELiteraciones.TextMatrix(i + 1, j + 1) = Fg2.TextMatrix(i + 1, j + 1)
   Next j
Next i
Case Is <= -1
Exit Sub
Case Else

k = contador1 - 1
 contador1 = contador1 - 1
   For i = 0 To n - 1
      For j = 0 To n
      valor = r(i, j, k)
      If (valor <= 0 Or valor > 0.000000000000001) Then
      fgSELiteraciones.TextMatrix(i + 1, j + 1) = valor
      Else
      fgSELiteraciones.TextMatrix(i + 1, j + 1) = 0
      End If
      'fgSELiteraciones.TextMatrix(i + 1, j + 1) = r(i, j, k)
      Next j
   Next i
End Select

Case Is = 4
Select Case (contador1)
```

```
Case Is = 0
contador1 = contador1 - 1
For i = 0 To n - 1
  For j = 0 To n
    fgSELiteraciones.TextMatrix(i + 1, j + 1) = Fg2.TextMatrix(i + 1, j + 1)
    Next j
Next i
Case Is <= -1
Exit Sub
Case Else

k = contador1 - 1
 contador1 = contador1 - 1
    For i = 0 To n - 1
       For j = 0 To n
       valor = r(i, j, k)
       If (valor <= 0 Or valor > 0.000000000000001) Then
       fgSELiteraciones.TextMatrix(i + 1, j + 1) = valor
       Else
       fgSELiteraciones.TextMatrix(i + 1, j + 1) = 0
       End If

       'fgSELiteraciones.TextMatrix(i + 1, j + 1) = r(i, j, k)
       Next j
    Next i
End Select



Case Is = 5
Select Case (contador1)

Case Is = 0
contador1 = contador1 - 1
For i = 0 To n - 1
  For j = 0 To n
    fgSELiteraciones.TextMatrix(i + 1, j + 1) = Fg2.TextMatrix(i + 1, j + 1)
    Next j
Next i
Case Is <= -1
Exit Sub
Case Else

k = contador1 - 1
 contador1 = contador1 - 1
    For i = 0 To n - 1
```

```
        For j = 0 To n
        valor = r(i, j, k)
        If (valor <= 0 Or valor > 0.000000000000001) Then
        fgSELiteraciones.TextMatrix(i + 1, j + 1) = valor
        Else
        fgSELiteraciones.TextMatrix(i + 1, j + 1) = 0
        End If

        'fgSELiteraciones.TextMatrix(i + 1, j + 1) = r(i, j, k)
        Next j
    Next i
End Select


Case Is = 6
Case Is = 7
Case Is = 8
Case Else
End Select
Exit Sub
err1:
End Sub

Private Sub CmdSELcalcular_Click()
Dim limitesup As Integer
On Error GoTo err1
Select Case (tipoalg)

Case Is = 1
Enviar
If aviso = True Then
aviso = False
Exit Sub
End If

SEL.EliGaussSimple
'Asignamos los resultados parciales a la matriz de resultados local R

ReDim r(0 To n - 1, 0 To n, 0 To n - 2)
For k = 0 To n - 2
    For j = 0 To n
        For i = 0 To n - 1
r(i, j, k) = SEL.ReturSELMR(i, j, k)
        Next i
    Next j
Next k
```

```
' Asignamos los resultados a la matriz local XR
ReDim XR(0 To n - 1)
For i = 0 To n - 1
XR(i) = SEL.ReturSELX(i)
Next i

SELpresentar
fgSELresultados.Visible = True
CmdSELiteraciones.Visible = True
Case Is = 2
Enviar
If aviso = True Then
aviso = False
Exit Sub
End If

SEL.EliGaussPivotación
'Asignamos los resultados parciales a la matriz de resultados local R
limitesup = ((n - 1) * 2) - 1
ReDim r(0 To n - 1, 0 To n, 0 To limitesup)
For k = 0 To limitesup
   For j = 0 To n
      For i = 0 To n - 1
r(i, j, k) = SEL.ReturSELMR(i, j, k)
      Next i
   Next j
Next k
' Asignamos los resultados a la matriz local XR
ReDim XR(0 To n - 1)
For i = 0 To n - 1
XR(i) = SEL.ReturSELX(i)
Next i

SELpresentar
fgSELresultados.Visible = True
CmdSELiteraciones.Visible = True
Case Is = 3
Enviar
If aviso = True Then
aviso = False

Exit Sub
End If

SEL.GaussJordan
'Asignamos los resultados parciales a la matriz de resultados local R
```

```
ReDim r(0 To n - 1, 0 To n, 0 To n - 1)
For k = 0 To n - 1
   For j = 0 To n
      For i = 0 To n - 1
r(i, j, k) = SEL.ReturSELMR(i, j, k)
      Next i
   Next j
Next k
' Asignamos los resultados a la matriz local XR
ReDim XR(0 To n - 1)
For i = 0 To n - 1
XR(i) = SEL.ReturSELX(i)
Next i

SELpresentar
fgSELresultados.Visible = True
CmdSELiteraciones.Visible = True
Case Is = 4
Enviar
If aviso = True Then
aviso = False

Exit Sub
End If

SEL.Dollittle
'Asignamos los resultados parciales a la matriz de resultados local R

ReDim r(0 To n - 1, 0 To n, 0 To 1)
For k = 0 To 1
   For j = 0 To n
      For i = 0 To n - 1
r(i, j, k) = SEL.ReturSELMR(i, j, k)
      Next i
   Next j
Next k
' Asignamos los resultados a la matriz local XR
ReDim XR(0 To n - 1)
For i = 0 To n - 1
XR(i) = SEL.ReturSELX(i)
Next i

SELpresentar
fgSELresultados.Visible = True
CmdSELiteraciones.Visible = True
Case Is = 5
```

```
Enviar
If aviso = True Then
aviso = False

Exit Sub
End If

SEL.Cholesky
'Asignamos los resultados parciales a la matriz de resultados local R

ReDim r(0 To n - 1, 0 To n, 0 To 1)
For k = 0 To 1
   For j = 0 To n
      For i = 0 To n - 1
r(i, j, k) = SEL.ReturSELMR(i, j, k)
      Next i
   Next j
Next k
' Asignamos los resultados a la matriz local XR
ReDim XR(0 To n - 1)
For i = 0 To n - 1
XR(i) = SEL.ReturSELX(i)
Next i

SELpresentar
fgSELresultados.Visible = True
CmdSELiteraciones.Visible = True
Case Is = 6
err = OptSELEa
Enviar
If aviso = True Then
aviso = False

Exit Sub
End If

SEL.Jacobi
'Asignamos los resultados parciales a la matriz de resultados local R

ReDim r(0 To itermax - 1, 0 To n, 0)

   For j = 0 To n
      For i = 0 To itermax - 1
r(i, j, 0) = SEL.ReturSELMR(i, j, 0)
      Next i
   Next j
```

```
' Asignamos los resultados a la matriz local XR
ReDim XR(0 To n - 1)
For i = 0 To n - 1
XR(i) = SEL.ReturSELX(i)
Next i

SELpresentar
fgSELresultados.Visible = True
CmdSELiteraciones.Visible = True

Case Is = 7
err = OptSELEa
Enviar
If aviso = True Then
aviso = False

Exit Sub
End If

SEL.GaussSeidel
'Asignamos los resultados parciales a la matriz de resultados local R

ReDim r(0 To itermax - 1, 0 To n, 0)

   For j = 0 To n
      For i = 0 To itermax - 1
r(i, j, 0) = SEL.ReturSELMR(i, j, 0)
      Next i
   Next j

' Asignamos los resultados a la matriz local XR
ReDim XR(0 To n - 1)
For i = 0 To n - 1
XR(i) = SEL.ReturSELX(i)
Next i

SELpresentar
fgSELresultados.Visible = True
CmdSELiteraciones.Visible = True



Case Is = 8
err = OptSELEa
Enviar
```

```
If aviso = True Then
aviso = False

Exit Sub
End If

SEL.JacobiSNL
'Asignamos los resultados parciales a la matriz de resultados local R

ReDim r(0 To itermax - 1, 0 To n, 0)

   For j = 0 To n
      For i = 0 To itermax - 1
r(i, j, 0) = SEL.ReturSELMR(i, j, 0)
      Next i
   Next j

' Asignamos los resultados a la matriz local XR
ReDim XR(0 To n - 1)
For i = 0 To n - 1
XR(i) = SEL.ReturSELX(i)
Next i

SELpresentar
fgSELresultados.Visible = True
CmdSELiteraciones.Visible = True

Case Is = 9
err = OptSELEa
Enviar
If aviso = True Then
aviso = False

Exit Sub
End If

SEL.GaussSeidelSNL
'Asignamos los resultados parciales a la matriz de resultados local R

ReDim r(0 To itermax - 1, 0 To n, 0)

   For j = 0 To n
      For i = 0 To itermax - 1
r(i, j, 0) = SEL.ReturSELMR(i, j, 0)
      Next i
   Next j
```

```
' Asignamos los resultados a la matriz local XR
ReDim XR(0 To n - 1)
For i = 0 To n - 1
XR(i) = SEL.ReturSELX(i)
Next i

SELpresentar
fgSELresultados.Visible = True
CmdSELiteraciones.Visible = True

Case Else
err = OptSELEa
Enviar
If aviso = True Then
aviso = False

Exit Sub
End If

SEL.NewtonSNL
'Asignamos los resultados parciales a la matriz de resultados local R

ReDim r(0 To itermax - 1, 0 To n, 0)

   For j = 0 To n
      For i = 0 To itermax - 1
r(i, j, 0) = SEL.ReturSELMR(i, j, 0)
      Next i
   Next j

' Asignamos los resultados a la matriz local XR
ReDim XR(0 To n - 1)
For i = 0 To n - 1
XR(i) = SEL.ReturSELX(i)
Next i

SELpresentar
fgSELresultados.Visible = True
CmdSELiteraciones.Visible = True


End Select
SNLimprimir.Enabled = True
Exit Sub
err1:
```

```
End Sub

Private Sub CmdSELiteraciones_Click()
Label20.Visible = True
contador1 = -1
If tipoalg > 5 Then
fgSELiteraciones.Visible = True
Else
fgSELiteraciones.Visible = True
CmdSELatras.Visible = True
CmdSELadelante.Visible = True
End If
End Sub

Private Sub Command1_Click()
   PLImprimir
End Sub

Sub Fg2_KeyPress(KeyAscii As Integer)
  MSHFlexGridEdit Fg2, TxtEdit, KeyAscii
End Sub

Sub Fg2_DblClick()
  MSHFlexGridEdit Fg2, TxtEdit, 32 ' Simula un espacio.
End Sub

Sub MSHFlexGridEdit(MSHFlexGrid As Control, _
Edt As Control, KeyAscii As Integer)

  ' Usar el carácter escrito.
  Select Case KeyAscii

  ' Un espacio significa modificar el texto actual.
  Case 0 To 32
    Edt = MSHFlexGrid
    Edt.SelStart = 1000

  ' Otro carácter reemplaza el texto actual.
  Case Else
    Edt = Chr(KeyAscii)
    Edt.SelStart = 1
  End Select

  ' Mostrar Edt en la posición correcta.
  Edt.Move MSHFlexGrid.Left + MSHFlexGrid.CellLeft, _
    MSHFlexGrid.Top + MSHFlexGrid.CellTop, _
```

```
      MSHFlexGrid.CellWidth - 8, _
      MSHFlexGrid.CellHeight - 8
   Edt.Visible = True

   ' Y hacer que funcione.
   Edt.SetFocus
End Sub



Private Sub Form_Load()
   SNLabril.Enabled = False
   SNLguardarcomo.Enabled = False
   SNLimprimir.Enabled = False
   VGForma = "SEL"
End Sub




Private Sub SNLabril_Click()
   FrmRecuperaTrabajo.Show vbModal
   If Acepta = True Then
      PLConexion True
      PLRecupera
   End If
End Sub

Private Sub SNlayuda_Click()
'Así se llamaría para mostrar un tópico de la ayuda
   Dim h As Long

   'h = HtmlHelp(Me.hWnd, "Milton.hlp", HH_DISPLAY_TOPIC, 0&)

   h = HtmlHelp(Me.hWnd, "Milton.hlp", HH_HELP_CONTEXT, 2040&)
End Sub

Private Sub SNLGR_Click()
Frmescalas.Show
End Sub

Private Sub SNLguardarcomo_Click()
   FrmNombre.Show vbModal
   If Acepta = True Then
      PLConexion True
```

```
      PLGuardarBDD
   End If
End Sub

Private Sub SNLimprimir_Click()
   Dim EndTime As Date
   Sincolor

      EndTime = DateAdd("s", 1 / 1E+16, Now)
      Do Until Now > EndTime
        DoEvents
      Loop
   'Form1.Command3_Click
   Set Form1.Picture1.Picture = CaptureClient(Me)
   Form1.Visible = True
   FrmPrincipalalgoritmos.Visible = False
   Concolor
End Sub

Private Sub SNLsalir_Click()
Unload Me
End Sub

Private Sub SNLSM_Click()
'Seteamos la variable tipoalg para indicar que tipo de algoritmos es
tipoalg = 1
   SNLabril.Enabled = True
   SNLguardarcomo.Enabled = True
   FrmRecuperaTrabajo.ver = tipoalg
' Borramos y presentamos todos los contorles necesarios en el formulario
LblSELtitulo.Caption = "Ingrese los coeficientes de las ecuaciones a resolver"
LblSELtitulo.Visible = False
LblSELnumecuaciones.Visible = True
LblSELtolerancia.Visible = False
LblSELiteraciones.Visible = False
TxtSELnumecuaciones.Visible = True
TxtSELtolerancia.Text = " "
TxtSELtolerancia.Visible = False
TxtSELiteraciones.Text = " "
TxtSELiteraciones.Visible = False
CmdSELcalcular.Visible = True
CmdSELiteraciones.Visible = False
CmdSELgraficar.Visible = False
FraSELerrores.Visible = False
OptSELEa.Visible = False
OptSELEr.Visible = False
```

```
Fg2.Visible = False
Fg2.Clear
TxtEdit.Visible = False
fgSELiteraciones.Clear
fgSELiteraciones.Visible = False
fgSELresultados.Clear
CmdSELadelante.Visible = False
CmdSELatras.Visible = False
LblNomalg.Caption = "Algoritmo de Eliminación Gaussiana Simple"
LblSelSintaxis.Caption = "Número de ecuaciones= número natural, representa el
número de ecuaciones a resolver" & Chr(10) & "X1,X2,..,Xn son las variables de las
ecuaciones a resolver se debe ingresar los coeficientes en la variable
correspondiente"
Label20.Visible = False
Label20.Caption = "Calculos"
CmdSELiteraciones.Caption = "Calculos"
End Sub
Private Sub SNLPV_Click()
'Seteamos la variable tipoalg para indicar que tipo de algoritmos es
tipoalg = 2
   SNLabril.Enabled = True
   SNLguardarcomo.Enabled = True
   FrmRecuperaTrabajo.ver = tipoalg
' Borramos y presentamos todos los contorles necesarios en el formulario
LblSELtitulo.Caption = "Ingrese los coeficientes de las ecuaciones a resolver"
LblSELtitulo.Visible = False
LblSELnumecuaciones.Visible = True
LblSELtolerancia.Visible = False
LblSELiteraciones.Visible = False
TxtSELnumecuaciones.Visible = True
TxtSELtolerancia.Text = " "
TxtSELtolerancia.Visible = False
TxtSELiteraciones.Text = " "
TxtSELiteraciones.Visible = False
CmdSELcalcular.Visible = True
CmdSELiteraciones.Visible = False
CmdSELgraficar.Visible = False
FraSELerrores.Visible = False
OptSELEa.Visible = False
OptSELEr.Visible = False
Fg2.Visible = False
Fg2.Clear
TxtEdit.Visible = False
fgSELiteraciones.Clear
fgSELiteraciones.Visible = False
fgSELresultados.Clear
```

```
CmdSELadelante.Visible = False
CmdSELatras.Visible = False
LblNomalg.Caption = "Algoritmo de Eliminación Gaussiana con Pivotación"
LblSelSintaxis.Caption = "Número de ecuaciones= número natural, representa el
número de ecuaciones a resolver" & Chr(10) & "X1,X2,..,Xn son las variables de las
ecuaciones a resolver se debe ingresar los coeficientes en la variable
correspondiente"
Label20.Visible = False
Label20.Caption = "Calculos"
CmdSELiteraciones.Caption = "Calculos"
End Sub
Private Sub SNLGJ_Click()
'Seteamos la variable tipoalg para indicar que tipo de algoritmos es
tipoalg = 3
   SNLabril.Enabled = True
   SNLguardarcomo.Enabled = True
   FrmRecuperaTrabajo.ver = tipoalg
' Borramos y presentamos todos los contorles necesarios en el formulario
LblSELtitulo.Caption = "Ingrese los coeficientes de las ecuaciones a resolver"
LblSELtitulo.Visible = False
LblSELnumecuaciones.Visible = True
LblSELtolerancia.Visible = False
LblSELiteraciones.Visible = False
TxtSELnumecuaciones.Visible = True
TxtSELtolerancia.Text = " "
TxtSELtolerancia.Visible = False
TxtSELiteraciones.Text = " "
TxtSELiteraciones.Visible = False
CmdSELcalcular.Visible = True
CmdSELiteraciones.Visible = False
CmdSELgraficar.Visible = False
FraSELerrores.Visible = False
OptSELEa.Visible = False
OptSELEr.Visible = False
Fg2.Visible = False
Fg2.Clear
TxtEdit.Visible = False
fgSELiteraciones.Clear
fgSELiteraciones.Visible = False
fgSELresultados.Clear
CmdSELadelante.Visible = False
CmdSELatras.Visible = False
LblNomalg.Caption = "Algoritmo de Gauss-Jordan"
LblSelSintaxis.Caption = "Número de ecuaciones= número natural, representa el
número de ecuaciones a resolver" & Chr(10) & "X1,X2,..,Xn son las variables de las
```

```
ecuaciones a resolver se debe ingresar los coeficientes en la variable
correspondiente"
Label20.Visible = False
Label20.Caption = "Calculos"
CmdSELiteraciones.Caption = "Calculos"
End Sub
Private Sub SNLDL_Click()
'Seteamos la variable tipoalg para indicar que tipo de algoritmos es
tipoalg = 4
   SNLabril.Enabled = True
   SNLguardarcomo.Enabled = True
   FrmRecuperaTrabajo.ver = tipoalg
' Borramos y presentamos todos los contorles necesarios en el formulario
LblSELtitulo.Caption = "Ingrese los coeficientes de las ecuaciones a resolver"
LblSELtitulo.Visible = False
LblSELnumecuaciones.Visible = True
LblSELtolerancia.Visible = False
LblSELiteraciones.Visible = False
TxtSELnumecuaciones.Visible = True
TxtSELtolerancia.Text = " "
TxtSELtolerancia.Visible = False
TxtSELiteraciones.Text = " "
TxtSELiteraciones.Visible = False
CmdSELcalcular.Visible = True
CmdSELiteraciones.Visible = False
CmdSELgraficar.Visible = False
FraSELerrores.Visible = False
OptSELEa.Visible = False
OptSELEr.Visible = False
Fg2.Visible = False
Fg2.Clear
TxtEdit.Visible = False
fgSELiteraciones.Clear
fgSELiteraciones.Visible = False
fgSELresultados.Clear
CmdSELadelante.Visible = False
CmdSELatras.Visible = False
LblNomalg.Caption = "Algoritmo de Dollittle"
LblSelSintaxis.Caption = "Número de ecuaciones= número natural, representa el
número de ecuaciones a resolver" & Chr(10) & "X1,X2,..,Xn son las variables de las
ecuaciones a resolver se debe ingresar los coeficientes en la variable
correspondiente"
Label20.Visible = False
Label20.Caption = "Calculos"
CmdSELiteraciones.Caption = "Calculos"
End Sub
```

```
Private Sub SNLCH_Click()
'Seteamos la variable tipoalg para indicar que tipo de algoritmos es
tipoalg = 5
   SNLabril.Enabled = True
   SNLguardarcomo.Enabled = True
   FrmRecuperaTrabajo.ver = tipoalg
' Borramos y presentamos todos los contorles necesarios en el formulario
LblSELtitulo.Caption = "Ingrese los coeficientes de las ecuaciones a resolver"
LblSELtitulo.Visible = False
LblSELnumecuaciones.Visible = True
LblSELtolerancia.Visible = False
LblSELiteraciones.Visible = False
TxtSELnumecuaciones.Visible = True
TxtSELtolerancia.Text = " "
TxtSELtolerancia.Visible = False
TxtSELiteraciones.Text = " "
TxtSELiteraciones.Visible = False
CmdSELcalcular.Visible = True
CmdSELiteraciones.Visible = False
CmdSELgraficar.Visible = False
FraSELerrores.Visible = False
OptSELEa.Visible = False
OptSELEr.Visible = False
Fg2.Clear
Fg2.Visible = False
TxtEdit.Visible = False
fgSELiteraciones.Visible = False
fgSELiteraciones.Clear
fgSELresultados.Clear
CmdSELadelante.Visible = False
CmdSELatras.Visible = False
LblNomalg.Caption = "Algoritmo de Choleski"
LblSelSintaxis.Caption = "Número de ecuaciones= número natural, representa el
número de ecuaciones a resolver" & Chr(10) & "X1,X2,..,Xn son las variables de las
ecuaciones a resolver se debe ingresar los coeficientes en la variable
correspondiente"
Label20.Visible = False
Label20.Caption = "Calculos"
CmdSELiteraciones.Caption = "Calculos"
End Sub


Private Sub SNLJA_Click()
'Seteamos la variable tipoalg para indicar que tipo de algoritmos es
tipoalg = 6
   SNLabril.Enabled = True
```

```
    SNLguardarcomo.Enabled = True
    FrmRecuperaTrabajo.ver = tipoalg

' Borramos y presentamos todos los contorles necesarios en el formulario
LblSELtitulo.Caption = "Ingrese los coeficientes de las ecuaciones a resolver"
LblSELtitulo.Visible = False
LblSELnumecuaciones.Visible = True
LblSELtolerancia.Visible = True
LblSELiteraciones.Visible = True
TxtSELnumecuaciones.Visible = True
TxtSELtolerancia.Text = " "
TxtSELtolerancia.Visible = True
TxtSELiteraciones.Text = " "
TxtSELiteraciones.Visible = True
CmdSELcalcular.Visible = True
CmdSELiteraciones.Visible = False
CmdSELgraficar.Visible = False
FraSELerrores.Visible = True
OptSELEa.Visible = True
OptSELEr.Visible = True
Fg2.Clear
Fg2.Visible = False
TxtEdit.Visible = False
fgSELiteraciones.Visible = False
fgSELiteraciones.Clear
fgSELresultados.Clear

CmdSELadelante.Visible = False
CmdSELatras.Visible = False
LblNomalg.Caption = "Algoritmo de Jacobi"
LblSelSintaxis.Caption = "Número de ecuaciones= número natural, representa el
número de ecuaciones a resolver" & Chr(10) & "X1,X2,..,Xn son las variables de las
ecuaciones a resolver se debe ingresar los coeficientes en la variable
correspondiente" & Chr(10) & _
"Vector inicial= valores para X1,X2,..,Xn cercanos a la solución del sistema" &
Chr(10) & " Tolerancia =0.001" & Chr(10) & "Iteraciones máximas= número natural,
representa el número de iteraciones que el algoritmo realizará" & Chr(10) & " Ea=
error absoluto; Er= error realtivo"
Label20.Visible = False
Label20.Caption = "Iteraciones"
CmdSELiteraciones.Caption = "Iteraciones"
End Sub

Private Sub SNLGS_Click()
'Seteamos la variable tipoalg para indicar que tipo de algoritmos es
tipoalg = 7
```

```
    SNLabril.Enabled = True
    SNLguardarcomo.Enabled = True
    FrmRecuperaTrabajo.ver = tipoalg

' Borramos y presentamos todos los contorles necesarios en el formulario
LblSELtitulo.Caption = "Ingrese los coeficientes de las ecuaciones a resolver"
LblSELtitulo.Visible = False
LblSELnumecuaciones.Visible = True
LblSELtolerancia.Visible = True
LblSELiteraciones.Visible = True
TxtSELnumecuaciones.Visible = True
TxtSELtolerancia.Text = " "
TxtSELtolerancia.Visible = True
TxtSELiteraciones.Text = " "
TxtSELiteraciones.Visible = True
CmdSELcalcular.Visible = True
CmdSELiteraciones.Visible = False
CmdSELgraficar.Visible = False
FraSELerrores.Visible = True
OptSELEa.Visible = True
OptSELEr.Visible = True
Fg2.Clear
Fg2.Visible = False
TxtEdit.Visible = False
fgSELiteraciones.Visible = False
fgSELiteraciones.Clear
fgSELresultados.Clear
CmdSELadelante.Visible = False
CmdSELatras.Visible = False
LblNomalg.Caption = "Algoritmo de Gauss-Seidel"
LblSelSintaxis.Caption = "Número de ecuaciones= número natural, representa el
número de ecuaciones a resolver" & Chr(10) & "X1,X2,..,Xn son las variables de las
ecuaciones a resolver se debe ingresar los coeficientes en la variable
correspondiente" & Chr(10) & _
"Vector inicial= valores para X1,X2,..,Xn cercanos a la solución del sistema" &
Chr(10) & " Tolerancia =0.001" & Chr(10) & "Iteraciones máximas= número natural,
representa el número de iteraciones que el algoritmo realizará" & Chr(10) & " Ea=
error absoluto; Er= error realtivo"
Label20.Visible = False
Label20.Caption = "Iteraciones"
CmdSELiteraciones.Caption = "Iteraciones"
End Sub
Private Sub SNLNJAi_Click()
'Seteamos la variable tipoalg para indicar que tipo de algoritmos es
tipoalg = 8
    SNLabril.Enabled = True
```

```
    SNLguardarcomo.Enabled = True
    FrmRecuperaTrabajo.ver = tipoalg

' Borramos y presentamos todos los contorles necesarios en el formulario
LblSELtitulo.Caption = "Ingrese las ecuaciones despejadas y el vector inicial"
LblSELtitulo.Visible = False
LblSELnumecuaciones.Visible = True
LblSELtolerancia.Visible = True
LblSELiteraciones.Visible = True
TxtSELnumecuaciones.Visible = True
TxtSELtolerancia.Text = " "
TxtSELtolerancia.Visible = True
TxtSELiteraciones.Text = " "
TxtSELiteraciones.Visible = True
CmdSELcalcular.Visible = True
CmdSELiteraciones.Visible = False
CmdSELgraficar.Visible = False
FraSELerrores.Visible = True
OptSELEa.Visible = True
OptSELEr.Visible = True
Fg2.Clear
Fg2.Visible = False
TxtEdit.Visible = False
fgSELiteraciones.Visible = False
fgSELiteraciones.Clear
fgSELresultados.Clear
CmdSELadelante.Visible = False
CmdSELatras.Visible = False
LblNomalg.Caption = "Algoritmo de Jacobi(SENL)"
LblSelSintaxis.Caption = "Número de ecuaciones= número natural, representa el
número de ecuaciones a resolver" & Chr(10) & "Xn= es la ecuación En despejada
para la Xn variable (n=1,2,..)" & Chr(10) & _
"Vector inicial= valores para X1,X2,..,Xn cercanos a la solución del sistema" &
Chr(10) & " Tolerancia =0.001" & Chr(10) & "Iteraciones máximas= número natural,
representa el número de iteraciones que el algoritmo realizará" & Chr(10) & " Ea=
error absoluto; Er= error realtivo"
Label20.Visible = False
Label20.Caption = "Iteraciones"
CmdSELiteraciones.Caption = "Iteraciones"
End Sub
Private Sub SNLNGS_Click()
'Seteamos la variable tipoalg para indicar que tipo de algoritmos es
tipoalg = 9
    SNLabril.Enabled = True
    SNLguardarcomo.Enabled = True
    FrmRecuperaTrabajo.ver = tipoalg
```

```
' Borramos y presentamos todos los contorles necesarios en el formulario
LblSELtitulo.Caption = "Ingrese las ecuaciones despejadas y el vector inicial"
LblSELtitulo.Visible = False
LblSELnumecuaciones.Visible = True
LblSELtolerancia.Visible = True
LblSELiteraciones.Visible = True
TxtSELnumecuaciones.Visible = True
TxtSELtolerancia.Text = " "
TxtSELtolerancia.Visible = True
TxtSELiteraciones.Text = " "
TxtSELiteraciones.Visible = True
CmdSELcalcular.Visible = True
CmdSELiteraciones.Visible = False
CmdSELgraficar.Visible = False
FraSELerrores.Visible = True
OptSELEa.Visible = True
OptSELEr.Visible = True
Fg2.Clear
Fg2.Visible = False
TxtEdit.Visible = False
fgSELiteraciones.Visible = False
fgSELiteraciones.Clear
fgSELresultados.Clear
CmdSELadelante.Visible = False
CmdSELatras.Visible = False
LblNomalg.Caption = "Algoritmo de Gauss-Seidel (SENL)"
LblSelSintaxis.Caption = "Número de ecuaciones= número natural, representa el
número de ecuaciones a resolver" & Chr(10) & "Xn= es la ecuación En despejada
para la Xn variable (n=1,2,..)" & Chr(10) & _
"Vector inicial= valores para X1,X2,..,Xn cercanos a la solución del sistema" &
Chr(10) & " Tolerancia =0.001" & Chr(10) & "Iteraciones máximas= número natural,
representa el número de iteraciones que el algoritmo realizará" & Chr(10) & " Ea=
error absoluto; Er= error realtivo"
Label20.Visible = False
Label20.Caption = "Iteraciones"
CmdSELiteraciones.Caption = "Iteraciones"
End Sub
Private Sub SNLNW_Click()
'Seteamos la variable tipoalg para indicar que tipo de algoritmos es
tipoalg = 10
   SNLabril.Enabled = True
   SNLguardarcomo.Enabled = True
   FrmRecuperaTrabajo.ver = tipoalg

' Borramos y presentamos todos los contorles necesarios en el formulario
```

```
LblSELtitulo.Caption = "Ingrese ingrese el Jacobiano, las ecuaciones y el vector
inicial"
LblSELtitulo.Visible = False
LblSELnumecuaciones.Visible = True
LblSELtolerancia.Visible = True
LblSELiteraciones.Visible = True
TxtSELnumecuaciones.Visible = True
TxtSELtolerancia.Text = " "
TxtSELtolerancia.Visible = True
TxtSELiteraciones.Text = " "
TxtSELiteraciones.Visible = True
CmdSELcalcular.Visible = True
CmdSELiteraciones.Visible = False
CmdSELgraficar.Visible = False
FraSELerrores.Visible = True
OptSELEa.Visible = True
OptSELEr.Visible = True
Fg2.Clear
Fg2.Visible = False
TxtEdit.Visible = False
fgSELiteraciones.Visible = False
fgSELiteraciones.Clear
fgSELresultados.Clear
CmdSELadelante.Visible = False
CmdSELatras.Visible = False
LblNomalg.Caption = "Algoritmo de Newton (SENL)"
LblSelSintaxis.Caption = "Número de ecuaciones= número natural, representa el
número de ecuaciones a resolver" & Chr(10) & "En:df/Xn= es la ecuación derivada
para la Xn variable (n=1,2,..)" & Chr(10) & "En:f(X1,X2,..,Xn)= es la ecuación En
(n=1, 2,... )" & Chr(10) & _
"Vector inicial= valores para X1,X2,..,Xn cercanos a la solución del sistema" &
Chr(10) & " Tolerancia =0.001" & Chr(10) & "Iteraciones máximas= número natural,
representa el número de iteraciones que el algoritmo realizará" & Chr(10) & " Ea=
error absoluto; Er= error realtivo"
Label20.Visible = False
Label20.Caption = "Iteraciones"
CmdSELiteraciones.Caption = "Iteraciones"
End Sub
Sub txtEdit_KeyPress(KeyAscii As Integer)



If tipoalg < 8 Then
'filtra la entrada de datos que no sean números incluye el signo - y el punto
  If (KeyAscii < 48 Or KeyAscii > 57) Then
      If (KeyAscii = 45) Then
```

```
      Exit Sub
      End If
      If (KeyAscii <> 46) Then
         If (KeyAscii <> 8) Then
         KeyAscii = 0
         End If
      End If
   End If
   If KeyAscii = Asc(vbCr) Then ' Elimina los retornos para quitar los pitidos.
   KeyAscii = 0
   End If

Else
    ' Elimina los retornos para quitar los pitidos.
    If KeyAscii = Asc(vbCr) Then KeyAscii = 0

End If
End Sub

Sub txtEdit_KeyDown(KeyCode As Integer, _
Shift As Integer)
   EditKeyCode Fg2, TxtEdit, KeyCode, Shift
End Sub

Sub EditKeyCode(MSHFlexGrid As Control, Edt As _
Control, KeyCode As Integer, Shift As Integer)

   ' Procesamiento del control de edición estándar.
   Select Case KeyCode

   Case 27   ' ESC: ocultar, devuelve el enfoque a          ' MSFlexGrid.
      Edt.Visible = False
      MSHFlexGrid.SetFocus

   Case 13   ' ENTRAR devuelve el enfoque a MSHFlexGrid.
      MSHFlexGrid.SetFocus

   Case 38      ' Arriba.
      MSHFlexGrid.SetFocus
      DoEvents
      If MSHFlexGrid.Row > MSHFlexGrid.FixedRows Then
         MSHFlexGrid.Row = MSHFlexGrid.Row - 1
      End If

   Case 40      ' Abajo.
      MSHFlexGrid.SetFocus
```

```
    DoEvents
    If MSHFlexGrid.Row < MSHFlexGrid.Rows - 1 Then
      MSHFlexGrid.Row = MSHFlexGrid.Row + 1
    End If
  End Select
End Sub

Sub Fg2_GotFocus()
  If TxtEdit.Visible = False Then Exit Sub

  Fg2 = TxtEdit
  TxtEdit.Visible = False
End Sub

Sub Fg2_LeaveCell()
  If TxtEdit.Visible = False Then Exit Sub

  Fg2 = TxtEdit
  TxtEdit.Visible = False
End Sub



Private Sub titulosenfg()
Dim i As Integer
On Error GoTo err1
 Select Case (tipoalg)

 Case Is = 1
  For i = Fg2.FixedRows To Fg2.Rows - 1
    Fg2.TextArray(Fgi(i, 0)) = "E" & i
  Next i
  For i = Fg2.FixedCols To Fg2.Cols - 1
    If i = Fg2.Cols - 1 Then
    Fg2.TextArray(Fgi(0, i)) = "b"
    Else
    Fg2.TextArray(Fgi(0, i)) = "X" & i
    End If
  Next i


Case Is = 2
For i = Fg2.FixedRows To Fg2.Rows - 1
    Fg2.TextArray(Fgi(i, 0)) = "E" & i
  Next i
  For i = Fg2.FixedCols To Fg2.Cols - 1
```

```
      If i = Fg2.Cols - 1 Then
      Fg2.TextArray(Fgi(0, i)) = "b"
      Else
      Fg2.TextArray(Fgi(0, i)) = "X" & i
      End If
    Next i

 Case Is = 3
 For i = Fg2.FixedRows To Fg2.Rows - 1
      Fg2.TextArray(Fgi(i, 0)) = "E" & i
    Next i
    For i = Fg2.FixedCols To Fg2.Cols - 1
      If i = Fg2.Cols - 1 Then
      Fg2.TextArray(Fgi(0, i)) = "b"
      Else
      Fg2.TextArray(Fgi(0, i)) = "X" & i
      End If
    Next i


 Case Is = 4
 For i = Fg2.FixedRows To Fg2.Rows - 1
      Fg2.TextArray(Fgi(i, 0)) = "E" & i
    Next i
    For i = Fg2.FixedCols To Fg2.Cols - 1
      If i = Fg2.Cols - 1 Then
      Fg2.TextArray(Fgi(0, i)) = "b"
      Else
      Fg2.TextArray(Fgi(0, i)) = "X" & i
      End If
    Next i

 Case Is = 5
 For i = Fg2.FixedRows To Fg2.Rows - 1
      Fg2.TextArray(Fgi(i, 0)) = "E" & i
    Next i
    For i = Fg2.FixedCols To Fg2.Cols - 1
      If i = Fg2.Cols - 1 Then
      Fg2.TextArray(Fgi(0, i)) = "b"
      Else
      Fg2.TextArray(Fgi(0, i)) = "X" & i
      End If
    Next i

 Case Is = 6
 For i = Fg2.FixedRows To Fg2.Rows - 1
```

```
    If i = Fg2.Rows - 1 Then
    Fg2.TextArray(Fgi(i, 0)) = "Vector Inicial"
    Else
    Fg2.TextArray(Fgi(i, 0)) = "E" & i
    End If
  Next i
  For i = Fg2.FixedCols To Fg2.Cols - 1
    If i = Fg2.Cols - 1 Then
    Fg2.TextArray(Fgi(0, i)) = "b"
    Else
    Fg2.TextArray(Fgi(0, i)) = "X" & i
    End If
  Next i

Case Is = 7
For i = Fg2.FixedRows To Fg2.Rows - 1
    If i = Fg2.Rows - 1 Then
    Fg2.TextArray(Fgi(i, 0)) = "Vector Inicial"
    Else
    Fg2.TextArray(Fgi(i, 0)) = "E" & i
    End If
  Next i
  For i = Fg2.FixedCols To Fg2.Cols - 1
    If i = Fg2.Cols - 1 Then
    Fg2.TextArray(Fgi(0, i)) = "b"
    Else
    Fg2.TextArray(Fgi(0, i)) = "X" & i
    End If
  Next i


Case Is = 8
Fg2.ColWidth(1) = 2300
  Fg2.ColAlignment(1) = 1   ' Centrado.
Fg2.ColWidth(2) = 1100
  Fg2.ColAlignment(2) = 1   ' Centrado.

For i = Fg2.FixedRows To Fg2.Rows - 1

    Fg2.TextArray(Fgi(i, 0)) = "X" & i & "="

  Next i
  For i = Fg2.FixedCols To Fg2.Cols - 1
    If i = Fg2.Cols - 1 Then
    Fg2.TextArray(Fgi(0, i)) = "Vector inicial"
    Else
```

```
      Fg2.TextArray(Fgi(0, i)) = "Ecuaciones despejadas"
      End If
    Next i
Case Is = 9
Fg2.ColWidth(1) = 2300
    Fg2.ColAlignment(1) = 1   ' Centrado.
Fg2.ColWidth(2) = 1100
    Fg2.ColAlignment(2) = 1   ' Centrado.

For i = Fg2.FixedRows To Fg2.Rows - 1

      Fg2.TextArray(Fgi(i, 0)) = "X" & i & "="

    Next i
    For i = Fg2.FixedCols To Fg2.Cols - 1
      If i = Fg2.Cols - 1 Then
      Fg2.TextArray(Fgi(0, i)) = "Vector inicial"
      Else
      Fg2.TextArray(Fgi(0, i)) = "Ecuaciones despejadas"
      End If
    Next i
Case Else
For i = 0 To Fg2.Cols - 1
Fg2.ColWidth(i) = 1100
    Fg2.ColAlignment(i) = 1   ' Centrado.
Next i

For i = Fg2.FixedRows To Fg2.Rows - 1
      If i = Fg2.Rows - 1 Then
      Fg2.TextArray(Fgi(i, 0)) = "Vector Inicial"
      Else
      Fg2.TextArray(Fgi(i, 0)) = "E" & i
      End If
    Next i
    For i = Fg2.FixedCols To Fg2.Cols - 1
      If i = Fg2.Cols - 1 Then
      Fg2.TextArray(Fgi(0, i)) = "f(X1,X2,..Xn)"
      Else
      Fg2.TextArray(Fgi(0, i)) = "df/dX" & i
      End If
    Next i

End Select
```

```
  ' Inicializar el cuadro de edición (lo carga ahora).
  TxtEdit = ""
Exit Sub
err1:
End Sub

Private Sub Enviar()
Dim i As Integer
Dim j As Integer
'Seteamos para los algoritmos que necesitan vector inicial
'Capturamos los valores de los texbox y del MSHFlexgrid, para asignarlos
'a las varibles generales del formulario para pasarlas al módulo ClasSEL
On Error GoTo err1

If tipoalg < 6 Then
   If TxtSELnumecuaciones = "" Then
   mensaje = MsgBox("Parámetros del algortimo incompletos", 48, "Error en el
ingreso")
   aviso = True
   Exit Sub
   End If
Else
   If (TxtSELnumecuaciones = "" Or TxtSELtolerancia = "" Or TxtSELiteraciones = "")
Then
   mensaje = MsgBox("Parámetros del algortimo incompletos", 48, "Error en el
ingreso")
   aviso = True
   Exit Sub
   End If
End If


If tipoalg > 5 Then
   If tipoalg <= 7 Then
   n = Val(TxtSELnumecuaciones.Text)
   ReDim RS(0) As String
   ReDim A(n, n - 1) As Double
   ReDim b(0 To n - 1) As Double
   ReDim Vinicial(0 To n - 1) As Double
   For i = 0 To n - 1
      For j = 0 To n - 1
      A(i, j) = Val(Fg2.TextMatrix(i + 1, j + 1))
      Next j
   Next i
   For i = 0 To n - 1
   b(i) = Val(Fg2.TextMatrix(i + 1, n + 1))
```

```
Next i
For j = 0 To n - 1
Vinicial(j) = Val(Fg2.TextMatrix(n + 1, j + 1))
Next j
err = OptSELEa.Value
tol = Val(TxtSELtolerancia.Text)
itermax = Val(TxtSELiteraciones.Text)
RS(0) = " "

'Pasamos los valores al módulo
SEL.setearSENL(n, tol, itermax, Vinicial(), RS(), err, A()) = b()
Else
   If tipoalg < 10 Then

   n = Val(TxtSELnumecuaciones.Text)
   ReDim RS(0 To n - 1) As String
   ReDim A(0) As Double
   ReDim b(0) As Double
   ReDim Vinicial(0 To n - 1) As Double

   A(0) = 0
   b(0) = 0

   For i = 0 To n - 1
   RS(i) = Fg2.TextMatrix(i + 1, 1)
   Next i
   For i = 0 To n - 1
   Vinicial(i) = Val(Fg2.TextMatrix(i + 1, 2))
   Next i
   err = OptSELEa.Value
   tol = Val(TxtSELtolerancia.Text)
   itermax = Val(TxtSELiteraciones.Text)


   'Pasamos los valores al módulo
   SEL.setearSENL(n, tol, itermax, Vinicial(), RS(), err, A()) = b()

   Else
   'algnewton
   n = Val(TxtSELnumecuaciones.Text)
   ReDim RS(0 To n, 0 To n) As String
   ReDim A(0) As Double
   ReDim b(0) As Double
   ReDim Vinicial(0 To n - 1) As Double

   A(0) = 0
```

```
        b(0) = 0

        For i = 0 To n - 1
          For j = 0 To n
          RS(i, j) = Fg2.TextMatrix(i + 1, j + 1)
          Next j
        Next i
        For j = 0 To n - 1
        Vinicial(j) = Val(Fg2.TextMatrix(n + 1, j + 1))
        Next j
        err = OptSELEa.Value
        tol = Val(TxtSELtolerancia.Text)
        itermax = Val(TxtSELiteraciones.Text)


        'Pasamos los valores al módulo
        SEL.setearSENL(n, tol, itermax, Vinicial(), RS(), err, A()) = b()

        End If
    End If
Else

'Seteamos para los algoritmos que no necesitan vector inicial
'Capturamos los valores de los texbox y del MSHFlexgrid, para asignarlos
'a las varibles generales del formulario para pasarlas al módulo ClasSEL
n = Val(TxtSELnumecuaciones.Text)
ReDim A(n - 1, n - 1) As Double
ReDim b(0 To n - 1) As Double

For i = 0 To n - 1
    For j = 0 To n - 1
    A(i, j) = Val(Fg2.TextMatrix(i + 1, j + 1))
    Next j
Next i
For i = 0 To n - 1
b(i) = Val(Fg2.TextMatrix(i + 1, n + 1))
Next i
err = OptSELEa.Value
tol = Val(TxtSELtolerancia.Text)
itermax = Val(TxtSELiteraciones.Text)
'Pasamos los valores al módulo
SEL.setearSEL(n, tol, itermax, err, A()) = b()
End If
Exit Sub
err1:
End Sub
```

```
Private Sub TxtSELiteraciones_KeyPress(KeyAscii As Integer)
If Val(TxtSELiteraciones) > 32767 Then
mensaje = MsgBox("Fuera del Rango", 48, "Error en el ingreso")
Exit Sub
End If

'filtra la entrada de datos que no sean números enteros y positivos
  If (KeyAscii < 48 Or KeyAscii > 57) Then

        If (KeyAscii <> 8) Then
        KeyAscii = 0
        End If

   End If
  If KeyAscii = Asc(vbCr) Then ' Elimina los retornos para quitar los pitidos.
  KeyAscii = 0
  End If

End Sub

Private Sub TxtSELnumecuaciones_Change()
On Error GoTo err1
LblSELtitulo.Visible = True
n = Val(TxtSELnumecuaciones.Text)

If n > 50 Then
mensaje = MsgBox("Fuera del rango", 48, "Error en el ingreso")
Exit Sub
End If
If n <> 0 Then
   If tipoalg > 5 Then
      If tipoalg <= 7 Then
      Fg2.Cols = n + 2
      Fg2.Rows = n + 2
      Fg2.FixedCols = 1
      Fg2.FixedRows = 1
      titulosenfg
      Fg2.Visible = True
      Else
         If tipoalg < 10 Then
         Fg2.Cols = 3
         Fg2.Rows = n + 1
         Fg2.FixedCols = 1
```

```
        Fg2.FixedRows = 1
        titulosenfg
        Fg2.Visible = True

        Else
        'algnewton
        Fg2.Cols = n + 2
        Fg2.Rows = n + 2
        Fg2.FixedCols = 1
        Fg2.FixedRows = 1
        titulosenfg
        Fg2.Visible = True
        End If
      End If
   Else
   Fg2.Cols = n + 2
   Fg2.Rows = n + 1
   Fg2.FixedCols = 1
   Fg2.FixedRows = 1
   titulosenfg
   Fg2.Visible = True
   End If
End If
Exit Sub
err1:
End Sub

Private Sub SELpresentar()
Dim i As Integer
Dim j As Integer
Dim valor As Double
 On Error GoTo err1

 Select Case (tipoalg)

 Case Is = 1
  'Presentación de los resultados
  fgSELresultados.Cols = 2
  fgSELresultados.Rows = n + 1
  fgSELresultados.FixedCols = 1
  fgSELresultados.FixedRows = 1

  For i = fgSELresultados.FixedRows To fgSELresultados.Rows - 1
    fgSELresultados.TextArray(Fgr(i, 0)) = "X" & i
  Next i
```

```
    fgSELresultados.TextArray(Fgi(0, 1)) = "Solución"

    For i = 0 To n - 1

    fgSELresultados.TextMatrix(i + 1, 1) = XR(i)
    Next i

   'Presentación de las iteraciones

   fgSELiteraciones.Cols = n + 2
   fgSELiteraciones.Rows = n + 1
   fgSELiteraciones.FixedCols = 1
   fgSELiteraciones.FixedRows = 1
   fgSELiteraciones.TextArray(Fgit(0, 0)) = ""
   For i = fgSELiteraciones.FixedRows To fgSELiteraciones.Rows - 1
     fgSELiteraciones.TextArray(Fgit(i, 0)) = "E" & i
   Next i
   For i = fgSELiteraciones.FixedCols To fgSELiteraciones.Cols - 1
     If i = fgSELiteraciones.Cols - 1 Then
     fgSELiteraciones.TextArray(Fgit(0, i)) = "b"
     Else
     fgSELiteraciones.TextArray(Fgit(0, i)) = "X" & i
     End If
   Next i
For i = 0 To n - 1
For j = 0 To n
    fgSELiteraciones.TextMatrix(i + 1, j + 1) = Fg2.TextMatrix(i + 1, j + 1)
    Next j
Next i

Case Is = 2
'Presentación de los resultados
  fgSELresultados.Cols = 2
  fgSELresultados.Rows = n + 1
  fgSELresultados.FixedCols = 1
  fgSELresultados.FixedRows = 1

  For i = fgSELresultados.FixedRows To fgSELresultados.Rows - 1
    fgSELresultados.TextArray(Fgr(i, 0)) = "X" & i
  Next i

   fgSELresultados.TextArray(Fgi(0, 1)) = "Solución"

   For i = 0 To n - 1
   fgSELresultados.TextMatrix(i + 1, 1) = XR(i)
   Next i
```

```
  'Presentación de las iteraciones

  fgSELiteraciones.Cols = n + 2
  fgSELiteraciones.Rows = n + 1
  fgSELiteraciones.FixedCols = 1
  fgSELiteraciones.FixedRows = 1
  fgSELiteraciones.TextArray(Fgit(0, 0)) = ""
  For i = fgSELiteraciones.FixedRows To fgSELiteraciones.Rows - 1
    fgSELiteraciones.TextArray(Fgit(i, 0)) = "E" & i
  Next i
  For i = fgSELiteraciones.FixedCols To fgSELiteraciones.Cols - 1
    If i = fgSELiteraciones.Cols - 1 Then
    fgSELiteraciones.TextArray(Fgit(0, i)) = "b"
    Else
    fgSELiteraciones.TextArray(Fgit(0, i)) = "X" & i
    End If
  Next i
For i = 0 To n - 1
For j = 0 To n
   fgSELiteraciones.TextMatrix(i + 1, j + 1) = Fg2.TextMatrix(i + 1, j + 1)
   Next j
Next i
Case Is = 3
'Presentación de los resultados
  fgSELresultados.Cols = 2
  fgSELresultados.Rows = n + 1
  fgSELresultados.FixedCols = 1
  fgSELresultados.FixedRows = 1

  For i = fgSELresultados.FixedRows To fgSELresultados.Rows - 1
    fgSELresultados.TextArray(Fgr(i, 0)) = "X" & i
  Next i

   fgSELresultados.TextArray(Fgi(0, 1)) = "Solución"

   For i = 0 To n - 1
   fgSELresultados.TextMatrix(i + 1, 1) = XR(i)
   Next i

  'Presentación de las iteraciones

  fgSELiteraciones.Cols = n + 2
  fgSELiteraciones.Rows = n + 1
  fgSELiteraciones.FixedCols = 1
  fgSELiteraciones.FixedRows = 1
```

```
  fgSELiteraciones.TextArray(Fgit(0, 0)) = ""
  For i = fgSELiteraciones.FixedRows To fgSELiteraciones.Rows - 1
    fgSELiteraciones.TextArray(Fgit(i, 0)) = "E" & i
  Next i
  For i = fgSELiteraciones.FixedCols To fgSELiteraciones.Cols - 1
    If i = fgSELiteraciones.Cols - 1 Then
    fgSELiteraciones.TextArray(Fgit(0, i)) = "b"
    Else
    fgSELiteraciones.TextArray(Fgit(0, i)) = "X" & i
    End If
  Next i
For i = 0 To n - 1
For j = 0 To n
   fgSELiteraciones.TextMatrix(i + 1, j + 1) = Fg2.TextMatrix(i + 1, j + 1)
   Next j
Next i


Case ls = 4
 'Presentación de los resultados
  fgSELresultados.Cols = 2
  fgSELresultados.Rows = n + 1
  fgSELresultados.FixedCols = 1
  fgSELresultados.FixedRows = 1

  For i = fgSELresultados.FixedRows To fgSELresultados.Rows - 1
    fgSELresultados.TextArray(Fgr(i, 0)) = "X" & i
  Next i

   fgSELresultados.TextArray(Fgi(0, 1)) = "Solución"

   For i = 0 To n - 1
   fgSELresultados.TextMatrix(i + 1, 1) = XR(i)
   Next i

  'Presentación de las iteraciones

  fgSELiteraciones.Cols = n + 2
  fgSELiteraciones.Rows = n + 1
  fgSELiteraciones.FixedCols = 1
  fgSELiteraciones.FixedRows = 1
  fgSELiteraciones.TextArray(Fgit(0, 0)) = ""
  For i = fgSELiteraciones.FixedRows To fgSELiteraciones.Rows - 1
    fgSELiteraciones.TextArray(Fgit(i, 0)) = "E" & i
  Next i
  For i = fgSELiteraciones.FixedCols To fgSELiteraciones.Cols - 1
```

```
      If i = fgSELiteraciones.Cols - 1 Then
      fgSELiteraciones.TextArray(Fgit(0, i)) = "b"
      Else
      fgSELiteraciones.TextArray(Fgit(0, i)) = "X" & i
      End If
    Next i
 For i = 0 To n - 1
 For j = 0 To n
    fgSELiteraciones.TextMatrix(i + 1, j + 1) = Fg2.TextMatrix(i + 1, j + 1)
    Next j
 Next i
 Case Is = 5
 'Presentación de los resultados
   fgSELresultados.Cols = 2
   fgSELresultados.Rows = n + 1
   fgSELresultados.FixedCols = 1
   fgSELresultados.FixedRows = 1
   For i = fgSELresultados.FixedRows To fgSELresultados.Rows - 1
     fgSELresultados.TextArray(Fgr(i, 0)) = "X" & i
   Next i

    fgSELresultados.TextArray(Fgi(0, 1)) = "Solución"

    For i = 0 To n - 1
    fgSELresultados.TextMatrix(i + 1, 1) = XR(i)
    Next i

   'Presentación de las iteraciones

   fgSELiteraciones.Cols = n + 2
   fgSELiteraciones.Rows = n + 1
   fgSELiteraciones.FixedCols = 1
   fgSELiteraciones.FixedRows = 1
   fgSELiteraciones.TextArray(Fgit(0, 0)) = ""
   For i = fgSELiteraciones.FixedRows To fgSELiteraciones.Rows - 1
     fgSELiteraciones.TextArray(Fgit(i, 0)) = "E" & i
   Next i
   For i = fgSELiteraciones.FixedCols To fgSELiteraciones.Cols - 1
     If i = fgSELiteraciones.Cols - 1 Then
     fgSELiteraciones.TextArray(Fgit(0, i)) = "b"
     Else
     fgSELiteraciones.TextArray(Fgit(0, i)) = "X" & i
     End If
   Next i
 For i = 0 To n - 1
 For j = 0 To n
```

```
    fgSELiteraciones.TextMatrix(i + 1, j + 1) = Fg2.TextMatrix(i + 1, j + 1)
    Next j
Next i

Case ls = 6
'Presentación de los resultados
  fgSELresultados.Cols = 2
  fgSELresultados.Rows = n + 1
  fgSELresultados.FixedCols = 1
  fgSELresultados.FixedRows = 1
  For i = fgSELresultados.FixedRows To fgSELresultados.Rows - 1
    fgSELresultados.TextArray(Fgr(i, 0)) = "X" & i
  Next i

   fgSELresultados.TextArray(Fgi(0, 1)) = "Solución"

   For i = 0 To n - 1
   fgSELresultados.TextMatrix(i + 1, 1) = XR(i)
   Next i

  'Presentación de las iteraciones

  fgSELiteraciones.Cols = n + 2
  fgSELiteraciones.Rows = itermax + 1
  fgSELiteraciones.FixedCols = 1
  fgSELiteraciones.FixedRows = 1
  fgSELiteraciones.TextArray(Fgit(0, 0)) = "Iteraciones"
  For i = fgSELiteraciones.FixedRows To fgSELiteraciones.Rows - 1
    fgSELiteraciones.TextArray(Fgit(i, 0)) = i
  Next i
  For i = fgSELiteraciones.FixedCols To fgSELiteraciones.Cols - 1
    If i = fgSELiteraciones.Cols - 1 Then
    fgSELiteraciones.TextArray(Fgit(0, i)) = "Error"
    Else
    fgSELiteraciones.TextArray(Fgit(0, i)) = "X" & i
    End If
  Next i
For i = 0 To itermax - 1
     For j = 0 To n
     valor = r(i, j, 0)
     If (valor <= 0 Or valor > 0.000000000000001) Then
     fgSELiteraciones.TextMatrix(i + 1, j + 1) = valor
     Else
     fgSELiteraciones.TextMatrix(i + 1, j + 1) = 0
     End If
```

```
        'fgSELiteraciones.TextMatrix(i + 1, j + 1) = r(i, j, 0)
        Next j
    Next i
Case Is = 7
'Presentación de los resultados
  fgSELresultados.Cols = 2
  fgSELresultados.Rows = n + 1
  fgSELresultados.FixedCols = 1
  fgSELresultados.FixedRows = 1
  For i = fgSELresultados.FixedRows To fgSELresultados.Rows - 1
    fgSELresultados.TextArray(Fgr(i, 0)) = "X" & i
  Next i

   fgSELresultados.TextArray(Fgi(0, 1)) = "Solución"

   For i = 0 To n - 1
   fgSELresultados.TextMatrix(i + 1, 1) = XR(i)
   Next i

  'Presentación de las iteraciones

  fgSELiteraciones.Cols = n + 2
  fgSELiteraciones.Rows = itermax + 1
  fgSELiteraciones.FixedCols = 1
  fgSELiteraciones.FixedRows = 1
  fgSELiteraciones.TextArray(Fgit(0, 0)) = "Iteraciones"
  For i = fgSELiteraciones.FixedRows To fgSELiteraciones.Rows - 1
    fgSELiteraciones.TextArray(Fgit(i, 0)) = i
  Next i
  For i = fgSELiteraciones.FixedCols To fgSELiteraciones.Cols - 1
    If i = fgSELiteraciones.Cols - 1 Then
    fgSELiteraciones.TextArray(Fgit(0, i)) = "Error"
    Else
    fgSELiteraciones.TextArray(Fgit(0, i)) = "X" & i
    End If
  Next i
For i = 0 To itermax - 1
      For j = 0 To n
      valor = r(i, j, 0)
      If (valor <= 0 Or valor > 0.000000000000001) Then
      fgSELiteraciones.TextMatrix(i + 1, j + 1) = valor
      Else
      fgSELiteraciones.TextMatrix(i + 1, j + 1) = 0
      End If

      'fgSELiteraciones.TextMatrix(i + 1, j + 1) = r(i, j, 0)
```

```
      Next j
    Next i

Case Is = 8
'Presentación de los resultados
  fgSELresultados.Cols = 2
  fgSELresultados.Rows = n + 1
  fgSELresultados.FixedCols = 1
  fgSELresultados.FixedRows = 1
  For i = fgSELresultados.FixedRows To fgSELresultados.Rows - 1
    fgSELresultados.TextArray(Fgr(i, 0)) = "X" & i
  Next i

   fgSELresultados.TextArray(Fgi(0, 1)) = "Solución"

   For i = 0 To n - 1
   fgSELresultados.TextMatrix(i + 1, 1) = XR(i)
   Next i

  'Presentación de las iteraciones

  fgSELiteraciones.Cols = n + 2
  fgSELiteraciones.Rows = itermax + 1
  fgSELiteraciones.FixedCols = 1
  fgSELiteraciones.FixedRows = 1
  fgSELiteraciones.TextArray(Fgit(0, 0)) = "Iteraciones"
  For i = fgSELiteraciones.FixedRows To fgSELiteraciones.Rows - 1
    fgSELiteraciones.TextArray(Fgit(i, 0)) = i
  Next i
  For i = fgSELiteraciones.FixedCols To fgSELiteraciones.Cols - 1
    If i = fgSELiteraciones.Cols - 1 Then
    fgSELiteraciones.TextArray(Fgit(0, i)) = "Error"
    Else
    fgSELiteraciones.TextArray(Fgit(0, i)) = "X" & i
    End If
  Next i
For i = 0 To itermax - 1
      For j = 0 To n
      valor = r(i, j, 0)
      If (valor <= 0 Or valor > 0.000000000000001) Then
      fgSELiteraciones.TextMatrix(i + 1, j + 1) = valor
      Else
      fgSELiteraciones.TextMatrix(i + 1, j + 1) = 0
      End If

      'fgSELiteraciones.TextMatrix(i + 1, j + 1) = r(i, j, 0)
```

```
      Next j
    Next i


Case Is = 9
'Presentación de los resultados
  fgSELresultados.Cols = 2
  fgSELresultados.Rows = n + 1
  fgSELresultados.FixedCols = 1
  fgSELresultados.FixedRows = 1
  For i = fgSELresultados.FixedRows To fgSELresultados.Rows - 1
    fgSELresultados.TextArray(Fgr(i, 0)) = "X" & i
  Next i

   fgSELresultados.TextArray(Fgi(0, 1)) = "Solución"

   For i = 0 To n - 1
   fgSELresultados.TextMatrix(i + 1, 1) = XR(i)
   Next i

  'Presentación de las iteraciones

  fgSELiteraciones.Cols = n + 2
  fgSELiteraciones.Rows = itermax + 1
  fgSELiteraciones.FixedCols = 1
  fgSELiteraciones.FixedRows = 1
  fgSELiteraciones.TextArray(Fgit(0, 0)) = "Iteraciones"
  For i = fgSELiteraciones.FixedRows To fgSELiteraciones.Rows - 1
    fgSELiteraciones.TextArray(Fgit(i, 0)) = i
  Next i
  For i = fgSELiteraciones.FixedCols To fgSELiteraciones.Cols - 1
    If i = fgSELiteraciones.Cols - 1 Then
    fgSELiteraciones.TextArray(Fgit(0, i)) = "Error"
    Else
    fgSELiteraciones.TextArray(Fgit(0, i)) = "X" & i
    End If
  Next i
For i = 0 To itermax - 1
      For j = 0 To n
      valor = r(i, j, 0)
      If (valor <= 0 Or valor > 0.000000000000001) Then
      fgSELiteraciones.TextMatrix(i + 1, j + 1) = valor
      Else
      fgSELiteraciones.TextMatrix(i + 1, j + 1) = 0
      End If
```

```
        'fgSELiteraciones.TextMatrix(i + 1, j + 1) = r(i, j, 0)
        Next j
    Next i


Case Else
'Presentación de los resultados
  fgSELresultados.Cols = 2
  fgSELresultados.Rows = n + 1
  fgSELresultados.FixedCols = 1
  fgSELresultados.FixedRows = 1
  For i = fgSELresultados.FixedRows To fgSELresultados.Rows - 1
    fgSELresultados.TextArray(Fgr(i, 0)) = "X" & i
  Next i

   fgSELresultados.TextArray(Fgi(0, 1)) = "Solución"

   For i = 0 To n - 1
   fgSELresultados.TextMatrix(i + 1, 1) = XR(i)
   Next i

  'Presentación de las iteraciones

  fgSELiteraciones.Cols = n + 2
  fgSELiteraciones.Rows = itermax + 1
  fgSELiteraciones.FixedCols = 1
  fgSELiteraciones.FixedRows = 1
  fgSELiteraciones.TextArray(Fgit(0, 0)) = "Iteraciones"
  For i = fgSELiteraciones.FixedRows To fgSELiteraciones.Rows - 1
    fgSELiteraciones.TextArray(Fgit(i, 0)) = i
  Next i
  For i = fgSELiteraciones.FixedCols To fgSELiteraciones.Cols - 1
    If i = fgSELiteraciones.Cols - 1 Then
    fgSELiteraciones.TextArray(Fgit(0, i)) = "Error"
    Else
    fgSELiteraciones.TextArray(Fgit(0, i)) = "X" & i
    End If
  Next i
For i = 0 To itermax - 1
      For j = 0 To n
      valor = r(i, j, 0)
      If (valor <= 0 Or valor > 0.000000000000001) Then
      fgSELiteraciones.TextMatrix(i + 1, j + 1) = valor
      Else
      fgSELiteraciones.TextMatrix(i + 1, j + 1) = 0
      End If
```

```
        'fgSELiteraciones.TextMatrix(i + 1, j + 1) = r(i, j, 0)
        Next j
    Next i

End Select
Exit Sub
err1:
End Sub

'****** Conexion a la BDD
Private Sub PLConexion(parBDD As Boolean)
Dim VTCad As String
Dim VTBase As Database
Dim VTcadena As String
    VTcadena = App.Path & "/Seguridad.mdb"
    Set VLConexion = New ADODB.Connection
    If parBDD = False Then
        VTCad = "Provider=SQLOLEDB.1;Integrated Security=SSPI;Persist Security
Info=False;Initial Catalog=Seguridad"
    Else
        VTCad = "Provider=Microsoft.Jet.OLEDB.4.0;Data Source='" & VTcadena &
"';Persist Security Info=False"
    End If
    VLConexion.Open VTCad
End Sub

Private Sub PLGuardarBDD()
    Dim VTCad As String
    Dim VTReg As ADODB.Recordset
    Dim VTID As Integer
    Dim VB As Integer
On Error GoTo error1
    PLVerificaNombre Trim(VLNombre)
    If tipoalg = 1 Or tipoalg = 2 Then
        If VLVerifica = True Then   'Si existe un nombre con el mismo nombre actualiza
            VB = MsgBox("El nombre de ese trabajo ya existe, desea reemplazarlo?",
vbYesNo, "Guardar Trabajo Como")
            If VB = 6 Then
                PLGuardaJacobiano
                VTCad = "update SEL set " _
                    & " Nume= '" & Trim(TxtSELnumecuaciones.Text) & "', " _
                    & " jacobiano='" & VLJacobiano & "' " _
                    & " where Nombre = '" & VLNombre & "'"
                VLConexion.Execute (VTCad)
            End If
```

```
      Else    'Guarda el trabajo
         VTID = PLRecuperaCodigo
         PLGuardaJacobiano
         VTCad = "insert into SEL values " _
               & "(" & CInt(VTID) & ", " _
               & " '" & TxtSELnumecuaciones.Text & "', '" & TxtSELtolerancia.Text & "',
" _
               & " '" & TxtSELiteraciones.Text & "', '" & VLJacobiano & "', " _
               & " '" & Trim(VLNombre) & "', " & 1 & ", " & CInt(tipoalg) & ")"
               '& " '" & Trim(VLNombre) & "', " & CInt(VGIDUsuario) & ", " & CInt(tipoalg
) & ")"
         VLConexion.Execute (VTCad)
         MsgBox "Trabajo guardado exitosamente", vbInformation, "Guardar trabajo
como"
      End If
   Else
      If VLVerifica = True Then   'Si existe un nombre con el mismo nombre actualiza
         VB = MsgBox("El nombre de ese trabajo ya existe, desea reemplazarlo?",
vbYesNo, "Guardar Trabajo Como")
         If VB = 6 Then
            PLGuardaJacobiano
            VTCad = "update SEL set " _
                  & " Nume= '" & Trim(TxtSELnumecuaciones.Text) & "', " _
                  & " tol='" & TxtSELtolerancia.Text & "', Imax='" &
TxtSELiteraciones.Text & "', " _
                  & " jacobiano='" & VLJacobiano & "' " _
                  & " where Nombre = '" & VLNombre & "'"
            VLConexion.Execute (VTCad)
         End If
      Else    'Guarda el trabajo
         VTID = PLRecuperaCodigo
         PLGuardaJacobiano
         VTCad = "insert into SEL values " _
               & "(" & CInt(VTID) & ", " _
               & " '" & TxtSELnumecuaciones.Text & "', '" & TxtSELtolerancia.Text & "',
" _
               & " '" & TxtSELiteraciones.Text & "', '" & VLJacobiano & "', " _
               & " '" & Trim(VLNombre) & "', " & 1 & ", " & CInt(tipoalg) & ")"
               '& " '" & Trim(VLNombre) & "', " & CInt(VGIDUsuario) & ", " & CInt(tipoalg
) & ")"
         VLConexion.Execute (VTCad)
         MsgBox "Trabajo guardado exitosamente", vbInformation, "Guardar trabajo
como"
      End If
   End If
```

```
Exit Sub
error1:
   'If err <> 0 Then
      'MsgBox "Se ha producido el siguiente error: " & err.Description & "." _
            & "Por favor consultar con el administrador", vbCritical
'   End If
End Sub

Private Sub PLVerificaNombre(parNombre As String)
Dim VTReg As ADODB.Recordset
Dim VTCad As String
On Error GoTo error1
   VTCad = "select Nombre from SEL where Nombre = '" & parNombre & "' and tipo =
" & tipoalg
      Set VTReg = VLConexion.Execute(VTCad)
      With VTReg
         If Not VTReg.EOF Then
            .MoveFirst
            Do Until .EOF
               If Trim(VTReg(0)) = parNombre Then
                  VLVerifica = True
                  Exit Do
               Else
                  VLVerifica = False
               End If
               .MoveNext
            Loop
         Else
            VLVerifica = False
         End If
      End With
Exit Sub
error1:

End Sub
Private Function PLRecuperaCodigo() As Integer
   Dim VTCad As String
   Dim VTReg As ADODB.Recordset
On Error GoTo err1
      VTCad = "select max(IDSel) from SEL"
      Set VTReg = VLConexion.Execute(VTCad)
      If Not IsNull(VTReg(0)) Then
         PLRecuperaCodigo = VTReg(0) + 1
      Else
         PLRecuperaCodigo = 1
      End If
```

```
Exit Function
err1:

End Function

Private Sub PLGuardaJacobiano()
Dim i, j As Integer
Dim Nume As Integer
On Error GoTo error1
   VLJacobiano = ""
   Nume = CInt(TxtSELnumecuaciones.Text)
   If tipoalg >= 1 And tipoalg <= 5 Then
      For i = 1 To Nume
         For j = 1 To Nume + 1
            VLJacobiano = VLJacobiano & Fg2.TextMatrix(i, j) & ";"
         Next j
      Next i
   ElseIf tipoalg >= 6 Then
      For i = 1 To Nume + 1
         For j = 1 To Nume + 1
            VLJacobiano = VLJacobiano & Fg2.TextMatrix(i, j) & ";"
         Next j
      Next i
   End If
'VLJacobiano
Exit Sub
error1:

End Sub

Private Sub PLRecupera()
   Dim VTCad As String
   Dim VTReg As ADODB.Recordset
On Error GoTo err1
   '******** APROXIMACIONES SUCESIVAS
      VTCad = "select * from SEL where Nombre = '" & VLNombreTrabajo & "'"
      Set VTReg = VLConexion.Execute(VTCad)
      If Not VTReg.EOF Then
         TxtSELnumecuaciones.Text = Trim(VTReg(1))
         TxtSELtolerancia.Text = Trim(VTReg(2))
         TxtSELiteraciones.Text = Trim(VTReg(3))
         PLRecuperaDatos CInt(TxtSELnumecuaciones.Text), Trim(VTReg(4))
      End If
Exit Sub
err1:
   'If err.Number <> 0 Then
```

```
    '  MsgBox "Se ha producido el siguiente error: " & err.Description & "." _
          & "Por favor consultar con el administrador", vbCritical
   'End If

End Sub

Private Sub PLRecuperaDatos(parNume As Integer, parJacobiano As String)
Dim i, j, k As Integer
Dim VTCont As Integer
Dim VTPos As Integer
Dim VTValor As String
Dim VTLimite As Integer
Dim VTJacob As String
On Error GoTo error1
   VTCont = 1
   VTJacob = parJacobiano
   If tipoalg >= 1 And tipoalg <= 5 Then
      VTLimite = (parNume * parNume) + parNume
      For k = 1 To VTLimite
         VLJacobianoMatriz(k) = 0
      Next k

      For k = 1 To VTLimite
         VTPos = InStr(1, parJacobiano, ";")
         VTValor = Mid(parJacobiano, 1, VTPos - 1)
         parJacobiano = Mid(parJacobiano, VTPos + 1)
         VLJacobianoMatriz(k) = VTValor
      Next k
      For i = 1 To parNume
         For j = 1 To parNume + 1
            Fg2.TextMatrix(i, j) = VLJacobianoMatriz(VTCont)
            VTCont = VTCont + 1
         Next j
      Next i
   ElseIf tipoalg >= 6 Then
      VTLimite = (parNume * parNume) + (parNume * 2)
      For k = 1 To VTLimite
         VLJacobianoMatriz(k) = 0
      Next k

      For k = 1 To VTLimite
         VTPos = InStr(1, parJacobiano, ";")
         VTValor = Mid(parJacobiano, 1, VTPos - 1)
         parJacobiano = Mid(parJacobiano, VTPos + 1)
         VLJacobianoMatriz(k) = VTValor
      Next k
```

```
        For i = 1 To parNume + 1
            For j = 1 To parNume + 1
                Fg2.TextMatrix(i, j) = VLJacobianoMatriz(VTCont)
                VTCont = VTCont + 1
            Next j
        Next i
    End If
Exit Sub
error1:
    'If err.Number <> 0 Then
     '   MsgBox "Se ha producido el siguiente error: " & err.Description & "." _
            & "Por favor consultar con el administrador", vbCritical
    'End If
End Sub

Private Sub PLImprimir()
    PLCambio
        'Form1.Command1_Click
        'PrintForm
        'Form1.Show
    PLOriginal
Form1.Visible = False
End Sub

Private Sub PLCambio()
    Me.BackColor = &HFFFFFF
    Me.BorderStyle = 0

    'Frame grande
'   frapar.BackColor = &HFFFFFF
'   'frapar.BorderStyle = 0
'   frapar.Top = 10
'   frapar.Width = 11550
'   frapar.Height = 10000
'   'Grilla
'   Fg2.Top = 3000
'   Fg2.Left = 40
'   Fg2.Width = 11450
'   Fg2.Height = 5000
'   Fg2.BackColorBkg = &HFFFFFF
'   'Fraerror
'   FraSELerrores.Top = 1300
'   FraSELerrores.Left = 1500
'   FraSELerrores.BackColor = &HFFFFFF
'   'Options
'   OptSELEa.BackColor = &HFFFFFF
```

```
'    OptSELEr.BackColor = &HFFFFFF
'
'    CmdSELcalcular.Visible = False
'    CmdSELgraficar.Visible = False
'    'SSTab1.Visible = False
'    LblSELtitulo.Visible = False
End Sub

Private Sub PLOriginal()
    Me.BackColor = &H8000000F
    Me.BorderStyle = 2
'    frapar.BackColor = &H8000000F
'    'frapar.BorderStyle = 1
'    frapar Top = 1080
'    frapar.Height = 4500
'    frapar.Width = 8500
'
'    Fg2.Top = 1320
'    Fg2.Left = 480
'    Fg2.Width = 4455
'    Fg2.Height = 2775
'    Fg2.BackColorBkg = &H8000000F
'    'Fraerror
'    FraSELerrores.Top = 1080
'    FraSELerrores.Left = 6800
'    FraSELerrores.BackColor = &H8000000F
'    'Options
'    OptSELEa.BackColor = &H8000000F
'    OptSELEr.BackColor = &H8000000F
'
'    CmdSELcalcular.Visible = True
'    CmdSELgraficar.Visible = False
'    frapar.Visible = True
'    SSTab1.Visible = True
'    LblSELtitulo.Visible = True
End Sub

Private Sub TxtSELnumecuaciones_KeyPress(KeyAscii As Integer)
'filtra la entrada de datos que no sean números enteros y positivos
  If (KeyAscii < 48 Or KeyAscii > 57) Then

        If (KeyAscii <> 8) Then
        KeyAscii = 0
        End If

  End If
```

```
   If KeyAscii = Asc(vbCr) Then ' Elimina los retornos para quitar los pitidos.
   KeyAscii = 0
   End If

End Sub



Private Sub TxtSELtolerancia_KeyPress(KeyAscii As Integer)
If Val(TxtSELtolerancia) > 32767 Then
mensaje = MsgBox("Fuera del Rango", 48, "Error en el ingreso")
Exit Sub
End If

'filtra la entrada de datos que no sean números no incluye signo - y si el punto
   If (KeyAscii < 48 Or KeyAscii > 57) Then
       If (KeyAscii <> 46) Then
          If (KeyAscii <> 8) Then
          KeyAscii = 0
          End If
       End If
    End If
   If KeyAscii = Asc(vbCr) Then ' Elimina los retornos para quitar los pitidos.
   KeyAscii = 0
   End If

End Sub

Private Sub Sincolor()
With Me
arribaImprimir = .FraImprimir.Top
arribaSstab = .SSTab1.Top
izquierdaSstab = .SSTab1.Left
anchoSstab = .SSTab1.Width
altoSstab = .SSTab1.Height
arribaResul = .FraResul.Top
anchoResul = .FraResul.Width
altoResul = .FraResul.Height
arribaFgitera = .fgSELiteraciones.Top
izquierdaFgitera = .fgSELiteraciones.Left
anchoFgitera = .fgSELiteraciones.Width
altoFgitera = .fgSELiteraciones.Height
arribaFgresul = .fgSELresultados.Top
anchoFgresul = .fgSELresultados.Width
altoFgresul = .fgSELresultados.Height
VLabel = .Label20.Visible
```

```
Vadelante = .CmdSELadelante.Visible
Vatras = .CmdSELatras.Visible
.FraImprimir.Top = 0
.SSTab1.Top = .SSTab1.Top + 1000
.SSTab1.Left = .SSTab1.Left + 1000
.SSTab1.Width = .SSTab1.Width / 4
.SSTab1.Height = .SSTab1.Height / 4
.FraResul.Top = .FraImprimir.Top + .FraImprimir.Height
.FraResul.Width = .FraResul.Width + 7000
.FraResul.Height = .FraResul.Height - 200
.fgSELresultados.Width = .fgSELresultados.Width + 7000
.fgSELresultados.Height = .fgSELresultados.Height - 200
.fgSELiteraciones.Left = .FraImprimir.Left
.fgSELiteraciones.Height = .fgSELiteraciones.Height + 50
.fgSELiteraciones.Top = .FraResul.Top + .FraResul.Height
.fgSELiteraciones.Width = .FraResul.Width
colororg = .BackColor
.BackColor = &H80000009
.FraImprimir.BackColor = &H80000009
.frapar.BackColor = &H80000009
.OptSELEa.BackColor = &H80000009
.OptSELEr.BackColor = &H80000009
.TxtEdit.BorderStyle = 0
.TxtSELiteraciones.BorderStyle = 0
.TxtSELnumecuaciones.BorderStyle = 0
.TxtSELtolerancia.BorderStyle = 0
.Fg2.BackColorBkg = &H80000009
.Fg2.BackColorFixed = &H80000009
.frapar.BorderStyle = 1
.FraResul.BackColor = &H80000009
.fgSELresultados.BackColorBkg = &H80000009
.fgSELresultados.BackColorFixed = &H80000009
.fgSELiteraciones.BackColorBkg = &H80000009
.fgSELiteraciones.BackColorFixed = &H80000009
.Shape1.Visible = False
.Label20.Visible = False
.CmdSELadelante.Visible = False
.CmdSELatras.Visible = False
.CmdSELiteraciones.Visible = False
.CmdSELcalcular.Visible = False
.Fg2.ScrollBars = 0
.FraSELerrores.BackColor = &H80000009
.LblNomalg.BackColor = &H80000009
.LblSELtitulo.Visible = False
FraSelSintaxis.Visible = False
End With
```

```
End Sub

Private Sub Concolor()
With Me
 .FraImprimir.Top = arribaImprimir
 .SSTab1.Top = arribaSstab
.SSTab1.Left = izquierdaSstab
 .SSTab1.Width = anchoSstab
 .SSTab1.Height = altoSstab
.FraResul.Top = arribaResul
 .FraResul.Width = anchoResul
.FraResul.Height = altoResul
 .fgSELiteraciones.Top = arribaFgitera
 .fgSELiteraciones.Left = izquierdaFgitera
 .fgSELiteraciones.Width = anchoFgitera
 .fgSELiteraciones.Height = altoFgitera
 .fgSELresultados.Top = arribaFgresul
 .fgSELresultados.Width = anchoFgresul
 .fgSELresultados.Height = altoFgresul

 .BackColor = colororg
.BackColor = colororg
.FraImprimir.BackColor = colororg
.frapar.BackColor = colororg
.OptSELEa.BackColor = colororg
.OptSELEr.BackColor = colororg
.TxtEdit.BorderStyle = 1
.TxtSELiteraciones.BorderStyle = 1
.TxtSELnumecuaciones.BorderStyle = 1
.TxtSELtolerancia.BorderStyle = 1
.Fg2.BackColorBkg = colororg
.Fg2.BackColorFixed = colororg
.frapar.BorderStyle = 0
.FraResul.BackColor = colororg
.fgSELresultados.BackColorBkg = colororg
.fgSELresultados.BackColorFixed = colororg
.Shape1.Visible = True
If VLabel = True Then
.Label20.Visible = True
End If
.fgSELiteraciones.BackColorBkg = colororg
.fgSELiteraciones.BackColorFixed = colororg
.Fg2.ScrollBars = 3
.FraSELerrores.BackColor = colororg
If Vadelante = True Then
```

```
.CmdSELadelante.Visible = True
End If
If Vatras = True Then
.CmdSELatras.Visible = True
End If
.CmdSELcalcular.Visible = True
.CmdSELiteraciones.Visible = True
.LblNomalg.BackColor = colororg
.LblSELtitulo.Visible = True
FraSelSintaxis.Visible = True
End With

End Sub
```

Módulo de Clase ClasSEL

```
Dim numecuaciones As Integer
Dim tolerancia As Double
Dim Maxitera As Integer
Dim Tipoerror As Boolean
Dim Ma() As Double
Dim Mb() As Double
Dim X() As Double
Dim vectorini() As Double
Dim MRS() As String
Dim MR() As Double



Public Property Let setearSEL(n As Integer, T As Double, itera As Integer, Ea As
Boolean, A() As Double, b() As Double)
On Error GoTo err1
numecuaciones = n
tolerancia = T
Maxitera = itera
Tipoerror = Ea
Ma = A
Mb = b
Exit Property
err1:
End Property
Public Property Get ReturSELMR() As Variant
On Error GoTo err1
ReturSELMR = MR
Exit Property
err1:
```

```
End Property

Public Property Get ReturSELX() As Variant
On Error GoTo err1
ReturSELX = X
err1:
End Property
Public Property Let setearSENL(n As Integer, T As Double, itera As Integer, Vinicial()
As Double, Ecdespejadas() As String, Ea As Boolean, A() As Double, b() As Double)
On Error GoTo err1:
numecuaciones = n
tolerancia = T
Maxitera = itera
Tipoerror = Ea
Ma = A
Mb = b
vectorini = Vinicial
MRS = Ecdespejadas
Exit Property
err1:
End Property

Public Sub EliGaussSimple()
Dim i As Integer
Dim j As Integer
Dim k As Integer
Dim mensaje As String
Dim factor As Double
Dim sum As Double
On Error GoTo err1:
ReDim X(0 To numecuaciones - 1) As Double
ReDim MR(0 To numecuaciones - 1, 0 To numecuaciones, 0 To numecuaciones - 2)
'Eliminación hacia adelante
For k = 0 To numecuaciones - 2
   For i = k + 1 To numecuaciones - 1
   If Ma(k, k) <> 0 Then
   factor = Ma(i, k) / Ma(k, k)
   Else
   mensaje = MsgBox("Si los coeficientes han sido ingresados entonces: A es
singular, no hay solución o no es única", 16, "Fracaso")
   Exit Sub
   End If
      For j = 0 To numecuaciones - 1
      sum = factor * Ma(k, j)
      Ma(i, j) = Ma(i, j) - sum
      Next j
```

```
    Mb(i) = Mb(i) - factor * Mb(k)
    Next i
' No es parte del algoritmo permite guardar los estados de las matrices
'durante el proceso de eliminación hacia adelante para luego presentarle
'al usuario
For i = 0 To numecuaciones - 1
   For j = 0 To numecuaciones - 1
   MR(i, j, k) = Ma(i, j)
   Next j
Next i
For i = 0 To numecuaciones - 1
MR(i, numecuaciones, k) = Mb(i)
Next i
'Las instrucciónes siguientes si pertenecen al algoritmo
Next k

'Sustitución hacia atrás
X(numecuaciones - 1) = Mb(numecuaciones - 1) / Ma(numecuaciones - 1,
numecuaciones - 1)
For i = numecuaciones - 2 To 0 Step -1
sum = 0
   For j = i + 1 To numecuaciones - 1
   sum = sum + Ma(i, j) * X(j)
   Next j
   If Ma(i, i) <> 0 Then
   X(i) = (Mb(i) - sum) / Ma(i, i)
   Else
   mensaje = MsgBox("No es posible encontrar la solución, los coeficientes de la
diagonal deben ser diferentes de cero", 16, "Fracaso")
   Exit Sub
   End If
Next i
Exit Sub
err1:
If err.Number <> 0 Then
   mensaje = MsgBox("Parámetros del algoritmo incompletos", 48, "Error en el
ingreso")
   End If

End Sub
Public Sub EliGaussPivotación()
Dim i As Integer
Dim j As Integer
Dim k As Integer
Dim mensaje As String
```

```
Dim indice As Integer
Dim factor As Double
Dim sum As Double
Dim limite As Integer
On Error GoTo err1
ReDim X(0 To numecuaciones - 1) As Double
limite = ((numecuaciones - 1) * 2) - 1
ReDim MR(0 To numecuaciones - 1, 0 To numecuaciones, 0 To limite)
'Eliminación hacia adelante
For k = 0 To numecuaciones - 2
   Pivotaciónparcial (k)
' No es parte del algoritmo permite guardar los estados de las matrices
'durante el proceso de eliminación hacia adelante para luego presentarle
'al usuario
   indice = 2 * k
   For i = 0 To numecuaciones - 1
      For j = 0 To numecuaciones - 1
      MR(i, j, indice) = Ma(i, j)
      Next j
   Next i
   For i = 0 To numecuaciones - 1
   MR(i, numecuaciones, indice) = Mb(i)
   Next i
'Las siguientes instrucciones son parte del algoritmo
'Se obtiene el factor multiplicativo para la fila
   For i = k + 1 To numecuaciones - 1
   If Ma(k, k) <> 0 Then
   factor = Ma(i, k) / Ma(k, k)
   Else
   mensaje = MsgBox("Si los coeficientes han sido ingresados entonces:A es
singular, no hay solución o no es única", 16, "Fracaso")

   Exit Sub
   End If
'Se obtiene la fila resultado de la eliminación hacia adelante
      For j = 0 To numecuaciones - 1
      sum = factor * Ma(k, j)
      Ma(i, j) = Ma(i, j) - sum
      Next j

   Mb(i) = Mb(i) - factor * Mb(k)
   Next i
' No es parte del algoritmo permite guardar los estados de las matrices
'durante el proceso de eliminación hacia adelante para luego presentarle
'al usuario
For i = 0 To numecuaciones - 1
```

```
    For j = 0 To numecuaciones - 1
    MR(i, j, indice + 1) = Ma(i, j)
    Next j
Next i
For i = 0 To numecuaciones - 1
MR(i, numecuaciones, indice + 1) = Mb(i)
Next i
'Las instrucciónes siguientes si pertenecen al algoritmo
Next k

'Sustitución hacia atrás
'Se obtiene el resultado de X
X(numecuaciones - 1) = Mb(numecuaciones - 1) / Ma(numecuaciones - 1,
numecuaciones - 1)
For i = numecuaciones - 2 To 0 Step -1
sum = 0
    For j = i + 1 To numecuaciones - 1
    sum = sum + Ma(i, j) * X(j)
    Next j
    If Ma(i, i) <> 0 Then
    X(i) = (Mb(i) - sum) / Ma(i, i)
    Else
    mensaje = MsgBox("No es posible encontrar la solución, los coeficientes de la
diagonal deben ser diferentes de cero", 16, "Fracaso")

    Exit Sub
    End If
Next i
Exit Sub
err1:
If err.Number <> 0 Then
    mensaje = MsgBox("Parámetros del algoritmo incompletos", 48, "Error en el
ingreso")
    End If

End Sub

Public Sub GaussJordan()
Dim i As Integer
Dim j As Integer
Dim k As Integer
Dim mensaje As String
Dim factor As Double
Dim Ampanterior As Double
Dim Amp() As Double
On Error GoTo err1
```

```
ReDim Amp(0 To numecuaciones - 1, 0 To numecuaciones) As Double
ReDim MR(0 To numecuaciones - 1, 0 To numecuaciones, 0 To numecuaciones - 1)
As Double
ReDim X(0 To numecuaciones - 1) As Double
'Realizamos la matriz ampliada Amp con las matrices Ma, Mb
For i = 0 To numecuaciones - 1
   For j = 0 To numecuaciones - 1
   Amp(i, j) = Ma(i, j)
   Next j
Next i
For i = 0 To numecuaciones - 1
Amp(i, numecuaciones) = Mb(i)
Next i


'Se realiza la eliminación de las columnas
For k = 0 To numecuaciones - 1
   For i = 0 To numecuaciones - 1
   If k <> i Then
   If Amp(k, k) <> 0 Then
   Ampanterior = Amp(k, k)
   factor = Amp(i, k) / Amp(k, k)
   Else
   mensaje = MsgBox("Si los coeficientes han sido ingresados entonces:A es
singular, no hay solución o no es única", 16, "Fracaso")

   Exit Sub
   End If
'Se obtiene las filas de la matriz
   For j = 0 To numecuaciones
   Amp(i, j) = Amp(i, j) - factor * Amp(k, j)
   Next j
   End If
   Next i
'Normalización de las filas
   For j = 0 To numecuaciones
   Amp(k, j) = Amp(k, j) / Ampanterior
   Next j
'Asignamos los resultados parciales a la matriz MR para su posterior
'presentación al usuario
For i = 0 To numecuaciones - 1
   For j = 0 To numecuaciones
   MR(i, j, k) = Amp(i, j)
   Next j
Next i
Next k
```

```
' Guardamos los resultados
For i = 0 To numecuaciones - 1
X(i) = MR(i, numecuaciones, k - 1)
Next i
Exit Sub
err1:
If err.Number <> 0 Then
   mensaje = MsgBox("Parámetros del algoritmo incompletos", 48, "Error en el
ingreso")
   End If

End Sub
Public Sub Dollittle()
Dim i As Integer
Dim j As Integer
Dim k As Integer
Dim mensaje As String
Dim factor As Double
Dim sum As Double
On Error GoTo err1
ReDim X(0 To numecuaciones - 1) As Double
ReDim MR(0 To numecuaciones - 1, 0 To numecuaciones, 0 To 1)
'Inicializamos la matriz L
For i = 0 To numecuaciones - 1
   For j = 0 To numecuaciones
   If i = j Then
   MR(i, j, 0) = 1
   Else
   MR(i, j, 0) = 0
   End If
   Next j
Next i
For i = 0 To numecuaciones - 1
MR(i, numecuaciones, 0) = Mb(i)
Next i

'Eliminación hacia adelante
For k = 0 To numecuaciones - 2
   For i = k + 1 To numecuaciones - 1
   If Ma(k, k) <> 0 Then
   factor = Ma(i, k) / Ma(k, k)
   Else
   mensaje = MsgBox("Si los coeficientes han sido ingresados entonces:A es
singular, no hay solución o no es única", 16, "Fracaso")

   Exit Sub
```

```
    End If
       For j = 0 To numecuaciones - 1
       sum = factor * Ma(k, j)
       Ma(i, j) = Ma(i, j) - sum
       Next j

     Mb(i) = Mb(i) - factor * Mb(k)
     MR(i, k, 0) = factor
     Next i

'Las instrucciónes siguientes si pertenecen al algoritmo
Next k
' No es parte del algoritmo permite guardar los estados de las matrices
'durante el proceso de eliminación hacia adelante para luego presentarle
'al usuario
For i = 0 To numecuaciones - 1
    For j = 0 To numecuaciones - 1
    MR(i, j, 1) = Ma(i, j)
    Next j
Next i
For i = 0 To numecuaciones - 1
MR(i, numecuaciones, 1) = Mb(i)
Next i



'Sustitución hacia atrás
X(numecuaciones - 1) = Mb(numecuaciones - 1) / Ma(numecuaciones - 1,
numecuaciones - 1)
For i = numecuaciones - 2 To 0 Step -1
sum = 0
    For j = i + 1 To numecuaciones - 1
    sum = sum + Ma(i, j) * X(j)
    Next j
    If Ma(i, i) <> 0 Then
    X(i) = (Mb(i) - sum) / Ma(i, i)
    Else
    mensaje = MsgBox("No es posible encontrar la solución, los coeficientes de la
diagonal deben ser diferentes de cero", 16, "Fracaso")

    Exit Sub
    End If
Next i
Exit Sub
err1:
If err.Number <> 0 Then
```

```
   mensaje = MsgBox("Parámetros del algoritmo incompletos", 48, "Error en el
ingreso")
   End If

End Sub
Public Sub Jacobi()

Dim i As Integer
Dim k As Integer
Dim j As Integer
Dim suma As Double
Dim mensaje As String
Dim e As Double
Dim XT() As Double
Dim numorden As Integer
Dim cambio As Double
Dim etemp(1 To 2) As Double
On Error GoTo err1
ReDim MR(0 To Maxitera - 1, 0 To numecuaciones, 0)
ReDim XT(0 To numecuaciones - 1) As Double
ReDim X(0 To numecuaciones - 1) As Double
k = 0
Do
k = k + 1
For i = 0 To numecuaciones - 1
suma = 0
   For j = 0 To numecuaciones - 1
   If i <> j Then
   suma = vectorini(j) * Ma(i, j) + suma
   End If
   Next j
   If Ma(i, i) <> 0 Then
   X(i) = (Mb(i) - suma) / Ma(i, i)
   Else
   mensaje = MsgBox(" Se debe ingresar las ecuaciones de modo que exista el la
variable X1 en la Ecuación 1 y asi con las demas si lo a hecho A es singular, no hay
solución o no es única", 16, "Fracaso")

   Exit Sub
   End If
Next i
'Ordenamos los vectores X y Vectorini con el uso de el vector XT para sacar las
normas  y los
' errores

XT = X
```

```
For i = 0 To numecuaciones - 1
XT(i) = Abs(XT(i))
Next i


For j = 1 To 2
numorden = 1
Do
For i = 0 To numecuaciones - 1 - numorden
   If XT(i) <= XT(i + 1) Then
   cambio = XT(i)
   XT(i) = XT(i + 1)
   XT(i + 1) = cambio
   End If
Next i
numorden = numorden + 1
Loop Until numorden > (numecuaciones - 1)
etemp(j) = XT(0)

For i = 0 To numecuaciones - 1
vectorini(i) = Abs(vectorini(i))
Next i
XT = vectorini
Next j

'Se obtiene el error selecionado por el usuario
If Tipoerror = True Then
e = etemp(1) - etemp(2)
Else
e = (etemp(1) - etemp(2)) / etemp(1)
End If

For j = 0 To numecuaciones - 1
MR(k - 1, j, 0) = X(j)
Next j
MR(k - 1, numecuaciones, 0) = Abs(e)
vectorini = X


Loop Until Abs(e) < tolerancia Or k >= Maxitera
If k >= Maxitera Then
mensaje = MsgBox("Número de iteraciones máximas superado, no fue posible
encontrar la respuesta; Sugerencia: incremente el número de iteraciones", 48,
"Iteraciones")

End If
```

```
Exit Sub
err1:
If err.Number <> 0 Then
   mensaje = MsgBox("Parámetros del algoritmo incompletos", 48, "Error en el
ingreso")
   End If

End Sub
Public Sub GaussSeidel()
Dim i As Integer
Dim k As Integer
Dim j As Integer
Dim suma As Double
Dim mensaje As String
Dim e As Double
Dim XT() As Double
Dim vectorinitemp() As Double
Dim numorden As Integer
Dim cambio As Double
Dim etemp(1 To 2) As Double
On Error GoTo err1
ReDim MR(0 To Maxitera - 1, 0 To numecuaciones, 0)
ReDim XT(0 To numecuaciones - 1) As Double
ReDim X(0 To numecuaciones - 1) As Double
ReDim vectorinitemp(0 To numecuaciones - 1) As Double


k = 0
Do
vectorinitemp = vectorini
k = k + 1
For i = 0 To numecuaciones - 1
suma = 0
   For j = 0 To numecuaciones - 1
   If i <> j Then
   suma = vectorinitemp(j) * Ma(i, j) + suma
   End If
   Next j
   If Ma(i, i) <> 0 Then
   X(i) = (Mb(i) - suma) / Ma(i, i)
   vectorinitemp(i) = X(i)
   Else
   mensaje = MsgBox(" Se debe ingresar las ecuaciones de modo que exista el la
variable X1 en la Ecuación 1 y asi con las demas si lo a hecho A es singular, no hay
solución o no es única", 16, "Fracaso")
   Exit Sub
```

```
   End If
Next i
'Ordenamos los vectores X y Vectorini con el uso de el vector XT para sacar las
normas  y los
' errores

XT = X
For i = 0 To numecuaciones - 1
XT(i) = Abs(XT(i))
Next i


For j = 1 To 2
numorden = 1
Do
For i = 0 To numecuaciones - 1 - numorden
   If XT(i) <= XT(i + 1) Then
   cambio = XT(i)
   XT(i) = XT(i + 1)
   XT(i + 1) = cambio
   End If
Next i
numorden = numorden + 1
Loop Until numorden > (numecuaciones - 1)
etemp(j) = XT(0)

For i = 0 To numecuaciones - 1
vectorini(i) = Abs(vectorini(i))
Next i
XT = vectorini
Next j

'Se obtiene el error selecionado por el usuario
If Tipoerror = True Then
e = etemp(1) - etemp(2)
Else
e = (etemp(1) - etemp(2)) / etemp(1)
End If

For j = 0 To numecuaciones - 1
MR(k - 1, j, 0) = X(j)
Next j
MR(k - 1, numecuaciones, 0) = Abs(e)
vectorini = X
```

```
Loop Until Abs(e) < tolerancia Or k >= Maxitera
If k >= Maxitera Then
mensaje = MsgBox("Número de iteraciones máximas superado, no fue posible
encontrar la respuesta; Sugerencia: incremente el número de iteraciones", 48,
"Iteraciones")
End If
Exit Sub
err1:
If err.Number <> 0 Then
   mensaje = MsgBox("Parámetros del algoritmo incompletos", 48, "Error en el
ingreso")
   End If

End Sub
Public Sub JacobiSNL()
Dim Evaluarfr As New McsExcel
Dim fx As String
Dim fr As String
Dim i As Integer
Dim k As Integer
Dim j As Integer
Dim mensaje As String
Dim e As Double
Dim XT() As Double
Dim numorden As Integer
Dim cambio As Double
Dim etemp(1 To 2) As Double
On Error GoTo err1
'Inicializamos las matrices de resultados y temporal
ReDim MR(0 To Maxitera - 1, 0 To numecuaciones, 0)
ReDim XT(0 To numecuaciones - 1) As Double
ReDim X(0 To numecuaciones - 1) As Double
k = 0
Do
k = k + 1
'Creamos un laso para reeemplazar todas las ecuaciones despejadas
'con los respectivos valores de las incognitas; para lo cual nos valemos
'de la función ReemplazarX para luego evaluarla con la ayuda del módulo
'MscExcel
For i = 0 To numecuaciones - 1
fx = MRS(i)
fr = ReemplazarX(fx, vectorini)
Evaluarfr.Tipox = True
Evaluarfr.fa = fr
Evaluarfr.evaluarf
X(i) = Evaluarfr.Fdy
```

```
Next i
'Ordenamos los vectores X y vectorini para sacar las normas para obtener
'el error
XT = X
For i = 0 To numecuaciones - 1
XT(i) = Abs(XT(i))
Next i

For j = 1 To 2
numorden = 1
Do
For i = 0 To numecuaciones - 1 - numorden
   If XT(i) <= XT(i + 1) Then
   cambio = XT(i)
   XT(i) = XT(i + 1)
   XT(i + 1) = cambio
   End If
Next i
numorden = numorden + 1
Loop Until numorden > (numecuaciones - 1)
etemp(j) = XT(0)

For i = 0 To numecuaciones - 1
vectorini(i) = Abs(vectorini(i))
Next i
XT = vectorini
Next j

'Se obtiene el error selecionado por el usuario
If Tipoerror = True Then
e = etemp(1) - etemp(2)
Else
e = (etemp(1) - etemp(2)) / etemp(1)
End If

For j = 0 To numecuaciones - 1
MR(k - 1, j, 0) = X(j)
Next j
MR(k - 1, numecuaciones, 0) = Abs(e)
vectorini = X


Loop Until Abs(e) < tolerancia Or k >= Maxitera
If k >= Maxitera Then
```

```
mensaje = MsgBox("Número de iteraciones máximas superado, no fue posible
encontrar la respuesta; Sugerencia: incremente el número de iteraciones", 48,
"Iteraciones")
End If
Exit Sub
err1:
If err.Number <> 0 Then
   mensaje = MsgBox("Parámetros del algoritmo incompletos", 48, "Error en el
ingreso")
   End If

End Sub
Public Sub GaussSeidelSNL()
Dim Evaluarfr As New McsExcel
Dim fx As String
Dim fr As String
Dim i As Integer
Dim k As Integer
Dim j As Integer
Dim mensaje As String
Dim e As Double
Dim XT() As Double
Dim vectorinitemp() As Double
Dim numorden As Integer
Dim cambio As Double
Dim etemp(1 To 2) As Double
On Error GoTo err1
'Inicializamos las matrices de resultados y temporal
ReDim MR(0 To Maxitera - 1, 0 To numecuaciones, 0)
ReDim XT(0 To numecuaciones - 1) As Double
ReDim X(0 To numecuaciones - 1) As Double
ReDim vectorinitemp(0 To numecuaciones - 1) As Double
k = 0
Do
vectorinitemp = vectorini
k = k + 1
'Creamos un laso para reeemplazar todas las ecuaciones despejadas
'con los respectivos valores de las incognitas; para lo cual nos valemos
'de la función ReemplazarX para luego evaluarla con la ayuda del módulo
'MscExcel
For i = 0 To numecuaciones - 1
fx = MRS(i)
fr = ReemplazarX(fx, vectorinitemp)
Evaluarfr.Tipox = True
Evaluarfr.fa = fr
Evaluarfr.evaluarf
```

```
cambio = Evaluarfr.Fdy
X(i) = cambio
vectorinitemp(i) = cambio
Next i
'Ordenamos los vectores X y vectorini para sacar las normas para obtener
'el error
XT = X
For i = 0 To numecuaciones - 1
XT(i) = Abs(XT(i))
Next i

For j = 1 To 2
numorden = 1
Do
For i = 0 To numecuaciones - 1 - numorden
   If XT(i) <= XT(i + 1) Then
   cambio = XT(i)
   XT(i) = XT(i + 1)
   XT(i + 1) = cambio
   End If
Next i
numorden = numorden + 1
Loop Until numorden > (numecuaciones - 1)
etemp(j) = XT(0)

For i = 0 To numecuaciones - 1
vectorini(i) = Abs(vectorini(i))
Next i
XT = vectorini
Next j

'Se obtiene el error selecionado por el usuario
If Tipoerror = True Then
e = etemp(1) - etemp(2)
Else
e = (etemp(1) - etemp(2)) / etemp(1)
End If

For j = 0 To numecuaciones - 1
MR(k - 1, j, 0) = X(j)
Next j
MR(k - 1, numecuaciones, 0) = Abs(e)
vectorini = X


Loop Until Abs(e) < tolerancia Or k >= Maxitera
```

```
If k >= Maxitera Then
mensaje = MsgBox("Número de iteraciones máximas superado, no fue posible
encontrar la respuesta; Sugerencia: incremente el número de iteraciones", 48,
"Iteraciones")
End If
Exit Sub
err1:
If err.Number <> 0 Then
   mensaje = MsgBox("Parámetros del algoritmo incompletos", 48, "Error en el
ingreso")
   End If

End Sub
Public Sub NewtonSNL()
Dim MRt() As Double
Dim Evaluarfr As New McsExcel
Dim fx As String
Dim fr As String
Dim i As Integer
Dim k As Integer
Dim j As Integer
Dim mensaje As String
Dim e As Double
Dim XT() As Double
Dim numorden As Integer
Dim cambio As Double
Dim etemp(1 To 2) As Double
On Error GoTo err1
'Inicializamos las matrices de resultados y temporal
ReDim MR(0 To Maxitera - 1, 0 To numecuaciones, 0)
ReDim MRt(0 To Maxitera - 1, 0 To numecuaciones, 0)
ReDim XT(0 To numecuaciones - 1) As Double
ReDim X(0 To numecuaciones - 1) As Double
ReDim Ma(0 To numecuaciones - 1, 0 To numecuaciones - 1) As Double
ReDim Mb(0 To numecuaciones - 1) As Double
k = 0
Do
k = k + 1
'Creamos un laso para reeemplazar todas las ecuaciones despejadas
'con los respectivos valores de las incognitas; para lo cual nos valemos
'de la función ReemplazarX para luego evaluarla con la ayuda del módulo
'MscExcel
'Llenamos la matriz Ma despues de evaluar el Jacobiano
For i = 0 To numecuaciones - 1
   For j = 0 To numecuaciones - 1
   fx = MRS(i, j)
```

```
    fr = ReemplazarX(fx, vectorini)
    Evaluarfr.Tipox = True
    Evaluarfr.fa = fr
    Evaluarfr.evaluarf
    Ma(i, j) = Evaluarfr.Fdy
    Next j
Next i
'Llenamos la matriz Mb despues de evaluar las funciones
For i = 0 To numecuaciones - 1
    fx = MRS(i, numecuaciones)
    fr = ReemplazarX(fx, vectorini)
    Evaluarfr.Tipox = True
    Evaluarfr.fa = fr
    Evaluarfr.evaluarf
    Mb(i) = -1 * Evaluarfr.Fdy
Next i
'Resolvemos el sistema de ecuaciones mediante Eliminación Gaussiana con
'pivotacion escalonada de columna

 EliGaussPivotación

'Obtenemos el nuevo punto sumando el resultado del sistema resuelto
'al vector inicial

For i = 0 To numecuaciones - 1
X(i) = X(i) + vectorini(i)
Next i

'Ordenamos los vectores X y vectorini para sacar las normas para obtener
'el error
XT = X
For i = 0 To numecuaciones - 1
XT(i) = Abs(XT(i))
Next i

For j = 1 To 2
numorden = 1
Do
For i = 0 To numecuaciones - 1 - numorden
    If XT(i) <= XT(i + 1) Then
    cambio = XT(i)
    XT(i) = XT(i + 1)
    XT(i + 1) = cambio
    End If
Next i
numorden = numorden + 1
```

```
Loop Until numorden > (numecuaciones - 1)
etemp(j) = XT(0)

For i = 0 To numecuaciones - 1
vectorini(i) = Abs(vectorini(i))
Next i
XT = vectorini
Next j

'Se obtiene el error selecionado por el usuario
If Tipoerror = True Then
e = etemp(1) - etemp(2)
Else
e = (etemp(1) - etemp(2)) / etemp(1)
End If

For j = 0 To numecuaciones - 1
MRt(k - 1, j, 0) = X(j)
Next j
MRt(k - 1, numecuaciones, 0) = Abs(e)
vectorini = X


Loop Until Abs(e) < tolerancia Or k >= Maxitera
If k >= Maxitera Then
mensaje = MsgBox("Número de iteraciones máximas superado, no fue posible
encontrar la respuesta; Sugerencia: incremente el número de iteraciones", 48,
"Iteraciones")
End If

MR = MRt




Exit Sub
err1:
If err.Number <> 0 Then
   mensaje = MsgBox("Parámetros del algoritmo incompletos", 48, "Error en el
ingreso")
   End If

End Sub
'Permite la realización de la pivotación parcial escalonada de columna para disminuir
el error
'de redondeo para lo cual se le pasa la columna actual para realizar la pivotación
```

```
'y esta se encarga de realizar el cambio de filas correspondiente
Private Sub Pivotaciónparcial(colum As Integer)
Dim k As Integer
Dim i As Integer
Dim cambio As Double
Dim filapivote As Integer
Dim filatmp() As Double
Dim Pivote() As Double
Dim Registro() As Integer
On Error GoTo err1
ReDim filatmp(0 To numecuaciones - 1)
ReDim Pivote(colum To numecuaciones - 1)
ReDim Registro(colum To numecuaciones - 1)
'Inicializamos la matriz Registro para conocer la fila quqe contiene el pivote
For i = colum To numecuaciones - 1
Registro(i) = i
Next i

'Hacemos una copia del valor absoluto de la fila de Ma a filatmp
For k = colum To numecuaciones - 1
   For i = 0 To numecuaciones - 1
   filatmp(i) = Abs(Ma(k, i))
   Next i
'Ordenamos la filatmp para obtener el máximo coeficiente de la fila
numorden = 1
Do
For i = 0 To numecuaciones - 1 - numorden
   If filatmp(i) <= filatmp(i + 1) Then
   cambio = filatmp(i)
   filatmp(i) = filatmp(i + 1)
   filatmp(i + 1) = cambio
   End If
Next i
numorden = numorden + 1
Loop Until numorden > (numecuaciones - 1)
'Guardamos los máximos de las filas en Pivote
Pivote(k) = Abs(Ma(k, colum)) / filatmp(0)
Next k
'Ordenamos las los máximos de las filas
numorden = 1
Do
For i = colum To numecuaciones - 1 - numorden

   If Pivote(i) < Pivote(i + 1) Then
   cambio = Pivote(i)
   Pivote(i) = Pivote(i + 1)
```

```
    Pivote(i + 1) = cambio
    cambio = Registro(i)
    Registro(i) = Registro(i + 1)
    Registro(i + 1) = cambio
    End If
Next i
numorden = numorden + 1
Loop Until numorden > (numecuaciones - 1)
'Cambiamos la fila actual por la que tiene el pivote
filapivote = Registro(colum)
If filapivote <> colum Then
For j = 0 To numecuaciones - 1
  cambio = Ma(colum, j)
  Ma(colum, j) = Ma(filapivote, j)
  Ma(filapivote, j) = cambio
Next j
cambio = Mb(colum)
Mb(colum) = Mb(filapivote)
Mb(filapivote) = cambio
End If
Exit Sub
err1:

End Sub




Private Function ReemplazarX(f As String, valores() As Double) As String
  Dim nx
  Dim NF
  Dim nL
  Dim Ns
  Dim nr
  Dim variables
On Error GoTo err1
  NF = f
  For i = 0 To numecuaciones - 1

  nx = valores(i)
  nL = Len(nx)
  Ns = InStr(1, nx, ",")
  nr = nL - Ns
  variables = "X" & i + 1
```

```
  If Ns <> 0 Then


  nx = Mid(nx, 1, InStr(1, nx, ",") - 1) & "." & Mid(nx, InStr(1, nx, ",") + 1, Len(Mid(nx, 1,
nr)))
  End If

  While InStr(1, NF, variables) > 0
    NF = Mid(NF, 1, InStr(1, NF, variables) - 1) & nx & Mid(NF, InStr(1, NF, variables)
+ 2, Len(NF))
  Wend
 Next i
  ReemplazarX = UCase(NF)
Exit Function
err1:
End Function


Public Sub Cholesky()
Dim i As Integer
Dim j As Integer
Dim k As Integer
Dim mensaje As String
Dim sum As Double
Dim LT() As Double
Dim Ltmp() As Double
Dim L() As Double
Dim Xtmp() As Double
On Error GoTo err1
'Implementamos la descomposición
ReDim LT(1 To numecuaciones, 1 To numecuaciones) As Double
ReDim L(1 To numecuaciones, 1 To numecuaciones) As Double
ReDim Ltmp(1 To numecuaciones, 1 To numecuaciones) As Double
ReDim MR(0 To numecuaciones - 1, 0 To numecuaciones, 0 To 1)
For i = 1 To numecuaciones
   For j = 1 To numecuaciones
   Ltmp(i, j) = Ma(i - 1, j - 1)
   Next j
Next i

' Obtenemos L
For k = 1 To numecuaciones
   For i = 1 To k - 1
   sum = 0
      For j = 1 To i - 1
      sum = sum + Ltmp(i, j) * Ltmp(k, j)
```

```
        Next j
    If Ltmp(i, i) <> 0 Then
    Ltmp(k, i) = (Ltmp(k, i) - sum) / Ltmp(i, i)
    Else
    mensaje = MsgBox("A es singular, no hay solución o no es única ", 16, "Fracaso")
    End If
    L(k, i) = Ltmp(k, i)
    Next i
    sum = 0
    For j = 1 To k - 1
    sum = sum + L(k, j) ^ 2
    Next j
    If (Ltmp(k, k) > sum) Then
    Ltmp(k, k) = (Ltmp(k, k) - sum) ^ (1 / 2)
    Else
    mensaje = MsgBox("A debe ser definida positiva y simétrica ", 16, "Fracaso")
    End If
    L(k, k) = Ltmp(k, k)
Next k
'Asignamos la respuesta a la matriz de resultados
For i = 1 To numecuaciones - 1
    For j = 1 To numecuaciones
    MR(i - 1, j - 1, 0) = L(i, j)
    Next j
Next i

' Obtenemos la L transpuesta
For i = 1 To numecuaciones
    For j = 1 To numecuaciones
    LT(j, i) = L(i, j)
    Next j
Next i
'Asignamos la respuesta a la matriz de resultados
For i = 1 To numecuaciones - 1
    For j = 1 To numecuaciones
    MR(i - 1, j - 1, 1) = LT(i, j)
    Next j
Next i

'Asignamos la matriz Mb a la matriz de resultados
For i = 1 To numecuaciones
MR(i - 1, numecuaciones, 0) = Mb(i - 1)
Next i


'Sustitución hacia adelante
```

```
For i = 2 To numecuaciones
sum = Mb(i - 1)
   For j = 1 To i - 1
   sum = sum - L(i, j) * Mb(j - 1)
   Next j
   Mb(i - 1) = sum
Next i
'Asignamos la respuesta a la matriz de resultados
For i = 1 To numecuaciones
MR(i - 1, numecuaciones, 1) = Mb(i - 1)
Next i


'Sustitución hacia atrás
ReDim X(0 To numecuaciones - 1) As Double
ReDim Xtmp(1 To numecuaciones) As Double

Xtmp(numecuaciones) = Mb(numecuaciones - 1) / LT(numecuaciones,
numecuaciones)
For i = numecuaciones To 1 Step -1
sum = 0
   For j = i + 1 To n
   sum = sum + LT(i, j) * Xtmp(j)
   Next j
Xtmp(i) = (Mb(i - 1) - sum) / LT(i, i)
Next i

For i = 1 To numecuaciones
X(i - 1) = Xtmp(i)
Next i

Exit Sub
err1:
If err.Number <> 0 Then
   mensaje = MsgBox("Parámetros del algoritmo incompletos", 48, "Error en el
ingreso")
   End If

End Sub
```

Graficar una función
Interfaz FrmGraficando

```
Option Explicit
'En Microsoft TechNet puedes encontrar este artículo:
'HOWTO: Use HTML Help API in a Visual Basic 5.0 Application
```

```
'PSS ID Number: Q183434
'
'Aunque la definición de la Enumeración y la primera declaración
'es de las news
'
'Htmlhelp consts
Private Enum HH_COMMAND
   HH_DISPLAY_TOPIC = &H0
   HH_HELP_FINDER = &H0       ' WinHelp equivalent
   HH_DISPLAY_TOC = &H1       ' not currently implemented
   HH_DISPLAY_INDEX = &H2     ' not currently implemented
   HH_DISPLAY_SEARCH = &H3    ' not currently implemented
   HH_SET_WIN_TYPE = &H4
   HH_GET_WIN_TYPE = &H5
   HH_GET_WIN_HANDLE = &H6
   HH_GET_INFO_TYPES = &H7    ' not currently implemented
   HH_SET_INFO_TYPES = &H8    ' not currently implemented
   HH_SYNC = &H9
   HH_ADD_NAV_UI = &HA        ' not currently implemented
   HH_ADD_BUTTON = &HB        ' not currently implemented
   HH_GETBROWSER_APP = &HC    ' not currently implemented
   HH_KEYWORD_LOOKUP = &HD
   HH_DISPLAY_TEXT_POPUP = &HE ' display string resource id
                       ' or text in a popup window
   HH_HELP_CONTEXT = &HF      ' display mapped numeric value
                       ' in dwData
   HH_TP_HELP_CONTEXTMENU     ' Text pop-up help, similar to
                       ' WinHelp's HELP_CONTEXTMENU.
   HH_TP_HELP_WM_HELP = &H11  ' text pop-up help, similar to
                       ' WinHelp's HELP_WM_HELP.
   HH_CLOSE_ALL = &H12        ' close all windows opened directly
                       ' or indirectly by the caller
   HH_ALINK_LOOKUP = &H13     ' ALink version of HH_KEYWORD_LOOKUP
End Enum

'HtmlHelp api call
'NOTA: Si se usa esta forma, hay que indicar el último parámetro
'     con la palabra ByVal delante...
'Private Declare Function HtmlHelp Lib "hhctrl.ocx" Alias "HtmlHelpA" _
   (ByVal hwndCaller As Long, ByVal pszFile As String, _
   ByVal uCommand As HH_COMMAND, dwData As Any) As Long
'Con esta funciona perfectamente
Private Declare Function HtmlHelp Lib "hhctrl.ocx" Alias "HtmlHelpA" _
   (ByVal hwndCaller As Long, ByVal pszFile As String, _
   ByVal uCommand As HH_COMMAND, ByVal dwData As Long) As Long
```

```
Dim appExcel As Excel.Application
Dim wbExcel As Excel.Workbook
Dim shtExcel As Excel.Worksheet
Dim n As Integer
Dim Xmin, XLng, Ymax, YLng, xdiv, ydiv
Dim XMind, XMaxd, YMaxd, YMind, XDivd, YDivd, Fd, Fd2, Fd3
Dim X, Y, xa, XT, Yt, Xmax, Ymin
Dim r, g, b As Integer
Dim f
Dim nodibuja As String
Dim altoDatos
Dim colororg
Private Acepta As Boolean
Private VLConexion As ADODB.Connection
Private VLRegistro As ADODB.Recordset
Private VLNombre As String
Private VLNombreTrabajo As String
Private VLVerifica As Boolean
Private VLValores As String
Private VLValoresMatriz(1 To 1000) As Integer
Private VLRecupera As Boolean

Property Let Aceptar(parAceptar As Boolean)
   Acepta = parAceptar
End Property

Property Let Nombre(parNombre As String)
   VLNombre = parNombre
End Property

Property Let NombreTrabajo(parNombreTrabajo As String)
   VLNombreTrabajo = parNombreTrabajo
End Property



Private Sub CheckGraficandofunción_Click()
If CheckGraficandofunción.Value = 1 Then
   LblGraficandof1.Visible = True
   LblGraficandof2.Visible = True
   LblGraficandof3.Visible = True
   If VLRecupera = True Then
   Else
      TxtF.Text = " "
```

```
        TxtF2.Text = " "
        TxtF3.Text = " "
    End If
        TxtF3.Visible = True
        TxtF2.Visible = True
        TxtF.Visible = True
    Shapef1.Visible = True
    Shapef2.Visible = True
    Shapef3.Visible = True
    Command1.Visible = True
    graficarabrir.Enabled = True
    VLRecupera = False
Else
LblGraficandof1.Visible = False
LblGraficandof2.Visible = False
LblGraficandof3.Visible = False
TxtF.Visible = False
TxtF2.Visible = False
TxtF3.Visible = False
Shapef1.Visible = False
Shapef2.Visible = False
Shapef3.Visible = False
    If CheckGraficandodatos.Value = 1 Then
        graficarabrir.Enabled = True
    Else
        graficarabrir.Enabled = False
    End If
End If

End Sub

Private Sub CheckGraficandodatos_Click()
If CheckGraficandodatos.Value = 1 Then
    LblGraficandoN.Visible = True
    If VLRecupera = True Then
    Else
        TxtGraficandoN.Text = ""
        Fg2.Visible = False
    End If
    TxtGraficandoN.Visible = True
    Command1.Visible = True
    graficarabrir.Enabled = True
    VLRecupera = False
Else
    LblGraficandoN.Visible = False
    TxtGraficandoN.Visible = False
```

```
    TxtGraficandoN.Text = ""
    Fg2.Clear
    Fg2.Visible = False
    If CheckGraficandofunción.Value = 1 Then
       graficarabrir.Enabled = True
    Else
       graficarabrir.Enabled = False
    End If

End If

End Sub

Private Sub Form_Load()

   Dim i As Integer
    graficarabrir.Enabled = False
   ' Estrechar la primera columna.
   Fg2.ColWidth(0) = Fg2.ColWidth(0) / 2
   Fg2.ColAlignment(0) = 1   ' Centrado.

   ' Etiquetas de filas y de columnas.
   For i = Fg2.FixedRows To Fg2.Rows - 1
      Fg2.TextArray(Fgi(i, 0)) = i
   Next
   For i = Fg2.FixedCols To Fg2.Cols - 1
      Fg2.TextArray(Fgi(0, i)) = i
   Next

   ' Inicializar el cuadro de edición (lo carga ahora).
   txtEdit = ""
Graficarimprimir.Enabled = False
Graficandoguardarcomo.Enabled = False
graficarabrir.Enabled = False
End Sub

Private Sub Form_QueryUnload(Cancel As Integer, UnloadMode As Integer)
Frmmodulos.Visible = True
End Sub

Private Sub Graficandoguardarcomo_Click()
If PLCamposVacios = True Then
   MsgBox "Llene los campos necesarios para guardar el trabajo", vbInformation,
"Campos vacios"
   Exit Sub
End If
```

```
    FrmNombre.Show vbModal
    If Acepta = True Then
       PLConexion True
       PLGuardarBDD
    End If
End Sub

Private Sub graficarabrir_Click()
    FrmRecuperaTrabajo.VerGrafico = "Graficar"
    FrmRecuperaTrabajo.Show vbModal
    If Acepta = True Then
       PLConexion True
       PLRecupera
    End If
End Sub

Private Sub Graficarayuda_Click()
'Así se llamaría para mostrar un tópico de la ayuda
    Dim h As Long

    'h = HtmlHelp(Me.hWnd, "Milton.hlp", HH_DISPLAY_TOPIC, 0&)

    h = HtmlHelp(Me.hWnd, "Milton.hlp", HH_HELP_CONTEXT, 2010&)
End Sub

Private Sub Graficarimprimir_Click()
Dim EndTime As Date
    Form1.SetearVariable = 1
    Sincolor

        EndTime = DateAdd("s", 1 / 1E+16, Now)
        Do Until Now > EndTime
          DoEvents
        Loop

    Set Form1.Picture1.Picture = CaptureClient(Me)
    Form1.Visible = True
    FrmGraficando.Visible = False
    Concolor
End Sub

Private Sub Graficarsalir_Click()
Unload Me
Frmmodulos.Visible = True
End Sub
```

```
'Manejo de datos para graficar
Function Fgi(r As Integer, c As Integer) As Integer
  Fgi = c + Fg2.Cols * r
End Function

Sub Fg2_KeyPress(KeyAscii As Integer)
  MSHFlexGridEdit Fg2, txtEdit, KeyAscii
End Sub

Sub Fg2_DblClick()
  MSHFlexGridEdit Fg2, txtEdit, 32 ' Simula un espacio.
End Sub

Sub MSHFlexGridEdit(MSHFlexGrid As Control, _
Edt As Control, KeyAscii As Integer)
Dim p As Double
  ' Usar el carácter escrito.
  Select Case KeyAscii

  ' Un espacio significa modificar el texto actual.
  Case 0 To 32
    Edt = MSHFlexGrid
    Edt.SelStart = 1000

  ' Otro carácter reemplaza el texto actual.
  Case Else
    Edt = Chr(KeyAscii)
    Edt.SelStart = 1
  End Select

 ' Mostrar Edt en la posición correcta.
  p = MSHFlexGrid.CellLeft
  If p = 45 Then
  Edt.Move MSHFlexGrid.Left, _
    MSHFlexGrid.Top + MSHFlexGrid.CellTop + 210, _
    2000, _
    MSHFlexGrid.CellHeight - 8
  Else
  Edt.Move MSHFlexGrid.Left + 2000, _
    MSHFlexGrid.Top + MSHFlexGrid.CellTop + 180, _
    2000, _
    MSHFlexGrid.CellHeight - 8
  End If
  Edt.Visible = True
```

```
  ' Y hacer que funcione.
  Edt.SetFocus
End Sub

Sub txtEdit_KeyPress(KeyAscii As Integer)
  ' Elimina los retornos para quitar los pitidos.
  If KeyAscii = Asc(vbCr) Then KeyAscii = 0
End Sub

Sub txtEdit_KeyDown(KeyCode As Integer, _
Shift As Integer)
  EditKeyCode Fg2, txtEdit, KeyCode, Shift
End Sub

Sub EditKeyCode(MSHFlexGrid As Control, Edt As _
Control, KeyCode As Integer, Shift As Integer)

  ' Procesamiento del control de edición estándar.
  Select Case KeyCode

  Case 27   ' ESC: ocultar, devuelve el enfoque a          ' MSFlexGrid.
    Edt.Visible = False
    MSHFlexGrid.SetFocus

  Case 13   ' ENTRAR devuelve el enfoque a MSHFlexGrid.
    MSHFlexGrid.SetFocus

  Case 38     ' Arriba.
    MSHFlexGrid.SetFocus
    DoEvents
    If MSHFlexGrid.Row > MSHFlexGrid.FixedRows Then
      MSHFlexGrid.Row = MSHFlexGrid.Row - 1
    End If

  Case 40     ' Abajo.
    MSHFlexGrid.SetFocus
    DoEvents
    If MSHFlexGrid.Row < MSHFlexGrid.Rows - 1 Then
      MSHFlexGrid.Row = MSHFlexGrid.Row + 1
    End If
  End Select
End Sub

Sub Fg2_GotFocus()
  If txtEdit.Visible = False Then Exit Sub
  Fg2 = txtEdit
```

```
  txtEdit.Visible = False
End Sub

Sub Fg2_LeaveCell()
  If txtEdit.Visible = False Then Exit Sub
  Fg2 = txtEdit
  txtEdit.Visible = False
End Sub



'muestra el número de celdas para el ingreso de datos
Private Sub TxtGraficandoN_Change()
Dim mensaje As String

If Val(TxtGraficandoN) < 32767 Then
n = Val(TxtGraficandoN.Text)

If n = 0 Then
n = 1
End If
   Fg2.Clear
   Fg2.Cols = 2
   Fg2.Rows = n + 1
   Fg2.FixedCols = 0
   Fg2.FixedRows = 1
   Fg2.ColAlignmentFixed(0) = 4
   Fg2.ColAlignmentFixed(1) = 4
   Fg2.ColAlignment(0) = 4
   Fg2.ColAlignment(1) = 4
   Fg2.TextArray(Fgi(0, 0)) = "X"
   Fg2.TextArray(Fgi(0, 1)) = "Y"
   Fg2.ColWidth(0) = 800
   Fg2.ColWidth(1) = 800

   Fg2.Visible = True
Else
mensaje = MsgBox("Fuera del rango ", 48, "Error en el ingreso")
End If
End Sub
Private Function ReemplazarF(X) As String
 Dim nx
 Dim NF
 Dim nL
 Dim Ns
 Dim nr
```

```
  NF = f
  nx = X
  nL = Len(nx)
  Ns = InStr(1, nx, ",")
  nr = nL - Ns
  If Ns <> 0 Then


  nx = Mid(nx, 1, InStr(1, nx, ",") - 1) & "." & Mid(nx, InStr(1, nx, ",") + 1, Len(Mid(nx, 1,
nr)))
  End If

  While InStr(1, NF, "X") > 0
    NF = Mid(NF, 1, InStr(1, NF, "X") - 1) & nx & Mid(NF, InStr(1, NF, "X") + 1,
Len(NF))
  Wend

  ReemplazarF = UCase(NF)
End Function

Private Sub PrepararPlano()
   Dim mensaje As String
   On Error GoTo err1

  Xmin = CDbl(TxtXMin)
  Xmax = CDbl(TxtXMax)
  xdiv = CDbl(TxtXDiv)
  Ymax = CDbl(TxtYMax)
  Ymin = CDbl(TxtYMin)
  ydiv = CDbl(TxtYDiv)
  XLng = Xmax - Xmin
  YLng = Ymax - Ymin

  PBPlano.ScaleLeft = Xmin
  PBPlano.ScaleWidth = XLng
  PBPlano.ScaleTop = Ymax
  PBPlano.ScaleHeight = -YLng
  PBPlano.Cls
  'Dibuja el eje X
  PBPlano.Line (Xmin, 0)-(Xmin + XLng, 0), vbRed
  For X = xdiv To XLng Step xdiv
    PBPlano.Line (X, -YLng / 100)-(X, YLng / 100), vbRed
    PBPlano.Line (-X, -YLng / 100)-(-X, YLng / 100), vbRed
  Next
  'Dibuja el eje Y
```

```
  PBPlano.Line (0, Ymax)-(0, Ymax - YLng), vbRed
  For Y = ydiv To YLng / ydiv Step ydiv
    PBPlano.Line (-XLng / 100, Y)-(XLng / 100, Y), vbRed
    PBPlano.Line (-XLng / 100, -Y)-(XLng / 100, -Y), vbRed
  Next
Exit Sub
err1:
mensaje = MsgBox("No ha ingresado las escalas para la graficaión", 48, "Error en el
ingreso")
End Sub



Private Sub Command1_Click()
' Presentamos los valores de las escalas el gráfico
LblGraficandoizquierda.Caption = TxtXMin.Text
LblGraficandoderecha.Caption = TxtXMax.Text
LblGraficandoarriba.Caption = TxtYMax.Text
LblGraficandoabajo.Caption = TxtYMin.Text
XMind = TxtXMin.Text
XMaxd = TxtXMax.Text
XDivd = TxtXDiv.Text
YMaxd = TxtYMax.Text
YMind = TxtYMin.Text
YDivd = TxtYDiv.Text
Fd = TxtF
Fd2 = TxtF2
Fd3 = TxtF3
PBPlano.DrawWidth = 1
PrepararPlano
'Enviamos a graficar
 If CheckGraficandofunción.Value = 1 Then

r = 0
g = 0
b = 255
f = TxtF
If (TxtF = "" Or TxtF = " ") Then
nodibuja = True
Else
nodibuja = False
End If
If nodibuja = False Then
Graficar
End If
```

```
r = 0
g = 0
b = 0
f = TxtF2
If (TxtF2 = "" Or TxtF2 = " ") Then
nodibuja = True
Else
nodibuja = False
End If
If nodibuja = False Then
Graficar
End If

r = 0
g = 255
b = 0
f = TxtF3
If (TxtF3 = "" Or TxtF3 = " ") Then
nodibuja = True
Else
nodibuja = False
End If
If nodibuja = False Then
Graficar
End If
End If
If CheckGraficandodatos.Value = 1 Then
PBPlano.DrawWidth = 2
Graficarpuntos
End If
'Presentamos controles  de grafico
LblGraficandox.Visible = True
LblGraficandoy.Visible = True
CmdGraficandodesahacer.Visible = True
Graficarimprimir.Enabled = True
Graficandoguardarcomo.Enabled = True

End Sub

Private Sub Graficar()
  Dim xa, ya, Xu, Yu

    On Error Resume Next 'ignorar errores
    Set appExcel = GetObject(, "Excel.Application") 'buscar una copia de Excel en
ejecución
    If err.Number <> 0 Then 'Si no se ejecuta Excel
```

```
    Set appExcel = CreateObject("Excel.Application") 'ejcutarlo
  End If
  err.Clear   ' Borrar el objeto Err si se produce un error.

  On Error GoTo 0 'Reaunudar el procesamiento normal de errores

  Set wbExcel = appExcel.Workbooks.Add
  Set shtExcel = wbExcel.Worksheets(1)


 'Dibuja la curva
 xa = Xmin
 'shtExcel.Application.Visible = True
 On Error Resume Next
 shtExcel.Range("A1").Formula = "= " & ReemplazarF(xa)
 PBPlano.PSet (X, Y), vbBlue
 ya = Cells(1, 1)
 For X = Xmin + 0.1 To Xmin + XLng Step 0.1
  shtExcel.Range("A1").Formula = "= " & ReemplazarF(X)
  Y = Cells(1, 1)
  PBPlano.Line (X, Y)-(xa, ya), RGB(r, g, b)
  xa = X: ya = Y
 Next
 err.Clear
 On Error GoTo 0

 wbExcel.Close False
 appExcel.Quit
 Set shtExcel = Nothing
 Set wbExcel = Nothing
 Set appExcel = Nothing
End Sub

Private Sub CmdGraficandodesahacer_Click()
 Dim nodibuja As Boolean
 TxtXMin.Text = XMind
  TxtXMax.Text = XMaxd
  TxtXDiv.Text = XDivd
  TxtYMax.Text = YMaxd
  TxtYMin.Text = YMind
  TxtYDiv.Text = YDivd
  TxtF = Fd
  TxtF2 = Fd2
  TxtF3 = Fd3
LblGraficandoizquierda.Caption = TxtXMin.Text
LblGraficandoderecha.Caption = TxtXMax.Text
```

```
LblGraficandoarriba.Caption = TxtYMax.Text
LblGraficandoabajo.Caption = TxtYMin.Text

PBPlano.DrawWidth = 1
PrepararPlano
'Enviamos a graficar
If CheckGraficandofunción.Value = 1 Then

r = 0
g = 0
b = 255
f = TxtF
If (TxtF = "" Or TxtF = " ") Then
nodibuja = True
Else
nodibuja = False
End If
If nodibuja = False Then
Graficar
End If
r = 0
g = 0
b = 0
f = TxtF2
If (TxtF2 = "" Or TxtF2 = " ") Then
nodibuja = True
Else
nodibuja = False
End If
If nodibuja = False Then
Graficar
End If

r = 0
g = 255
b = 0
f = TxtF3
If (TxtF3 = "" Or TxtF3 = " ") Then
nodibuja = True
Else
nodibuja = False
End If
If nodibuja = False Then
Graficar
End If
End If
```

```
If CheckGraficandodatos.Value = 1 Then
PBPlano.DrawWidth = 2
Graficarpuntos
End If
End Sub


Private Sub PBPlano_MouseDown(Button As Integer, Shift As Integer, X As Single, Y
As Single)
 If Button = vbLeftButton Then
 PBPlano.MousePointer = 2
XT = X
Yt = Y
End If
End Sub

Private Sub PBPlano_MouseMove(Button As Integer, Shift As Integer, X As Single, Y
As Single)
LblGraficandox.Caption = " Coordenadas= (" & X
LblGraficandoy.Caption = Y & ")"
End Sub

Private Sub PBPlano_MouseUp(Button As Integer, Shift As Integer, X As Single, Y
As Single)
If Button = vbLeftButton Then
   If PBPlano.MousePointer = 2 Then
   PBPlano.MousePointer = 0
   End If
TxtXMin.Text = XT
TxtYMin.Text = Y
TxtXMax.Text = X
TxtYMax.Text = Yt

LblGraficandoizquierda.Caption = XT
LblGraficandoabajo.Caption = Y
LblGraficandoderecha.Caption = X
LblGraficandoarriba.Caption = Yt
PBPlano.DrawWidth = 1
PrepararPlano
'Enviamos a graficar
 If CheckGraficandofunción.Value = 1 Then

r = 0
g = 0
b = 255
f = TxtF
```

```
If (TxtF = "" Or TxtF = " ") Then
nodibuja = True
Else
nodibuja = False
End If
If nodibuja = False Then
Graficar
End If
r = 0
g = 0
b = 0
f = TxtF2
If (TxtF2 = "" Or TxtF2 = " ") Then
nodibuja = True
Else
nodibuja = False
End If
If nodibuja = False Then
Graficar
End If

r = 0
g = 255
b = 0
f = TxtF3
If (TxtF3 = "" Or TxtF3 = " ") Then
nodibuja = True
Else
nodibuja = False
End If
If nodibuja = False Then
Graficar
End If
End If
If CheckGraficandodatos.Value = 1 Then
PBPlano.DrawWidth = 2
Graficarpuntos
End If
End If
End Sub

Private Sub TxtGraficandoN_KeyPress(KeyAscii As Integer)
'filtra la entrada de datos que no sean números enteros y positivos
  If (KeyAscii < 48 Or KeyAscii > 57) Then

        If (KeyAscii <> 8) Then
```

```
        KeyAscii = 0
        End If

    End If
   If KeyAscii = Asc(vbCr) Then ' Elimina los retornos para quitar los pitidos.
   KeyAscii = 0
   End If

End Sub

Private Sub Graficarpuntos()
Dim i As Integer
Dim j As Integer
Dim dx As Double
Dim dy As Double
For i = 1 To n
dx = Val(Fg2.TextMatrix(i, 0))
dy = Val(Fg2.TextMatrix(i, 1))
PBPlano.PSet (dx, dy), RGB(125, 125, 255)
Next i
End Sub



Private Sub TxtXDiv_KeyPress(KeyAscii As Integer)
'filtra la entrada de datos que no sean números incluye el signo - y el punto
   If (KeyAscii < 48 Or KeyAscii > 57) Then
       If (KeyAscii = 45) Then
       Exit Sub
       End If
       If (KeyAscii <> 46) Then
          If (KeyAscii <> 8) Then
          KeyAscii = 0
          End If
       End If
    End If
   If KeyAscii = Asc(vbCr) Then ' Elimina los retornos para quitar los pitidos.
   KeyAscii = 0
   End If
End Sub

Private Sub TxtXMax_KeyPress(KeyAscii As Integer)
'filtra la entrada de datos que no sean números
```

```
  If (KeyAscii < 48 Or KeyAscii > 57) Then
      If (KeyAscii = 45) Then
      Exit Sub
      End If
      If (KeyAscii <> 46) Then
         If (KeyAscii <> 8) Then
         KeyAscii = 0
         End If
      End If
   End If
  If KeyAscii = Asc(vbCr) Then ' Elimina los retornos para quitar los pitidos.
  KeyAscii = 0
  End If
End Sub

Private Sub TxtXMin_KeyPress(KeyAscii As Integer)
'filtra la entrada de datos que no sean números
  If (KeyAscii < 48 Or KeyAscii > 57) Then
      If (KeyAscii = 45) Then
      Exit Sub
      End If
      If (KeyAscii <> 46) Then
         If (KeyAscii <> 8) Then
         KeyAscii = 0
         End If
      End If
   End If
  If KeyAscii = Asc(vbCr) Then ' Elimina los retornos para quitar los pitidos.
  KeyAscii = 0
  End If
End Sub



Private Sub TxtYDiv_KeyPress(KeyAscii As Integer)
'filtra la entrada de datos que no sean números
  If (KeyAscii < 48 Or KeyAscii > 57) Then
      If (KeyAscii = 45) Then
      Exit Sub
      End If
      If (KeyAscii <> 46) Then
         If (KeyAscii <> 8) Then
         KeyAscii = 0
```

```
        End If
      End If
    End If
   If KeyAscii = Asc(vbCr) Then ' Elimina los retornos para quitar los pitidos.
   KeyAscii = 0
   End If
End Sub

Private Sub TxtYMax_KeyPress(KeyAscii As Integer)
'filtra la entrada de datos que no sean números
   If (KeyAscii < 48 Or KeyAscii > 57) Then
       If (KeyAscii = 45) Then
       Exit Sub
       End If
       If (KeyAscii <> 46) Then
          If (KeyAscii <> 8) Then
          KeyAscii = 0
          End If
       End If
    End If
   If KeyAscii = Asc(vbCr) Then ' Elimina los retornos para quitar los pitidos.
   KeyAscii = 0
   End If
End Sub

Private Sub TxtYMin_KeyPress(KeyAscii As Integer)
'filtra la entrada de datos que no sean números
   If (KeyAscii < 48 Or KeyAscii > 57) Then
       If (KeyAscii = 45) Then
       Exit Sub
       End If
       If (KeyAscii <> 46) Then
          If (KeyAscii <> 8) Then
          KeyAscii = 0
          End If
       End If
    End If
   If KeyAscii = Asc(vbCr) Then ' Elimina los retornos para quitar los pitidos.
   KeyAscii = 0
   End If
End Sub

'****** Conexion a la BDD
Private Sub PLConexion(parBDD As Boolean)
Dim VTCad As String
Dim VTBase As Database
```

```
Dim VTcadena As String
   VTcadena = App.Path & "/Seguridad.mdb"
   Set VLConexion = New ADODB.Connection
   VTCad = "Provider=Microsoft.Jet.OLEDB.4.0;Data Source='" & VTcadena &
"';Persist Security Info=False"
   VLConexion.Open VTCad
End Sub

Private Sub PLGuardarBDD()
   Dim VTCad As String
   Dim VTReg As ADODB.Recordset
   Dim VTID As Integer
   Dim VB As Integer
   Dim VTNumero As String
On Error GoTo error1
   PLVerificaNombre Trim(VLNombre)
      If VLVerifica = True Then   'Si existe un nombre con el mismo nombre actualiza
         VB = MsgBox("El nombre de ese trabajo de graficación ya existe en este
tema o en otro tema, desea reemplazarlo?", vbYesNo, "Guardar Trabajo Como")
         If VB = 6 Then
            If Trim(TxtGraficandoN.Text) = "" Then
               VLValores = ""
            Else
               PLGuardaDatos CInt(TxtGraficandoN.Text)
            End If
               VTCad = "update Graficacion set " _
                     & " Xmin= '" & Trim(TxtXMin.Text) & "', " _
                     & " Xmax='" & TxtXMax.Text & "', " _
                     & " Xdiv='" & TxtXDiv.Text & "', " _
                     & " Ymin= '" & Trim(TxtYMin.Text) & "', " _
                     & " Ymax='" & TxtYMax.Text & "', " _
                     & " Ydiv='" & TxtYDiv.Text & "', " _
                     & " F1= '" & TxtF.Text & "', " _
                     & " F2='" & TxtF2.Text & "', " _
                     & " F3='" & TxtF3.Text & "', " _
                     & " Numi='" & TxtGraficandoN.Text & "', " _
                     & " Datos='" & Trim(VLValores) & "' " _
                     & " where Nombre = '" & VLNombre & "'"
            VLConexion.Execute (VTCad)
            'cmdNuevo_Click
            MsgBox "Trabajo reemplazado correctamente", vbInformation, "Trabajo
reemplazado"
         End If
      Else   'Guarda el trabajo
         VTID = PLRecuperaCodigo
         If Trim(TxtGraficandoN.Text) = "" Then
```

```
            VLValores = ""
        Else
            PLGuardaDatos CInt(TxtGraficandoN.Text)
        End If
        If Trim(TxtGraficandoN.Text) = "" Then
            TxtGraficandoN.Text = 0
        End If
        VTCad = "insert into Graficacion values " _
              & "(" & CInt(VTID) & ", " _
              & " '" & TxtXMin.Text & "', '" & TxtXMax.Text & "', " _
              & " '" & TxtXDiv.Text & "', '" & TxtYMin.Text & "', " _
              & " '" & TxtYMax.Text & "', '" & TxtYDiv.Text & "', " _
              & " '" & TxtF.Text & "', '" & TxtF2.Text & "', " _
              & " '" & TxtF3.Text & "', '" & TxtGraficandoN.Text & "', " _
              & " '" & Trim(VLNombre) & "', '" & VGForma & "', " _
              & " " & VGIDUsuario & ", '" & VLValores & "')"
              '& " '" & Trim(VLNombre) & "', " & CInt(tipoalg) & ", " & CInt(VGIDUsuario)
& ")"
        VLConexion.Execute (VTCad)
        MsgBox "Trabajo guardado exitosamente", vbInformation, "Guardar trabajo
como"
    End If

Exit Sub
error1:
  If err.Number <> 0 Then
    MsgBox "Se ha producido el siguiente error: " & err.Description & "." _
          & "Por favor consultar con el administrador", vbCritical
  End If

End Sub

Private Sub PLVerificaNombre(parNombre As String)
Dim VTReg As ADODB.Recordset
Dim VTCad As String
Dim VTTabla As String
On Error GoTo error1
    VTCad = "select Nombre from Graficacion where Nombre = '" & parNombre & "'"
      Set VTReg = VLConexion.Execute(VTCad)
      With VTReg
        If Not VTReg.EOF Then
          .MoveFirst
          Do Until .EOF
            If Trim(VTReg(0)) = parNombre Then
              VLVerifica = True
              Exit Do
```

```
                    Else
                       VLVerifica = False
                    End If
                    .MoveNext
                 Loop
              Else
                 VLVerifica = False
              End If
           End With
Exit Sub
error1:

End Sub

Private Function PLRecuperaCodigo() As Integer
   Dim VTCad As String
   Dim VTReg As ADODB.Recordset
   Dim VTTabla As String
On Error GoTo err1
      VTCad = "select max(IDGraficacion) from Graficacion"
      Set VTReg = VLConexion.Execute(VTCad)
      If Not IsNull(VTReg(0)) Then
         PLRecuperaCodigo = VTReg(0) + 1
      Else
         PLRecuperaCodigo = 1
      End If
Exit Function
err1:

End Function


Private Sub PLGuardaDatos(parNume As String)
Dim i, j As Integer
Dim Nume As Integer
On Error GoTo error1
   VLValores = ""
   Nume = CInt(parNume)
      For i = 1 To Nume
         For j = 0 To 1
            VLValores = VLValores & Fg2.TextMatrix(i, j) & ";"
         Next j
      Next i
Exit Sub
error1:
   If err.Number <> 0 Then
```

```
      MsgBox "Se ha producido el siguiente error: " & err.Description & "." _
            & "Por favor consultar con el administrador", vbCritical
   End If
End Sub

Private Sub PLRecupera()
   Dim VTCad As String
   Dim VTReg As ADODB.Recordset
On Error GoTo err1

      VTCad = "select * from Graficacion where Nombre = '" & VLNombreTrabajo & "'
and Tema = '" & VGForma & "'"
      Set VTReg = VLConexion.Execute(VTCad)
      If Not VTReg.EOF Then
         TxtXMin.Text = Trim(VTReg(1))
         TxtXMax.Text = Trim(VTReg(2))
         TxtXDiv.Text = Trim(VTReg(3))
         TxtYMin.Text = Trim(VTReg(4))
         TxtYMax.Text = Trim(VTReg(5))
         TxtYDiv = Trim(VTReg(6))
         TxtF.Text = Trim(VTReg(7))
         TxtF2.Text = Trim(VTReg(8))
         TxtF3.Text = Trim(VTReg(9))
         If Trim(TxtF.Text) = "" Or Trim(TxtF2.Text) = "" Or Trim(TxtF3.Text) = "" Then
            VLRecupera = True
         Else
            VLRecupera = False
         End If
         TxtGraficandoN.Text = Trim(VTReg(10))
         PLRecuperaDatos CInt(VTReg(10)), Trim(VTReg(14))
      End If
Exit Sub
err1:
   If err.Number <> 0 Then
      MsgBox "Se ha producido el siguiente error: " & err.Description & "." _
            & "Por favor consultar con el administrador", vbCritical
   End If
End Sub


Private Sub PLRecuperaDatos(parNume As Integer, parDatos As String)
Dim i, j, k As Integer
Dim VTCont As Integer
Dim VTPos As Integer
Dim VTValor As Integer
Dim VTLimite As Integer
```

```
Dim VTJacob As String
On Error GoTo error1
   VTCont = 1
   VTJacob = parDatos
      VTLimite = parNume * 2
      For k = 1 To VTLimite
         VLValoresMatriz(k) = 0
      Next k

      For k = 1 To VTLimite
         VTPos = InStr(1, parDatos, ";")
         VTValor = Mid(parDatos, 1, VTPos - 1)
         parDatos = Mid(parDatos, VTPos + 1)
         VLValoresMatriz(k) = VTValor
      Next k
      For i = 1 To parNume
         For j = 0 To 1
            Fg2.TextMatrix(i, j) = VLValoresMatriz(VTCont)
            VTCont = VTCont + 1
         Next j
      Next i

Exit Sub
error1:
   If err.Number <> 0 Then
      MsgBox "Se ha producido el siguiente error: " & err.Description & "." _
            & "Por favor consultar con el administrador", vbCritical
   End If
End Sub


Private Function PLCamposVacios() As Boolean
   If TxtXMin.Text = "" Or TxtXMax.Text = "" _
      Or TxtXDiv.Text = "" Or TxtYMin.Text = "" _
      Or TxtYMax.Text = "" Or TxtYDiv.Text = "" Then
         PLCamposVacios = True
   Else
         PLCamposVacios = False
   End If
End Function

Private Sub PLCambio()
   Me.BackColor = &HFFFFFF
   Me.BorderStyle = 0
   Fg2.BackColorBkg = &HFFFFFF
End Sub
```

```
Private Sub PLOriginal()
   Me.BackColor = &H8000000F
   Me.BorderStyle = 2
   Fg2.BackColorBkg = &H8000000F
End Sub



Private Sub Sincolor()
With Me
altoDatos = .Fg2.Height
colororg = .BackColor
.Fg2.Height = .Fg2.Height + 2000
.BackColor = &H80000009
.Label1.BackColor = &H80000009
.Label2.BackColor = &H80000009
.Label3.BackColor = &H80000009
.Label4.BackColor = &H80000009
.Label5.BackColor = &H80000009
.Label6.BackColor = &H80000009
.Label8.BackColor = &H80000009
.Label9.BackColor = &H80000009
.LblGraficandoabajo.BackColor = &H80000009
.LblGraficandoarriba.BackColor = &H80000009
.LblGraficandoderecha.BackColor = &H80000009
.LblGraficandof2.BackColor = &H80000009
.LblGraficandof1.BackColor = &H80000009
.LblGraficandoizquierda.BackColor = &H80000009
.LblGraficandoN.BackColor = &H80000009
.LblGraficandox.BackColor = &H80000009
.LblGraficandoy.BackColor = &H80000009
.txtEdit.BorderStyle = 0
.TxtF.BorderStyle = 0
.TxtF2.BorderStyle = 0
.TxtF3.BorderStyle = 0
.TxtGraficandoN.BorderStyle = 0
.TxtXDiv.BorderStyle = 0
.TxtXMax.BorderStyle = 0
.TxtXMin.BorderStyle = 0
.TxtYDiv.BorderStyle = 0
.TxtYMax.BorderStyle = 0
.TxtYMin.BorderStyle = 0
.Fg2.BackColorBkg = &H80000009
.Fg2.BackColorFixed = &H80000009
.Fg2.ScrollBars = 0
```

```
.CmdGraficandodesahacer.Visible = False
.Command1.Visible = False
.LblGraficandof3.BackColor = &H80000009
.CheckGraficandodatos.BackColor = &H80000009
.CheckGraficandofunción.BackColor = &H80000009
End With
End Sub

Private Sub Concolor()
With Me
 .Fg2.Height = altoDatos
 .BackColor = colororg

.Label1.BackColor = colororg
.Label2.BackColor = colororg
.Label3.BackColor = colororg
.Label4.BackColor = colororg
.Label5.BackColor = colororg
.Label6.BackColor = colororg
.Label8.BackColor = colororg
.Label9.BackColor = colororg
.LblGraficandoabajo.BackColor = colororg
.LblGraficandoarriba.BackColor = colororg
.LblGraficandoderecha.BackColor = colororg
.LblGraficandof2.BackColor = colororg
.LblGraficandof1.BackColor = colororg
.LblGraficandoizquierda.BackColor = colororg
.LblGraficandoN.BackColor = colororg
.LblGraficandox.BackColor = colororg
.LblGraficandoy.BackColor = colororg
.txtEdit.BorderStyle = 1
.TxtF.BorderStyle = 1
.TxtF2.BorderStyle = 1
.TxtF3.BorderStyle = 1
.TxtGraficandoN.BorderStyle = 1
.TxtXDiv.BorderStyle = 1
.TxtXMax.BorderStyle = 1
.TxtXMin.BorderStyle = 1
.TxtYDiv.BorderStyle = 1
.TxtYMax.BorderStyle = 1
.TxtYMin.BorderStyle = 1
.Fg2.BackColorBkg = colororg
.Fg2.BackColorFixed = colororg
.Fg2.ScrollBars = 3
.CmdGraficandodesahacer.Visible = True
.Command1.Visible = True
```

```
.LblGraficandof3.BackColor = colororg
.CheckGraficandodatos.BackColor = colororg
.CheckGraficandofunción.BackColor = colororg
End With
End Sub
```

Módulo de clase Graficar

```
Dim Flagerror As Boolean
Dim datos() As Double
Dim numdat As Integer
Dim tipográficof As Boolean
Dim tipográficod As Boolean
Dim Xmin, XLng, Ymax, YLng, xdiv, ydiv As Double
Dim XMind, XMaxd, YMaxd, YMind, XDivd, YDivd, XLngd, YLngd As Double
Dim Fd, Fd2, Fd3, fr1, fr2, fr3 As String
Dim X, Y, xa, Xmax, Ymin As Double
Dim f As String
Dim r, v, A As Integer
Dim nodibuja As Boolean
Dim Plano As Object


Public Property Let Seteargrafico(num As Integer, f1 As String, f2 As String, f3 As
String, Xmi As Double, Xmai As Double _
, Xdvi As Double, Ymi As Double, Ymai As Double, Ydvi As Double, tg As Boolean,
tgd As Boolean, dat() As Double)
On Error GoTo err1
numdat = num
fr1 = f1
fr2 = f2
fr3 = f3
Xmin = Xmi
Xmax = Xmai
xdiv = Xdvi
Ymin = Ymi
Ymax = Ymai
ydiv = Ydvi
XLng = Xmai - Xmi
YLng = Ymai - Ymi
tipográficof = tg
tipográficod = tgd

datos = dat

 XMind = Xmin
 XMaxd = Xmax
```

```
  XDivd = xdiv
  XLngd = XLng
  YMaxd = Ymax
  YMind = Ymin
  YDivd = ydiv
  YLngd = YLng
  Fd = fr1
  Fd2 = fr2
  Fd3 = fr3
Exit Property
err1:
End Property

Public Sub setarpicturebox()
On Error GoTo err1
Set Plano = FrmGráfico.PBGrafico
Exit Sub
err1:

End Sub

Public Sub Gráfico()
On Error GoTo err1
Set Plano = FrmGráfico.PBGrafico


Flagerror = False
PrepararPlano
If tipográficof = True Then
If (Frmescalas.TxtGraficof1 = "" Or Frmescalas.TxtGraficof1 = " ") Then
nodibuja = True
Else
nodibuja = False
End If
If nodibuja = False Then
f = fr1
r = 0
v = 0
A = 255
Graficar
End If
If Flagerror = True Then
Exit Sub
End If
If (Frmescalas.TxtGraficof2 = "" Or Frmescalas.TxtGraficof2 = " ") Then
nodibuja = True
```

```
Else
nodibuja = False
End If
If nodibuja = False Then
f = fr2
r = 0
v = 0
A = 0

Graficar
End If
If (Frmescalas.TxtGraficof3 = "" Or Frmescalas.TxtGraficof3 = " ") Then
nodibuja = True
Else
nodibuja = False
End If
If nodibuja = False Then

f = fr3
r = 0
v = 255
A = 0
Graficar
End If
End If
If tipográficod = True Then
Plano.DrawWidth = 2
Graficarpuntos
Plano.DrawWidth = 1
End If
Exit Sub
err1:
End Sub

Public Function ReemplazarF(X) As String
  Dim nx
  Dim NF
  Dim nL
  Dim Ns
  Dim nr

  NF = f
  nx = X
  nL = Len(nx)
  Ns = InStr(1, nx, ",")
  nr = nL - Ns
```

```
  If Ns <> 0 Then


  nx = Mid(nx, 1, InStr(1, nx, ",") - 1) & "." & Mid(nx, InStr(1, nx, ",") + 1, Len(Mid(nx, 1,
nr)))
  End If

  While InStr(1, NF, "X") > 0
    NF = Mid(NF, 1, InStr(1, NF, "X") - 1) & nx & Mid(NF, InStr(1, NF, "X") + 1,
Len(NF))
  Wend

  ReemplazarF = UCase(NF)
End Function

Private Sub PrepararPlano()
  On Error GoTo err1

  Plano.ScaleLeft = Xmin
  Plano.ScaleWidth = XLng
  Plano.ScaleTop = Ymax
  Plano.ScaleHeight = -YLng
  If id <> 0 Then
  Plano.Cls
  End If
  'Dibuja el eje X
  Plano.Line (Xmin, 0)-(Xmin + XLng, 0), vbRed
  For X = xdiv To XLng Step xdiv
    Plano.Line (X, -YLng / 100)-(X, YLng / 100), vbRed
    Plano.Line (-X, -YLng / 100)-(-X, YLng / 100), vbRed
  Next
  'Dibuja el eje Y
  Plano.Line (0, Ymax)-(0, Ymax - YLng), vbRed
  For Y = ydiv To YLng / ydiv Step ydiv
    Plano.Line (-XLng / 100, Y)-(XLng / 100, Y), vbRed
    Plano.Line (-XLng / 100, -Y)-(XLng / 100, -Y), vbRed
  Next
Exit Sub
err1:
End Sub
Private Sub Graficar()
  Dim xa, ya, Xu, Yu
  Dim mensaje As String
    On Error Resume Next 'ignorar errores
    Set appExcel = GetObject(, "Excel.Application") 'buscar una copia de Excel en
ejecución
```

```
    If err.Number <> 0 Then 'Si no se ejecuta Excel
       Set appExcel = CreateObject("Excel.Application") 'ejcutarlo
    End If
    err.Clear   ' Borrar el objeto Err si se produce un error.

    On Error GoTo 0 'Reaunudar el procesamiento normal de errores

    Set wbExcel = appExcel.Workbooks.Add
    Set shtExcel = wbExcel.Worksheets(1)
  'Dibuja la curva
  xa = Xmin
  'shtExcel.Application.Visible = True

  On Error Resume Next
  shtExcel.Range("A1").Formula = "= " & ReemplazarF(xa)
  Plano.PSet (X, Y), vbBlue
  ya = Cells(1, 1)


  For X = Xmin + 0.1 To Xmin + XLng Step 0.1
    shtExcel.Range("A1").Formula = "= " & ReemplazarF(X)
    Y = Cells(1, 1)
    Plano.Line (X, Y)-(xa, ya), RGB(r, v, A)
    xa = X: ya = Y
  Next
  err.Clear
  On Error GoTo 0

  wbExcel.Close False
  appExcel.Quit
  Set shtExcel = Nothing
  Set wbExcel = Nothing
  Set appExcel = Nothing

End Sub


Public Sub ampliar(Xmit As Double, Xmat As Double, Xdivt As Double, Ymit As
Double, Ymat As Double, Ydivt As Double _
, fa1 As String, fa2 As String, fa3 As String, tg As Boolean, tgd As Boolean, dat() As
Double, nds As Integer)
On Error GoTo err1
Xmin = Xmit
Xmax = Xmat
xdiv = Xdivt
Ymin = Ymit
```

```
Ymax = Ymat
ydiv = Ydivt
XLng = Xmat - Xmit
YLng = Ymat - Ymit
fr1 = fa1
fr2 = fa2
fr3 = fa3
tipográficof = tg
tipográficod = tgd
datos = dat
numdat = nds
Gráfico
Exit Sub
err1:

End Sub
Public Sub desahacer(Xmit As Double, Xmat As Double, Xdivt As Double, Ymit As
Double, Ymat As Double, Ydivt As Double _
, fa1 As String, fa2 As String, fa3 As String, tg As Boolean, tgd As Boolean, dat() As
Double, nds As Integer)
On Error GoTo err1
Xmin = Xmit
Xmax = Xmat
xdiv = Xdivt
Ymin = Ymit
Ymax = Ymat
ydiv = Ydivt
XLng = Xmat - Xmit
YLng = Ymat - Ymit
fr1 = fa1
fr2 = fa2
fr3 = fa3
tipográficof = tg
tipográficod = tgd
datos = dat
numdat = nds
Gráfico
Exit Sub
err1:
End Sub

Private Sub Graficarpuntos()
Dim i As Integer
Dim dx As Double
Dim dy As Double
On Error GoTo err1
```

```
For i = 0 To numdat - 1
dx = datos(i, 0)
dy = datos(i, 1)
Plano.PSet (dx, dy), RGB(125, 125, 255)
Next i
Exit Sub
err1:
End Sub
```

Interfaz Frmescalas

```
Option Explicit
Dim MG As New Graficar

Dim XR As Double
Dim Yr As Double
Dim XT As Double
Dim Yt As Double
Dim Xmtmp As Double
Dim Xmatmp As Double
Dim Xdivtmp As Double
Dim Ymtmp As Double
Dim Ymatmp As Double
Dim Ydivtmp As Double
Dim n As Integer
Dim tipoalg As Integer
Dim datosgrafico() As Double
Dim f1s As String
Dim f2s As String
Dim f3s As String

Private Acepta As Boolean
Private VLConexion As ADODB.Connection
Private VLRegistro As ADODB.Recordset
Private VLNombre As String
Private VLNombreTrabajo As String
Private VLVerifica As Boolean
Private VLValores As String
Private VLValoresMatriz(1 To 1000) As Integer
Private VLRecupera As Boolean

Property Let Aceptar(parAceptar As Boolean)
   Acepta = parAceptar
End Property

Property Let Nombre(parNombre As String)
   VLNombre = parNombre
```

```
End Property

Property Let NombreTrabajo(parNombreTrabajo As String)
   VLNombreTrabajo = parNombreTrabajo
End Property


Private Sub cdmSalir_Click()
   Unload Me
End Sub

Private Sub cmdAbrir_Click()
   FrmRecuperaTrabajo.VerGrafico = "Graficar"
   FrmRecuperaTrabajo.Show vbModal
   FrmRecuperaTrabajo.VerGrafico = ""
   If Acepta = True Then
      PLConexion True
      PLRecupera
   End If

End Sub

'Manejo de graficos

Private Sub CmdGraficograficar_Click()
Dim n As Integer
Dim i As Integer
Dim datosgrafico() As Double
Dim flagf As Boolean
Dim flagd As Boolean
Dim mensaje As String

On Error GoTo err1
n = Val(TxtGraficoN)

If Val(TxtGraficoXmax) <= Val(TxtGraficoXmin) Then
mensaje = MsgBox("Xmax debe ser mayor que Xmin", 48, "Error en el ingreso")
Exit Sub
End If
If Val(TxtGraficoYmax) <= Val(TxtGraficoYmin) Then
mensaje = MsgBox("Ymax debe ser mayor que Ymin", 48, "Error en el ingreso")
Exit Sub
End If


If CheckGraficodatos.Value = 1 Then
```

```
ReDim datosgrafico(0 To n - 1, 0 To 1) As Double
For i = 0 To n - 1
datosgrafico(i, 0) = Val(Fg2.TextMatrix(i + 1, 0))
datosgrafico(i, 1) = Val(Fg2.TextMatrix(i + 1, 1))
Next i
Else
n = 2
ReDim datosgrafico(0 To n - 1, 0 To 1) As Double
End If
If CheckGraficofunción.Value = 1 Then
flagf = True
End If
If CheckGraficodatos.Value = 1 Then
flagd = True
End If



Xmtmp = TxtGraficoXmin.Text
Xmatmp = TxtGraficoXmax.Text
Xdivtmp = Txt
Ymatmp = TxtGraficoYmax.Text
Ymtmp = TxtGraficoYmin.Text
Ydivtmp = TxtGraficoYdiv.Text
Enviargráfico
MG.setarpicturebox
FrmGráfico.seteografico(n, datosgrafico(), flagf, flagd, Xmtmp, Xmatmp, Xdivtmp,
Ymtmp, Ymatmp, Ydivtmp, f1s, f2s) = f3s
FrmGráfico.Show

MG.Gráfico
Exit Sub
err1:
If err.Number <> 0 Then
   mensaje = MsgBox("Parámetros del algoritmo incompletos", 48, "Error en el
ingreso")
   End If
End Sub


Private Sub CheckGraficofunción_Click()
' Si se seleciona función se presentan los controles necesarios
If CheckGraficofunción.Value = 1 Then
   LblGraficof1.Visible = True
   LblGraficof2.Visible = True
   LblGraficof3.Visible = True
```

```
    TxtGraficof1.Visible = True
    TxtGraficof2.Visible = True
    TxtGraficof3.Visible = True
    ShapeGraficof1.Visible = True
    ShapeGraficof2.Visible = True
    ShapeGraficof3.Visible = True
    If VLRecupera = True Then
    Else
       TxtGraficof1.Text = " "
       TxtGraficof2.Text = " "
       TxtGraficof3.Text = " "
    End If
    cmdAbrir.Enabled = True
    VLRecupera = False
 Else
    LblGraficof1.Visible = False
    LblGraficof2.Visible = False
    LblGraficof3.Visible = False
    TxtGraficof1.Visible = False
    TxtGraficof2.Visible = False
    TxtGraficof3.Visible = False
    ShapeGraficof1.Visible = False
    ShapeGraficof2.Visible = False
    ShapeGraficof3.Visible = False
    TxtGraficof1.Text = " "
    TxtGraficof2.Text = " "
    TxtGraficof3.Text = " "
    If CheckGraficodatos.Value = 1 Then
       cmdAbrir.Enabled = True
    Else
       cmdAbrir.Enabled = False
    End If
 End If

 End Sub

 'Si  elige datos se muestran los controles necesarios
 Private Sub CheckGraficodatos_Click()
 If CheckGraficodatos.Value = 1 Then
    LblGraficoN.Visible = True
    TxtGraficoN.Visible = True

    If VLRecupera = True Then
    Else
       TxtGraficoN.Text = ""
    End If
```

```
    cmdAbrir.Enabled = True
    VLRecupera = False
Else
n = 4
    LblGraficoN.Visible = False
    TxtGraficoN.Visible = False
    Fg2.Visible = False
    If CheckGraficofunción.Value = 1 Then
        cmdAbrir.Enabled = True
    Else
        cmdAbrir.Enabled = False
    End If
End If
End Sub

'Manejo de datos para graficar

Function Fgi(r As Integer, c As Integer) As Integer
  Fgi = c + Fg2.Cols * r
End Function

Private Sub cmdGuardar_Click()
If PLCamposVacios = True Then
    MsgBox "Llene los campos necesarios para guardar el trabajo", vbInformation,
"Campos vacios"
    Exit Sub
End If
    FrmNombre.Show vbModal
    If Acepta = True Then
        PLConexion True
        PLGuardarBDD
    End If

End Sub

Private Sub cmdNuevo_Click()
TxtGraficoXmin.Text = ""
TxtGraficoXmax.Text = ""
Txt.Text = ""
TxtGraficoYmin.Text = ""
TxtGraficoYmax.Text = ""
TxtGraficoYdiv.Text = ""
TxtGraficof1.Text = ""
TxtGraficof2.Text = ""
TxtGraficof3.Text = ""
TxtGraficoN.Text = ""
```

```
cmdGuardar.Enabled = True
CheckGraficofunción.Value = 0
CheckGraficodatos.Value = 0

End Sub

Sub Fg2_KeyPress(KeyAscii As Integer)
  MSHFlexGridEdit Fg2, TxtEdit, KeyAscii
End Sub

Sub Fg2_DblClick()
  MSHFlexGridEdit Fg2, TxtEdit, 32 ' Simula un espacio.
End Sub

Sub MSHFlexGridEdit(MSHFlexGrid As Control, _
Edt As Control, KeyAscii As Integer)

  ' Usar el carácter escrito.
  Select Case KeyAscii

  ' Un espacio significa modificar el texto actual.
  Case 0 To 32
    Edt = MSHFlexGrid
    Edt.SelStart = 1000

  ' Otro carácter reemplaza el texto actual.
  Case Else
    Edt = Chr(KeyAscii)
    Edt.SelStart = 1
  End Select

  ' Mostrar Edt en la posición correcta.
  Edt.Move MSHFlexGrid.Left + MSHFlexGrid.CellLeft, _
    MSHFlexGrid.Top + MSHFlexGrid.CellTop, _
    MSHFlexGrid.CellWidth - 8, _
    MSHFlexGrid.CellHeight - 8
  Edt.Visible = True

  ' Y hacer que funcione.
  Edt.SetFocus
End Sub



Private Sub Txt_KeyPress(KeyAscii As Integer)
'filtra la entrada de datos que no sean números
```

```
  If (KeyAscii < 48 Or KeyAscii > 57) Then
      If (KeyAscii = 45) Then
      Exit Sub
      End If
      If (KeyAscii <> 46) Then
         If (KeyAscii <> 8) Then
         KeyAscii = 0
         End If
      End If
   End If
  If KeyAscii = Asc(vbCr) Then ' Elimina los retornos para quitar los pitidos.
  KeyAscii = 0
  End If
End Sub

Sub txtEdit_KeyPress(KeyAscii As Integer)
 'filtra la entrada de datos que no sean números
  If (KeyAscii < 48 Or KeyAscii > 57) Then
      If (KeyAscii = 45) Then
      Exit Sub
      End If
      If (KeyAscii <> 46) Then
         If (KeyAscii <> 8) Then
         KeyAscii = 0
         End If
      End If
   End If
  If KeyAscii = Asc(vbCr) Then ' Elimina los retornos para quitar los pitidos.
  KeyAscii = 0
  End If
End Sub

Sub txtEdit_KeyDown(KeyCode As Integer, _
Shift As Integer)
  EditKeyCode Fg2, TxtEdit, KeyCode, Shift
End Sub

Sub EditKeyCode(MSHFlexGrid As Control, Edt As _
Control, KeyCode As Integer, Shift As Integer)

  ' Procesamiento del control de edición estándar.
  Select Case KeyCode

  Case 27   ' ESC: ocultar, devuelve el enfoque a          ' MSFlexGrid.
    Edt.Visible = False
    MSHFlexGrid.SetFocus
```

```
    Case 13   ' ENTRAR devuelve el enfoque a MSHFlexGrid.
      MSHFlexGrid.SetFocus

    Case 38     ' Arriba.
      MSHFlexGrid.SetFocus
      DoEvents
      If MSHFlexGrid.Row > MSHFlexGrid.FixedRows Then
        MSHFlexGrid.Row = MSHFlexGrid.Row - 1
      End If

    Case 40     ' Abajo.
      MSHFlexGrid.SetFocus
      DoEvents
      If MSHFlexGrid.Row < MSHFlexGrid.Rows - 1 Then
        MSHFlexGrid.Row = MSHFlexGrid.Row + 1
      End If
    End Select
End Sub

Sub Fg2_GotFocus()
    If TxtEdit.Visible = False Then Exit Sub
    Fg2 = TxtEdit
    TxtEdit.Visible = False
End Sub

Sub Fg2_LeaveCell()
    If TxtEdit.Visible = False Then Exit Sub
    Fg2 = TxtEdit
    TxtEdit.Visible = False
End Sub



Private Sub TxtGraficof1_Change()
LTrim (TxtGraficof1)
End Sub

Private Sub TxtGraficof2_Change()
LTrim (TxtGraficof2)
End Sub

Private Sub TxtGraficof3_Change()
LTrim (TxtGraficof3)
End Sub
```

```
Private Sub TxtGraficoN_Change()
n = Val(TxtGraficoN.Text)
If n = 0 Then
n = 1
End If
   Fg2.Cols = 2
   Fg2.Rows = n + 1
   Fg2.FixedCols = 0
   Fg2.FixedRows = 1
   Fg2.ColAlignmentFixed(0) = 4
   Fg2.ColAlignmentFixed(1) = 4
   Fg2.ColAlignment(0) = 4
   Fg2.ColAlignment(1) = 4
   Fg2.TextArray(Fgi(0, 0)) = "X"
   Fg2.TextArray(Fgi(0, 1)) = "Y"
   Fg2.ColWidth(0) = 800
   Fg2.ColWidth(1) = 800
   Fg2.Visible = True
End Sub
'Envia los parámetros de graficación  al módulo de graficación
Private Sub Enviargráfico()
Dim i As Integer
Dim j As Integer
Dim xm As Double
Dim Xma As Double
Dim Xdv As Double
Dim ym As Double
Dim Yma As Double
Dim Ydv As Double
Dim flagf As Boolean
Dim flagd As Boolean

On Error GoTo err1
n = Val(TxtGraficoN)

If CheckGraficodatos.Value = 1 Then
ReDim datosgrafico(0 To n - 1, 0 To 1) As Double
For i = 0 To n - 1
datosgrafico(i, 0) = Val(Fg2.TextMatrix(i + 1, 0))
datosgrafico(i, 1) = Val(Fg2.TextMatrix(i + 1, 1))
Next i
Else
n = 2
ReDim datosgrafico(0 To n - 1, 0 To 1) As Double
End If
If CheckGraficofunción.Value = 1 Then
```

```
flagf = True
End If
If CheckGraficodatos.Value = 1 Then
flagd = True
End If

Xdv = Val(Txt)
xm = Val(TxtGraficoXmin)
Xma = Val(TxtGraficoXmax)
ym = Val(TxtGraficoYmin)
Yma = Val(TxtGraficoYmax)
Ydv = Val(TxtGraficoYdiv)
f1s = TxtGraficof1
f2s = TxtGraficof2
f3s = TxtGraficof3
MG.Seteargrafico(n, f1s, f2s, f3s, xm, Xma, Xdv, ym, Yma, Ydv, flagf, flagd) =
datosgrafico
Exit Sub
err1:
End Sub


Private Sub TxtGraficoN_KeyPress(KeyAscii As Integer)
'filtra la entrada de datos que no sean números enteros y positivos
  If (KeyAscii < 48 Or KeyAscii > 57) Then

        If (KeyAscii <> 8) Then
        KeyAscii = 0
        End If

   End If
  If KeyAscii = Asc(vbCr) Then ' Elimina los retornos para quitar los pitidos.
  KeyAscii = 0
  End If

End Sub

Private Sub TxtGraficoXmax_KeyPress(KeyAscii As Integer)
'filtra la entrada de datos que no sean números
  If (KeyAscii < 48 Or KeyAscii > 57) Then
      If (KeyAscii = 45) Then
      Exit Sub
      End If
      If (KeyAscii <> 46) Then
         If (KeyAscii <> 8) Then
         KeyAscii = 0
```

```
        End If
      End If
    End If
   If KeyAscii = Asc(vbCr) Then ' Elimina los retornos para quitar los pitidos.
   KeyAscii = 0
   End If
End Sub

Private Sub TxtGraficoXmin_KeyPress(KeyAscii As Integer)
'filtra la entrada de datos que no sean números
   If (KeyAscii < 48 Or KeyAscii > 57) Then
       If (KeyAscii = 45) Then
       Exit Sub
       End If
       If (KeyAscii <> 46) Then
          If (KeyAscii <> 8) Then
          KeyAscii = 0
          End If
       End If
    End If
   If KeyAscii = Asc(vbCr) Then ' Elimina los retornos para quitar los pitidos.
   KeyAscii = 0
   End If
End Sub

Private Sub TxtGraficoYdiv_KeyPress(KeyAscii As Integer)
'filtra la entrada de datos que no sean números
   If (KeyAscii < 48 Or KeyAscii > 57) Then
       If (KeyAscii = 45) Then
       Exit Sub
       End If
       If (KeyAscii <> 46) Then
          If (KeyAscii <> 8) Then
          KeyAscii = 0
          End If
       End If
    End If
   If KeyAscii = Asc(vbCr) Then ' Elimina los retornos para quitar los pitidos.
   KeyAscii = 0
```

```
   End If
End Sub

Private Sub TxtGraficoYmax_KeyPress(KeyAscii As Integer)
'filtra la entrada de datos que no sean números
   If (KeyAscii < 48 Or KeyAscii > 57) Then
       If (KeyAscii = 45) Then
       Exit Sub
       End If
       If (KeyAscii <> 46) Then
          If (KeyAscii <> 8) Then
          KeyAscii = 0
          End If
       End If
    End If
   If KeyAscii = Asc(vbCr) Then ' Elimina los retornos para quitar los pitidos.
   KeyAscii = 0
   End If
End Sub

Private Sub TxtGraficoYmin_KeyPress(KeyAscii As Integer)
'filtra la entrada de datos que no sean números
   If (KeyAscii < 48 Or KeyAscii > 57) Then
       If (KeyAscii = 45) Then
       Exit Sub
       End If
       If (KeyAscii <> 46) Then
          If (KeyAscii <> 8) Then
          KeyAscii = 0
          End If
       End If
    End If
   If KeyAscii = Asc(vbCr) Then ' Elimina los retornos para quitar los pitidos.
   KeyAscii = 0
   End If
End Sub

'****** Conexion a la BDD
Private Sub PLConexion(parBDD As Boolean)
Dim VTCad As String
Dim VTBase As Database
Dim VTcadena As String
   VTcadena = App.Path & "/Seguridad.mdb"
   Set VLConexion = New ADODB.Connection
   VTCad = "Provider=Microsoft.Jet.OLEDB.4.0;Data Source='" & VTcadena &
"';Persist Security Info=False"
```

```
   VLConexion.Open VTCad
End Sub

Private Sub PLGuardarBDD()
   Dim VTCad As String
   Dim VTReg As ADODB.Recordset
   Dim VTID As Integer
   Dim VB As Integer
   Dim VTNumero As String
On Error GoTo error1
   PLVerificaNombre Trim(VLNombre)
      If VLVerifica = True Then   'Si existe un nombre con el mismo nombre actualiza
         VB = MsgBox("El nombre de ese trabajo de graficación ya existe en este
tema o en otro tema, desea reemplazarlo?", vbYesNo, "Guardar Trabajo Como")
         If VB = 6 Then
            If Trim(TxtGraficoN.Text) = "" Then
               VLValores = ""
            Else
               PLGuardaDatos CInt(TxtGraficoN.Text)
            End If
               VTCad = "update Graficacion set " _
                     & " Xmin= '" & Trim(TxtGraficoXmin.Text) & "', " _
                     & " Xmax='" & TxtGraficoXmax.Text & "', " _
                     & " Xdiv='" & Txt.Text & "', " _
                     & " Ymin= '" & Trim(TxtGraficoYmin.Text) & "', " _
                     & " Ymax='" & TxtGraficoYmax.Text & "', " _
                     & " Ydiv='" & TxtGraficoYdiv.Text & "', " _
                     & " F1= '" & Trim(TxtGraficof1.Text) & "', " _
                     & " F2='" & TxtGraficof2.Text & "', " _
                     & " F3='" & TxtGraficof3.Text & "', " _
                     & " Numi='" & TxtGraficoN.Text & "', " _
                     & " Datos='" & Trim(VLValores) & "' " _
                     & " where Nombre = '" & VLNombre & "'"
            VLConexion.Execute (VTCad)
            cmdNuevo_Click
            MsgBox "Trabajo reemplazado correctamente", vbInformation, "Trabajo
reemplazado"
         End If
      Else   'Guarda el trabajo
         VTID = PLRecuperaCodigo
         If Trim(TxtGraficoN.Text) = "" Then
            VLValores = ""
         Else
            PLGuardaDatos CInt(TxtGraficoN.Text)
         End If
         If Trim(TxtGraficoN.Text) = "" Then
```

```
            TxtGraficoN.Text = 0
        End If
        VTCad = "insert into Graficacion values " _
            & "(" & CInt(VTID) & ", " _
            & " '" & TxtGraficoXmin.Text & "', '" & TxtGraficoXmax.Text & "', " _
            & " '" & Txt.Text & "', '" & TxtGraficoYmin.Text & "', " _
            & " '" & TxtGraficoYmax.Text & "', '" & TxtGraficoYdiv.Text & "', " _
            & " '" & TxtGraficof1.Text & "', '" & TxtGraficof2.Text & "', " _
            & " '" & TxtGraficof3.Text & "', '" & TxtGraficoN.Text & "', " _
            & " '" & Trim(VLNombre) & "', '" & VGForma & "', " _
            & " " & VGIDUsuario & ", '" & VLValores & "')"
            '& " '" & Trim(VLNombre) & "', " & CInt(tipoalg) & ", " & CInt(VGIDUsuario)
& ")"
        VLConexion.Execute (VTCad)
        MsgBox "Trabajo guardado exitosamente", vbInformation, "Guardar trabajo
como"
    End If

Exit Sub
error1:
  If err.Number <> 0 Then
    MsgBox "Se ha producido el siguiente error: " & err.Description & "." _
        & "Por favor consultar con el administrador", vbCritical
  End If
End Sub


Private Sub PLVerificaNombre(parNombre As String)
Dim VTReg As ADODB.Recordset
Dim VTCad As String
Dim VTTabla As String
On Error GoTo error1
    VTCad = "select Nombre from Graficacion where Nombre = '" & parNombre & "'"
      Set VTReg = VLConexion.Execute(VTCad)
      With VTReg
        If Not VTReg.EOF Then
          .MoveFirst
          Do Until .EOF
            If Trim(VTReg(0)) = parNombre Then
              VLVerifica = True
              Exit Do
            Else
              VLVerifica = False
            End If
            .MoveNext
          Loop
```

```
            Else
               VLVerifica = False
            End If
         End With
Exit Sub
error1:

End Sub
Private Function PLRecuperaCodigo() As Integer
   Dim VTCad As String
   Dim VTReg As ADODB.Recordset
   Dim VTTabla As String
On Error GoTo err1
      VTCad = "select max(IDGraficacion) from Graficacion"
      Set VTReg = VLConexion.Execute(VTCad)
      If Not IsNull(VTReg(0)) Then
         PLRecuperaCodigo = VTReg(0) + 1
      Else
         PLRecuperaCodigo = 1
      End If
Exit Function
err1:

End Function

Private Sub PLGuardaDatos(parNume As String)
Dim i, j As Integer
Dim Nume As Integer
On Error GoTo error1
   VLValores = ""
   Nume = CInt(parNume)
      For i = 1 To Nume
         For j = 0 To 1
            VLValores = VLValores & Fg2.TextMatrix(i, j) & ";"
         Next j
      Next i
Exit Sub
error1:
   If err.Number <> 0 Then
      MsgBox "Se ha producido el siguiente error: " & err.Description & "." _
            & "Por favor consultar con el administrador", vbCritical
   End If
End Sub

Private Function PLCamposVacios() As Boolean
   If TxtGraficoXmin.Text = "" Or TxtGraficoXmax.Text = "" _
```

```
      Or Txt.Text = "" Or TxtGraficoYmin.Text = "" _
      Or TxtGraficoYmax.Text = "" Or TxtGraficoYdiv.Text = "" Then
      PLCamposVacios = True
   Else
      PLCamposVacios = False
   End If
End Function

Private Sub PLRecupera()
   Dim VTCad As String
   Dim VTReg As ADODB.Recordset
On Error GoTo err1

      VTCad = "select * from Graficacion where Nombre = '" & VLNombreTrabajo & "'
and Tema = '" & VGForma & "'"
      Set VTReg = VLConexion.Execute(VTCad)
      If Not VTReg.EOF Then

         TxtGraficoXmin.Text = Trim(VTReg(1))
         TxtGraficoXmax.Text = Trim(VTReg(2))
         Txt.Text = Trim(VTReg(3))
         TxtGraficoYmin.Text = Trim(VTReg(4))
         TxtGraficoYmax.Text = Trim(VTReg(5))
         TxtGraficoYdiv.Text = Trim(VTReg(6))
         TxtGraficof1.Text = Trim(VTReg(7))
         TxtGraficof2.Text = Trim(VTReg(8))
         TxtGraficof3.Text = Trim(VTReg(9))
         If Trim(TxtGraficof1.Text) = "" Or Trim(TxtGraficof2.Text) = "" Or
Trim(TxtGraficof3.Text) = "" Then
            VLRecupera = True
         Else
            VLRecupera = False
         End If
         TxtGraficoN.Text = Trim(VTReg(10))
         PLRecuperaDatos CInt(VTReg(10)), Trim(VTReg(14))
      End If
Exit Sub
err1:
   If err.Number <> 0 Then
      MsgBox "Se ha producido el siguiente error: " & err.Description & "." _
            & "Por favor consultar con el administrador", vbCritical
   End If
End Sub


Private Sub PLRecuperaDatos(parNume As Integer, parDatos As String)
```

```
Dim i, j, k As Integer
Dim VTCont As Integer
Dim VTPos As Integer
Dim VTValor As Integer
Dim VTLimite As Integer
Dim VTJacob As String
On Error GoTo error1
   VTCont = 1
   VTJacob = parDatos
      VTLimite = parNume * 2
      For k = 1 To VTLimite
         VLValoresMatriz(k) = 0
      Next k

      For k = 1 To VTLimite
         VTPos = InStr(1, parDatos, ";")
         VTValor = Mid(parDatos, 1, VTPos - 1)
         parDatos = Mid(parDatos, VTPos + 1)
         VLValoresMatriz(k) = VTValor
      Next k
      For i = 1 To parNume
         For j = 0 To 1
            Fg2.TextMatrix(i, j) = VLValoresMatriz(VTCont)
            VTCont = VTCont + 1
         Next j
      Next i

Exit Sub
error1:
   If err.Number <> 0 Then
      MsgBox "Se ha producido el siguiente error: " & err.Description & "." _
            & "Por favor consultar con el administrador", vbCritical
   End If
End Sub
```

Interfaz Gráfico

```
Option Explicit
Dim MG As New Graficar
Dim Xming As Double
Dim Xmaxg As Double
Dim Xdivg As Double
Dim Yming As Double
Dim Ymaxg As Double
Dim Ydivg As Double
Dim f1 As String
Dim f2 As String
```

```
Dim f3 As String
Dim Xmtmp As Double
Dim Xmatmp As Double
Dim Xdivtmp As Double
Dim Ymtmp As Double
Dim Ymatmp As Double
Dim Ydivtmp As Double
Dim numd As Integer
Dim datosg() As Double
Dim tipof As Boolean
Dim tipod As Boolean
Dim XT As Double
Dim colororg
Dim Yt As Double
Public Property Let seteografico(nd As Integer, dats() As Double, tf As Boolean, td As
Boolean, xm As Double, xa As Double, xdiv As Double, ym As Double, ya As Double
_
, ydiv As Double, fx1 As String, fx2 As String, fx3 As String)
On Error GoTo err1
Xming = xm
Xmaxg = xa
Xdivg = xdiv
Yming = ym
Ymaxg = ya
Ydivg = ydiv
f1 = fx1
f2 = fx2
f3 = fx3
Xmtmp = Xming
Xmatmp = Xmaxg
Xdivtmp = Xdivg
Ymtmp = Yming
Ymatmp = Ymaxg
Ydivtmp = Ydivg
numd = nd
datosg = dats
tipof = tf
tipod = td

PBGrafico.BackColor = RGB(255, 255, 255)

LblGraficof1.Caption = f1
LblGraficof2.Caption = f2
LblGraficof3.Caption = f3
LblGraficoizquierda.Caption = Xming
LblGraficoderecha.Caption = Xmaxg
```

```
LblGraficoarriba.Caption = Ymaxg
LblGraficoabajo.Caption = Yming
Exit Property
err1:

End Property

'Manejo de graficos
Public Sub titulos()


End Sub

Private Sub CmdGraficoundo_Click()
On Error GoTo err1
LblGraficoizquierda.Caption = Xmtmp
LblGraficoderecha.Caption = Xmatmp
LblGraficoarriba.Caption = Ymatmp
LblGraficoabajo.Caption = Ymtmp
PBGrafico.Cls
MG.desahacer Xmtmp, Xmatmp, Xdivtmp, Ymtmp, Ymatmp, Ydivtmp, f1, f2, f3, tipof,
tipod, datosg(), numd
Exit Sub
err1:
End Sub

Private Sub Command1_Click()
   Unload Me
   Frmescalas.Show
End Sub




Private Sub Opimprimir_Click()
 Dim EndTime As Date
  Form1.SetearVariable = 2
   Sincolor

      EndTime = DateAdd("s", 1 / 1E+16, Now)
      Do Until Now > EndTime
        DoEvents
      Loop

   Set Form1.Picture1.Picture = CaptureClient(Me)
```

```
    Form1.Visible = True
    FrmPrincipalalgoritmos.Visible = False
    Frmescalas.Visible = False
    FrmGráfico.Visible = False
    Concolor
End Sub

Private Sub Opsalir_Click()
Unload Me
End Sub

'Manejo de ampliación del gráfico
Private Sub PBGrafico_MouseDown(Button As Integer, Shift As Integer, X As Single,
Y As Single)
If Button = vbLeftButton Then
 PBGrafico.MousePointer = 2
XT = X
Yt = Y
End If
End Sub

Private Sub PBGrafico_MouseMove(Button As Integer, Shift As Integer, X As Single,
Y As Single)

LblGraficox.Caption = "Coordenadas(" & X
LblGraficoy.Caption = Y & ")"


End Sub

Private Sub PBGrafico_MouseUp(Button As Integer, Shift As Integer, X As Single, Y
As Single)
Dim XR As Double
Dim Yr As Double
On Error GoTo err1
If Button = vbLeftButton Then
  If PBGrafico.MousePointer = 2 Then
  PBGrafico.MousePointer = 0
  End If

LblGraficoizquierda.Caption = XT
LblGraficoderecha.Caption = X
LblGraficoarriba.Caption = Yt
LblGraficoabajo.Caption = Y

XR = X
```

```
Yr = Y
PBGrafico.Cls
MG.ampliar XT, XR, Xdivtmp, Yr, Yt, Ydivtmp, f1, f2, f3, tipof, tipod, datosg(), numd

End If
Exit Sub
err1:

End Sub



Private Sub Sincolor()
With Me
colororg = .BackColor
.BackColor = &H80000009
.LblGraficoabajo.BackColor = &H80000009
.LblGraficoarriba.BackColor = &H80000009
.LblGraficoderecha.BackColor = &H80000009
.LblGraficof1.BackColor = &H80000009
.LblGraficof2.BackColor = &H80000009
.LblGraficof3.BackColor = &H80000009
.LblGraficoizquierda.BackColor = &H80000009
.LblGraficox.BackColor = &H80000009
.LblGraficoy.BackColor = &H80000009
.Cmdgraficoundo.Visible = False
End With
End Sub

Private Sub Concolor()
 With Me
 .BackColor = colororg

.LblGraficoabajo.BackColor = colororg
.LblGraficoarriba.BackColor = colororg
.LblGraficoderecha.BackColor = colororg
.LblGraficof1.BackColor = colororg
.LblGraficof2.BackColor = colororg
.LblGraficof3.BackColor = colororg
.LblGraficoizquierda.BackColor = colororg
.LblGraficox.BackColor = colororg
.LblGraficoy.BackColor = colororg
.Cmdgraficoundo.Visible = True
End With
```

```
End Sub
```

Módulo de clase MscExcel

```
Private tipo As Boolean

Private ye, XE, f


Public Property Get Fdy() As Double
On Error GoTo err1
Fdy = ye
Exit Property
err1:

If err.Number <> 0 Then
   'mensaje = MsgBox("Parámetros del algoritmo incompletos", 48, "Error en el
ingreso")
   Fdy = 8121975
   End If
End Property

Public Property Let PtX(ByVal NXE As Double)
XE = NXE
End Property

Public Property Let fa(ByVal NF As Variant)
f = NF
End Property
Public Property Let Tipox(ByVal xvariable As Boolean)
tipo = xvariable
End Property
Public Sub evaluarf()

Dim appExcel As Excel.Application
Dim wbExcel As Excel.Workbook
Dim shtExcel As Excel.Worksheet


   On Error Resume Next 'ignorar errores
   Set appExcel = GetObject(, "Excel.Application") 'buscar una copia de Excel en
ejecución
   If err.Number <> 0 Then 'Si no se ejecuta Excel
      Set appExcel = CreateObject("Excel.Application") 'ejcutarlo
   End If
   err.Clear   ' Borrar el objeto Err si se produce un error.
```

```
    On Error GoTo 0 'Reaunudar el procesamiento normal de errores

    Set wbExcel = appExcel.Workbooks.Add
    Set shtExcel = wbExcel.Worksheets(1)
  'shtExcel.Application.Visible = True
  On Error Resume Next
  'Seleccionamos el tipo de evaluación a realizar si es solo una variable o
  'varias variables
  If tipo = True Then
  shtExcel.Range("A1").Formula = "= " & f
  Else
  shtExcel.Range("A1").Formula = "= " & ReemplazarF(XE)
  End If
  ye = Cells(1, 1)

  err.Clear
  On Error GoTo 0

  wbExcel.Close False
  appExcel.Quit
  Set shtExcel = Nothing
  Set wbExcel = Nothing
  Set appExcel = Nothing
End Sub
Private Function ReemplazarF(X) As String
  Dim nx
  Dim NF
  Dim nL
  Dim Ns
  Dim nr

  NF = f
  nx = X
  nL = Len(nx)
  Ns = InStr(1, nx, ",")
  nr = nL - Ns
  If Ns <> 0 Then


  nx = Mid(nx, 1, InStr(1, nx, ",") - 1) & "." & Mid(nx, InStr(1, nx, ",") + 1, Len(Mid(nx, 1,
nr)))
  End If

  While InStr(1, NF, "X") > 0
    NF = Mid(NF, 1, InStr(1, NF, "X") - 1) & nx & Mid(NF, InStr(1, NF, "X") + 1,
Len(NF))
```

```
  Wend

  ReemplazarF = UCase(NF)
End Function
```

CODIGO DEL CLUSTER (3)
Realizar evaluación

Interfaz FrmEvaluación

```
Option Explicit
Dim id As Form
Dim informe As Form
Dim contestadas() As Integer
Dim cadenaconsulta As String
Dim Tipo_de_Evaluación As Boolean 'Si es true es parcial
Dim respuesta As Integer 'respuesta elgida por el usuario
Dim numpregunta As Integer 'número de pregunta elegida por el ususario
Dim Resultado As Integer
Dim respuestacorrecta As Integer
Dim mensaje As String
Dim pruebatipo As Long
Dim alarma
Dim Inicio As Boolean
Dim Tema_a_ser_evaluado As String

Private Sub Command1_Click()
'Sse guarda que pregunta se ha contestado, la respuesta elegida, y la correcta
contestadas(numpregunta, 1) = 1
contestadas(numpregunta, 3) = respuestacorrecta
If Opt1.Value = True Then
contestadas(numpregunta, 2) = 1
Else
   If Opt2.Value = True Then
   contestadas(numpregunta, 2) = 2
   Else
      If Opt3.Value = True Then
      contestadas(numpregunta, 2) = 3
      Else
         If Opt4.Value = True Then
         contestadas(numpregunta, 2) = 4
         Else
         mensaje = MsgBox("Debe contestar la pregunta antes de ingresar la
respuesta", 16, "Error de Ingreso")
         contestadas(numpregunta, 2) = 0
         End If
```

```
        End If
    End If
End If
End Sub

Private Sub Command2_Click()
Dim i As Integer
'Verificamos si todas las preguntas estan contestadas
If Tipo_de_Evaluación = True Then
    For i = 1 To 20
    If contestadas(i, 1) = 0 Then
    mensaje = MsgBox("No todas las preguntas han sido contestadas", 16, "Error de
Evaluación")
    Exit Sub
    End If
    Next i
Else
    For i = 1 To 30
    If contestadas(i, 1) = 0 Then
    mensaje = MsgBox("No todas las preguntas han sido contestadas", 16, "Error de
Evaluación")
    Exit Sub
    End If
    Next i
End If
'Realizamos la evaluación
Resultado = 0
If Tipo_de_Evaluación = True Then
    For i = 1 To 20
        If contestadas(i, 2) = contestadas(i, 3) Then
        Resultado = Resultado + 1
        End If
    Next i
Else
    For i = 1 To 30
        If contestadas(i, 2) = contestadas(i, 3) Then
        Resultado = Resultado + 1
        End If
    Next i
End If
Set id = FrmSeguridad
Set informe = FrmInformeevaluación
informe.Show

If Tipo_de_Evaluación = True Then
informe.LblTipo = LblTitulo3.Caption
```

```
informe.LblTema = LblTitulo4.Caption
informe.LblNombre = id.Nombre
informe.LblNumero = Resultado
informe.lblprocentaje = (Resultado * 70) / 100
informe.lblprocentaje = Format(informe.lblprocentaje, "###.##")
   If Resultado > 13 Then
   informe.LblAprobación = "ACEPTADA"
   informe.lblRecomendación = " Felicitaciones usted ha alcanzado comprensión
corecta del Tema, continue así"
   Else
   informe.LblAprobación = "DENEGADA"
   informe.lblRecomendación = "Su nivel de conocimiento no es suficiente repase
nuevamente el Tema poniendo enfasís en los conceptos"
   End If
Else
informe.LblTipo = LblTitulo3.Caption
informe.LblTema = " "
informe.LblNumero = Resultado
informe.lblprocentaje = (Resultado * 70) / 100
informe.lblprocentaje = Format(informe.lblprocentaje, "###.##")
   If Resultado > 13 Then
   informe.LblAprobación = "ACEPTADA"
   informe.lblRecomendación = " Felicitaciones usted ha alcanzado comprensión
corecta del Tema, continue así"
   Else
   informe.LblAprobación = "DENEGADA"
   informe.lblRecomendación = "Su nivel de conocimiento no es suficiente repase
nuevamente el Tema poniendo enfasís en los conceptos"
   End If

End If
Eparcial.Enabled = False
Etotal.Enabled = False
LblTema.Visible = False
LblTitulo.Visible = False
LblNumero.Visible = False
LblTipo.Visible = False
Lbltitulo2.Visible = False
LblTitulo3.Visible = False
LblTitulo4.Visible = False
Fraopciones.Visible = False
Opt1.Visible = False
Opt2.Visible = False
Opt3.Visible = False
Opt4.Visible = False
FgEnunciados.Visible = False
```

```
FgTabla.Visible = False
LblTitulo5.Visible = False
DlstTema.Visible = False
DlstNumero.Visible = False
DlstTipo.Visible = False
Command1.Visible = False
Command2.Visible = False
LblTime.Visible = False
End Sub

' Se accede a la base de datos para conseguir el tipo de pregunta
Private Sub DlstTema_Click()
LblTitulo4.Caption = DlstTema.Text
DlstTema.Visible = False
cadenaconsulta = "Select * From Tipodepregunta Where IDTema =" & "'" & _
DlstTema.BoundText & "' "
AdoTipoPregunta.RecordSource = cadenaconsulta
AdoTipoPregunta.Refresh
DlstTipo.ListField = "Tipo de Pregunta"
DlstTipo.BoundColumn = "IDTipo"
DlstTipo.Visible = True

End Sub
' Se accede a la base de datos para conseguir el número de la pregunta
Private Sub DlstTipo_Click()

If Tipo_de_Evaluación = True Then
LblNumero.Visible = True
cadenaconsulta = "Select * From Preguntas Where IDTipo =" & "'" & _
DlstTipo.BoundText & "' "
pruebatipo = InStr(1, cadenaconsulta, "v", 1)

AdoNumpregunta.RecordSource = cadenaconsulta
AdoNumpregunta.Refresh
DlstNumero.ListField = "Número de Pregunta"
DlstNumero.BoundColumn = "IDPregunta"
DlstNumero.Visible = True
Else
LblNumero.Visible = True
cadenaconsulta = "Select * From Totalpreguntas Where IDTipo =" & "'" & _
DlstTipo.BoundText & "' "
pruebatipo = InStr(1, cadenaconsulta, "v", 1)

AdoNumpregunta.RecordSource = cadenaconsulta
AdoNumpregunta.Refresh
DlstNumero.ListField = "Número de Pregunta"
```

```
DlstNumero.BoundColumn = "IDPregunta"
DlstNumero.Visible = True
End If
LblTime.Visible = True
End Sub
' Se accede a la base de datos para conseguir el enunciado  de la pregunta y las
respuestas
Private Sub DlstNumero_Click()
If Tipo_de_Evaluación = True Then
cadenaconsulta = "Select Enunciados From Alternativas Where IDPregunta =" & "'" &
_
DlstNumero.BoundText & "' "
AdoEnunciados.RecordSource = cadenaconsulta
AdoEnunciados.Refresh
Fraopciones.Visible = True

If pruebatipo <> 0 Then
Opt3.Visible = False
Opt4.Visible = False
Else
Opt3.Visible = True
Opt4.Visible = True
End If
Opt1.Visible = True
Opt2.Visible = True
'Opt3.Visible = False
'Opt4.Visible = False
FgEnunciados.Visible = True
cadenaconsulta = "Select X,Y From Paresdedatos Where IDPregunta =" & "'" & _
DlstNumero.BoundText & "' "
AdoTabla.RecordSource = cadenaconsulta
AdoTabla.Refresh
LblTitulo5.Visible = True
FgTabla.Visible = True
cadenaconsulta = "Select Respuesta From Preguntas Where IDPregunta =" & "'" & _
DlstNumero.BoundText & "' "
AdoRespuestas.RecordSource = cadenaconsulta
AdoRespuestas.Refresh
respuestacorrecta = Val(lblRespuestas.Caption)
numpregunta = Val(DlstNumero.Text)
Else
cadenaconsulta = "Select Enunciado From Totalalternativas Where IDPregunta =" &
"'" & _
DlstNumero.BoundText & "' "
AdoEnunciados.RecordSource = cadenaconsulta
AdoEnunciados.Refresh
```

```
Fraopciones.Visible = True
If pruebatipo <> 0 Then
Opt3.Visible = False
Opt4.Visible = False
Else
Opt3.Visible = True
Opt4.Visible = True
End If
Opt1.Visible = True
Opt2.Visible = True
'Opt3.Visible = False
'Opt4.Visible = False
FgEnunciados.Visible = True
cadenaconsulta = "Select X,Y From Totalparesdedatos Where IDPregunta =" & "'" &
_
DlstNumero.BoundText & "' "
AdoTabla.RecordSource = cadenaconsulta
AdoTabla.Refresh
LblTitulo5.Visible = True
FgTabla.Visible = True
cadenaconsulta = "Select Respuesta From Totalpreguntas Where IDPregunta =" & "'"
& _
DlstNumero.BoundText & "' "
AdoRespuestas.RecordSource = cadenaconsulta
AdoRespuestas.Refresh
respuestacorrecta = Val(lblRespuestas.Caption)
numpregunta = Val(DlstNumero.Text)
End If
Command1.Visible = True
Command2.Visible = True
End Sub

'seleccion del tipo de evaluación y presentación de controles

Private Sub Eparcial_Click()
Tipo_de_Evaluación = True
Eparcial.Enabled = False
Etotal.Enabled = False
LblTitulo.Visible = True
Lbltitulo2.Visible = True
LblTitulo3.Caption = "Parcial"
LblTitulo3.Visible = True
LblTitulo4.Visible = True
LblTema.Visible = True
DlstTema.Visible = True
ReDim contestadas(1 To 20, 1 To 3) As Integer
```

```
alarma = DateAdd("h", 1, Time)
Inicio = True
End Sub

Private Sub Etotal_Click()
Tipo_de_Evaluación = False
Eparcial.Enabled = False
Etotal.Enabled = False
LblTitulo.Visible = True
Lbltitulo2.Visible = False
LblTitulo3.Caption = "Total"
LblTitulo3.Visible = True
LblTitulo4.Visible = False
cadenaconsulta = "Select * From Totaltipodepregunta;"
AdoTipoPregunta.RecordSource = cadenaconsulta
AdoTipoPregunta.Refresh
DlstTipo.ListField = "Tipo de Pregunta"
DlstTipo.BoundColumn = "IDTipo"
DlstTipo.Visible = True
ReDim contestadas(1 To 30, 1 To 3) As Integer
alarma = DateAdd("h", 1, Time)
alarma = alarma + DateAdd("n", 30, Time)
Inicio = True
End Sub

Private Sub Form_QueryUnload(Cancel As Integer, UnloadMode As Integer)
Frmmodulos.Visible = True
End Sub

Private Sub salir_Click()
Frmmodulos.Visible = True
Unload Me
End Sub

'Manejo del tiempo de la evaluación
'Reloj
Private Sub Timer1_Timer()
Static bander As Boolean
If Inicio = True Then

   If LblTime.Caption <> CStr(Time) Then
     LblTime.Caption = Format(Time, "Medium Time")
   End If
   If bander = True Then
      alarma = DateAdd("n", 1, Time)
      bander = False
```

```
    Else
      If Time > alarma Then
        MsgBox "Tiempo de la evaluación cumplido complete las preguntas "
        bander = True
      End If
    End If
End If
End Sub

Private Sub Form_Load()
Inicio = False

FgEnunciados.ColWidth(0) = 10300
FgEnunciados.RowHeight(-1) = 350

End Sub
```

CODIGO DEL CLUSTER (4)
Encontrar raíces de un polinómio

Interfaz FrmMP

```
Option Explicit
'En Microsoft TechNet puedes encontrar este artículo:
'HOWTO: Use HTML Help API in a Visual Basic 5.0 Application
'PSS ID Number: Q183434
'
'Aunque la definición de la Enumeración y la primera declaración
'es de las news
'
'Htmlhelp consts
Private Enum HH_COMMAND
   HH_DISPLAY_TOPIC = &H0
   HH_HELP_FINDER = &H0       ' WinHelp equivalent
   HH_DISPLAY_TOC = &H1       ' not currently implemented
   HH_DISPLAY_INDEX = &H2     ' not currently implemented
   HH_DISPLAY_SEARCH = &H3    ' not currently implemented
   HH_SET_WIN_TYPE = &H4
   HH_GET_WIN_TYPE = &H5
   HH_GET_WIN_HANDLE = &H6
   HH_GET_INFO_TYPES = &H7    ' not currently implemented
   HH_SET_INFO_TYPES = &H8    ' not currently implemented
   HH_SYNC = &H9
   HH_ADD_NAV_UI = &HA        ' not currently implemented
   HH_ADD_BUTTON = &HB        ' not currently implemented
   HH_GETBROWSER_APP = &HC    ' not currently implemented
   HH_KEYWORD_LOOKUP = &HD
```

```
    HH_DISPLAY_TEXT_POPUP = &HE ' display string resource id
                          ' or text in a popup window
    HH_HELP_CONTEXT = &HF       ' display mapped numeric value
                          ' in dwData
    HH_TP_HELP_CONTEXTMENU      ' Text pop-up help, similar to
                          ' WinHelp's HELP_CONTEXTMENU.
    HH_TP_HELP_WM_HELP = &H11   ' text pop-up help, similar to
                          ' WinHelp's HELP_WM_HELP.
    HH_CLOSE_ALL = &H12         ' close all windows opened directly
                          ' or indirectly by the caller
    HH_ALINK_LOOKUP = &H13      ' ALink version of HH_KEYWORD_LOOKUP
End Enum

'HtmlHelp api call
'NOTA: Si se usa esta forma, hay que indicar el último parámetro
'     con la palabra ByVal delante...
'Private Declare Function HtmlHelp Lib "hhctrl.ocx" Alias "HtmlHelpA" _
    (ByVal hwndCaller As Long, ByVal pszFile As String, _
    ByVal uCommand As HH_COMMAND, dwData As Any) As Long
'Con esta funciona perfectamente
Private Declare Function HtmlHelp Lib "hhctrl.ocx" Alias "HtmlHelpA" _
    (ByVal hwndCaller As Long, ByVal pszFile As String, _
    ByVal uCommand As HH_COMMAND, ByVal dwData As Long) As Long
Dim n As Integer
Dim ClaseMP As New MP
Dim mensaje As String
Dim seleccion As Integer
Dim anchoSstab
Dim altoSstab
Dim arribaSstab
Dim izquierdaSstab
Dim arribaImprimir
Dim arribaResul
Dim anchoResul
Dim altoResul
Dim anchoitera
Dim altoitera
Dim arribaltera
Dim izquierdaltera
Dim arribaFgresul
Dim altoFgresul
Dim anchoFgresul
Dim colororg
Dim VLabel As Boolean
Public limpiar As Boolean
Private Acepta As Boolean
```

```
Private VLConexion As ADODB.Connection
Private VLRegistro As ADODB.Recordset
Private VLNombre As String
Private VLNombreTrabajo As String
Private VLVerifica As Boolean
Private VLBDD As Boolean

Property Let Aceptar(parAceptar As Boolean)
   Acepta = parAceptar
End Property

Property Let Nombre(parNombre As String)
   VLNombre = parNombre
End Property

Property Let NombreTrabajo(parNombreTrabajo As String)
   VLNombreTrabajo = parNombreTrabajo
End Property


Private Sub CmdMPcalcular_Click()

Dim i As Integer
Dim count As Integer
count = LstMPresultados.ListCount
For i = 1 To count
LstMPresultados.RemoveItem count - i
Next i
Select Case seleccion
Case Is = 1
If (TxtMPpolinomio = "" Or TxtMPpolinomio = " " Or TxtMPr = "" Or TxtMPr = " ") Then
mensaje = MsgBox("Parámetros del algoritmo incompletos", 48, "Error en el ingreso")
LstMPresultados.Clear
FgMPiteraciones.Clear
Exit Sub
End If
ClaseMP.Flag = False
ClaseMP.AlgoritmodeHorner
If limpiar = True Then
LstMPresultados.Clear
FgMPiteraciones.Clear
limpiar = False
Exit Sub
Else
PresentarMP
```

```
End If
CmdMPiteraciones.Visible = True
Case Is = 2
If (TxtMPpolinomio = "" Or TxtMPpolinomio = " " Or TxtMPr = "" Or TxtMPr = " " Or _
TxtMPs = "" Or TxtMPs = " " Or TxtMPe = "" Or TxtMPe = " ") Then
mensaje = MsgBox("Parámetros del algoritmo incompletos", 48, "Error en el ingreso")
LstMPresultados.Clear
FgMPiteraciones.Clear
TxtMPver.Visible = False
Exit Sub
End If


ClaseMP.AlgoritmodeNewtonHorner
If limpiar = True Then
LstMPresultados.Clear
FgMPiteraciones.Clear
limpiar = False
TxtMPver.Visible = False
Exit Sub
Else
PresentarMP
TxtMPver = " "
TxtMPver.Visible = True
End If
CmdMPiteraciones.Visible = True
Case Is = 3
If (TxtMPpolinomio = "" Or TxtMPpolinomio = " " Or TxtMPr = "" Or TxtMPr = " " Or _
TxtMPs = "" Or TxtMPs = " " Or TxtMPe = "" Or TxtMPe = " " Or TxtMPiteraciones = ""
Or TxtMPiteraciones = " ") Then
mensaje = MsgBox("Parámetros del algoritmo incompletos", 48, "Error en el ingreso")
LstMPresultados.Clear
FgMPiteraciones.Clear
Exit Sub
End If


ClaseMP.AlgoritmodeNewtonBaristow
If limpiar = True Then
LstMPresultados.Clear
FgMPiteraciones.Clear
limpiar = False
Exit Sub
Else
PresentarMP
```

```
End If
CmdMPiteraciones.Visible = True
Case Else
If (TxtMPpolinomio = "" Or TxtMPpolinomio = " " Or TxtMPr = "" Or TxtMPr = " " Or _
TxtMPs = "" Or TxtMPs = " ") Then
mensaje = MsgBox("Parámetros del algoritmo incompletos", 48, "Error en el ingreso")
LstMPresultados.Clear
FgMPiteraciones.Clear
Exit Sub
End If


ClaseMP.AplicacióndeNewtonBairstow
End Select
MPimprimir.Enabled = True
End Sub

Private Sub Command1_Click()
   PLImprimir
End Sub

Private Sub CmdMPiteraciones_Click()
FgMPiteraciones.Visible = True
LblMPitera.Visible = True

End Sub

Private Sub Form_Load()
   MPabrir.Enabled = False
   MPguardarcomo.Enabled = False
   MPimprimir.Enabled = False
   VGForma = "MP"
End Sub




Private Sub LstMPresultados_Click()
If seleccion = 2 Then
TxtMPver = LstMPresultados.Text
End If

End Sub

Private Sub MPabrir_Click()
```

```
    FrmRecuperaTrabajo.Show vbModal
    If Acepta = True Then
        PLConexion True
        PLRecupera
    End If
End Sub

Private Sub MPalgH_Click()
'borramos todos los controles
Dim i As Integer
Dim count As Integer
count = LstMPresultados.ListCount
For i = 1 To count
LstMPresultados.RemoveItem count - i
Next i
TxtMPpolinomio.Text = " "
TxtMPr.Text = " "
TxtMPs.Text = " "
TxtMPe.Text = " "
TxtMPiteraciones.Text = " "
'valor  para  elegir el algoritmo
seleccion = 1
    MPabrir.Enabled = True
    MPguardarcomo.Enabled = True
    FrmRecuperaTrabajo.ver = seleccion

'muestran los controles necesarios para la entrada de datos
'para el algoritmo
LblMPpolinomio.Visible = True
TxtMPpolinomio.Visible = True
LblMPr.Caption = "Valor inicial Xo="
LblMPr.Visible = True
TxtMPr.Visible = True
LblMPs.Visible = False
TxtMPs.Visible = False
LblMPe.Visible = False
TxtMPe.Visible = False
LblMPiteraciones.Visible = False
TxtMPiteraciones.Visible = False
FraMPerrores.Visible = False
OptMPEw.Visible = False
OptMPEH.Visible = False
MPalgH.Checked = True
MPalgNB.Checked = False
MPalgNH.Checked = False
MPalgANB.Checked = False
```

```
FgMPiteraciones.Visible = False
FgMPiteraciones.Clear
CmdMPiteraciones.Visible = False
CmdMPgraficar.Visible = False
LblMPitera.Visible = False
LblNomalg.Caption = "Algoritmo de Horner"
LblSintaxis.Caption = "Polinomio= AnX^n+An-1X^n-1+.....+A1Xo+Ao (2X^3-4X^2+X-
5)" & Chr(10) & "Xo= valor en el que se evalua el polinomio y sus derivadas"
CmdMPiteraciones.Caption = "Cálculos"
LblMPitera.Caption = "Cálculos"
TxtMPver.Visible = False
End Sub



Private Sub MPalgNH_Click()
'borramos todos los controles
Dim i As Integer
Dim count As Integer
count = LstMPresultados.ListCount
For i = 1 To count
LstMPresultados.RemoveItem count - i
Next i
TxtMPpolinomio.Text = " "
TxtMPr.Text = " "
TxtMPs.Text = " "
TxtMPe.Text = " "
TxtMPiteraciones.Text = " "
'valor  para  elegir el algoritmo
seleccion = 2
   MPabrir.Enabled = True
   MPguardarcomo.Enabled = True
   FrmRecuperaTrabajo.ver = seleccion
'muestran los controles necesarios para la entrada de datos
'para el algoritmo
LblMPpolinomio.Visible = True
TxtMPpolinomio.Visible = True
LblMPr.Caption = "Valor inicial Xo="
LblMPr.Visible = True
TxtMPr.Visible = True
LblMPs.Caption = "Tolerancia ="
LblMPs.Visible = True
TxtMPs.Visible = True
LblMPe.Caption = "Iteraciones máximas="
LblMPe.Visible = True
```

```
TxtMPe.Visible = True
LblMPiteraciones.Visible = False
TxtMPiteraciones.Visible = False
FraMPerrores.Visible = True
OptMPEw.Caption = "Ea"
OptMPEw.Visible = True
OptMPEH.Caption = "Ep"
OptMPEH.Visible = True
MPalgH.Checked = False
MPalgNB.Checked = False
MPalgNH.Checked = True
MPalgANB.Checked = False
FgMPiteraciones.Visible = False
FgMPiteraciones.Clear
CmdMPiteraciones.Visible = False
CmdMPgraficar.Visible = False
LblMPitera.Visible = False
LblNomalg.Caption = "Algoritmo de Newton-Horner"
LblSintaxis.Caption = "Polinomio= AnX^n+An-1X^n-1+.....+A1Xo+Ao (2X^3-4X^2+X-
5)" & Chr(10) & "Xo= " & Chr(177) & "Ao/A1 (correspondiente al polinomio
normalizado)" & Chr(10) & " Pn(X)= X^n+An-1X^n-1+....+A1Xo+Ao)" & Chr(10) & _
"Tolerancia= 0.001" & Chr(10) & "Iteraciones máximas= número natural, representa
el número de iteraciones que el algoritmo realizará" & Chr(10) & "Ea = error absoluto;
Ep= error en el polinomio"
CmdMPiteraciones.Caption = " iteraciones"
LblMPitera.Caption = "Iteraciones"
TxtMPver.Visible = False
End Sub

Private Sub MPalgNB_Click()
'borramos todos los controles
Dim i As Integer
Dim count As Integer
count = LstMPresultados.ListCount
For i = 1 To count
LstMPresultados.RemoveItem count - i
Next i
TxtMPpolinomio.Text = " "
TxtMPr.Text = " "
TxtMPs.Text = " "
TxtMPe.Text = " "
TxtMPiteraciones.Text = " "
'valor  para  elegir el algoritmo
seleccion = 3
   MPabrir.Enabled = True
   MPguardarcomo.Enabled = True
```

```
    FrmRecuperaTrabajo.ver = seleccion

'muestran los controles necesarios para la entrada de datos
'para el algoritmo
LblMPpolinomio.Visible = True
TxtMPpolinomio.Visible = True
LblMPr.Caption = "Valor de r="
LblMPr.Visible = True
TxtMPr.Visible = True
LblMPs.Caption = "Valor de s="
LblMPs.Visible = True
TxtMPs.Visible = True
LblMPe.Caption = "Tolerancia="
LblMPe.Visible = True
TxtMPe.Visible = True
LblMPiteraciones.Caption = "Iteraciones máximas="
LblMPiteraciones.Visible = True
TxtMPiteraciones.Visible = True
FraMPerrores.Visible = True
OptMPEw.Caption = "Ew"
OptMPEw.Visible = True
OptMPEH.Caption = "EH"
OptMPEH.Visible = True
MPalgH.Checked = False
MPalgNB.Checked = True
MPalgNH.Checked = False
MPalgANB.Checked = False
FgMPiteraciones.Visible = False
FgMPiteraciones.Clear
CmdMPiteraciones.Visible = False
CmdMPgraficar.Visible = False
LblMPitera.Visible = False
LblNomalg.Caption = "Algoritmo de Newton-Bairstow"
LblSintaxis.Caption = "Polinomio= AnX^n+An-1X^n-1+.....+A1Xo+Ao (2X^3-4X^2+X-
5)" & Chr(10) & "r= valor real del punto cercano a la raíz; s= valor imaginario del
punto cercano a la raíz" & Chr(10) & _
"Tolerancia= 0.001" & Chr(10) & "Iteraciones máximas= número natural, representa
el número de iteraciones que el algoritmo realizará" & Chr(10) & "Ew = error absoluto
relativo al valor inicial" & Chr(10) & " EH= error en las funciones f,g"
CmdMPiteraciones.Caption = " iteraciones"
LblMPitera.Caption = "Iteraciones"
TxtMPver.Visible = False
End Sub
Private Sub MPalgANB_Click()
'borramos todos los controles
Dim i As Integer
```

```
Dim count As Integer
count = LstMPresultados.ListCount
For i = 1 To count
LstMPresultados.RemoveItem count - i
Next i
TxtMPpolinomio.Text = " "
TxtMPr.Text = " "
TxtMPs.Text = " "
TxtMPe.Text = " "
TxtMPiteraciones.Text = " "
'valor  para  elegir el algoritmo
seleccion = 4
   MPabrir.Enabled = True
   MPguardarcomo.Enabled = True
   FrmRecuperaTrabajo.ver = seleccion

'muestran los controles necesarios para la entrada de datos
'para el algoritmo
LblMPpolinomio.Visible = True
TxtMPpolinomio.Visible = True
LblMPr.Caption = "Valor real Re="
LblMPr.Visible = True
TxtMPr.Visible = True
LblMPs.Caption = "Valor imaginario Im="
LblMPs.Visible = True
TxtMPs.Visible = True
LblMPe.Visible = False
TxtMPe.Visible = False
LblMPiteraciones.Visible = False
TxtMPiteraciones.Visible = False
FraMPerrores.Visible = False
OptMPEw.Visible = False
OptMPEH.Visible = False
MPalgH.Checked = False
MPalgNB.Checked = False
MPalgNH.Checked = False
MPalgANB.Checked = True
FgMPiteraciones.Visible = False
FgMPiteraciones.Clear
CmdMPiteraciones.Visible = False
CmdMPgraficar.Visible = False
LblMPitera.Visible = False
LblNomalg.Caption = "Aplicación de Newton-Bairstow"
LblSintaxis.Caption = "Polinomio= AnX^n+An-1X^n-1+.....+A1Xo+Ao (2X^3-4X^2+X-
5)" & Chr(10) & "Re valor real del punto en el que se evalua el polinomio y su primera
```

```
derivada" & Chr(10) & "Im= valor imaginario del punto en el que se evalua el
polinomio y su primera derivada"
TxtMPver.Visible = False
End Sub

Private Sub MPayuda_Click()
'Así se llamaría para mostrar un tópico de la ayuda
   Dim h As Long

   'h = HtmlHelp(Me.hWnd, "Milton.hlp", HH_DISPLAY_TOPIC, 0&)

   h = HtmlHelp(Me.hWnd, "Milton.hlp", HH_HELP_CONTEXT, 2030&)
End Sub

Private Sub MPgraficar_Click()
Frmescalas.Show
End Sub

'Conexion a la BDD
Private Sub PLConexion(parBDD As Boolean)
Dim VTCad As String
Dim VTBase As Database
Dim VTcadena As String
   VTcadena = App.Path & "/Seguridad.mdb"
   Set VLConexion = New ADODB.Connection
   If parBDD = False Then
      VTCad = "Provider=SQLOLEDB.1;Integrated Security=SSPI;Persist Security
Info=False;Initial Catalog=Seguridad"
   Else
      VTCad = "Provider=Microsoft.Jet.OLEDB.4.0;Data Source='" & VTcadena &
"';Persist Security Info=False"
   End If
   VLConexion.Open VTCad
End Sub

Private Sub MPguardarcomo_Click()
   FrmNombre.Show vbModal
   If Acepta = True Then
      PLConexion True
      PLGuardarBDD
   End If
End Sub

Private Sub PLGuardarBDD()
   Dim VTCad As String
   Dim VTReg As ADODB.Recordset
```

```
   Dim VTID As Integer
   Dim VB As Integer
On Error GoTo err1
   PLVerificaNombre Trim(VLNombre)
   '******* APROXIMACIONES SUCESIVAS, MIUB
   If seleccion = 1 Then
      If VLVerifica = True Then   'Si existe un nombre con el mismo nombre actualiza
         VB = MsgBox("El nombre de ese trabajo ya existe, desea reemplazarlo?",
vbYesNo, "Guardar Trabajo Como")
         If VB = 6 Then
            VTCad = "update MPol set " _
                  & " polinomio= '" & Trim(TxtMPpolinomio.Text) & "', r_Xo_real='" &
Trim(TxtMPr.Text) & "' " _
                  & " where Nombre = '" & VLNombre & "'"
            VLConexion.Execute (VTCad)
         End If
      Else    'Guarda el trabajo
         VTID = PLRecuperaCodigo
         VTCad = "insert into MPol values " _
               & "(" & CInt(VTID) & ", " _
               & " '" & TxtMPpolinomio.Text & "', '" & TxtMPr.Text & "', " _
               & " '" & TxtMPs.Text & "', '" & TxtMPe.Text & "', '" &
TxtMPiteraciones.Text & "', " _
               & " '" & Trim(VLNombre) & "', " & 1 & ", " & CInt(seleccion) & ")"
               '& " '" & Trim(VLNombre) & "', " & CInt(VGIDUsuario) & ", " &
CInt(seleccion ) & ")"
         VLConexion.Execute (VTCad)
         MsgBox "Trabajo guardado exitosamente", vbInformation, "Guardar trabajo
como"
      End If

   ElseIf seleccion = 2 Then
      If VLVerifica = True Then   'Si existe un nombre con el mismo nombre actualiza
         VB = MsgBox("El nombre de ese trabajo ya existe, desea reemplazarlo?",
vbYesNo, "Guardar Trabajo Como")
         If VB = 6 Then
            VTCad = "update MPol set " _
                  & " polinomio= '" & Trim(TxtMPpolinomio.Text) & "', r_Xo_real='" &
Trim(TxtMPr.Text) & "', " _
                  & " Tol= '" & Trim(TxtMPe.Text) & "', Imax='" &
Trim(TxtMPiteraciones.Text) & "' " _
                  & " where Nombre = '" & VLNombre & "'"
            VLConexion.Execute (VTCad)
         End If
      Else    'Guarda el trabajo
         VTID = PLRecuperaCodigo
```

```
        VTCad = "insert into MPol values " _
              & "(" & CInt(VTID) & ", " _
              & " '" & TxtMPpolinomio.Text & "', '" & TxtMPr.Text & "', " _
              & " '" & TxtMPs.Text & "', '" & TxtMPe.Text & "', '" &
TxtMPiteraciones.Text & "', " _
              & " '" & Trim(VLNombre) & "', " & 1 & ", " & CInt(seleccion) & ")"
              '& " '" & Trim(VLNombre) & "', " & CInt(VGIDUsuario) & ", " &
CInt(seleccion ) & ")"
        VLConexion.Execute (VTCad)
        MsgBox "Trabajo guardado exitosamente", vbInformation, "Guardar trabajo
como"
      End If

   ElseIf seleccion = 3 Then
      If VLVerifica = True Then   'Si existe un nombre con el mismo nombre actualiza
        VB = MsgBox("El nombre de ese trabajo ya existe, desea reemplazarlo?",
vbYesNo, "Guardar Trabajo Como")
        If VB = 6 Then
          VTCad = "update MPol set " _
              & " polinomio= '" & Trim(TxtMPpolinomio.Text) & "', r_Xo_real='" &
Trim(TxtMPr.Text) & "', " _
              & " s_img= '" & Trim(TxtMPs.Text) & "', " _
              & " Tol= '" & Trim(TxtMPe.Text) & "', Imax='" &
Trim(TxtMPiteraciones.Text) & "' " _
              & " where Nombre = '" & VLNombre & "'"
          VLConexion.Execute (VTCad)
        End If
      Else   'Guarda el trabajo
        VTID = PLRecuperaCodigo
        VTCad = "insert into MPol values " _
              & "(" & CInt(VTID) & ", " _
              & " '" & TxtMPpolinomio.Text & "', '" & TxtMPr.Text & "', " _
              & " '" & TxtMPs.Text & "', '" & TxtMPe.Text & "', '" &
TxtMPiteraciones.Text & "', " _
              & " '" & Trim(VLNombre) & "', " & 1 & ", " & CInt(seleccion) & ")"
              '& " '" & Trim(VLNombre) & "', " & CInt(VGIDUsuario) & ", " &
CInt(seleccion ) & ")"
        VLConexion.Execute (VTCad)
        MsgBox "Trabajo guardado exitosamente", vbInformation, "Guardar trabajo
como"
      End If

   ElseIf seleccion = 4 Then
      If VLVerifica = True Then   'Si existe un nombre con el mismo nombre actualiza
        VB = MsgBox("El nombre de ese trabajo ya existe, desea reemplazarlo?",
vbYesNo, "Guardar Trabajo Como")
```

```
        If VB = 6 Then
           VTCad = "update MPol set " _
                 & " polinomio= '" & Trim(TxtMPpolinomio.Text) & "', r_Xo_real='" &
Trim(TxtMPr.Text) & "', " _
                 & " s_img= '" & Trim(TxtMPs.Text) & "' " _
                 & " where Nombre = '" & VLNombre & "'"
           VLConexion.Execute (VTCad)
        End If
     Else    'Guarda el trabajo
        VTID = PLRecuperaCodigo
        VTCad = "insert into MPol values " _
              & "(" & CInt(VTID) & ", " _
              & " '" & TxtMPpolinomio.Text & "', '" & TxtMPr.Text & "', " _
              & " '" & TxtMPs.Text & "', '" & TxtMPe.Text & "', '" &
TxtMPiteraciones.Text & "', " _
              & " '" & Trim(VLNombre) & "', " & 1 & ", " & CInt(seleccion) & ")"
              '& " '" & Trim(VLNombre) & "', " & CInt(VGIDUsuario) & ", " &
CInt(seleccion ) & ")"
        VLConexion.Execute (VTCad)
        MsgBox "Trabajo guardado exitosamente", vbInformation, "Guardar trabajo
como"
     End If

  End If
Exit Sub
err1:
  If err.Number <> 0 Then
     MsgBox "Se ha producido el siguiente error: " & err.Description & "." _
          & "Por favor consultar con el administrador", vbCritical
  End If
End Sub


Private Sub PLVerificaNombre(parNombre As String)
Dim VTReg As ADODB.Recordset
Dim VTCad As String
     VTCad = "select Nombre from MPol where Nombre = '" & parNombre & "' and
tipo = " & seleccion
     Set VTReg = VLConexion.Execute(VTCad)
     With VTReg
        If Not VTReg.EOF Then
           .MoveFirst
           Do Until .EOF
              If Trim(VTReg(0)) = parNombre Then
                 VLVerifica = True
                 Exit Do
```

```
                Else
                   VLVerifica = False
                End If
                .MoveNext
             Loop
          Else
             VLVerifica = False
          End If
       End With

End Sub

Private Function PLRecuperaCodigo() As Integer
   Dim VTCad As String
   Dim VTReg As ADODB.Recordset
On Error GoTo err1
      VTCad = "select max(IDMPol) from MPol where tipo = " & seleccion
      Set VTReg = VLConexion.Execute(VTCad)
      If Not IsNull(VTReg(0)) Then
         PLRecuperaCodigo = VTReg(0) + 1
      Else
         PLRecuperaCodigo = 1
      End If
Exit Function
err1:

End Function

Private Sub PLRecupera()
   Dim VTCad As String
   Dim VTReg As ADODB.Recordset
On Error GoTo err1
      VTCad = "select * from MPol where Nombre = '" & VLNombreTrabajo & "'"
      Set VTReg = VLConexion.Execute(VTCad)
      If Not VTReg.EOF Then
         TxtMPpolinomio.Text = Trim(VTReg(1))
         TxtMPs.Text = Trim(VTReg(3))
         TxtMPr.Text = Trim(VTReg(2))
         TxtMPe.Text = Trim(VTReg(4))
         TxtMPiteraciones.Text = Trim(VTReg(5))
      End If
Exit Sub
err1:

End Sub
```

```
Private Sub PLImprimir()
   PLCambio
      PrintForm
   PLOriginal
End Sub

Private Sub PLCambio()
   Me.BackColor = &HFFFFFF
   Me.BorderStyle = 0
'   frapar.BackColor = &HFFFFFF
'   frapar.BorderStyle = 0
'   If seleccion = 2 Or seleccion = 3 Then
'      FraMPerrores.Visible = False
'   End If
'   LstMPresultados.BackColor = &HFFFFFF
'   CmdMPcalcular.Visible = False
'   CmdMPiteraciones.Visible = False
End Sub

Private Sub PLOriginal()
   Me.BackColor = &H8000000F
   Me.BorderStyle = 2
'   frapar.BackColor = &H8000000F
'   frapar.BorderStyle = 1
'   If seleccion = 2 Or seleccion = 3 Then
'      FraMPerrores.Visible = True
'   End If
'   LstMPresultados.BackColor = &H8000000F
'   CmdMPcalcular.Visible = True
'   CmdMPiteraciones.Visible = True
End Sub

Private Sub MPimprimir_Click()
      Dim EndTime As Date
   Sincolor

      EndTime = DateAdd("s", 1 / 1E+16, Now)
      Do Until Now > EndTime
        DoEvents
      Loop
   'Form1.Command3_Click
   Set Form1.Picture1.Picture = CaptureClient(Me)
   Form1.Visible = True
   FrmPrincipalalgoritmos.Visible = False
   Concolor
End Sub
```

```
Function Fgit(r As Integer, c As Integer) As Integer
  Fgit = c + FgMPiteraciones.Cols * r
End Function
Private Sub PresentarMP()
Dim i As Integer
Dim j As Integer
Dim valor As Double
On Error GoTo err1
n = ClaseMP.grado

Select Case seleccion
Case Is = 1

FgMPiteraciones.Cols = n + 2
FgMPiteraciones.Rows = n + 2
FgMPiteraciones.FixedCols = 1
FgMPiteraciones.FixedRows = 1
FgMPiteraciones.TextArray(Fgit(0, 0)) = "i/j"

For i = 1 To n + 1
FgMPiteraciones.TextArray(Fgit(i, 0)) = i
Next i
For i = 1 To n + 1
FgMPiteraciones.TextArray(Fgit(0, i)) = i
Next i
For i = 1 To n + 1
FgMPiteraciones.ColWidth(i) = 800
Next i
For i = 0 To n + 1
FgMPiteraciones.ColAlignmentFixed(i) = 4
Next i
For i = 1 To n + 1
   For j = 1 To n + 1
   valor = ClaseMP.regresariteraciones(i - 1, j - 1)
      If (valor <= 0 Or valor > 0.000000000000001) Then
      FgMPiteraciones.TextMatrix(i, j) = valor
      Else
      FgMPiteraciones.TextMatrix(i, j) = 0
      End If
  Next j
Next i

Case Is = 2
LstMPresultados.Clear
LstMPresultados.AddItem "Raiz aproximada= " & ClaseMP.Rraiz
```

```
LstMPresultados.AddItem "Polinomio deflexionado= " & ClaseMP.polinomio

FgMPiteraciones.Cols = 6
FgMPiteraciones.Rows = Val(TxtMPe) + 1
FgMPiteraciones.FixedCols = 1
FgMPiteraciones.FixedRows = 1
FgMPiteraciones.TextArray(Fgit(0, 0)) = "i/j"

For i = 1 To Val(TxtMPe)
FgMPiteraciones.TextArray(Fgit(i, 0)) = i
Next i

FgMPiteraciones.TextArray(Fgit(0, 0)) = "Iteraciones"
FgMPiteraciones.TextArray(Fgit(0, 1)) = "Xi-1"
FgMPiteraciones.TextArray(Fgit(0, 2)) = "P(x)"
FgMPiteraciones.TextArray(Fgit(0, 3)) = "P'(x)"
FgMPiteraciones.TextArray(Fgit(0, 4)) = "Xi"
FgMPiteraciones.TextArray(Fgit(0, 5)) = "Error"



For i = 1 To 5
FgMPiteraciones.ColWidth(i) = 800
Next i
For i = 0 To 5
FgMPiteraciones.ColAlignmentFixed(i) = 4
Next i
For i = 1 To Val(TxtMPe)
   For j = 1 To 5
      valor = ClaseMP.Riteraciones(i - 1, j - 1)
      If (valor <= 0 Or valor > 0.000000000000001) Then
      FgMPiteraciones.TextMatrix(i, j) = valor
      Else
      FgMPiteraciones.TextMatrix(i, j) = 0
      End If
  Next j
Next i




Case Is = 3
FgMPiteraciones.Cols = 2 * n + 6
FgMPiteraciones.Rows = Val(TxtMPiteraciones) + 1
```

```
FgMPiteraciones.FixedCols = 1
FgMPiteraciones.FixedRows = 1
FgMPiteraciones.TextArray(Fgit(0, 0)) = "Iteraciones"

For i = 1 To Val(TxtMPiteraciones)
FgMPiteraciones.TextArray(Fgit(i, 0)) = i
Next i
For i = 1 To n + 1
FgMPiteraciones.TextArray(Fgit(0, i)) = "b" & i - 1
Next i
For i = n + 2 To (2 * n) + 2
FgMPiteraciones.TextArray(Fgit(0, i)) = "C" & i - n - 2
Next i
FgMPiteraciones.TextArray(Fgit(0, (2 * n) + 3)) = "DELTAJ"
FgMPiteraciones.TextArray(Fgit(0, (2 * n) + 4)) = "r"
FgMPiteraciones.TextArray(Fgit(0, (2 * n) + 5)) = "s"


For i = 1 To (2 * n) + 5
FgMPiteraciones.ColWidth(i) = 800
Next i
For i = 0 To (2 * n) + 5
FgMPiteraciones.ColAlignmentFixed(i) = 4
Next i
For i = 1 To Val(TxtMPiteraciones)
   For j = 1 To (2 * n) + 5
   valor = ClaseMP.regresariteraciones(i - 1, j - 1)
      If (valor <= 0 Or valor > 0.000000000000001) Then
      FgMPiteraciones.TextMatrix(i, j) = valor
      Else
      FgMPiteraciones.TextMatrix(i, j) = 0
      End If
  Next j
Next i

Case Is = 4
End Select
Exit Sub
err1:
End Sub

Private Sub MPsalir_Click()
Unload Me
End Sub

Private Sub TxtMPe_Change()
```

```
Dim mensaje As String
LTrim (TxtMPe)
If seleccion = 2 Then
If Val(TxtMPe.Text) > 32767 Then
mensaje = MsgBox("Fuera del rango", 48, "Error en el ingreso")
End If
End If

If seleccion = 3 Then
If Val(TxtMPe.Text) > 32767 Then
mensaje = MsgBox("Fuera del rango", 48, "Error en el ingreso")
End If
End If



End Sub

Private Sub TxtMPe_KeyPress(KeyAscii As Integer)

If seleccion = 2 Then
'filtra la entrada de datos que no sean números enteros y positivos
  If (KeyAscii < 48 Or KeyAscii > 57) Then

        If (KeyAscii <> 8) Then
        KeyAscii = 0
        End If

   End If
  If KeyAscii = Asc(vbCr) Then ' Elimina los retornos para quitar los pitidos.
  KeyAscii = 0
  End If
End If
If seleccion = 3 Then
'filtra la entrada de datos que no sean números no incluye signo - y si el punto
  If (KeyAscii < 48 Or KeyAscii > 57) Then
      If (KeyAscii <> 46) Then
        If (KeyAscii <> 8) Then
        KeyAscii = 0
        End If
      End If
   End If
  If KeyAscii = Asc(vbCr) Then ' Elimina los retornos para quitar los pitidos.
  KeyAscii = 0
  End If
```

```
End If


End Sub

Private Sub TxtMPiteraciones_Change()
Dim mensaje As String
LTrim (TxtMPiteraciones)
If seleccion = 3 Then
If Val(TxtMPiteraciones.Text) > 32767 Then
mensaje = MsgBox("Fuera del rango", 48, "Error en el ingreso")
End If
End If
End Sub

Private Sub TxtMPiteraciones_KeyPress(KeyAscii As Integer)
If seleccion = 3 Then
'filtra la entrada de datos que no sean números enteros y positivos
  If (KeyAscii < 48 Or KeyAscii > 57) Then

        If (KeyAscii <> 8) Then
        KeyAscii = 0
        End If

   End If
  If KeyAscii = Asc(vbCr) Then ' Elimina los retornos para quitar los pitidos.
  KeyAscii = 0
  End If
End If
End Sub


Private Sub TxtMPpolinomio_Change()
LTrim (TxtMPpolinomio)
End Sub

Private Sub TxtMPpolinomio_KeyPress(KeyAscii As Integer)

'filtra la entrada de datos que  sean números incluye el signo - , el signo + y el punto
y la letra X
  If (KeyAscii < 48 Or KeyAscii > 57) Then
      If (KeyAscii = 88 Or KeyAscii = 94 Or KeyAscii = 43) Then
      Exit Sub
      End If
      If (KeyAscii = 45) Then
```

```
        Exit Sub
        End If
        If (KeyAscii <> 46) Then
           If (KeyAscii <> 8) Then
           KeyAscii = 0
           End If
        End If
     End If
    If KeyAscii = Asc(vbCr) Then ' Elimina los retornos para quitar los pitidos.
    KeyAscii = 0
    End If
End Sub

Private Sub TxtMPr_Change()
LTrim (TxtMPr)
End Sub

Private Sub TxtMPr_KeyPress(KeyAscii As Integer)
'filtra la entrada de datos que no sean números incluye el signo - y el punto
    If (KeyAscii < 48 Or KeyAscii > 57) Then
        If (KeyAscii = 45) Then
        Exit Sub
        End If
        If (KeyAscii <> 46) Then
           If (KeyAscii <> 8) Then
           KeyAscii = 0
           End If
        End If
     End If
    If KeyAscii = Asc(vbCr) Then ' Elimina los retornos para quitar los pitidos.
    KeyAscii = 0
    End If

End Sub

Private Sub TxtMPs_Change()
Dim mensaje As String
LTrim (TxtMPs)
If seleccion = 2 Then
If Val(TxtMPs.Text) > 32767 Then
mensaje = MsgBox("Fuera del rango", 48, "Error en el ingreso")
End If
End If
End Sub

Private Sub TxtMPs_KeyPress(KeyAscii As Integer)
```

```
If seleccion = 2 Then
'filtra la entrada de datos que no sean números no incluye signo - y si el punto
  If (KeyAscii < 48 Or KeyAscii > 57) Then
      If (KeyAscii <> 46) Then
         If (KeyAscii <> 8) Then
         KeyAscii = 0
         End If
      End If
   End If
  If KeyAscii = Asc(vbCr) Then ' Elimina los retornos para quitar los pitidos.
  KeyAscii = 0
  End If
End If



'filtra la entrada de datos que no sean números incluye el signo - y el punto
  If (KeyAscii < 48 Or KeyAscii > 57) Then
      If (KeyAscii = 45) Then
      Exit Sub
      End If
      If (KeyAscii <> 46) Then
         If (KeyAscii <> 8) Then
         KeyAscii = 0
         End If
      End If
   End If
  If KeyAscii = Asc(vbCr) Then ' Elimina los retornos para quitar los pitidos.
  KeyAscii = 0
  End If

End Sub

Public Sub Sincolor()
With Me
arribaImprimir = .Fraimprimir.Top
altoResul = .Fraresul.Height
anchoResul = .Fraresul.Width
arribaResul = .Fraresul.Top
arribaItera = .FgMPiteraciones.Top
altoitera = .FgMPiteraciones.Height
anchoitera = .FgMPiteraciones.Width
izquierdaItera = .FgMPiteraciones.Left
arribaSstab = .SSTab1.Top
izquierdaSstab = .SSTab1.Left
anchoSstab = .SSTab1.Width
```

```
altoSstab = .SSTab1.Height
arribaFgresul = .LstMPresultados.Top
altoFgresul = .LstMPresultados.Height
anchoFgresul = .LstMPresultados.Width
colororg = .BackColor
VLabel = .LblMPitera.Visible
.Fraimprimir.Top = 0
.SSTab1.Top = .SSTab1.Top + 1000
.SSTab1.Left = .SSTab1.Left + 1000
.SSTab1.Width = .SSTab1.Width / 4
.SSTab1.Height = .SSTab1.Height / 4
.Fraresul.Top = .Fraimprimir.Top + .Fraimprimir.Height
.Fraresul.Width = .Fraresul.Width + 7000
.Fraresul.Height = .Fraresul.Height - 500
.LstMPresultados.Height = .LstMPresultados.Height - 500
.LstMPresultados.Width = .LstMPresultados.Width + 7000
.FgMPiteraciones.Top = .Fraresul.Top + .Fraresul.Height
.FgMPiteraciones.Left = .Fraimprimir.Left
.FgMPiteraciones.Width = .Fraresul.Width
.FgMPiteraciones.Height = .FgMPiteraciones.Height - 500
.BackColor = &H80000009
.SSTab1.BackColor = &H80000009
.Fraimprimir.BackColor = &H80000009
.Fraresul.BackColor = &H80000009
.frapar.BackColor = &H80000009
.FraMPerrores.BackColor = &H80000009
.frapar.BorderStyle = 1
.TxtMPe.BorderStyle = 0
.TxtMPiteraciones.BorderStyle = 0
.TxtMPpolinomio.BorderStyle = 0
.TxtMPr.BorderStyle = 0
.TxtMPs.BorderStyle = 0
.Fraresul.BackColor = &H80000009
.LstMPresultados.BackColor = &H80000009
.Shape1.Visible = False
.LblNomalg.BackColor = &H80000009

.FgMPiteraciones.BackColorBkg = &H80000009
.FgMPiteraciones.BackColorFixed = &H80000009
.FgMPiteraciones.ScrollBars = 0
.CmdMPcalcular.Visible = False
.CmdMPiteraciones.Visible = False
.LblMPitera.Visible = False
.OptMPEH.BackColor = &H80000009
.OptMPEw.BackColor = &H80000009
FraMPSintaxis.Visible = False
```

```
End With
End Sub

Public Sub Concolor()
With Me
 .Fraimprimir.Top = arribaImprimir
 .Fraresul.Height = altoResul
 .Fraresul.Width = anchoResul
 .Fraresul.Top = arribaResul
 .FgMPiteraciones.Top = arribaItera
 .FgMPiteraciones.Height = altoItera
 .FgMPiteraciones.Width = anchoItera
 .FgMPiteraciones.Left = izquierdaItera
 .SSTab1.Top = arribaSstab
 .SSTab1.Left = izquierdaSstab
 .SSTab1.Width = anchoSstab
.SSTab1.Height = altoSstab
.LstMPresultados.Top = arribaFgresul
 .LstMPresultados.Height = altoFgresul
 .LstMPresultados.Width = anchoFgresul
.BackColor = colororg
.BackColor = colororg
.SSTab1.BackColor = colororg
.Fraimprimir.BackColor = colororg
.frapar.BorderStyle = 0
.frapar.BackColor = colororg
.FraMPerrores.BackColor = colororg
.frapar.BorderStyle = 1
.TxtMPe.BorderStyle = 1
.TxtMPiteraciones.BorderStyle = 1
.TxtMPpolinomio.BorderStyle = 1
.TxtMPr.BorderStyle = 1
.TxtMPs.BorderStyle = 1
.Fraresul.BackColor = colororg
.LstMPresultados.BackColor = colororg
.Shape1.Visible = True
.LblNomalg.BackColor = colororg
.FgMPiteraciones.BackColorBkg = colororg
.FgMPiteraciones.BackColorFixed = colororg
.FgMPiteraciones.ScrollBars = 3
.CmdMPcalcular.Visible = True
.CmdMPiteraciones.Visible = True
.OptMPEH.BackColor = colororg
.OptMPEw.BackColor = colororg
If VLabel = True Then
.LblMPitera = True
```

```
End If
FraMPSintaxis.Visible = True
End With
End Sub
```

Módulo de clase MP

```
' matriz de resultados del algortimo de Horner
Dim gradoP As Integer
Public Flag As Boolean
Dim Polideflexionado As String
Dim raíz As Double
Dim MResul() As Double
Dim Resulitera() As Double
Dim aviso As Boolean
Dim mensaje As String
Dim RH() As Double
Public Property Get regresariteraciones() As Variant
regresariteraciones = Resulitera
End Property
Public Property Get grado() As Variant
grado = gradoP
End Property
Public Property Get polinomio() As Variant
polinomio = Polideflexionado
End Property
Public Property Get Riteraciones() As Variant
Riteraciones = MResul
End Property
Public Property Get Rraiz() As Variant
Rraiz = raíz
End Property


Public Function Factorial(A As Integer) As Double
If A = 0 Or A = 1 Then
Factorial = 1
Else
Factorial = A * Factorial(A - 1)
End If
End Function


Public Sub AlgoritmodeHorner()
```

```
Dim A() As Double
Dim Atmp() As Double
Dim px As String
Dim T As Integer
Dim i As Integer
Dim n As Integer
Dim k As Integer
Dim Xo As Double
Dim K1 As Integer
Dim parciales() As Double
Dim Resultados() As Double
Dim b() As Double
Dim TxtPoli As TextBox
Dim txtxo As TextBox

Dim Lstresul As ListBox
On Error GoTo err1
Set TxtPoli = FrmMP.TxtMPpolinomio
Set txtxo = FrmMP.TxtMPr
Set Lstresul = FrmMP.LstMPresultados
If (txtxo = "" Or TxtPoli = "") Then
mensaje = MsgBox("Parámetros del algoritmo incompletos", 48, "Error en el ingreso")
FrmMP.limpiar = True
Exit Sub
End If
Xo = Val(txtxo)
px = TxtPoli

'Vector de coeficientes
A = Pn(px)
Atmp = A

' buscamos grado n del polinomio
T = 0
i = 0
Do While T = 0
T = A(20 - i)
i = i + 1
Loop
'For i = 0 To 20
' t = a(i)
' Next i
n = 20 - i + 1
gradoP = n
' dimensionamos matrices
ReDim parciales(n, n + 1) As Double 'resultados paciales
```

```
ReDim b(n) As Double ' vector de residuos
ReDim Resultado(n) As Double ' vector resultados
'implementamos algoritmo
b(n) = A(n)
K1 = Factorial(n)
Resultado(n) = K1 * A(n)
For i = 0 To n + 1
parciales(n, i) = A(n)
Next i
For i = 0 To n
parciales(i, 1) = A(i)
Next i
For k = 0 To n - 1

   For i = n - 1 To k Step -1
   b(i) = A(i) + b(i + 1) * Xo
   parciales(i, k + 1) = b(i)
   Next i
   K1 = Factorial(k)
   Resultado(k) = K1 * b(k)
   A = b
Next k

For i = 0 To n
parciales(i, 0) = Atmp(i)
Next i
Resulitera = parciales

' Presentación de resultados
RH = Resultado
If Flag = False Then
For i = 0 To n
   If i = 0 Then
   Lstresul.AddItem "P(" & n & ") = " & Resultado(i), 0
   Else
   Lstresul.AddItem "dP(" & n & ")/dX(" & i & ") = " & Resultado(i), i
   End If
Next i
End If
Exit Sub

err1:
If aviso = False Then
   mensaje = MsgBox("Función no ingresada según sintaxis o Parámetros del
algoritmo incompletos", 48, "Error en el ingreso")
FrmMP.limpiar = True
```

```
Else
aviso = False
End If


End Sub
Public Sub AlgoritmodeNewtonHorner()
Dim TxtPolino As TextBox
Dim TxtX As TextBox
Dim OptNH As OptionButton
Dim NHPolino As String
Dim TxtTolerancia As TextBox
Dim Iter As TextBox
Dim NHX As Double
Dim i As Integer
Dim T1 As Integer
Dim n As Integer
Dim nt As Integer
Dim Pevaluada As Double
Dim Pderivada As Double
Dim Xactual As Double
Dim e As Double
Dim err As Double
Dim maxiteraciones As Integer
Dim k As Integer
Dim NHT As Double
Dim Coef() As Double
Dim RHT() As Double
Dim msj As String
Dim indice As Integer
Dim alfa() As Double
Dim beta() As Double
Dim j As Integer
Dim limite As Integer
Dim cambio As Double
Dim inferior As Double
Dim superior As Double
On Error GoTo err1
Set Iter = FrmMP.TxtMPe
Set TxtTolerancia = FrmMP.TxtMPs
Set TxtPolino = FrmMP.TxtMPpolinomio
Set TxtX = FrmMP.TxtMPr
Set OptNH = FrmMP.OptMPEw


NHPolino = TxtPolino.Text
```

```
NHX = Val(TxtX.Text)
NHT = NHX
err = Val(TxtTolerancia.Text)
maxiteraciones = Val(Iter.Text)
Coef = Pn(NHPolino)

If (TxtPolino = "" Or TxtX = "" Or TxtTolerancia = "" Or Iter = "") Then
mensaje = MsgBox("Parámetros del algoritmo incompletos", 48, "Error en el ingreso")
FrmMP.limpiar = True
Exit Sub
End If

' Buscamos grado n del polinomio
T1 = 0
i = 0
Do While T1 = 0
T1 = Coef(20 - i)
i = i + 1
Loop

n = 20 - i + 1
nt = n
k = 1
indice = 0
ReDim MResul(0 To maxiteraciones - 1, 0 To 4) As Double
If n < 3 Then
msj = MsgBox("Grado del polinomio debe ser mayor o igual a 3 ", 48, "Falla")
FrmMP.limpiar = True
Exit Sub
Else

'Normalizamos el polinomio
If Coef(n) <> 0 Then
For i = 0 To n
Coef(i) = Coef(i) / Coef(n)
Next i
Else
Exit Sub
End If
'Buscamos el anillo complejo donde se localizan las raíces
ReDim alfa(0 To n - 1) As Double
ReDim beta(0 To n - 1) As Double
For i = n To 1 Step -1
alfa(i - 1) = Abs(Coef(i))
Next i
For i = n - 1 To 0 Step -1
```

```
beta(i) = Abs(Coef(i))
Next i
'Ordenamos
limite = 1
Do
For j = 0 To n - 1 - limite
   If alfa(j) >= alfa(j + 1) Then
   cambio = alfa(j)
   alfa(j) = alfa(j + 1)
   alfa(j + 1) = cambio
   End If
Next j
limite = limite + 1
Loop Until limite > (n - 1)
limite = 1
Do
For j = 0 To n - 1 - limite
   If beta(j) >= beta(j + 1) Then
   cambio = beta(j)
   beta(j) = beta(j + 1)
   beta(j + 1) = cambio
   End If
Next j
limite = limite + 1
Loop Until limite > (n - 1)
inferior = Abs(Coef(0)) / (alfa(n - 1) + Abs(Coef(0)))
superior = (beta(n - 1) + Abs(Coef(n))) / Abs(Coef(n))
If (NHX < inferior Or NHX > superior) Then
   If (NHX > -1 * inferior Or NHX < -1 * superior) Then
   msj = MsgBox("EL valor inicial no corresponde al anillo complejo " & inferior & " <=
X <= " & superior & " o " & -1 * superior & "<= X <= " & -1 * inferior & "donde se
localizan las raíces, Sugerencia ingrese en Xo un valor que este dentro del anillo ",
48, "Error en el ingreso")
   FrmMP.limpiar = True
   Exit Sub
   End If
End If

Do

TxtX.Text = NHT
AlgoritmodeHorner
TxtX.Text = NHX
RHT = RH ' Se duplica para preservar los coeficientes del polinomio deflexionado
Pevaluada = RH(0)
Pderivada = RH(1)
```

```
If Pderivada <> 0 Then
Xactual = NHT - (Pevaluada / Pderivada)
   If OptNH.Value = True Then
   TxtX.Text = Xactual
   Flag = True
   AlgoritmodeHorner
   e = Abs(NHT - Xactual)
   TxtX.Text = NHX
   Else
   Flag = True
   TxtX.Text = Xactual
   AlgoritmodeHorner
   e = Abs(RH(0))
   TxtX.Text = NHX
   End If
Else
msj = MsgBox("No esposible encontrar la raíz, intente otro punto", 48, "Falla")
FrmMP.limpiar = True
End If
'guardado de resultados de iteraciones
MResul(k - 1, 0) = NHT
MResul(k - 1, 3) = Xactual
MResul(k - 1, 1) = Pevaluada
MResul(k - 1, 2) = Pderivada
MResul(k - 1, 4) = e
If err < e Then
NHT = Xactual
End If
k = k + 1
Loop Until (e < err Or k > maxiteraciones)
If k > maxiteraciones Then
msj = MsgBox("Número de iteraciones máximas superado,no fue posible encontrar la
raíz del polinomio; Sugerencia: incremente el número de iteraciones", 48,
"Iteraciones")
FrmMP.limpiar = True
End If
raíz = Xactual
If Resulitera(n, 1) = 1 Then
Polideflexionado = "X^" & n - 1
Else
Polideflexionado = Resulitera(n, 1) & "X^" & n - 1
End If
For i = n - 1 To 3
   If Resulitera(i, 1) < 0 Then
   Polideflexionado = Polideflexionado & Resulitera(i, 1) & "X^" & i - 1
   Else
```

```
      If Resulitera(i, 1) > 0 Then
         If Resulitera(i, 1) = 1 Then
         Polideflexionado = Polideflexionado & "+" & "X^" & i - 1
         Else
         Polideflexionado = Polideflexionado & "+" & Resulitera(i, 1) & "X^" & i - 1
         End If
      End If
   End If
Next i
   If Resulitera(2, 1) < 0 Then
   Polideflexionado = Polideflexionado & Resulitera(2, 1) & "X"
   Else
      If Resulitera(2, 1) > 0 Then
         If Resulitera(2, 1) = 1 Then
         Polideflexionado = Polideflexionado & "+" & "X"
         Else
         Polideflexionado = Polideflexionado & "+" & Resulitera(i, 1) & "X"
         End If
      End If
   End If
   If Resulitera(1, 1) < 0 Then
   Polideflexionado = Polideflexionado & Resulitera(1, 1)
   Else
      If Resulitera(1, 1) > 0 Then
      Polideflexionado = Polideflexionado & "+" & Resulitera(1, 1)
      End If
   End If
End If

Exit Sub

err1:
If aviso = False Then
   mensaje = MsgBox("Función no ingresada según sintaxis o Parámetros del
algoritmo incompletos", 48, "Error en el ingreso")
FrmMP.limpiar = True
Else
aviso = False
End If


End Sub
Public Sub AlgoritmodeNewtonBaristow()


Dim px As String
```

```
Dim T As Double
Dim T1 As Integer
Dim r As Double
Dim s As Double
Dim i As Integer
Dim n As Integer
Dim k As Integer
Dim Xo As Double
Dim dr As Double
Dim ds As Double
Dim det As Double
Dim er As Double
Dim es As Double
Dim Txtr As TextBox
Dim Txts As TextBox
Dim Txtp As TextBox
Dim TxtI As TextBox
Dim TxtT As TextBox
Dim Opte As OptionButton
Dim LstR As ListBox
Dim maxiter As Integer
Dim mensage As String
Dim bandera As Boolean
Dim disc As Double
Dim r1 As Double
Dim r2 As Double
Dim i1 As Double
Dim i2 As Double
Dim nr As Integer
Dim b() As Double
Dim c() As Double
Dim A() As Double
Dim Re() As Double
Dim Im() As Double

On Error GoTo err1
Set Txtr = FrmMP.TxtMPr
Set Txts = FrmMP.TxtMPs
Set Txtp = FrmMP.TxtMPpolinomio
Set TxtI = FrmMP.TxtMPiteraciones
Set TxtT = FrmMP.TxtMPe
Set Opte = FrmMP.OptMPEw
Set LstR = FrmMP.LstMPresultados

r = Val(Txtr.Text)
s = Val(Txts.Text)
```

```
maxiter = Val(Txtl.Text)
T = Val(TxtT.Text)
px = Txtp.Text

If (Txtr = " " Or Txts = " " Or Txtl = " " Or TxtT = " " Or Txtp = " ") Then
mensaje = MsgBox("Parámetros del algoritmo incompletos", 48, "Error en el ingreso")
FrmMP.limpiar = True
Exit Sub
End If


'Vector de coeficientes
A = Pn(px)
' buscamos grado n del polinomio
T1 = 0
i = 0
Do While T1 = 0
T1 = A(20 - i)
i = i + 1
Loop

n = 20 - i + 1
gradoP = n
nr = n
' Dimensionamos matrices
ReDim b(n) As Double
ReDim c(n) As Double
ReDim Re(n) As Double
ReDim Im(n) As Double
ReDim Resulitera(0 To maxiter - 1, (2 * n) + 4) As Double

k = 0
bandera = True
If n < 3 Then
mensage = MsgBox("polinomio de grado menor que 3 ", 48, "Fracaso")
FrmMP.limpiar = True
Exit Sub
End If
Do
If n < 3 Then
mensage = MsgBox("Número de iteraciones máximas superado,no fue posible
encontrar la raíz del polinomio; Sugerencia: incremente el número de iteraciones",
48, "Iteraciones")
FrmMP.limpiar = True
Exit Sub
Else
```

```
Do
k = k + 1
b(n) = A(n)
b(n - 1) = A(n - 1) + r * b(n)
c(n) = b(n)
c(n - 1) = b(n - 1) + r * c(n)
For i = n - 2 To 0 Step -1
b(i) = A(i) + r * b(i + 1) + s * b(i + 2)
Next i
For i = n - 2 To 1 Step -1
c(i) = b(i) + r * c(i + 1) + s * c(i + 2)
Next i
det = c(2) * c(2) - c(3) * c(1)
   If det <> 0 Then
   dr = (-b(1) * c(2) + b(0) * c(3)) / det
   ds = (-b(0) * c(2) + b(1) * c(1)) / det
   r = r + dr
   s = s + ds
      If Opte.Value = True Then
      er = Abs(dr)
      es = Abs(ds)
      Else
      er = Abs(A(1) + r * b(2) + s * b(3))
      es = Abs(A(0) + r * b(1) + s * b(2))
      End If
   Else
   mensage = MsgBox("No es posible encontrar la raiz pruebe otro par de valores
r,s cercanos a la raiz", 48, "Fracaso")
   FrmMP.limpiar = True
   Exit Sub
   End If
If n = 3 Then
For i = 0 To n
Resulitera(k - 1, i) = b(i)
Next i
For i = n + 1 To (2 * n) + 1
Resulitera(k - 1, i) = c(i - n - 1)
Next i
Resulitera(k - 1, (2 * n) + 2) = det
Resulitera(k - 1, (2 * n) + 3) = r
Resulitera(k - 1, (2 * n) + 4) = s
End If


Loop Until ((er < T And es < T) Or k > maxiter)
```

```
    If k > maxiter Then
    mensage = MsgBox("Número de iteraciones máximas superado,no fue posible encontrar la raíz del polinomio; Sugerencia: incremente el número de iteraciones", 48, "Iteraciones")
    FrmMP.limpiar = True
    Exit Sub
    End If
    'Raízcuadratica
    disc = r * r + 4 * s
    If disc > 0 Then
    r1 = (r + Sqr(disc)) / 2
    r2 = (r - Sqr(disc)) / 2
    i1 = 0
    i2 = 0
    Else
    r1 = r / 2
    r2 = r1
    i1 = Sqr(Abs(disc)) / 2
    i2 = -i1
    End If
    Re(n) = r1
    Re(n - 1) = r2
    Im(n) = i1
    Im(n - 1) = i2
    n = n - 2
    For i = 0 To n
    A(i) = b(i + 2)
    Next i
       If k < maxiter Then
          If n = 2 Then
          r = -A(1) / A(2)
          s = -A(0) / A(2)
          ' Raízcuadratica
          disc = r * r + 4 * s
          If disc > 0 Then
          r1 = (r + Sqr(disc)) / 2
          r2 = (r - Sqr(disc)) / 2
          i1 = 0
          i2 = 0
          Else
          r1 = r / 2
          r2 = r1
          i1 = Sqr(Abs(disc)) / 2
          i2 = -i1
          End If
          Re(n) = r1
```

```
            Re(n - 1) = r2
            Im(n) = i1
            Im(n - 1) = i2
            bandera = False
            Else
               If n = 1 Then
               Re(n) = -A(0) / A(1)
               Im(0) = 0
               bandera = False
               End If
            End If
         End If
End If
k = 0
Loop While (bandera)
' Presentación de resultados
For i = 1 To nr
If Im(i) >= 0 Then
LstR.AddItem "r" & i & "= " & Re(i) & "+" & Im(i) & "i", i - 1
Else
LstR.AddItem "r" & i & "= " & Re(i) & "-" & Abs(Im(i)) & "i", i - 1
End If
Next i
Exit Sub
err1:

If aviso = False Then
   mensage = MsgBox("Número de iteraciones máximas superado,no fue posible
encontrar la raíz del polinomio; Sugerencia: incremente el número de iteraciones",
48, "Iteraciones")
FrmMP.limpiar = True
Else
aviso = False
End If

End Sub
Public Sub AplicacióndeNewtonBairstow()

Dim px As String
Dim T1 As Integer
Dim r As Double
Dim s As Double
Dim U As Double
Dim v As Double
Dim pr As Double
Dim ps As Double
```

```
Dim dpr As Double
Dim dps As Double
Dim i As Integer
Dim n As Integer
Dim Txtr As TextBox
Dim Txts As TextBox
Dim Txtp As TextBox
Dim LstR As ListBox
Dim maxiter As Integer
Dim mensage As String
Dim b() As Double
Dim c() As Double
Dim A() As Double

On Error GoTo err1
Set Txtr = FrmMP.TxtMPr
Set Txts = FrmMP.TxtMPs
Set Txtp = FrmMP.TxtMPpolinomio
Set LstR = FrmMP.LstMPresultados

If (Txtr = "" Or Txts = "" Or Txtp = "") Then
mensaje = MsgBox("Parámetros del algoritmo incompletos", 48, "Error en el ingreso")
Exit Sub
End If



'Asignación del punto a evaluar
U = Val(Txtr.Text)
v = Val(Txts.Text)
r = 2 * U
s = -(U * U) - (v * v)
px = Txtp.Text
'Vector de coeficientes
A = Pn(px)
' buscamos grado n del polinomio
T1 = 0
i = 0
Do While T1 = 0
T1 = A(20 - i)
i = i + 1
Loop

n = 20 - i + 1
nr = n
' Dimensionamos matrices
```

```
ReDim b(n) As Double
ReDim c(n) As Double


If n < 3 Then
mensage = MsgBox("polinomio de grado menor que 3 ", 48, "Fracaso")
Exit Sub
Else
' Implementamos el algoritmo
   b(n) = A(n)
   b(n - 1) = A(n - 1) + r * b(n)
   c(n) = b(n)
   c(n - 1) = b(n - 1) + r * c(n)
   For i = n - 2 To 0 Step -1
   b(i) = A(i) + r * b(i + 1) + s * b(i + 2)
   Next i
   For i = n - 2 To 1 Step -1
   c(i) = b(i) + r * c(i + 1) + s * c(i + 2)
   Next i
End If

' Presentación de resultados
pr = b(0) - U * b(1)
ps = v * b(1)
dr = b(1) - (2 * v * v * c(3))
ds = (2 * v * c(2)) - ((2 * v * c(2)) - (2 * U * v * c(3)))
If ps >= 0 Then
LstR.AddItem "P" & n & "(Xo)=" & pr & "+" & ps & "i", 0
Else
LstR.AddItem "P" & n & "(Xo)=" & pr & ps & "i", 0
End If
If ds >= 0 Then
LstR.AddItem "P'" & n & "(Xo)=" & dr & "+" & ds & "i", 1
Else
LstR.AddItem "P'" & n & "(Xo)=" & dr & ds & "i", 1
End If
Exit Sub
err1:
If aviso = False Then
   mensaje = MsgBox("Función no ingresada según sintaxis o Parámetros del
algoritmo incompletos", 48, "Error en el ingreso")
Else
aviso = False
End If

End Sub
```

```
Public Function Pn(p As String) As Double()
Dim Coeficiente(20) As Double
Dim s As Integer
Dim r As Integer
Dim r1 As Integer
Dim L As Integer
Dim lugar As Integer
Dim lugarX As Integer
Dim lugarE As Integer
Dim cadena As String
Dim coe As Double
Dim Exp As Integer
Dim Flag As Boolean
Dim i As Integer
Dim T As Double
Dim resta As Boolean
Dim suma As Boolean
Dim signo As Integer
Dim su As Integer
Dim Re As Integer
Dim mensaje As String
On Error GoTo err1
' inicializamos con ceros el vector de coeficientes
For i = 0 To 20
Coeficiente(i) = 0
Next i
Flag = False

p = LTrim(p) ' quitamos los espacios iniciales
T = 0
'Buscamos el ligar donde se encuentra un + o -
s = InStr(T + 1, p, "+", 1)
r = InStr(T + 1, p, "-", 1)
' Caso cuando todos los signos son negativos
If Val(s) = 0 Then
 ' Si inica con el signo - lo quitamos para que cumpla el algoritmo
' e inicializamos Flag para restaurar el signo
 If Val(r) = 1 Then
 p = Mid(p, 2, Len(p))
 Flag = True
 End If
 If Val(r) <> 1 Then
 r = 0
 lugar = 1
```

```
Do While lugar <> 0
lugar = InStr(r + 1, p, "-", 1)
If lugar = 0 Then
lugar = Val(Len(p)) + 1
End If
'recuperamos el terminao del polinomoi hasta donde esta el signo menos
cadena = Mid(p, r + 1, lugar - r - 1)
L = Len(cadena)
'Encontramos la posición de X y del esponente
lugarX = InStr(1, cadena, "X", 1)
lugarE = InStr(1, cadena, "^", 1)
  ' Asignamos los coeficientes a al vector de coeficientes
  If lugarX <> 0 Then
     If lugarE <> 0 Then
        If L = 3 Then
        coe = -1
        Exp = Val(Right(cadena, L - lugarE))
        Coeficiente(Exp) = coe
        Else
        coe = -1 * Val(Left(cadena, lugarX - 1))
        Exp = Val(Right(cadena, L - lugarE))
        Coeficiente(Exp) = coe
        End If
     Else
        If L = 1 Then
        coe = -1
        Exp = 1
        Coeficiente(Exp) = coe

        Else
        coe = -1 * Val(Left(cadena, lugarX - 1))
        Exp = 1
        Coeficiente(Exp) = coe

        End If
     End If
  Else
  coe = -1 * Val(cadena)
  Exp = 0
  Coeficiente(Exp) = coe
  lugar = 0
  End If
r = lugar
Loop
' Restaura el signo - quitado al principio
If Flag = False Then
```

```
T = 0
i = 0
Do While T = 0
T = Coeficiente(20 - i)
i = i + 1
Loop
Coeficiente(20 - i + 1) = -1 * T
End If


End If

Else
   ' Caso de todos los signos positivos en el polinomio
   If Val(r) = 0 Then
   lugar = 1
   Do While lugar <> 0
   lugar = InStr(r + 1, p, "+", 1)
   If lugar = 0 Then
   lugar = Val(Len(p)) + 1
   End If
   cadena = Mid(p, r + 1, lugar - r - 1)
   L = Len(cadena)
   lugarX = InStr(1, cadena, "X", 1)
   lugarE = InStr(1, cadena, "^", 1)
      If lugarX <> 0 Then
         If lugarE <> 0 Then
            If L = 3 Then
            coe = 1
            Exp = Val(Right(cadena, L - lugarE))
            Coeficiente(Exp) = coe
            Else
            coe = Val(Left(cadena, lugarX - 1))
            Exp = Val(Right(cadena, L - lugarE))
            Coeficiente(Exp) = coe
            End If
         Else
            If L = 1 Then
            coe = 1
            Exp = 1
            Coeficiente(Exp) = coe
            Else
            coe = Val(Left(cadena, lugarX - 1))
            Exp = 1
            Coeficiente(Exp) = coe
            End If
```

```
        End If
    Else
    coe = Val(cadena)
    Exp = 0
    Coeficiente(Exp) = coe
    lugar = 0
    End If
r = lugar
Loop

Else
' Caso de signos diferentes en el polinomio
su = Val(s)
Re = Val(r)
resta = False
suma = True
    ' Quitamos el signo - del polinomio si empieza con el mismo
    If Val(r) = 1 Then
    p = Mid(p, 2, Len(p))
    Flag = True
    su = InStr(s + 1, p, "+", 1)
    Re = InStr(s + 1, p, "-", 1)
    'Inicializamos las varibles que nos indican que signo es
    'el que se debe poner en los coeficientes
    resta = True
    suma = False
    r = 0
    End If
    r = 0
    Do While (Val(su) Or Val(Re)) <> 0
        'Caso si se encuentra primero un signo - que el signo +
        If Val(su) > Val(Re) Then
            ' seleccion del signo del coeficiente
            If (resta = True And suma = False) Then
            signo = -1
            End If
            If (resta = False And suma = True) Then
            signo = 1
            resta = True
            suma = False
            End If
            lugar = InStr(r + 1, p, "-", 1)
            If lugar = 0 Then
            lugar = Val(Len(p)) + 1
            End If
            'recuperación de el coeficiente y su exponente
```

```
'asignandoles a la matriz de coeficientes
cadena = Mid(p, r + 1, lugar - r - 1)
L = Len(cadena)
lugarX = InStr(1, cadena, "X", 1)
lugarE = InStr(1, cadena, "^", 1)
If lugarX <> 0 Then
   If lugarE <> 0 Then
      If L = 3 Then
       coe = 1
       Exp = Val(Right(cadena, L - lugarE))
       Coeficiente(Exp) = signo * coe
       Else
       coe = Val(Left(cadena, lugarX - 1))
       Exp = Val(Right(cadena, L - lugarE))
       Coeficiente(Exp) = signo * coe
       End If
    Else
      If L = 1 Then
      coe = 1
      Exp = 1
      Coeficiente(Exp) = signo * coe
      Else
      coe = Val(Left(cadena, lugarX - 1))
      Exp = 1
      Coeficiente(Exp) = signo * coe
      End If
  End If
Else
coe = Val(cadena)
Exp = 0
Coeficiente(Exp) = signo * coe
lugar = 0
End If
r = lugar
s = su
T = Re
su = InStr(r + 1, p, "+", 1)
Re = InStr(r + 1, p, "-", 1)
  If su = 0 Then
     If Re <> 0 Then
     su = Re + 1
     End If
  Else
     If Re = 0 Then
        If su <> 0 Then
        Re = su + 1
```

```
            End If
          End If
        End If
  ' Obtención del ultimo término del polinomio
  If (Val(su) Or Val(Re)) = 0 Then
  cadena = Mid(p, r + 1, Len(p) - r)
  coe = Val(cadena)
  Exp = 0
  If T < s Then
  signo = -1
  Else
  signo = 1
  End If
  Coeficiente(Exp) = signo * coe
  End If


Else
  ' Caso cuando el signo encontrado primero es un +
  If (suma = True And resta = False) Then
  signo = 1
  End If
  If (suma = False And resta = True) Then
  signo = -1
  suma = True
  resta = False
  End If
  lugar = InStr(r + 1, p, "+", 1)
  s = lugar
  If lugar = 0 Then
  lugar = Val(Len(p)) + 1
  End If
  cadena = Mid(p, r + 1, lugar - r - 1)
  L = Len(cadena)
  lugarX = InStr(1, cadena, "X", 1)
  lugarE = InStr(1, cadena, "^", 1)
  If lugarX <> 0 Then
     If lugarE <> 0 Then
        If L = 3 Then
        coe = 1
        Exp = Val(Right(cadena, L - lugarE))
        Coeficiente(Exp) = signo * coe
        Else
        coe = Val(Left(cadena, lugarX - 1))
        Exp = Val(Right(cadena, L - lugarE))
        Coeficiente(Exp) = signo * coe
```

```
        End If
    Else
        If L = 1 Then
        coe = 1
        Exp = 1
        Coeficiente(Exp) = signo * coe
        Else
        coe = Val(Left(cadena, lugarX - 1))
        Exp = 1
        Coeficiente(Exp) = signo * coe
        End If
    End If
  Else
  coe = Val(cadena)
  Exp = 0
  Coeficiente(Exp) = signo * coe
  lugar = 0
  End If
  r = lugar
  s = su
  T = Re
  su = InStr(r + 1, p, "+", 1)
  Re = InStr(r + 1, p, "-", 1)
  If su = 0 Then
        If Re <> 0 Then
        su = Re + 1
        End If
    Else
        If Re = 0 Then
            If su <> 0 Then
            Re = su + 1
            End If
        End If
    End If
If (Val(su) Or Val(Re)) = 0 Then
cadena = Mid(p, r + 1, Len(p) - r)
coe = Val(cadena)
Exp = 0
If T < s Then
signo = -1
Else
signo = 1
End If
Coeficiente(Exp) = signo * coe
End If
```

```
        End If
        Loop
        ' Restauraqción del signo - sustraido
        If Flag = True Then
        T = 0
        i = 0
        Do While T = 0
        T = Coeficiente(20 - i)
        i = i + 1
        Loop
        Coeficiente(20 - i + 1) = -1 * T
    End If
End If
End If
'Asignación de la matriz de coeficientes a la variables de retorno
' de la función Pn
Pn = Coeficiente

Exit Function
err1:
mensaje = MsgBox("Grado del polinomio fuera de rango", 48, "Error en el ingreso")
aviso = True
End Function
```

Calcular la derivada de una función

Interfaz FrmDerivación

```
Option Explicit
'En Microsoft TechNet puedes encontrar este artículo:
'HOWTO: Use HTML Help API in a Visual Basic 5.0 Application
'PSS ID Number: Q183434
'
'Aunque la definición de la Enumeración y la primera declaración
'es de las news
'
'Htmlhelp consts
Private Enum HH_COMMAND
    HH_DISPLAY_TOPIC = &H0
    HH_HELP_FINDER = &H0        ' WinHelp equivalent
    HH_DISPLAY_TOC = &H1        ' not currently implemented
    HH_DISPLAY_INDEX = &H2      ' not currently implemented
    HH_DISPLAY_SEARCH = &H3     ' not currently implemented
```

```
    HH_SET_WIN_TYPE = &H4
    HH_GET_WIN_TYPE = &H5
    HH_GET_WIN_HANDLE = &H6
    HH_GET_INFO_TYPES = &H7     ' not currently implemented
    HH_SET_INFO_TYPES = &H8     ' not currently implemented
    HH_SYNC = &H9
    HH_ADD_NAV_UI = &HA         ' not currently implemented
    HH_ADD_BUTTON = &HB         ' not currently implemented
    HH_GETBROWSER_APP = &HC     ' not currently implemented
    HH_KEYWORD_LOOKUP = &HD
    HH_DISPLAY_TEXT_POPUP = &HE ' display string resource id
                        ' or text in a popup window
    HH_HELP_CONTEXT = &HF       ' display mapped numeric value
                        ' in dwData
    HH_TP_HELP_CONTEXTMENU      ' Text pop-up help, similar to
                        ' WinHelp's HELP_CONTEXTMENU.
    HH_TP_HELP_WM_HELP = &H11   ' text pop-up help, similar to
                        ' WinHelp's HELP_WM_HELP.
    HH_CLOSE_ALL = &H12         ' close all windows opened directly
                        ' or indirectly by the caller
    HH_ALINK_LOOKUP = &H13      ' ALink version of HH_KEYWORD_LOOKUP
End Enum

'HtmlHelp api call
'NOTA: Si se usa esta forma, hay que indicar el último parámetro
'     con la palabra ByVal delante...
'Private Declare Function HtmlHelp Lib "hhctrl.ocx" Alias "HtmlHelpA" _
    (ByVal hwndCaller As Long, ByVal pszFile As String, _
    ByVal uCommand As HH_COMMAND, dwData As Any) As Long
'Con esta funciona perfectamente
Private Declare Function HtmlHelp Lib "hhctrl.ocx" Alias "HtmlHelpA" _
    (ByVal hwndCaller As Long, ByVal pszFile As String, _
    ByVal uCommand As HH_COMMAND, ByVal dwData As Long) As Long

Dim exval As New McsExcel
Dim derivación As New ClasDerivación  'Creación de la insatancia ClasDerivación
Dim n As Double                 ' número de incrementos h
Dim tipoalg As Integer          'tipo de algoritmo
Dim arribaSstab
Dim izquierdaSstab
Dim anchoSstab
Dim altoSstab
Dim colororg
Private Acepta As Boolean
Private VLConexion As ADODB.Connection
Private VLRegistro As ADODB.Recordset
```

```
Private VLNombre As String
Private VLNombreTrabajo As String
Private VLVerifica As Boolean
Private VLBDD As Boolean
Private VLJacobiano As String
Private VLJacobianoMatriz(1 To 1000) As Double
Private VLDatos As String
Private VLDatosMatriz(1 To 1000) As Double
Private VLIndice As Integer
Private VLRec As Boolean

Property Let Aceptar(parAceptar As Boolean)
   Acepta = parAceptar
End Property

Property Let Nombre(parNombre As String)
   VLNombre = parNombre
End Property

Property Let NombreTrabajo(parNombreTrabajo As String)
   VLNombreTrabajo = parNombreTrabajo
End Property

Private Sub CmdDeriCalcular_Click()
Dim mensaje As String
Dim verifica As Double

If OptDeriFunción.Value = True Then
   If (TxtDeriFunción = " " Or TxtDeriFunción = " " Or TxtDeriPunto = "" Or
TxtDeriPunto = " " Or _
      TxtDeriNumincrementos = "" Or TxtDeriNumincrementos = " " Or TxtDeriEdit2 =
"" Or TxtDeriEdit2 = " ") Then
      mensaje = MsgBox("Parámetros del algortimo incompletos", 48, "Error en el
ingreso")
      Exit Sub
   End If
Else
   If (TxtDeriEdit2 = "" Or TxtDeriEdit2 = " ") Then
      mensaje = MsgBox("Parámetros del algortimo incompletos", 48, "Error en el
ingreso")
      Exit Sub
   End If
End If


      exval.Tipox = False
```

```
        exval.fa = TxtDeriFunción
        exval.PtX = 500
        exval.evaluarf
        verifica = exval.Fdy
        If verifica = 8121975 Then
           mensaje = MsgBox("Debe ingresar la función de acuerdo a la sintaxis
especificada", 16, "Error en el ingreso")
           Exit Sub
        End If

SetearDerivación
Select Case tipoalg
Case Is = 1
derivación.DiferenciasCentradas
DeriPresentar
Case Is = 2
derivación.DiferenciasCentradas
DeriPresentar
Case Is = 3
derivación.DiferenciasProgresivas
DeriPresentar
Case Is = 4
derivación.DiferenciasProgresivas
DeriPresentar
Case Is = 5
derivación.DiferenciasRegresivas
DeriPresentar
Case Is = 6
derivación.DiferenciasRegresivas
DeriPresentar
End Select
Deriimprimir.Enabled = True

Deriguardarcomo.Enabled = True

End Sub

Private Sub Command1_Click()
   PLImprimir
End Sub

Private Sub Deriabrir_Click()
   FrmRecuperaTrabajo.Show vbModal
   If Acepta = True Then
      PLConexion True
      PLRecupera
```

```
    End If
End Sub

Private Sub Deriayuda_Click()
'Así se llamaría para mostrar un tópico de la ayuda
    Dim h As Long

    'h = HtmlHelp(Me.hWnd, "Milton.hlp", HH_DISPLAY_TOPIC, 0&)

    h = HtmlHelp(Me.hWnd, "Milton.hlp", HH_HELP_CONTEXT, 2060&)
End Sub

'Manejo de la elección del tipo de lagoritmo
Private Sub Dericentrada1_Click()
'Seteamos la variable tipoalg para indicar que tipo de algoritmos es
tipoalg = 1
    Deriabrir.Enabled = False
    Deriguardarcomo.Enabled = False
    Deriimprimir.Enabled = False
    FrmRecuperaTrabajo.ver = tipoalg

' Presentamos todos los contorles necesarios en el formulario par elegir función o
datos
FraDeritipo.Visible = True
OptDeriFunción.Value = False
OptDeriFunción.Visible = True
OptDeriDatos.Value = False
OptDeriDatos.Visible = True
FraDeriTipoerror.Visible = False
OptDeriOh2.Visible = False
OptDeriOh4.Visible = False
LblDeriFunción.Visible = False
TxtDeriFunción.Text = " "
TxtDeriFunción.Visible = False
LblDeriPunto.Visible = False
TxtDeriPunto.Text = " "
TxtDeriPunto.Visible = False
TxtDeriNumincrementos.Text = " "
LblDeriNumincrementos.Visible = False
TxtDeriNumincrementos.Visible = False
CheckDeriError.Visible = False
FraDeriError.Visible = False
LblDeriReal.Visible = False
TxtDeriReal.Visible = False
LblDerititulo1.Visible = False
Fg2Deri.Clear
```

```
Fg2Deri.Visible = False
LblDeriTitulo2.Visible = False
Fg3Deri.Clear
Fg3Deri.Visible = False
fgDeriresultados.Clear
LblNomalg.Caption = "Diferencias centradas primera derivada"
LblDeriSintaxis.Caption = " Función= derivación de una función; Datos= derivación de
datos" & Chr(10) & "f(x)=función de la cual se obtendrá la derivada exp(-
1*X^2)*(SIN(X)/7)" & Chr(10) & "Puede usar las funciones:
Ln(X);COS(X);SQRT(X);Log(X); Tan(X); PI(), el resto de funciones sigue la sintaxis
de Excel" & Chr(10) & "O(h);O(h2); O(h4)= orden del error en la fórmula usada para
la derivación " & Chr(10) & _
"Punto de derivación= punto en el cual se desea hallar la derivada" & Chr(10) &
"Número de incrementos= número natural, representa el número de incrementos
para los cuales se calculará la derivada" & Chr(10) & "h= espaciamiento entre los
puntos usados en la formula para la integración" & Chr(10) & "Calcular error= permite
calcular el error absoluto si se conoce el valor de la integral real" & Chr(10) & _
" f-2, f-1, etc = Valor de la ordenada (función) en las abscisas: f-2= Punto de
derivación-2h; f-1= Punto de derivación-h; fo= Punto de derivación; f1= Punto de
derivación + h; f2= Punto de derivación + 2h"

End Sub


Private Sub Dericentrada2_Click()
'Seteamos la variable tipoalg para indicar que tipo de algoritmos es
tipoalg = 2
   Deriabrir.Enabled = False
   Deriguardarcomo.Enabled = False
   Deriimprimir.Enabled = False
   FrmRecuperaTrabajo.ver = tipoalg

' Presentamos todos los contorles necesarios en el formulario par elegir función o
datos
FraDeritipo.Visible = True
OptDeriFunción.Value = False
OptDeriFunción.Visible = True
OptDeriDatos.Value = False
OptDeriDatos.Visible = True
FraDeriTipoerror.Visible = False
OptDeriOh2.Visible = False
OptDeriOh4.Visible = False
LblDeriFunción.Visible = False
TxtDeriFunción.Text = " "
TxtDeriFunción.Visible = False
LblDeriPunto.Visible = False
```

```
TxtDeriPunto.Text = " "
TxtDeriPunto.Visible = False
TxtDeriNumincrementos.Text = " "
LblDeriNumincrementos.Visible = False
TxtDeriNumincrementos.Visible = False
CheckDeriError.Visible = False
FraDeriError.Visible = False
LblDeriReal.Visible = False
TxtDeriReal.Visible = False
LblDerititulo1.Visible = False
Fg2Deri.Clear
Fg2Deri.Visible = False
LblDeriTitulo2.Visible = False
Fg3Deri.Clear
Fg3Deri.Visible = False
fgDeriresultados.Clear
LblNomalg.Caption = "Diferencias centradas segunda derivada"
LblDeriSintaxis.Caption = " Función= derivación de una función; Datos= derivación de
datos" & Chr(10) & "f(x)=función de la cual se obtendrá la derivada exp(-
1*X^2)*(SIN(X)/7)" & Chr(10) & "Puede usar las funciones:
Ln(X);COS(X);SQRT(X);Log(X); Tan(X); PI(), el resto de funciones sigue la sintaxis
de Excel" & Chr(10) & "O(h);O(h2); O(h4)= orden del error en la fórmula usada para
la derivación " & Chr(10) & _
"Punto de derivación= punto en el cual se desea hallar la derivada" & Chr(10) &
"Número de incrementos= número natural, representa el número de incrementos
para los cuales se calculará la derivada" & Chr(10) & "h= espaciamiento entre los
puntos usados en la formula para la integración" & Chr(10) & "Calcular error= permite
calcular el error absoluto si se conoce el valor de la integral real" & Chr(10) & _
" f-2, f-1, etc = Valor de la ordenada (función) en las abscisas: f-2= Punto de
derivación-2h; f-1= Punto de derivación-h; fo= Punto de derivación; f1= Punto de
derivación + h; f2= Punto de derivación + 2h"
End Sub



Private Sub Derigraficar_Click()
Frmescalas.Show
End Sub

Private Sub Deriguardarcomo_Click()
   FrmNombre.Show vbModal
   If Acepta = True Then
      PLConexion True
      PLGuardarBDD
   End If
End Sub
```

```
Private Sub Deriimprimir_Click()
Dim EndTime As Date
   Sincolor

      EndTime = DateAdd("s", 1 / 1E+16, Now)
      Do Until Now > EndTime
        DoEvents
      Loop
   'Form1.Command3_Click
   Set Form1.Picture1.Picture = CaptureClient(Me)
   Form1.Visible = True
   FrmPrincipalalgoritmos.Visible = False
   Concolor
End Sub

Private Sub Deriprogresivas1_Click()
'Seteamos la variable tipoalg para indicar que tipo de algoritmos es
tipoalg = 3
   Deriabrir.Enabled = False
   Deriguardarcomo.Enabled = False
   Deriimprimir.Enabled = False
   FrmRecuperaTrabajo.ver = tipoalg

' Presentamos todos los contorles necesarios en el formulario par elegir función o
datos
FraDeritipo.Visible = True
OptDeriFunción.Value = False
OptDeriFunción.Visible = True
OptDeriDatos.Value = False
OptDeriDatos.Visible = True
FraDeriTipoerror.Visible = False
OptDeriOh2.Visible = False
OptDeriOh4.Visible = False
LblDeriFunción.Visible = False
TxtDeriFunción.Text = " "
TxtDeriFunción.Visible = False
LblDeriPunto.Visible = False
TxtDeriPunto.Text = " "
TxtDeriPunto.Visible = False
TxtDeriNumincrementos.Text = " "
LblDeriNumincrementos.Visible = False
TxtDeriNumincrementos.Visible = False
CheckDeriError.Visible = False
FraDeriError.Visible = False
LblDeriReal.Visible = False
```

```
TxtDeriReal.Visible = False
LblDerititulo1.Visible = False
Fg2Deri.Clear
Fg2Deri.Visible = False
LblDeriTitulo2.Visible = False
Fg3Deri.Clear
Fg3Deri.Visible = False
fgDeriresultados.Clear
LblNomalg.Caption = "Diferencias progresivas primera derivada"
LblDeriSintaxis.Caption = " Función= derivación de una función; Datos= derivación de
datos" & Chr(10) & "f(x)=función de la cual se obtendrá la derivada exp(-
1*X^2)*(SIN(X)/7)" & Chr(10) & "Puede usar las funciones:
Ln(X);COS(X);SQRT(X);Log(X); Tan(X); PI(), el resto de funciones sigue la sintaxis
de Excel" & Chr(10) & "O(h);O(h2); O(h4)= orden del error en la fórmula usada para
la derivación " & Chr(10) & _
"Punto de derivación= punto en el cual se desea hallar la derivada" & Chr(10) &
"Número de incrementos= número natural, representa el número de incrementos
para los cuales se calculará la derivada" & Chr(10) & "h= espaciamiento entre los
puntos usados en la formula para la integración" & Chr(10) & "Calcular error= permite
calcular el error absoluto si se conoce el valor de la integral real" & Chr(10) & _
" f-2, f-1, etc = Valor de la ordenada (función) en las abscisas: f-2= Punto de
derivación-2h; f-1= Punto de derivación-h; fo= Punto de derivación; f1= Punto de
derivación + h; f2= Punto de derivación + 2h"
End Sub

Private Sub Deriprogresivas2_Click()
'Seteamos la variable tipoalg para indicar que tipo de algoritmos es
tipoalg = 4
   Deriabrir.Enabled = False
   Deriguardarcomo.Enabled = False
   Deriimprimir.Enabled = False
   FrmRecuperaTrabajo.ver = tipoalg

' Presentamos todos los contorles necesarios en el formulario par elegir función o
datos
FraDeritipo.Visible = True
OptDeriFunción.Value = False
OptDeriFunción.Visible = True
OptDeriDatos.Value = False
OptDeriDatos.Visible = True
FraDeriTipoerror.Visible = False
OptDeriOh2.Visible = False
OptDeriOh4.Visible = False
LblDeriFunción.Visible = False
TxtDeriFunción.Text = " "
TxtDeriFunción.Visible = False
```

```
LblDeriPunto.Visible = False
TxtDeriPunto.Text = " "
TxtDeriPunto.Visible = False
TxtDeriNumincrementos.Text = " "
LblDeriNumincrementos.Visible = False
TxtDeriNumincrementos.Visible = False
CheckDeriError.Visible = False
FraDeriError.Visible = False
LblDeriReal.Visible = False
TxtDeriReal.Visible = False
LblDerititulo1.Visible = False
Fg2Deri.Clear
Fg2Deri.Visible = False
LblDeriTitulo2.Visible = False
Fg3Deri.Clear
Fg3Deri.Visible = False
fgDeriresultados.Clear
LblNomalg.Caption = "Diferencias progresivas segunda derivada"
LblDeriSintaxis.Caption = " Función= derivación de una función; Datos= derivación de
datos" & Chr(10) & "f(x)=función de la cual se obtendrá la derivada exp(-
1*X^2)*(SIN(X)/7)" & Chr(10) & "Puede usar las funciones:
Ln(X);COS(X);SQRT(X);Log(X); Tan(X); PI(), el resto de funciones sigue la sintaxis
de Excel" & Chr(10) & "O(h);O(h2); O(h4)= orden del error en la fórmula usada para
la derivación " & Chr(10) & _
"Punto de derivación= punto en el cual se desea hallar la derivada" & Chr(10) &
"Número de incrementos= número natural, representa el número de incrementos
para los cuales se calculará la derivada" & Chr(10) & "h= espaciamiento entre los
puntos usados en la formula para la integración" & Chr(10) & "Calcular error= permite
calcular el error absoluto si se conoce el valor de la integral real" & Chr(10) & _
" f-2, f-1, etc = Valor de la ordenada (función) en las abscisas: f-2= Punto de
derivación-2h; f-1= Punto de derivación-h; fo= Punto de derivación; f1= Punto de
derivación + h; f2= Punto de derivación + 2h"
End Sub

Private Sub Deriregresiva1_Click()
'Seteamos la variable tipoalg para indicar que tipo de algoritmos es
tipoalg = 5
   Deriabrir.Enabled = False
   Deriguardarcomo.Enabled = False
   Deriimprimir.Enabled = False
   FrmRecuperaTrabajo.ver = tipoalg

' Presentamos todos los contorles necesarios en el formulario par elegir función o
datos
FraDeritipo.Visible = True
OptDeriFunción.Value = False
```

```
OptDeriFunción.Visible = True
OptDeriDatos.Value = False
OptDeriDatos.Visible = True
FraDeriTipoerror.Visible = False
OptDeriOh2.Visible = False
OptDeriOh4.Visible = False
LblDeriFunción.Visible = False
TxtDeriFunción.Text = " "
TxtDeriFunción.Visible = False
LblDeriPunto.Visible = False
TxtDeriPunto.Text = " "
TxtDeriPunto.Visible = False
TxtDeriNumincrementos.Text = " "
LblDeriNumincrementos.Visible = False
TxtDeriNumincrementos.Visible = False
CheckDeriError.Visible = False
FraDeriError.Visible = False
LblDeriReal.Visible = False
TxtDeriReal.Visible = False
LblDerititulo1.Visible = False
Fg2Deri.Clear
Fg2Deri.Visible = False
LblDeriTitulo2.Visible = False
Fg3Deri.Clear
Fg3Deri.Visible = False
fgDeriresultados.Clear
LblNomalg.Caption = "Diferencias regresivas primera derivada"
LblDeriSintaxis.Caption = " Función= derivación de una función; Datos= derivación de
datos" & Chr(10) & "f(x)=función de la cual se obtendrá la derivada exp(-
1*X^2)*(SIN(X)/7)" & Chr(10) & "Puede usar las funciones:
Ln(X);COS(X);SQRT(X);Log(X); Tan(X); PI(), el resto de funciones sigue la sintaxis
de Excel" & Chr(10) & "O(h);O(h2); O(h4)= orden del error en la fórmula usada para
la derivación " & Chr(10) & _
"Punto de derivación= punto en el cual se desea hallar la derivada" & Chr(10) &
"Número de incrementos= número natural, representa el número de incrementos
para los cuales se calculará la derivada" & Chr(10) & "h= espaciamiento entre los
puntos usados en la formula para la integración" & Chr(10) & "Calcular error= permite
calcular el error absoluto si se conoce el valor de la integral real" & Chr(10) & _
" f-2, f-1, etc = Valor de la ordenada (función) en las abscisas: f-2= Punto de
derivación-2h; f-1= Punto de derivación-h; fo= Punto de derivación; f1= Punto de
derivación + h; f2= Punto de derivación + 2h"
End Sub

Private Sub Deriregresiva2_Click()
'Seteamos la variable tipoalg para indicar que tipo de algoritmos es
tipoalg = 6
```

```
    Deriabrir.Enabled = False
    Deriguardarcomo.Enabled = False
    Deriimprimir.Enabled = False
    FrmRecuperaTrabajo.ver = tipoalg

' Presentamos todos los contorles necesarios en el formulario par elegir función o
datos
FraDeritipo.Visible = True
OptDeriFunción.Value = False
OptDeriFunción.Visible = True
OptDeriDatos.Value = False
OptDeriDatos.Visible = True
FraDeriTipoerror.Visible = False
OptDeriOh2.Visible = False
OptDeriOh4.Visible = False
LblDeriFunción.Visible = False
TxtDeriFunción.Text = " "
TxtDeriFunción.Visible = False
LblDeriPunto.Visible = False
TxtDeriPunto.Text = " "
TxtDeriPunto.Visible = False
TxtDeriNumincrementos.Text = " "
LblDeriNumincrementos.Visible = False
TxtDeriNumincrementos.Visible = False
CheckDeriError.Visible = False
FraDeriError.Visible = False
LblDeriReal.Visible = False
TxtDeriReal.Visible = False
LblDerititulo1.Visible = False
Fg2Deri.Clear
Fg2Deri.Visible = False
LblDeriTitulo2.Visible = False
Fg3Deri.Clear
Fg3Deri.Visible = False
fgDeriresultados.Clear
LblNomalg.Caption = "Diferencias regresivas segunda derivada"
LblDeriSintaxis.Caption = " Función= derivación de una función; Datos= derivación de
datos" & Chr(10) & "f(x)=función de la cual se obtendrá la derivada exp(-
1*X^2)*(SIN(X)/7)" & Chr(10) & "Puede usar las funciones:
Ln(X);COS(X);SQRT(X);Log(X); Tan(X); PI(), el resto de funciones sigue la sintaxis
de Excel" & Chr(10) & "O(h);O(h2); O(h4)= orden del error en la fórmula usada para
la derivación " & Chr(10) & _
"Punto de derivación= punto en el cual se desea hallar la derivada" & Chr(10) &
"Número de incrementos= número natural, representa el número de incrementos
para los cuales se calculará la derivada" & Chr(10) & "h= espaciamiento entre los
```

```
puntos usados en la formula para la integración" & Chr(10) & "Calcular error= permite
calcular el error absoluto si se conoce el valor de la integral real" & Chr(10) & _
" f-2, f-1, etc = Valor de la ordenada (función) en las abscisas: f-2= Punto de
derivación-2h; f-1= Punto de derivación-h; fo= Punto de derivación; f1= Punto de
derivación + h; f2= Punto de derivación + 2h"
End Sub

Private Sub Derisalir_Click()
Unload Me
End Sub

Private Sub Form_Load()
   Deriabrir.Enabled = False
   Deriguardarcomo.Enabled = False
   Deriimprimir.Enabled = False
   VGForma = "Derivacion"
End Sub

'Presentamos los controles necesrios para obtener la derivada de Datos
Private Sub OptDeriDatos_Click()
LblDeriFunción.Visible = False
TxtDeriFunción.Text = " "
TxtDeriFunción.Visible = False
LblDeriPunto.Visible = False
TxtDeriPunto.Text = " "
TxtDeriPunto.Visible = False
LblDeriNumincrementos.Visible = False
TxtDeriNumincrementos.Text = "1"
TxtDeriNumincrementos.Visible = False
CheckDeriError.Value = 0
CheckDeriError.Visible = True
LblDeriTitulo2.Visible = True


Select Case tipoalg
'si a elegido formula de diferencias centradas
 'primera derivada
Case Is = 1
FraDeriTipoerror.Visible = True
OptDeriOh2.Caption = "O(h2)"
OptDeriOh2.Value = False
OptDeriOh2.Visible = True
OptDeriOh4.Caption = "O(h4)"
OptDeriOh4.Value = False
OptDeriOh4.Visible = True
```

```
'segunda derivada
Case Is = 2
FraDeriTipoerror.Visible = True
OptDeriOh2.Caption = "O(h2)"
OptDeriOh2.Value = False
OptDeriOh2.Visible = True
OptDeriOh4.Caption = "O(h4)"
OptDeriOh4.Value = False
OptDeriOh4.Visible = True

'si a elegido formula de diferenciación progresiva
 'primera derivada
Case Is = 3
FraDeriTipoerror.Visible = True
OptDeriOh2.Caption = "O(h)"
OptDeriOh2.Value = False
OptDeriOh2.Visible = True
OptDeriOh4.Caption = "O(h2)"
OptDeriOh4.Value = False
OptDeriOh4.Visible = True

 'segunda derivada
Case Is = 4
FraDeriTipoerror.Visible = True
OptDeriOh2.Caption = "O(h)"
OptDeriOh2.Value = False
OptDeriOh2.Visible = True
OptDeriOh4.Caption = "O(h2)"
OptDeriOh4.Value = False
OptDeriOh4.Visible = True

'si ha elegido formula de diferenicación regresiva
 'primera derivada
Case Is = 5
FraDeriTipoerror.Visible = True
OptDeriOh2.Caption = "O(h)"
OptDeriOh2.Value = False
OptDeriOh2.Visible = True
OptDeriOh4.Caption = "O(h2)"
OptDeriOh4.Value = False
OptDeriOh4.Visible = True

' segunda derivda
Case Is = 6
FraDeriTipoerror.Visible = True
OptDeriOh2.Caption = "O(h)"
```

```
OptDeriOh2.Value = False
OptDeriOh2.Visible = True
OptDeriOh4.Caption = "O(h2)"
OptDeriOh4.Value = False
OptDeriOh4.Visible = True

End Select

   Deriabrir.Enabled = True
   Deriguardarcomo.Enabled = True
   FrmRecuperaTrabajo.ver = tipoalg

End Sub

'Presentamos los controles necesrios para obtener la derivada de la función
Private Sub OptDeriFunción_Click()

LblDeriFunción.Visible = True
TxtDeriFunción.Text = " "
TxtDeriFunción.Visible = True
LblDeriPunto.Visible = True
TxtDeriPunto.Text = " "
TxtDeriPunto.Visible = True
LblDeriNumincrementos.Visible = True
TxtDeriNumincrementos.Text = " "
TxtDeriNumincrementos.Visible = True
CheckDeriError.Value = 0
CheckDeriError.Visible = True
LblDerititulo1.Visible = False
LblDeriTitulo2.Visible = False
Fg2Deri.Clear
Fg2Deri.Visible = False
Fg3Deri.Clear
Fg3Deri.Visible = False

'Se setea el titulo del orden del error dependiendo del tipo de algoritmo
Select Case tipoalg
'si a elegido formula de diferencias centradas
 'primera derivada
Case Is = 1
FraDeriTipoerror.Visible = True
OptDeriOh2.Caption = "O(h2)"
OptDeriOh2.Value = False
OptDeriOh2.Visible = True
OptDeriOh4.Caption = "O(h4)"
OptDeriOh4.Value = False
```

```
OptDeriOh4.Visible = True
'segunda derivada
Case Is = 2
FraDeriTipoerror.Visible = True
OptDeriOh2.Caption = "O(h2)"
OptDeriOh2.Value = False
OptDeriOh2.Visible = True
OptDeriOh4.Caption = "O(h4)"
OptDeriOh4.Value = False
OptDeriOh4.Visible = True
'si a elegido formula de diferenciación progresiva
 'primera derivada
Case Is = 3
FraDeriTipoerror.Visible = True
OptDeriOh2.Caption = "O(h)"
OptDeriOh2.Value = False
OptDeriOh2.Visible = True
OptDeriOh4.Caption = "O(h2)"
OptDeriOh4.Value = False
OptDeriOh4.Visible = True
'segunda derivada
Case Is = 4
FraDeriTipoerror.Visible = True
OptDeriOh2.Caption = "O(h)"
OptDeriOh2.Value = False
OptDeriOh2.Visible = True
OptDeriOh4.Caption = "O(h2)"
OptDeriOh4.Value = False
OptDeriOh4.Visible = True
'si a elegido formula de diferenciación regresivas
 'primera derivada
Case Is = 5
FraDeriTipoerror.Visible = True
OptDeriOh2.Caption = "O(h)"
OptDeriOh2.Value = False
OptDeriOh2.Visible = True
OptDeriOh4.Caption = "O(h2)"
OptDeriOh4.Value = False
OptDeriOh4.Visible = True
'segunda derivada
Case Is = 6
FraDeriTipoerror.Visible = True
OptDeriOh2.Caption = "O(h)"
OptDeriOh2.Value = False
OptDeriOh2.Visible = True
OptDeriOh4.Caption = "O(h2)"
```

```
OptDeriOh4.Value = False
OptDeriOh4.Visible = True
End Select

   Deriabrir.Enabled = True
   Deriguardarcomo.Enabled = True
   FrmRecuperaTrabajo.ver = tipoalg

End Sub
' Presentación de controles para el calculo del error
Private Sub CheckDeriError_Click()
If CheckDeriError.Value = 1 Then
FraDeriError.Visible = True
LblDeriReal.Visible = True
TxtDeriReal.Text = " "
TxtDeriReal.Visible = True
Else
FraDeriError.Visible = False
LblDeriReal.Visible = False
TxtDeriReal.Text = " "
TxtDeriReal.Visible = False
End If
End Sub

'Manejo del número de celdas en el Fg2Deri y sus titulos

Private Sub OptDeriOh2_Click()
Select Case tipoalg
'Fórmulas de diferencias centradas
 ' primera derivada
Case Is = 1
   If OptDeriOh2.Value = True Then
      If OptDeriDatos.Value = True Then
      LblDerititulo1.Visible = True
      Fg2Deri.Clear
      Fg2Deri.Cols = 2
      Fg2Deri.Rows = 3
      Fg2Deri.FixedRows = 1
      Fg2Deri.ColWidth(0) = 500
      Fg2Deri.ColAlignmentFixed(0) = 4
      Fg2Deri.ColAlignmentFixed(1) = 4
      Fg2Deri.TextArray(Fgi1(0, 1)) = "f(x)"
      Fg2Deri.TextArray(Fgi1(1, 0)) = "f-1"
      Fg2Deri.TextArray(Fgi1(2, 0)) = "f1"
      LblDerititulo1.Visible = True
      Fg2Deri.Visible = True
```

```
        End If
    End If
 'segunda derivada
Case Is = 2
    If OptDeriOh2.Value = True Then
        If OptDeriDatos.Value = True Then
        LblDerititulo1.Visible = True
        Fg2Deri.Clear
        Fg2Deri.Cols = 2
        Fg2Deri.Rows = 4
        Fg2Deri.FixedRows = 1
        Fg2Deri.ColWidth(0) = 500
        Fg2Deri.ColAlignmentFixed(0) = 4
        Fg2Deri.ColAlignmentFixed(1) = 4
        Fg2Deri.TextArray(Fgi1(0, 1)) = "f(x)"
        Fg2Deri.TextArray(Fgi1(1, 0)) = "f-1"
        Fg2Deri.TextArray(Fgi1(2, 0)) = "f0"
        Fg2Deri.TextArray(Fgi1(3, 0)) = "f1"
        LblDerititulo1.Visible = True
        Fg2Deri.Visible = True
        End If
    End If
'Fórmulas de diferencias progresivas
 'primera derivada
Case Is = 3
    If OptDeriOh2.Value = True Then
        If OptDeriDatos.Value = True Then
        LblDerititulo1.Visible = True
        Fg2Deri.Clear
        Fg2Deri.Cols = 2
        Fg2Deri.Rows = 3
        Fg2Deri.FixedRows = 1
        Fg2Deri.ColWidth(0) = 500
        Fg2Deri.ColAlignmentFixed(0) = 4
        Fg2Deri.ColAlignmentFixed(1) = 4
        Fg2Deri.TextArray(Fgi1(0, 1)) = "f(x)"
        Fg2Deri.TextArray(Fgi1(1, 0)) = "f0"
        Fg2Deri.TextArray(Fgi1(2, 0)) = "f1"
        LblDerititulo1.Visible = True
        Fg2Deri.Visible = True
        End If
    End If
'segunda derivada
Case Is = 4
    If OptDeriOh2.Value = True Then
        If OptDeriDatos.Value = True Then
```

```
        LblDerititulo1.Visible = True
        Fg2Deri.Clear
        Fg2Deri.Cols = 2
        Fg2Deri.Rows = 4
        Fg2Deri.FixedRows = 1
        Fg2Deri.ColWidth(0) = 500
        Fg2Deri.ColAlignmentFixed(0) = 4
        Fg2Deri.ColAlignmentFixed(1) = 4
        Fg2Deri.TextArray(Fgi1(0, 1)) = "f(x)"
        Fg2Deri.TextArray(Fgi1(1, 0)) = "f0"
        Fg2Deri.TextArray(Fgi1(2, 0)) = "f1"
        Fg2Deri.TextArray(Fgi1(3, 0)) = "f2"
        LblDerititulo1.Visible = True
        Fg2Deri.Visible = True
        End If
    End If
'Fórmulas de diferencias regresivas
 'primera derivada
Case Is = 5
    If OptDeriOh2.Value = True Then
        If OptDeriDatos.Value = True Then
        LblDerititulo1.Visible = True
        Fg2Deri.Clear
        Fg2Deri.Cols = 2
        Fg2Deri.Rows = 3
        Fg2Deri.FixedRows = 1
        Fg2Deri.ColWidth(0) = 500
        Fg2Deri.ColAlignmentFixed(0) = 4
        Fg2Deri.ColAlignmentFixed(1) = 4
        Fg2Deri.TextArray(Fgi1(0, 1)) = "f(x)"
        Fg2Deri.TextArray(Fgi1(1, 0)) = "f-1"
        Fg2Deri.TextArray(Fgi1(2, 0)) = "f0"
        LblDerititulo1.Visible = True
        Fg2Deri.Visible = True
        End If
    End If
'segunda derivada
Case Is = 6
    If OptDeriOh2.Value = True Then
        If OptDeriDatos.Value = True Then
        LblDerititulo1.Visible = True
        Fg2Deri.Clear
        Fg2Deri.Cols = 2
        Fg2Deri.Rows = 4
        Fg2Deri.FixedRows = 1
        Fg2Deri.ColWidth(0) = 500
```

```
        Fg2Deri.ColAlignmentFixed(0) = 4
        Fg2Deri.ColAlignmentFixed(1) = 4
        Fg2Deri.TextArray(Fgi1(0, 1)) = "f(x)"
        Fg2Deri.TextArray(Fgi1(1, 0)) = "f-2"
        Fg2Deri.TextArray(Fgi1(2, 0)) = "f-1"
        Fg2Deri.TextArray(Fgi1(3, 0)) = "f0"
        LblDerititulo1.Visible = True
        Fg2Deri.Visible = True
        End If
    End If
End Select
End Sub

Private Sub OptDeriOh4_Click()
Select Case tipoalg
'Fórmulas de diferencias centradas
 ' primera derivada
Case Is = 1
    If OptDeriOh4.Value = True Then
        If OptDeriDatos.Value = True Then
        LblDerititulo1.Visible = True
        Fg2Deri.Clear
        Fg2Deri.Cols = 2
        Fg2Deri.Rows = 5
        Fg2Deri.FixedRows = 1
        Fg2Deri.ColWidth(0) = 500
        Fg2Deri.ColAlignmentFixed(0) = 4
        Fg2Deri.ColAlignmentFixed(1) = 4
        Fg2Deri.TextArray(Fgi1(0, 1)) = "f(x)"
        Fg2Deri.TextArray(Fgi1(1, 0)) = "f-2"
        Fg2Deri.TextArray(Fgi1(2, 0)) = "f-1"
        Fg2Deri.TextArray(Fgi1(3, 0)) = "f1"
        Fg2Deri.TextArray(Fgi1(4, 0)) = "f2"
        LblDerititulo1.Visible = True
        Fg2Deri.Visible = True
        End If
    End If
 'segunda derivada
Case Is = 2
    If OptDeriOh4.Value = True Then
        If OptDeriDatos.Value = True Then
        LblDerititulo1.Visible = True
        Fg2Deri.Clear
        Fg2Deri.Cols = 2
        Fg2Deri.Rows = 6
        Fg2Deri.FixedRows = 1
```

```
        Fg2Deri.ColWidth(0) = 500
        Fg2Deri.ColAlignmentFixed(0) = 4
        Fg2Deri.ColAlignmentFixed(1) = 4
        Fg2Deri.TextArray(Fgi1(0, 1)) = "f(x)"
        Fg2Deri.TextArray(Fgi1(1, 0)) = "f-2"
        Fg2Deri.TextArray(Fgi1(2, 0)) = "f-1"
        Fg2Deri.TextArray(Fgi1(3, 0)) = "f0"
        Fg2Deri.TextArray(Fgi1(4, 0)) = "f1"
        Fg2Deri.TextArray(Fgi1(5, 0)) = "f2"
        LblDerititulo1.Visible = True
        Fg2Deri.Visible = True
        End If
    End If
'Fórmulas de diferencias progresivas
 'primera derivada
Case Is = 3
    If OptDeriOh4.Value = True Then
        If OptDeriDatos.Value = True Then
        LblDerititulo1.Visible = True
        Fg2Deri.Clear
        Fg2Deri.Cols = 2
        Fg2Deri.Rows = 4
        Fg2Deri.FixedRows = 1
        Fg2Deri.ColWidth(0) = 500
        Fg2Deri.ColAlignmentFixed(0) = 4
        Fg2Deri.ColAlignmentFixed(1) = 4
        Fg2Deri.TextArray(Fgi1(0, 1)) = "f(x)"
        Fg2Deri.TextArray(Fgi1(1, 0)) = "f0"
        Fg2Deri.TextArray(Fgi1(2, 0)) = "f1"
        Fg2Deri.TextArray(Fgi1(3, 0)) = "f2"
        LblDerititulo1.Visible = True
        Fg2Deri.Visible = True
        End If
    End If

 'segunda derivada
Case Is = 4
    If OptDeriOh4.Value = True Then
        If OptDeriDatos.Value = True Then
        LblDerititulo1.Visible = True
        Fg2Deri.Clear
        Fg2Deri.Cols = 2
        Fg2Deri.Rows = 5
        Fg2Deri.FixedRows = 1
        Fg2Deri.ColWidth(0) = 500
        Fg2Deri.ColAlignmentFixed(0) = 4
```

```
        Fg2Deri.ColAlignmentFixed(1) = 4
        Fg2Deri.TextArray(Fgi1(0, 1)) = "f(x)"
        Fg2Deri.TextArray(Fgi1(1, 0)) = "f0"
        Fg2Deri.TextArray(Fgi1(2, 0)) = "f1"
        Fg2Deri.TextArray(Fgi1(3, 0)) = "f2"
        Fg2Deri.TextArray(Fgi1(4, 0)) = "f3"
        LblDerititulo1.Visible = True
        Fg2Deri.Visible = True
        End If
    End If


'Fórmulas de diferencias regresivas
 'primera derivada
Case Is = 5
    If OptDeriOh4.Value = True Then
        If OptDeriDatos.Value = True Then
        LblDerititulo1.Visible = True
        Fg2Deri.Clear
        Fg2Deri.Cols = 2
        Fg2Deri.Rows = 4
        Fg2Deri.FixedRows = 1
        Fg2Deri.ColWidth(0) = 500
        Fg2Deri.ColAlignmentFixed(0) = 4
        Fg2Deri.ColAlignmentFixed(1) = 4
        Fg2Deri.TextArray(Fgi1(0, 1)) = "f(x)"
        Fg2Deri.TextArray(Fgi1(1, 0)) = "f-2"
        Fg2Deri.TextArray(Fgi1(2, 0)) = "f-1"
        Fg2Deri.TextArray(Fgi1(3, 0)) = "f0"
        LblDerititulo1.Visible = True
        Fg2Deri.Visible = True
        End If
    End If
 'segunda derivada
Case Is = 6
    If OptDeriOh4.Value = True Then
        If OptDeriDatos.Value = True Then
        LblDerititulo1.Visible = True
        Fg2Deri.Clear
        Fg2Deri.Cols = 2
        Fg2Deri.Rows = 5
        Fg2Deri.FixedRows = 1
        Fg2Deri.ColWidth(0) = 500
        Fg2Deri.ColAlignmentFixed(0) = 4
        Fg2Deri.ColAlignmentFixed(1) = 4
        Fg2Deri.TextArray(Fgi1(0, 1)) = "f(x)"
```

```
        Fg2Deri.TextArray(Fgi1(1, 0)) = "f-3"
        Fg2Deri.TextArray(Fgi1(2, 0)) = "f-2"
        Fg2Deri.TextArray(Fgi1(3, 0)) = "f-1"
        Fg2Deri.TextArray(Fgi1(4, 0)) = "f0"
        LblDerititulo1.Visible = True
        Fg2Deri.Visible = True
        End If
    End If
End Select

End Sub

Private Sub TxtDeriEdit2_Change()
LTrim (TxtDeriEdit2)
End Sub

Private Sub TxtDeriFunción_Change()
LTrim (TxtDeriFunción)
End Sub

Private Sub TxtDeriNumincrementos_KeyPress(KeyAscii As Integer)


If (KeyAscii < 48 Or KeyAscii > 57) Then
If (KeyAscii <> 8) Then KeyAscii = 0

End If

End Sub

Private Sub TxtDeriNumincrementos_Change()
Dim mensaje As String
Dim i As Integer
LTrim (TxtDeriNumincrementos)
If Val(TxtDeriNumincrementos) > 50 Then
mensaje = MsgBox("Fuera del rango", 16, "Error en el ingreso")
TxtDeriNumincrementos.Text = " "
Else
n = Val(TxtDeriNumincrementos)
   If n = 0 Then
   n = 1
   End If
Fg3Deri.Cols = 2
Fg3Deri.ColWidth(0) = 500
Fg3Deri.ColAlignmentFixed(0) = 4
Fg3Deri.ColAlignmentFixed(1) = 4
```

```
Fg3Deri.Rows = n + 1
Fg3Deri.FixedRows = 1
Fg3Deri.FixedCols = 1
Fg3Deri.TextArray(Fgi(0, 1)) = "h"
   For i = 1 To n
   Fg3Deri.TextArray(Fgi(i, 0)) = i
   Next i
Fg3Deri.Visible = True
LblDeriTitulo2.Visible = True
End If

End Sub

'Manejo de Fg3Deri  para ingreso de datos e incrementos respectivamente
Function Fgi(r As Integer, c As Integer) As Long
  Fgi = c + Fg3Deri.Cols * r
End Function

Sub Fg3Deri_KeyPress(KeyAscii As Integer)
  MSHFlexGridEdit Fg3Deri, TxtDeriEdit2, KeyAscii
End Sub

Sub Fg3Deri_DblClick()
  MSHFlexGridEdit Fg3Deri, TxtDeriEdit2, 32 ' Simula un espacio.
End Sub
Sub MSHFlexGridEdit(MSHFlexGrid As Control, _
Edt As Control, KeyAscii As Integer)

  ' Usar el carácter escrito.
  Select Case KeyAscii

  ' Un espacio significa modificar el texto actual.
  Case 0 To 32
    Edt = MSHFlexGrid
    Edt.SelStart = 1000

  ' Otro carácter reemplaza el texto actual.
  Case Else
    Edt = Chr(KeyAscii)
    Edt.SelStart = 1
  End Select

  ' Mostrar Edt en la posición correcta.
  Edt.Move MSHFlexGrid.Left + MSHFlexGrid.CellLeft, _
    MSHFlexGrid.Top + MSHFlexGrid.CellTop, _
```

```
    MSHFlexGrid.CellWidth - 8, _
    MSHFlexGrid.CellHeight - 8
  Edt.Visible = True

  ' Y hacer que funcione.
  Edt.SetFocus
End Sub

Sub TxtDeriEdit2_KeyPress(KeyAscii As Integer)
'filtra la entrada de datos que no sean números
  If (KeyAscii < 48 Or KeyAscii > 57) Then
      If (KeyAscii <> 46) Then
         If (KeyAscii <> 8) Then
         KeyAscii = 0
         End If
      End If
   End If
  If KeyAscii = Asc(vbCr) Then ' Elimina los retornos para quitar los pitidos.
  KeyAscii = 0
  End If
End Sub

Sub TxtDeriEdit2_KeyDown(KeyCode As Integer, _
Shift As Integer)
  EditKeyCode1 Fg3Deri, TxtDeriEdit2, KeyCode, Shift
End Sub

Sub EditKeyCode1(MSHFlexGrid As Control, Edt As _
Control, KeyCode As Integer, Shift As Integer)

  ' Procesamiento del control de edición estándar.
  Select Case KeyCode

  Case 27   ' ESC: ocultar, devuelve el enfoque a          ' MSFlexGrid.
    Edt.Visible = False
    MSHFlexGrid.SetFocus

  Case 13   ' ENTRAR devuelve el enfoque a MSHFlexGrid.
    MSHFlexGrid.SetFocus

  Case 38      ' Arriba.
    MSHFlexGrid.SetFocus
    DoEvents
    If MSHFlexGrid.Row > MSHFlexGrid.FixedRows Then
      MSHFlexGrid.Row = MSHFlexGrid.Row - 1
    End If
```

```
  Case 40      ' Abajo.
    MSHFlexGrid.SetFocus
    DoEvents
    If MSHFlexGrid.Row < MSHFlexGrid.Rows - 1 Then
      MSHFlexGrid.Row = MSHFlexGrid.Row + 1
    End If
  End Select
End Sub

Sub Fg3Deri_GotFocus()
  If TxtDeriEdit2.Visible = False Then Exit Sub
  Fg3Deri = TxtDeriEdit2
  TxtDeriEdit2.Visible = False
End Sub

Sub Fg3Deri_LeaveCell()
  If TxtDeriEdit2.Visible = False Then Exit Sub
  Fg3Deri = TxtDeriEdit2
  TxtDeriEdit2.Visible = False
End Sub
'Manejo de Fg2Deri  para ingreso de datos e incrementos respectivamente
Function Fgi1(r As Integer, c As Integer) As Integer
  Fgi1 = c + Fg2Deri.Cols * r
End Function

Sub Fg2Deri_KeyPress(KeyAscii As Integer)
  MSHFlexGridEdit1 Fg2Deri, TxtDeriEdit1, KeyAscii
End Sub

Sub Fg2Deri_DblClick()
  MSHFlexGridEdit1 Fg2Deri, TxtDeriEdit1, 32 ' Simula un espacio.
End Sub

Sub MSHFlexGridEdit1(MSHFlexGrid As Control, _
Edt As Control, KeyAscii As Integer)

  ' Usar el carácter escrito.
  Select Case KeyAscii

  ' Un espacio significa modificar el texto actual.
  Case 0 To 32
    Edt = MSHFlexGrid
    Edt.SelStart = 1000

  ' Otro carácter reemplaza el texto actual.
```

```
  Case Else
    Edt = Chr(KeyAscii)
    Edt.SelStart = 1
  End Select

  ' Mostrar Edt en la posición correcta.
  Edt.Move MSHFlexGrid.Left + MSHFlexGrid.CellLeft, _
    MSHFlexGrid.Top + MSHFlexGrid.CellTop, _
    MSHFlexGrid.CellWidth - 8, _
    MSHFlexGrid.CellHeight - 8
  Edt.Visible = True

  ' Y hacer que funcione.
  Edt.SetFocus
End Sub

Sub TxtDeriEdit1_KeyPress(KeyAscii As Integer)
  'filtra la entrada de datos que no sean números
  If (KeyAscii < 48 Or KeyAscii > 57) Then
      If (KeyAscii <> 46) Then
         If (KeyAscii <> 45) Then
            If (KeyAscii <> 8) Then
            KeyAscii = 0
            End If
         End If
      End If
   End If
  If KeyAscii = Asc(vbCr) Then ' Elimina los retornos para quitar los pitidos.
  KeyAscii = 0
  End If
End Sub

Sub TxtDeriEdit1_KeyDown(KeyCode As Integer, _
Shift As Integer)
  EditKeyCode Fg2Deri, TxtDeriEdit1, KeyCode, Shift
End Sub

Sub EditKeyCode(MSHFlexGrid As Control, Edt As _
Control, KeyCode As Integer, Shift As Integer)

  ' Procesamiento del control de edición estándar.
  Select Case KeyCode

  Case 27   ' ESC: ocultar, devuelve el enfoque a          ' MSFlexGrid.
    Edt.Visible = False
    MSHFlexGrid.SetFocus
```

```
  Case 13   ' ENTRAR devuelve el enfoque a MSHFlexGrid.
    MSHFlexGrid.SetFocus

  Case 38      ' Arriba.
    MSHFlexGrid.SetFocus
    DoEvents
    If MSHFlexGrid.Row > MSHFlexGrid.FixedRows Then
      MSHFlexGrid.Row = MSHFlexGrid.Row - 1
    End If

  Case 40      ' Abajo.
    MSHFlexGrid.SetFocus
    DoEvents
    If MSHFlexGrid.Row < MSHFlexGrid.Rows - 1 Then
      MSHFlexGrid.Row = MSHFlexGrid.Row + 1
    End If
  End Select
End Sub

Sub Fg2Deri_GotFocus()
  If TxtDeriEdit1.Visible = False Then Exit Sub
  Fg2Deri = TxtDeriEdit1
  TxtDeriEdit1.Visible = False
End Sub

Sub Fg2Deri_LeaveCell()
  If TxtDeriEdit1.Visible = False Then Exit Sub
  Fg2Deri = TxtDeriEdit1
  TxtDeriEdit1.Visible = False
End Sub

'Verifica si los parámetros del algoritmo estan completos
Private Sub ComprobarParametros()
Dim bandera As Boolean  ' indica si existe espacios en blanco entonces bantera es
true
Dim i As Long          'contador
Dim valor As Integer     'número de celdas que tiene Fg2Deri o Fg3Deri
Dim valortmp As String     'valor que tendra una celda de Fg2Deri o Fg3Deri
bandera = False
If OptDeriFunción.Value = True Then
   If Val(TxtDeriNumincrementos) = 0 Then
   bandera = True
   Else
   valor = Fg3Deri.Rows
      For i = 1 To valor - 1
```

```
        valortmp = Fg3Deri.TextMatrix(i, 1)
          If valortmp <> " " Then
          bandera = True
          End If
        Next i
    End If
End If
End Sub

Private Sub SetearDerivación()
On Error GoTo err1
Dim Pares() As Double
Dim intervalo() As Double
Dim i As Integer
Dim tope As Integer
If OptDeriFunción = False Then
'copiamos los valores de Fg2Deri a la matriz pares
tope = Fg2Deri.Rows
ReDim Pares(0 To tope - 2) As Double
For i = 1 To tope - 1
Pares(i - 1) = Val(Fg2Deri.TextMatrix(i, 1))
Next i
End If
'copiamos los valores de Fg3 a la matriz intervalo
tope = Fg3Deri.Rows
ReDim intervalo(0 To tope - 2) As Double
For i = 0 To tope - 2
intervalo(i) = Val(Fg3Deri.TextMatrix(i + 1, 1))
Next i


'Seteamos los atributos de la clase derivación

derivación.SetearDeri(tipoalg, OptDeriFunción.Value, OptDeriDatos.Value,
OptDeriOh2.Value, OptDeriOh4.Value, CheckDeriError.Value _
, Val(TxtDeriNumincrementos.Text), CDbl(Val(TxtDeriPunto.Text)),
CDbl(Val(TxtDeriReal.Text)) _
, TxtDeriFunción.Text, intervalo) = Pares
Exit Sub
err1:
End Sub
'Manejo de presentación de resultados
Function Fgr(r As Integer, c As Integer) As Long
  Fgr = c + fgDeriresultados.Cols * r
End Function
Private Sub DeriPresentar()
```

```
On Error GoTo err1
Dim i As Integer
Dim j As Integer
Dim valor As Double
fgDeriresultados.Cols = 3
fgDeriresultados.Rows = n + 1
fgDeriresultados.FixedCols = 1
fgDeriresultados.FixedRows = 1
fgDeriresultados.TextArray(Fgr(0, 0)) = "h"
fgDeriresultados.TextArray(Fgr(0, 1)) = "Derivada"
fgDeriresultados.TextArray(Fgr(0, 2)) = "Error"
fgDeriresultados.ColWidth(0) = 500
fgDeriresultados.ColWidth(1) = 3500
fgDeriresultados.ColWidth(2) = 3500
fgDeriresultados.ColAlignmentFixed(0) = 4
fgDeriresultados.ColAlignmentFixed(1) = 4
fgDeriresultados.ColAlignmentFixed(2) = 4

For i = 1 To n
fgDeriresultados.TextArray(Fgr(i, 0)) = i
Next i
For i = 1 To n
   For j = 1 To 2
   valor = derivación.Derivada(i - 1, j - 1)
      If (valor <= 0 Or valor > 0.000000000000001) Then
      fgDeriresultados.TextMatrix(i, j) = valor
      Else
      fgDeriresultados.TextMatrix(i, j) = 0
      End If
  Next j
Next i
Exit Sub
err1:
End Sub

'****** Conexion a la BDD
Private Sub PLConexion(parBDD As Boolean)
Dim VTCad As String
Dim VTBase As Database
Dim VTcadena As String
   VTcadena = App.Path & "/Seguridad.mdb"
   Set VLConexion = New ADODB.Connection
   If parBDD = False Then
      VTCad = "Provider=SQLOLEDB.1;Integrated Security=SSPI;Persist Security
Info=False;Initial Catalog=Seguridad"
   Else
```

```
      VTCad = "Provider=Microsoft.Jet.OLEDB.4.0;Data Source='" & VTcadena &
"';Persist Security Info=False"
   End If
   VLConexion.Open VTCad
End Sub

Private Sub PLGuardarBDD()
   Dim VTCad As String
   Dim VTReg As ADODB.Recordset
   Dim VTID As Integer
   Dim VB As Integer
   Dim VTNumero As String
   Dim VTDatoMatriz As String
   Dim VTIncrementoMatriz As String
On Error GoTo error1
   PLVerificaNombre Trim(VLNombre)
      If VLVerifica = True Then   'Si existe un nombre con el mismo nombre actualiza
         VB = MsgBox("El nombre de ese trabajo ya existe, desea reemplazarlo?",
vbYesNo, "Guardar Trabajo Como")
         If VB = 6 Then
            If OptDeriFunción.Value = True Then
               VTCad = "update Derivacion set " _
                     & " Fx= '" & Trim(TxtDeriFunción.Text) & "', " _
                     & " Pd='" & TxtDeriPunto.Text & "', " _
                     & " h='" & TxtDeriNumincrementos.Text & "' " _
                     & " where Nombre = '" & VLNombre & "'"
            End If
            If OptDeriDatos.Value = True Then
               If tipoalg = 1 And OptDeriOh2.Value = True Then
                  PLGuardaIncrementos 2
               End If
               If OptDeriOh4.Value = True Then
                  PLGuardaIncrementos 4
               End If
               'VTCad = "update Derivacion set " _
                     & " Fx= '" & Trim(TxtIntegrafunción.Text) & "', " _
                     & " Datos='" & Fg3Deri.TextMatrix(1, 1) & "', " _
                     & " Incrementos='" & VLJacobiano & "' " _
                     & " where Nombre = '" & VLNombre & "'"
            End If
            VLConexion.Execute (VTCad)
         End If
      Else   'Guarda el trabajo
         VTID = PLRecuperaCodigo
         If tipoalg = 1 Then
            If OptDeriFunción.Value = True Then
```

```
            PLGuardaDatos TxtDeriNumincrementos
         End If
         If OptDeriOh2.Value = True And Fg2Deri.Visible = True Then
            VLDatos = Fg3Deri.TextMatrix(1, 1)
            PLGuardaIncrementos 2
         End If
      ElseIf tipoalg = 2 And OptDeriOh4.Value = True Then
         PLGuardaIncrementos 4
      End If
      VTDatoMatriz = VLDatos
      VTIncrementoMatriz = VLJacobiano
      VTCad = "insert into Derivacion values " _
            & "(" & CInt(VTID) & ", " _
            & " '" & TxtDeriFunción.Text & "', '" & TxtDeriPunto.Text & "', " _
            & " '" & TxtDeriNumincrementos.Text & "', '" & VTDatoMatriz & "', " _
            & " '" & VTIncrementoMatriz & "', " _
            & " '" & Trim(VLNombre) & "', " & 1 & ", " & CInt(tipoalg) & ")"
            '& " '" & Trim(VLNombre) & "', " & CInt(tipoalg) & ", " & CInt(VGIDUsuario)
& ")"
      VLConexion.Execute (VTCad)
      MsgBox "Trabajo guardado exitosamente", vbInformation, "Guardar trabajo
como"
   End If

Exit Sub
error1:
   'If err <> 0 Then
      'MsgBox "Se ha producido el siguiente error: " & err.Description & "." _
            & "Por favor consultar con el administrador", vbCritical
'   End If
End Sub

Private Sub PLVerificaNombre(parNombre As String)
Dim VTReg As ADODB.Recordset
Dim VTCad As String
On Error GoTo error1
   VTCad = "select Nombre from Derivacion where Nombre = '" & parNombre & "' and
tipo = " & tipoalg
   Set VTReg = VLConexion.Execute(VTCad)
   With VTReg
      If Not VTReg.EOF Then
         .MoveFirst
         Do Until .EOF
            If Trim(VTReg(0)) = parNombre Then
               VLVerifica = True
               Exit Do
```

```
                Else
                   VLVerifica = False
                End If
                .MoveNext
             Loop
          Else
             VLVerifica = False
          End If
       End With
Exit Sub
error1:

End Sub
Private Function PLRecuperaCodigo() As Integer
   Dim VTCad As String
   Dim VTReg As ADODB.Recordset
On Error GoTo err1
       VTCad = "select max(IDDerivacion) from Derivacion"
       Set VTReg = VLConexion.Execute(VTCad)
       If Not IsNull(VTReg(0)) Then
          PLRecuperaCodigo = VTReg(0) + 1
       Else
          PLRecuperaCodigo = 1
       End If
Exit Function
err1:

End Function

Private Sub PLGuardaIncrementos(parNume As String)
Dim i, j As Integer
Dim Nume As Integer
On Error GoTo error1
   VLJacobiano = ""
   Nume = CInt(parNume)
   If tipoalg = 1 Then
      For i = 1 To Nume
         For j = 1 To Nume - 1
            VLJacobiano = VLJacobiano & Fg2Deri.TextMatrix(i, j) & ";"
         Next j
      Next i
   End If
Exit Sub
error1:

End Sub
```

```
Private Sub PLGuardaDatos(parNume As String)
Dim i, j As Integer
Dim Nume As Integer
On Error GoTo error1
   VLDatos = ""
   Nume = CInt(parNume)
   If tipoalg = 1 Then
      For i = 1 To Nume
         For j = 1 To Nume - 1
            VLDatos = VLDatos & Fg3Deri.TextMatrix(i, j) & ";"
         Next j
      Next i
   End If
Exit Sub
error1:

End Sub

Private Sub PLRecupera()
   Dim VTCad As String
   Dim VTReg As ADODB.Recordset
On Error GoTo err1

      VTCad = "select * from Derivacion where Nombre = '" & VLNombreTrabajo & "'"
      Set VTReg = VLConexion.Execute(VTCad)
      If Not VTReg.EOF Then
         TxtDeriFunción.Text = Trim(VTReg(1))
         TxtDeriPunto.Text = Trim(VTReg(2))
         TxtDeriNumincrementos.Text = Trim(VTReg(3))
         TxtDeriNumincrementos.Visible = True
         LblDeriNumincrementos.Visible = True
            'PLRecuperaDatos CInt(VTReg(3)), Trim(VTReg(4))
         Fg3Deri.TextMatrix(1, 1) = VTReg(4)

            'PLRecuperaIncrementos CInt(VTReg(3)), Trim(VTReg(4))

      End If
Exit Sub
err1:
   'If err.Number <> 0 Then
    '  MsgBox "Se ha producido el siguiente error: " & err.Description & "." _
          & "Por favor consultar con el administrador", vbCritical
   'End If

End Sub
```

```
Private Sub PLRecuperaDatos(parNume As Integer, parDatos As String)
Dim i, j, k As Integer
Dim VTCont As Integer
Dim VTPos As Integer
Dim VTValor As Integer
Dim VTLimite As Integer
Dim VTDat As String
On Error GoTo error1
   VTCont = 1
   VTDat = parDatos
      VTLimite = parNume
      For k = 1 To VTLimite
         VLDatosMatriz(k) = 0
      Next k

      For k = 1 To VTLimite
         VTPos = InStr(1, parDatos, ";")
         VTValor = Mid(parDatos, 1, VTPos - 1)
         parDatos = Mid(parDatos, VTPos + 1)
         VLDatosMatriz(k) = VTValor
      Next k
      For i = 1 To parNume
         For j = 1 To parNume - 1
            Fg3Deri.TextMatrix(i, j) = VLDatosMatriz(VTCont)
            VTCont = VTCont + 1
         Next j
      Next i

Exit Sub
error1:
   'If err.Number <> 0 Then
    '  MsgBox "Se ha producido el siguiente error: " & err.Description & "." _
            & "Por favor consultar con el administrador", vbCritical
   'End If
End Sub

Private Sub PLRecuperaIncrementos(parNume As Integer, parDatos As String)
Dim i, j, k As Integer
Dim VTCont As Integer
Dim VTPos As Integer
Dim VTValor As Double
Dim VTLimite As Integer
Dim VTDat As String
On Error GoTo error1
   VTCont = 1
```

```
    VTDat = parDatos
       VTLimite = parNume
       For k = 1 To VTLimite
          VLDatosMatriz(k) = 0
       Next k

       For k = 1 To VTLimite
          VTPos = InStr(1, parDatos, ";")
          VTValor = Mid(parDatos, 1, VTPos - 1)
          parDatos = Mid(parDatos, VTPos + 1)
          VLDatosMatriz(k) = VTValor
       Next k
       For i = 1 To parNume
          For j = 1 To parNume - 1
             Fg3Deri.TextMatrix(i, j) = VLDatosMatriz(VTCont)
             VTCont = VTCont + 1
          Next j
       Next i

Exit Sub
error1:
   'If err.Number <> 0 Then
    '  MsgBox "Se ha producido el siguiente error: " & err.Description & "." _
          & "Por favor consultar con el administrador", vbCritical
   'End If
End Sub


Private Sub PLImprimir()
   PLCambio
      PrintForm
   PLOriginal
End Sub

Private Sub PLCambio()
   Me.BackColor = &HFFFFFF
   Me.BorderStyle = 0
   CmdDeriCalcular.Visible = False
End Sub

Private Sub PLOriginal()
   Me.BackColor = &H8000000F
   Me.BorderStyle = 2
   CmdDeriCalcular.Visible = True
End Sub
```

```
Private Sub TxtDeriPunto_Change()
LTrim (TxtDeriPunto)
End Sub



Private Sub TxtDeriPunto_KeyPress(KeyAscii As Integer)
'filtra la entrada de datos que no sean números incluye el signo - y el punto
  If (KeyAscii < 48 Or KeyAscii > 57) Then
      If (KeyAscii = 45) Then
      Exit Sub
      End If
      If (KeyAscii <> 46) Then
         If (KeyAscii <> 8) Then
         KeyAscii = 0
         End If
      End If
   End If
  If KeyAscii = Asc(vbCr) Then ' Elimina los retornos para quitar los pitidos.
  KeyAscii = 0
  End If
End Sub

Private Sub TxtDeriReal_KeyPress(KeyAscii As Integer)
'filtra la entrada de datos que no sean números incluye el signo - y el punto
  If (KeyAscii < 48 Or KeyAscii > 57) Then
      If (KeyAscii = 45) Then
      Exit Sub
      End If
      If (KeyAscii <> 46) Then
         If (KeyAscii <> 8) Then
         KeyAscii = 0
         End If
      End If
   End If
  If KeyAscii = Asc(vbCr) Then ' Elimina los retornos para quitar los pitidos.
  KeyAscii = 0
  End If

End Sub

Private Sub Sincolor()
With Me
arribaSstab = .SSTab1.Top
izquierdaSstab = .SSTab1.Left
anchoSstab = .SSTab1.Width
```

```
altoSstab = .SSTab1.Height
colororg = .BackColor
.SSTab1.Top = .SSTab1.Top + 1000
.SSTab1.Left = .SSTab1.Left + 1000
.SSTab1.Width = .SSTab1.Width / 4
.SSTab1.Height = .SSTab1.Height / 4
.BackColor = &H80000009
.FraImprimir.BackColor = &H80000009
.FraDeriError.BackColor = &H80000009
.FraDeritipo.BackColor = &H80000009
.FraPara.BackColor = &H80000009
.Fg2Deri.BackColorBkg = &H80000009
.Fg2Deri.BackColorFixed = &H80000009
.fgDeriresultados.BackColorBkg = &H80000009
.fgDeriresultados.BackColorFixed = &H80000009
.Fg3Deri.BackColorBkg = &H80000009
.Fg3Deri.BackColorFixed = &H80000009
.OptDeriDatos.BackColor = &H80000009
.OptDeriFunción.BackColor = &H80000009
.OptDeriOh2.BackColor = &H80000009
.OptDeriOh4.BackColor = &H80000009
.CheckDeriError.BackColor = &H80000009
.FraPara.BorderStyle = 1
.TxtDeriEdit1.BorderStyle = 0
.TxtDeriEdit2.BorderStyle = 0
.TxtDeriFunción.BorderStyle = 0
.TxtDeriNumincrementos.BorderStyle = 0
.TxtDeriPunto.BorderStyle = 0
.TxtDeriReal.BorderStyle = 0
.Shape1.Visible = False
.CmdDeriCalcular.Visible = False
.LblNomalg.BackColor = &H80000009
.FraDeriTipoerror.BackColor = &H80000009
FraDeriSintaxis.Visible = False
End With
End Sub

Private Sub Concolor()
With Me
 .SSTab1.Top = arribaSstab
 .SSTab1.Left = izquierdaSstab
 .SSTab1.Width = anchoSstab
 .SSTab1.Height = altoSstab
 .BackColor = colororg
.FraImprimir.BackColor = colororg
.FraDeriError.BackColor = colororg
```

```
.FraDeritipo.BackColor = colororg
.FraPara.BackColor = colororg
.Fg2Deri.BackColorBkg = colororg
.Fg2Deri.BackColorFixed = colororg
.fgDeriresultados.BackColorBkg = colororg
.fgDeriresultados.BackColorFixed = colororg
.Fg3Deri.BackColorBkg = colororg
.Fg3Deri.BackColorFixed = colororg
.OptDeriDatos.BackColor = colororg
.OptDeriFunción.BackColor = colororg
.OptDeriOh2.BackColor = colororg
.OptDeriOh4.BackColor = colororg
.CheckDeriError.BackColor = colororg
.FraPara.BorderStyle = 0
.TxtDeriEdit1.BorderStyle = 1
.TxtDeriEdit2.BorderStyle = 1
.TxtDeriFunción.BorderStyle = 1
.TxtDeriNumincrementos.BorderStyle = 1
.TxtDeriPunto.BorderStyle = 1
.TxtDeriReal.BorderStyle = 1
.Shape1.Visible = True
.CmdDeriCalcular.Visible = True
.LblNomalg.BackColor = colororg
.FraDeriTipoerror.BackColor = colororg
FraDeriSintaxis.Visible = True
End With
End Sub
```

Módulo de clase ClasDerivación

```
Option Explicit
Dim Evaluar As New McsExcel
Dim Tipoalgoritmo As Integer 'Tipo de algoritmo
Dim Tipofunción As Boolean  'Sí se ha elegido función es true
Dim Tipodatos As Boolean    'Sí se ha elegido datos es true
Dim función As String       'función
Dim fx() As Double          'valores de fx
Dim h() As Double           'valores de  h
Dim derivadareal As Double  'valor de la dreivada real
Dim numh As Integer         'número de h
Dim punto As Double         'valor donde se hallara la derivada
Dim ResulDeri() As Double   'matriz de resultados
Dim TipoDerivada As Boolean 'indica el tipo de derivada si es de función es true
Dim error As Integer        'indica si ha elegido el calculo del error
Dim Oh2 As Boolean          'indica si ha elegido el error Oh2
Dim Oh4 As Boolean          'indica si ha elegido el error Oh4
```

```
Public Property Let SetearDeri(tipoalg As Integer, Optf As Boolean, OptDatos As
Boolean, h2 As Boolean, h4 As Boolean, calcularerror As Integer, num As Integer,
puntoevaluar As Double, _
real As Double, f As String, intervalo() As Double, pts() As Double)
On Error GoTo err1
Tipoalgoritmo = tipoalg
Tipofunción = Optf
Tipodatos = OptDatos
Oh2 = h2
Oh4 = h4
error = calcularerror
derivadareal = real
numh = num
función = f
h = intervalo
punto = puntoevaluar
fx = pts
Exit Property
err1:
End Property
Public Property Get Derivada() As Variant
On Error GoTo err1
Derivada = ResulDeri
Exit Property
err1:
End Property

'Permite obtener la primera derivada de orden Oh2 y Oh4
Private Sub Centradaprimeraderi()
Dim i As Integer
Dim j As Integer

Dim mensaje As String
'Comprobamos si algún incremento es igual a cero
On Error GoTo err1
If numh = 0 Then
   mensaje = MsgBox("Debe existir por al menos un incremento", 16, "Error en el
ingreso")
   Exit Sub
End If
For i = 0 To numh - 1
   If h(i) = 0 Then
   mensaje = MsgBox("El incremento no puede ser cero", 16, "Error en el ingreso")
   Exit Sub
   End If
Next i
```

```
If Tipofunción = True Then

'Redimensionamos la matriz de resultados
ReDim ResulDeri(0 To numh - 1, 0 To 1) As Double

'Calculamos la primera derivada para Oh2 y el error
If Oh2 = True Then
   For i = 0 To numh - 1
   ResulDeri(i, 0) = (feval(punto + h(i)) - feval(punto - h(i))) / (2 * h(i))
      If error = 1 Then
      ResulDeri(i, 1) = derivadareal - ResulDeri(i, 0)
      End If
   Next i
End If
'Calculamos la primera derivada para Oh4
If Oh4 = True Then
   For i = 0 To numh - 1
   ResulDeri(i, 0) = (8 * feval(punto + h(i)) - feval(punto + (2 * h(i))) - 8 * feval(punto -
h(i)) + feval(punto - (2 * h(i)))) / (12 * h(i))
      If error = 1 Then
      ResulDeri(i, 1) = derivadareal - ResulDeri(i, 0)
      End If
   Next i
End If

Else
'Redimensionamos la matriz de resultados
ReDim ResulDeri(0 To numh - 1, 0 To 1) As Double

'Calculamos la primera derivada para Oh2 y el error
If Oh2 = True Then
   For i = 0 To numh - 1
   ResulDeri(i, 0) = (fx(1) - fx(0)) / (2 * h(i))
      If error = 1 Then
      ResulDeri(i, 1) = derivadareal - ResulDeri(i, 0)
      End If
   Next i
End If
'Calculamos la primera derivada para Oh4
If Oh4 = True Then
For i = 0 To numh - 1
   ResulDeri(i, 0) = (8 * fx(2) - fx(3) - 8 * fx(1) + fx(1)) / (12 * h(i))
      If error = 1 Then
      ResulDeri(i, 1) = derivadareal - ResulDeri(i, 0)
      End If
```

```
    Next i
End If

End If
Exit Sub
err1:
If err.Number <> 0 Then
    mensaje = MsgBox("Parámetros del algoritmo incompletos", 48, "Error en el
ingreso")
    End If
End Sub
'Permite obtener la segunda derivada de orden Oh2 y Oh4
Private Sub Centradasegundaderi()

Dim i As Integer
Dim j As Integer

Dim mensaje As String
'Comprobamos si algún incremento es igual a cero
On Error GoTo err1
If numh = 0 Then
    mensaje = MsgBox("Debe existir por al menos un incremento", 16, "Error en el
ingreso")
    Exit Sub
End If
For i = 0 To numh - 1
    If h(i) = 0 Then
    mensaje = MsgBox("El incremento no puede ser cero", 16, "Error en el ingreso")
    Exit Sub
    End If
Next i

If Tipofunción = True Then
'Redimensionamos la matriz de resultados
ReDim ResulDeri(0 To numh - 1, 0 To 1) As Double

'Calculamos la segunda derivada para Oh2 y el error
If Oh2 = True Then
   For i = 0 To numh - 1
    ResulDeri(i, 0) = (feval(punto + h(i)) - 2 * feval(punto) + feval(punto - h(i))) / ((h(i)) ^
2)
       If error = 1 Then
       ResulDeri(i, 1) = derivadareal - ResulDeri(i, 0)
       End If
    Next i
End If
```

```
'Calculamos la segunda derivada para Oh4
If Oh4 = True Then
   For i = 0 To numh - 1
   ResulDeri(i, 0) = (16 * feval(punto + h(i)) - feval(punto + (2 * h(i))) - 30 *
feval(punto) + 16 * feval(punto - h(i)) - feval(punto - (2 * h(i)))) / (12 * ((h(i)) ^ 2))
      If error = 1 Then
      ResulDeri(i, 1) = derivadareal - ResulDeri(i, 0)
      End If
   Next i
End If


Else

'Redimensionamos la matriz de resultados
ReDim ResulDeri(0 To numh - 1, 0 To 1) As Double

'Calculamos la segunda derivada para Oh2 y el error
If Oh2 = True Then
  For i = 0 To numh - 1
   ResulDeri(i, 0) = (fx(2) - 2 * fx(1) + fx(0)) / ((h(i)) ^ 2)
      If error = 1 Then
      ResulDeri(i, 1) = derivadareal - ResulDeri(i, 0)
      End If
   Next i
End If
'Calculamos la segunda derivada para Oh4
If Oh4 = True Then
   For i = 0 To numh - 1
   ResulDeri(i, 0) = (16 * fx(3) - fx(4) - 30 * fx(2) + 16 * fx(1) - fx(0)) / (12 * ((h(i)) ^ 2))
      If error = 1 Then
      ResulDeri(i, 1) = derivadareal - ResulDeri(i, 0)
      End If
   Next i
End If

End If
Exit Sub
err1:
If err.Number <> 0 Then
   mensaje = MsgBox("Parámetros del algoritmo incompletos", 48, "Error en el
ingreso")
   End If
End Sub

'Permite obtener la primera derivada de orden Oh2  mediante diferencias progresivas
```

```
Private Sub Progresivasprimeraderi()
Dim i As Integer
Dim j As Integer

Dim mensaje As String
On Error GoTo err1
'Comprobamos si algún incremento es igual a cero
If numh = 0 Then
   mensaje = MsgBox("Debe existir por al menos un incremento", 16, "Error en el
ingreso")
   Exit Sub
End If
For i = 0 To numh - 1
   If h(i) = 0 Then
   mensaje = MsgBox("El incremento no puede ser cero", 16, "Error en el ingreso")
   Exit Sub
   End If
Next i
If Tipofunción = True Then
'Redimensionamos la matriz de resultados
ReDim ResulDeri(0 To numh - 1, 0 To 1) As Double

'Calculamos la primera derivada para Oh y el error
If Oh2 = True Then
   For i = 0 To numh - 1
      ResulDeri(i, 0) = (feval(punto + h(i)) - feval(punto)) / h(i)
         If error = 1 Then
         ResulDeri(i, 1) = derivadareal - ResulDeri(i, 0)
         End If
   Next i
End If
'Calculamos la primera derivada para Oh2 y el error
If Oh4 = True Then
   For i = 0 To numh - 1
      ResulDeri(i, 0) = (4 * feval(punto + h(i)) - 3 * feval(punto) - feval(punto + (2 *
h(i)))) / (2 * h(i))
         If error = 1 Then
         ResulDeri(i, 1) = derivadareal - ResulDeri(i, 0)
         End If
   Next i
End If

Else
'Redimensionamos la matriz de resultados
ReDim ResulDeri(0 To numh - 1, 0 To 1) As Double
```

```
'Calculamos la primera derivada para Oh y el error
If Oh2 = True Then
   For i = 0 To numh - 1
      ResulDeri(i, 0) = (fx(1) - fx(0)) / h(i)
         If error = 1 Then
         ResulDeri(i, 1) = derivadareal - ResulDeri(i, 0)
         End If
   Next i
End If
'Calculamos la primera derivada para Oh2 y el error
If Oh4 = True Then
   For i = 0 To numh - 1
      ResulDeri(i, 0) = (4 * fx(1) - 3 * fx(0) - fx(2)) / (2 * h(i))
         If error = 1 Then
         ResulDeri(i, 1) = derivadareal - ResulDeri(i, 0)
         End If
   Next i
End If

End If
Exit Sub
err1:
If err.Number <> 0 Then
   mensaje = MsgBox("Parámetros del algoritmo incompletos", 48, "Error en el
ingreso")
   End If
End Sub
'Permite obtener la segunda derivada de orden Oh2  mediante diferencias
progresivas
Private Sub Progresivassegundaderi()
Dim i As Integer
Dim j As Integer

Dim mensaje As String
'Comprobamos si algún incremento es igual a cero
On Error GoTo err1
If numh = 0 Then
   mensaje = MsgBox("Debe existir por al menos un incremento", 16, "Error en el
ingreso")
   Exit Sub
End If
For i = 0 To numh - 1
   If h(i) = 0 Then
   mensaje = MsgBox("El incremento no puede ser cero", 16, "Error en el ingreso")
   Exit Sub
   End If
```

```
Next i
If Tipofunción = True Then
'Redimensionamos la matriz de resultados
ReDim ResulDeri(0 To numh - 1, 0 To 1) As Double
'Calculamos la segunda derivada para Oh y el error
If Oh4 = True Then
   For i = 0 To numh - 1
      ResulDeri(i, 0) = (feval(punto + (2 * h(i))) - 2 * feval(punto + h(i)) + feval(punto)) /
((h(i)) ^ 2)
         If error = 1 Then
         ResulDeri(i, 1) = derivadareal - ResulDeri(i, 0)
         End If
   Next i
End If

'Calculamos la segunda derivada para Oh2 y el error
If Oh4 = True Then
   For i = 0 To numh - 1
      ResulDeri(i, 0) = (2 * feval(punto) - 5 * feval(punto + h(i)) + 4 * feval(punto + (2 *
h(i))) - fx(punto + (3 * h(i)))) / ((h(i)) ^ 2)
         If error = 1 Then
         ResulDeri(i, 1) = derivadareal - ResulDeri(i, 0)
         End If
   Next i
End If


Else
'Redimensionamos la matriz de resultados
ReDim ResulDeri(0 To numh - 1, 0 To 1) As Double
'Calculamos la segunda derivada para Oh y el error
If Oh4 = True Then
   For i = 0 To numh - 1
      ResulDeri(i, 0) = (fx(2) - 2 * fx(1) + fx(0)) / ((h(i)) ^ 2)
         If error = 1 Then
         ResulDeri(i, 1) = derivadareal - ResulDeri(i, 0)
         End If
   Next i
End If

'Calculamos la segunda derivada para Oh2 y el error
If Oh4 = True Then
   For i = 0 To numh - 1
      ResulDeri(i, 0) = (2 * fx(0) - 5 * fx(1) + 4 * fx(2) - fx(3)) / ((h(i)) ^ 2)
         If error = 1 Then
         ResulDeri(i, 1) = derivadareal - ResulDeri(i, 0)
```

```
        End If
    Next i
End If

End If
Exit Sub
err1:
If err.Number <> 0 Then
    mensaje = MsgBox("Parámetros del algoritmo incompletos", 48, "Error en el
ingreso")
    End If
End Sub

'Permite obtener la primera derivada de orden Oh2  mediante diferencias regresivas
Private Sub Regresivasprimeraderi()
Dim i As Integer
Dim j As Integer

Dim mensaje As String
'Comprobamos si algún incremento es igual a cero
On Error GoTo err1
If numh = 0 Then
    mensaje = MsgBox("Debe existir por al menos un incremento", 16, "Error en el
ingreso")
    Exit Sub
End If
For i = 0 To numh - 1
    If h(i) = 0 Then
    mensaje = MsgBox("El incremento no puede ser cero", 16, "Error en el ingreso")
    Exit Sub
    End If
Next i
If Tipofunción = True Then

'Redimensionamos la matriz de resultados
ReDim ResulDeri(0 To numh - 1, 0 To 1) As Double
'Calculamos la primera derivada para Oh y el error
If Oh2 = True Then
    For i = 0 To numh - 1
        ResulDeri(i, 0) = (feval(punto) - feval(punto - h(i))) / h(i)
            If error = 1 Then
            ResulDeri(i, 1) = derivadareal - ResulDeri(i, 0)
            End If
    Next i
End If
```

```
'Calculamos la primera derivada para Oh2 y el error
If Oh4 = True Then
   For i = 0 To numh - 1
      ResulDeri(i, 0) = (3 * feval(punto) - 4 * feval(punto - h(i)) - feval(punto - (2 * h(i))))
/ (2 * h(i))
         If error = 1 Then
         ResulDeri(i, 1) = derivadareal - ResulDeri(i, 0)
         End If
   Next i
End If

Else
'Redimensionamos la matriz de resultados
ReDim ResulDeri(0 To numh - 1, 0 To 1) As Double
'Calculamos la primera derivada para Oh y el error
If Oh2 = True Then
   For i = 0 To numh - 1
      ResulDeri(i, 0) = (fx(1) - fx(0)) / h(i)
         If error = 1 Then
         ResulDeri(i, 1) = derivadareal - ResulDeri(i, 0)
         End If
   Next i
End If

'Calculamos la primera derivada para Oh2 y el error
If Oh4 = True Then
   For i = 0 To numh - 1
      ResulDeri(i, 0) = (3 * fx(2) - 4 * fx(1) - fx(0)) / (2 * h(i))
         If error = 1 Then
         ResulDeri(i, 1) = derivadareal - ResulDeri(i, 0)
         End If
   Next i
End If

End If
Exit Sub
err1:
If err.Number <> 0 Then
   mensaje = MsgBox("Parámetros del algoritmo incompletos", 48, "Error en el
ingreso")
   End If
End Sub
'Permite obtener la segunda derivada de orden Oh2  mediante diferencias regresivas
Private Sub Regresivassegundaderi()
Dim i As Integer
Dim j As Integer
```

```
Dim mensaje As String
'Comprobamos si algún incremento es igual a cero
On Error GoTo err1
If numh = 0 Then
   mensaje = MsgBox("Debe existir por al menos un incremento", 16, "Error en el
ingreso")
   Exit Sub
End If
For i = 0 To numh - 1
   If h(i) = 0 Then
   mensaje = MsgBox("El incremento no puede ser cero", 16, "Error en el ingreso")
   Exit Sub
   End If
Next i
If Tipofunción = True Then

'Redimensionamos la matriz de resultados
ReDim ResulDeri(0 To numh - 1, 0 To 1) As Double
'Calculamos la primera derivada para Oh y el error
If Oh4 = True Then
   For i = 0 To numh - 1
      ResulDeri(i, 0) = (feval(punto) - 2 * feval(punto - h(i)) + feval(punto - (2 * h(i)))) /
((h(i)) ^ 2)
         If error = 1 Then
         ResulDeri(i, 1) = derivadareal - ResulDeri(i, 0)
         End If
   Next i
End If


'Calculamos la primera derivada para Oh2 y el error
If Oh4 = True Then
   For i = 0 To numh - 1
      ResulDeri(i, 0) = (2 * feval(punto) - 5 * feval(punto - h(i)) + 4 * feval(punto - (2 *
h(i))) - feval(punto - (3 * h(i)))) / ((h(i)) ^ 2)
         If error = 1 Then
         ResulDeri(i, 1) = derivadareal - ResulDeri(i, 0)
         End If
   Next i
End If

Else
'Redimensionamos la matriz de resultados
ReDim ResulDeri(0 To numh - 1, 0 To 1) As Double
'Calculamos la primera derivada para Oh y el error
```

```
If Oh4 = True Then
   For i = 0 To numh - 1
      ResulDeri(i, 0) = (fx(2) - 2 * fx(1) + fx(0)) / ((h(i)) ^ 2)
         If error = 1 Then
         ResulDeri(i, 1) = derivadareal - ResulDeri(i, 0)
         End If
   Next i
End If


'Calculamos la primera derivada para Oh2 y el error
If Oh4 = True Then
   For i = 0 To numh - 1
      ResulDeri(i, 0) = (2 * fx(3) - 5 * fx(2) + 4 * fx(1) - fx(0)) / ((h(i)) ^ 2)
         If error = 1 Then
         ResulDeri(i, 1) = derivadareal - ResulDeri(i, 0)
         End If
   Next i
End If

End If
Exit Sub
err1:
If err.Number <> 0 Then
   mensaje = MsgBox("Parámetros del algoritmo incompletos", 48, "Error en el
ingreso")
   End If
End Sub

Public Sub DiferenciasCentradas()
On Error GoTo err1
Select Case Tipoalgoritmo
'Primera derivada
Case Is = 1
Centradaprimeraderi
'Segunda derivada
Case Is = 2
Centradasegundaderi
End Select
Exit Sub
err1:
End Sub

Public Sub DiferenciasProgresivas()
On Error GoTo err1
Select Case Tipoalgoritmo
```

```
'Primera derivada
Case Is = 3
Progresivasprimeraderi
'Segunda derivada
Case Is = 4
Progresivassegundaderi
End Select
Exit Sub
err1:
End Sub

Public Sub DiferenciasRegresivas()
On Error GoTo err1
Select Case Tipoalgoritmo
'Primera derivada
Case Is = 5
Regresivasprimeraderi
'Segunda derivada
Case Is = 6
Regresivassegundaderi
End Select
Exit Sub
err1:
End Sub


Private Function feval(X As Double) As Double
On Error GoTo err1
Evaluar.Tipox = False
Evaluar.fa = función
Evaluar.PtX = X
Evaluar.evaluarf
feval = Evaluar.Fdy
Exit Function
err1:
End Function
```

CODIGO DEL CLUSTER (5)
Resolver un sistema de ecuaciones diferenciales ordinarias

Interfaz FrmEDO

```
Option Explicit
'En Microsoft TechNet puedes encontrar este articulo:
'HOWTO: Use HTML Help API in a Visual Basic 5.0 Application
'PSS ID Number: Q183434
'
'Aunque la definición de la Enumeración y la primera declaración
'es de las news
'
'Htmlhelp consts
Private Enum HH_COMMAND
   HH_DISPLAY_TOPIC = &H0
   HH_HELP_FINDER = &H0       ' WinHelp equivalent
   HH_DISPLAY_TOC = &H1       ' not currently implemented
   HH_DISPLAY_INDEX = &H2     ' not currently implemented
   HH_DISPLAY_SEARCH = &H3    ' not currently implemented
   HH_SET_WIN_TYPE = &H4
   HH_GET_WIN_TYPE = &H5
   HH_GET_WIN_HANDLE = &H6
   HH_GET_INFO_TYPES = &H7    ' not currently implemented
   HH_SET_INFO_TYPES = &H8    ' not currently implemented
   HH_SYNC = &H9
   HH_ADD_NAV_UI = &HA        ' not currently implemented
   HH_ADD_BUTTON = &HB        ' not currently implemented
   HH_GETBROWSER_APP = &HC    ' not currently implemented
   HH_KEYWORD_LOOKUP = &HD
   HH_DISPLAY_TEXT_POPUP = &HE ' display string resource id
                              ' or text in a popup window
   HH_HELP_CONTEXT = &HF      ' display mapped numeric value
                              ' in dwData
   HH_TP_HELP_CONTEXTMENU     ' Text pop-up help, similar to
                              ' WinHelp's HELP_CONTEXTMENU.
   HH_TP_HELP_WM_HELP = &H11  ' text pop-up help, similar to
                              ' WinHelp's HELP_WM_HELP.
   HH_CLOSE_ALL = &H12        ' close all windows opened directly
                              ' or indirectly by the caller
   HH_ALINK_LOOKUP = &H13     ' ALink version of HH_KEYWORD_LOOKUP
End Enum

'HtmlHelp api call
'NOTA: Si se usa esta forma, hay que indicar el último parámetro
'     con la palabra ByVal delante...
'Private Declare Function HtmlHelp Lib "hhctrl.ocx" Alias "HtmlHelpA" _
```

```
    (ByVal hwndCaller As Long, ByVal pszFile As String, _
    ByVal uCommand As HH_COMMAND, dwData As Any) As Long
'Con esta funciona perfectamente
Private Declare Function HtmlHelp Lib "hhctrl.ocx" Alias "HtmlHelpA" _
    (ByVal hwndCaller As Long, ByVal pszFile As String, _
    ByVal uCommand As HH_COMMAND, ByVal dwData As Long) As Long

Dim Evaluarfedot As New McsExcel
Dim Edo As New ClasEDO
Dim EuHe As Boolean   'Indica si se ha elegido Heuler modificado o Heun
Dim tipoalg As Integer
Dim n As Double
Dim arribaSstab
Dim izquierdaSstab
Dim anchoSstab
Dim altoSstab
Dim colororg
Private Acepta As Boolean
Private VLConexion As ADODB.Connection
Private VLRegistro As ADODB.Recordset
Private VLNombre As String
Private VLNombreTrabajo As String
Private VLVerifica As Boolean
Private VLBDD As Boolean
Private VLSistema As String
Private VLSistemaMatriz(1 To 1000) As Integer
Private VLValores As String
Private VLValoresMatriz(1 To 1000) As Integer
Private VLIndice As Integer
Private VLRec As Boolean

Property Let Aceptar(parAceptar As Boolean)
    Acepta = parAceptar
End Property

Property Let Nombre(parNombre As String)
    VLNombre = parNombre
End Property

Property Let NombreTrabajo(parNombreTrabajo As String)
    VLNombreTrabajo = parNombreTrabajo
End Property


Private Sub CmdEdocalcular_Click()
Dim mensaje As String
```

```
Dim verifica As Double
Dim valoresini() As Double
Dim i As Integer
Dim final As Integer
Dim Fun() As String

Select Case tipoalg
Case Is = 1
If (TxtEdofunción = "" Or TxtEdofunción = " " Or TxtEdoa = "" Or TxtEdoa = " " Or
TxtEdob = "" Or TxtEdob = " " Or _
TxtEdoh = "" Or TxtEdoh = " " Or TxtEdoinicial = "" Or TxtEdoinicial = " ") Then
mensaje = MsgBox("Parámetros del algoritmo incompletos", 48, "Error en el ingreso")
Exit Sub
End If

verifica = reemplazafedo(TxtEdofunción, 500, 500)
'manejo del error en el ingreso de la función
If TxtEdofunción = " " Then
mensaje = MsgBox("Debe ingresar una función", 16, "Error en el ingreso")
Exit Sub
Else
If verifica = 8121975 Then
mensaje = MsgBox("Debe ingresar la función de acuerdo a la sintaxis especificada",
16, "Error en el ingreso")
Exit Sub
End If
End If
SetearEdo
Edo.Euler
EdoPresentar
Case Is = 2
If (TxtEdofunción = "" Or TxtEdofunción = " " Or TxtEdoa = "" Or TxtEdoa = " " Or
TxtEdob = "" Or TxtEdob = " " Or _
TxtEdoh = "" Or TxtEdoh = " " Or TxtEdoinicial = "" Or TxtEdoinicial = " ") Then
mensaje = MsgBox("Parámetros del algoritmo incompletos", 48, "Error en el ingreso")
Exit Sub
End If
verifica = reemplazafedo(TxtEdofunción, 500, 500)
'manejo del error en el ingreso de la función
If TxtEdofunción = " " Then
mensaje = MsgBox("Debe ingresar una función", 16, "Error en el ingreso")
Exit Sub
Else
If verifica = 8121975 Then
mensaje = MsgBox("Debe ingresar la función de acuerdo a la sintaxis especificada",
16, "Error en el ingreso")
```

```
Exit Sub
End If
End If
SetearEdo
Edo.EulermodificadoyHeun
EdoPresentar
Case Is = 3
If (TxtEdofunción = "" Or TxtEdofunción = " " Or TxtEdoa = "" Or TxtEdoa = " " Or
TxtEdob = "" Or TxtEdob = " " Or _
TxtEdoh = "" Or TxtEdoh = " " Or TxtEdoinicial = "" Or TxtEdoinicial = " ") Then
mensaje = MsgBox("Parámetros del algoritmo incompletos", 48, "Error en el ingreso")
Exit Sub
End If
verifica = reemplazafedo(TxtEdofunción, 500, 500)
'manejo del error en el ingreso de la función
If TxtEdofunción = " " Then
mensaje = MsgBox("Debe ingresar una función", 16, "Error en el ingreso")
Exit Sub
Else
If verifica = 8121975 Then
mensaje = MsgBox("Debe ingresar la función de acuerdo a la sintaxis especificada",
16, "Error en el ingreso")
Exit Sub
End If
End If
SetearEdo
Edo.EulermodificadoyHeun
EdoPresentar
Case Is = 4
If (TxtEdofunción = "" Or TxtEdofunción = " " Or TxtEdoa = "" Or TxtEdoa = " " Or
TxtEdob = "" Or TxtEdob = " " Or _
TxtEdoh = "" Or TxtEdoh = " " Or TxtEdoinicial = "" Or TxtEdoinicial = " ") Then
mensaje = MsgBox("Parámetros del algoritmo incompletos", 48, "Error en el ingreso")
Exit Sub
End If
verifica = reemplazafedo(TxtEdofunción, 500, 500)
'manejo del error en el ingreso de la función
If TxtEdofunción = " " Then
mensaje = MsgBox("Debe ingresar una función", 16, "Error en el ingreso")
Exit Sub
Else
If verifica = 8121975 Then
mensaje = MsgBox("Debe ingresar la función de acuerdo a la sintaxis especificada",
16, "Error en el ingreso")
Exit Sub
End If
```

```
End If
SetearEdo
Edo.RKF4
EdoPresentar
Case Is = 5
If (TxtEdofunción = "" Or TxtEdofunción = " " Or TxtEdoa = "" Or TxtEdoa = " " Or
TxtEdob = "" Or TxtEdob = " " Or _
TxtEdoh = "" Or TxtEdoh = " " Or TxtEdoinicial = "" Or TxtEdoinicial = " ") Then
mensaje = MsgBox("Parámetros del algoritmo incompletos", 48, "Error en el ingreso")
Exit Sub
End If
verifica = reemplazafedo(TxtEdofunción, 500, 500)
'manejo del error en el ingreso de la función
If TxtEdofunción = " " Then
mensaje = MsgBox("Debe ingresar una función", 16, "Error en el ingreso")
Exit Sub
Else
If verifica = 8121975 Then
mensaje = MsgBox("Debe ingresar la función de acuerdo a la sintaxis especificada",
16, "Error en el ingreso")
Exit Sub
End If
End If
SetearEdo
Edo.AdamsBashforthMoulton
EdoPresentar
Case Is = 6
If (TxtEdofunción = "" Or TxtEdofunción = " " Or TxtEdoa = "" Or TxtEdoa = " " Or
TxtEdob = "" Or TxtEdob = " " Or _
TxtEdoh = "" Or TxtEdoh = " " Or TxtEdoinicial = "" Or TxtEdoinicial = " ") Then
mensaje = MsgBox("Parámetros del algoritmo incompletos", 48, "Error en el ingreso")
Exit Sub
End If
verifica = reemplazafedo(TxtEdofunción, 500, 500)
'manejo del error en el ingreso de la función
If TxtEdofunción = " " Then
mensaje = MsgBox("Debe ingresar una función", 16, "Error en el ingreso")
Exit Sub
Else
If verifica = 8121975 Then
mensaje = MsgBox("Debe ingresar la función de acuerdo a la sintaxis especificada",
16, "Error en el ingreso")
Exit Sub
End If
End If
SetearEdo
```

```
Edo.MileneSimpson
EdoPresentar
Case Is = 7
If (TxtEdonum = "" Or TxtEdonum = " " Or TxtEdoa = "" Or TxtEdoa = " " Or TxtEdob
= "" Or TxtEdob = " " Or _
TxtEdoh = "" Or TxtEdoh = " " Or TxtEdoEdit1 = "" Or TxtEdoEdit1 = " " Or
TxtEdoEdit2 = "" Or TxtEdoEdit2 = " ") Then
mensaje = MsgBox("Parámetros del algoritmo incompletos", 48, "Error en el ingreso")
Exit Sub
End If
If TxtEdonum = " " Then
mensaje = MsgBox("Debe ingresar el numero de función del sistema", 16, "Error en
el ingreso")
Exit Sub
Else
ReDim valoresini(0 To Val(TxtEdonum)) As Double
For i = 0 To Val(TxtEdonum)
valoresini(i) = i + 1
Next i
'copiamos los valores de FgEdosistema a la matriz pares
final = fgEdosistema.Rows
ReDim Fun(0 To final - 2) As String
For i = 1 To final - 1
Fun(i - 1) = fgEdosistema.TextMatrix(i, 1)
Next i
For i = 0 To Val(TxtEdonum) - 1
verifica = Evaluarsistema(Fun(i), valoresini)
If verifica = 8121975 Then
mensaje = MsgBox("Debe ingresar la función de acuerdo a la sintaxis especificada",
16, "Error en el ingreso")
Exit Sub
End If
Next i
End If
SetearEdo
Edo.SistemasEDO
EdoPresentar
Case Is = 8
If (TxtEdofinal = "" Or TxtEdofinal = " " Or TxtEdoa = "" Or TxtEdoa = " " Or TxtEdob =
"" Or TxtEdob = " " Or _
TxtEdoh = "" Or TxtEdoh = " " Or TxtEdop = "" Or TxtEdop = " " Or TxtEdoq = "" Or
TxtEdoq = " " Or Txtedor = "" Or Txtedor = " " Or TxtEdoinicial = "" Or TxtEdoinicial = "
") Then
mensaje = MsgBox("Parámetros del algoritmo incompletos", 48, "Error en el ingreso")
Exit Sub
End If
```

```
verifica = Evaluart(TxtEdop, 500)
'manejo del error en el ingreso de la función
If TxtEdop = " " Then
mensaje = MsgBox("Debe ingresar una función", 16, "Error en el ingreso")
Exit Sub
Else
If verifica = 8121975 Then
mensaje = MsgBox("Debe ingresar la función de acuerdo a la sintaxis especificada",
16, "Error en el ingreso")
Exit Sub
End If
End If

verifica = Evaluart(TxtEdoq, 500)
'manejo del error en el ingreso de la función
If TxtEdoq = " " Then
mensaje = MsgBox("Debe ingresar una función", 16, "Error en el ingreso")
Exit Sub
Else
If verifica = 8121975 Then
mensaje = MsgBox("Debe ingresar la función de acuerdo a la sintaxis especificada",
16, "Error en el ingreso")
Exit Sub
End If
End If

verifica = Evaluart(Txtedor, 500)
'manejo del error en el ingreso de la función
If Txtedor = " " Then
mensaje = MsgBox("Debe ingresar una función", 16, "Error en el ingreso")
Exit Sub
Else
If verifica = 8121975 Then
mensaje = MsgBox("Debe ingresar la función de acuerdo a la sintaxis especificada",
16, "Error en el ingreso")
Exit Sub
End If
End If
SetearEdo
Edo.DisparoLineal
EdoPresentar
End Select
   Edoimprimir.Enabled = True
End Sub
```

```
Private Sub Command1_Click()
   PLImprimir
End Sub

Private Sub Edoabrir_Click()
   FrmRecuperaTrabajo.Show vbModal
   If Acepta = True Then
      PLConexion True
      PLRecupera
   End If
End Sub

Private Sub Edoayuda_Click()
'Así se llamaría para mostrar un tópico de la ayuda
   Dim h As Long

   'h = HtmlHelp(Me.hWnd, "Milton.hlp", HH_DISPLAY_TOPIC, 0&)

   h = HtmlHelp(Me.hWnd, "Milton.hlp", HH_HELP_CONTEXT, 2080&)
End Sub

Private Sub Edoeuler_Click()
'Seteamos el tipo de algoritmo
tipoalg = 1
   Edoabrir.Enabled = True
   Edoguardarcomo.Enabled = True
   FrmRecuperaTrabajo.ver = tipoalg
LblNomalg.Caption = "Método de Euler"
'Presentamos los controles necesarios
LblEdofunción.Visible = True
TxtEdofunción.Text = " "
TxtEdofunción.Visible = True
LblEdonum.Visible = False
TxtEdonum.Visible = False
LblEdoa.Visible = True
TxtEdoa.Text = " "
TxtEdoa.Visible = True
LblEdob.Visible = True
TxtEdob.Text = " "
TxtEdob.Visible = True
LblEdoh.Visible = True
TxtEdoh.Text = " "
TxtEdoh.Visible = True
LblEdotitulo.Visible = False
LblEdotitulo2.Visible = False
```

```
fgEdosistema.Visible = False
fgEdoiniciales.Visible = False
LblEdoinicial.Visible = True
TxtEdoinicial.Text = " "
TxtEdoinicial.Visible = True
LblEdofinal.Visible = False
TxtEdofinal.Text = " "
TxtEdofinal.Visible = False
LblEdop.Visible = False
TxtEdop.Text = " "
TxtEdop.Visible = False
LblEdoq.Visible = False
TxtEdoq.Text = " "
TxtEdoq.Visible = False
LblEdor.Visible = False
Txtedor.Text = " "
Txtedor.Visible = False
fgEdoresultados.Clear
LblEdoSintaxis.Caption = "Y' = es la ecuación diferencial ordinaria Y'=f(t,y) y'=(t-y)/2"
& Chr(10) & "Puede usar las funciones con la variable t o y:
Ln(t);COS(t);SQRT(t);Log(t); Tan(t); PI(), el resto de funciones sigue la sintaxis de
Excel" & Chr(10) & _
"[a,b]= intervalo en el cual se busca la respuesta" & Chr(10) & "h= tamaño de paso" &
Chr(10) & " Y(a)= condición inicial, valor de la función Y a t=a"
End Sub

Private Sub Edograficar_Click()
Frmescalas.Show
End Sub

Private Sub Edoguardarcomo_Click()
   FrmNombre.Show vbModal
   If Acepta = True Then
      PLConexion True
      PLGuardarBDD
   End If
End Sub

Private Sub Edoimprimir_Click()
 Dim EndTime As Date
   Sincolor

      EndTime = DateAdd("s", 1 / 1E+16, Now)
      Do Until Now > EndTime
        DoEvents
      Loop
```

```
    'Form1.Command3_Click
    Set Form1.Picture1.Picture = CaptureClient(Me)
    Form1.Visible = True
    FrmPrincipalalgoritmos.Visible = False
    Concolor
End Sub

Private Sub Edomodificado_Click()
EuHe = True
'Seteamos el tipo de algoritmo
tipoalg = 2
    Edoabrir.Enabled = True
    Edoguardarcomo.Enabled = True
    FrmRecuperaTrabajo.ver = tipoalg
LblNomalg.Caption = "Método de Euler Modificado"
'Presentamos los controles necesarios
LblEdofunción.Visible = True
TxtEdofunción.Text = " "
TxtEdofunción.Visible = True
LblEdonum.Visible = False
TxtEdonum.Visible = False
LblEdoa.Visible = True
TxtEdoa.Text = " "
TxtEdoa.Visible = True
LblEdob.Visible = True
TxtEdob.Text = " "
TxtEdob.Visible = True
LblEdoh.Visible = True
TxtEdoh.Text = " "
TxtEdoh.Visible = True
LblEdotitulo.Visible = False
LblEdotitulo2.Visible = False
fgEdosistema.Visible = False
fgEdoiniciales.Visible = False
LblEdoinicial.Visible = True
TxtEdoinicial.Text = " "
TxtEdoinicial.Visible = True
LblEdofinal.Visible = False
TxtEdofinal.Text = " "
TxtEdofinal.Visible = False
LblEdop.Visible = False
TxtEdop.Text = " "
TxtEdop.Visible = False
LblEdoq.Visible = False
TxtEdoq.Text = " "
TxtEdoq.Visible = False
```

```
LblEdor.Visible = False
Txtedor.Text = " "
Txtedor.Visible = False
fgEdoresultados.Clear
LblEdoSintaxis.Caption = "Y' = es la ecuación diferencial ordinaria Y'=f(t,y) y'=(t-y)/2"
& Chr(10) & "Puede usar las funciones con la variable t o y:
Ln(t);COS(t);SQRT(t);Log(t); Tan(t); PI(), el resto de funciones sigue la sintaxis de
Excel" & Chr(10) & _
"[a,b]= intervalo en el cual se busca la respuesta" & Chr(10) & "h= tamaño de paso" &
Chr(10) & " Y(a)= condición inicial, valor de la función Y a t=a"
End Sub

Private Sub Edoheun_Click()
EuHe = False
'Seteamos el tipo de algoritmo
tipoalg = 3
    Edoabrir.Enabled = True
    Edoguardarcomo.Enabled = True
    FrmRecuperaTrabajo.ver = tipoalg
LblNomalg.Caption = "Método de Heun"
'Presentamos los controles necesarios
LblEdofunción.Visible = True
TxtEdofunción.Text = " "
TxtEdofunción.Visible = True
LblEdonum.Visible = False
TxtEdonum.Visible = False
LblEdoa.Visible = True
TxtEdoa.Text = " "
TxtEdoa.Visible = True
LblEdob.Visible = True
TxtEdob.Text = " "
TxtEdob.Visible = True
LblEdoh.Visible = True
TxtEdoh.Text = " "
TxtEdoh.Visible = True
LblEdotitulo.Visible = False
LblEdotitulo2.Visible = False
fgEdosistema.Visible = False
fgEdoiniciales.Visible = False
LblEdoinicial.Visible = True
TxtEdoinicial.Text = " "
TxtEdoinicial.Visible = True
LblEdofinal.Visible = False
TxtEdofinal.Text = " "
TxtEdofinal.Visible = False
LblEdop.Visible = False
```

```
TxtEdop.Text = " "
TxtEdop.Visible = False
LblEdoq.Visible = False
TxtEdoq.Text = " "
TxtEdoq.Visible = False
LblEdor.Visible = False
Txtedor.Text = " "
Txtedor.Visible = False
fgEdoresultados.Clear
LblEdoSintaxis.Caption = "Y' = es la ecuación diferencial ordinaria Y'=f(t,y) y'=(t-y)/2"
& Chr(10) & "Puede usar las funciones con la variable t o y:
Ln(t);COS(t);SQRT(t);Log(t); Tan(t); PI(), el resto de funciones sigue la sintaxis de
Excel" & Chr(10) & _
"[a,b]= intervalo en el cual se busca la respuesta" & Chr(10) & "h= tamaño de paso" &
Chr(10) & " Y(a)= condición inicial, valor de la función Y a t=a"
End Sub

Private Sub Edorunge_Click()
'Seteamos el tipo de algoritmo
tipoalg = 4
   Edoabrir.Enabled = True
   Edoguardarcomo.Enabled = True
   FrmRecuperaTrabajo.ver = tipoalg
LblNomalg.Caption = "Método de Runge-Kutta de cuarto orden"
'Presentamos los controles necesarios
LblEdofunción.Visible = True
TxtEdofunción.Text = " "
TxtEdofunción.Visible = True
LblEdonum.Visible = False
TxtEdonum.Visible = False
LblEdoa.Visible = True
TxtEdoa.Text = " "
TxtEdoa.Visible = True
LblEdob.Visible = True
TxtEdob.Text = " "
TxtEdob.Visible = True
LblEdoh.Visible = True
TxtEdoh.Text = " "
TxtEdoh.Visible = True
LblEdotitulo.Visible = False
LblEdotitulo2.Visible = False
fgEdosistema.Visible = False
fgEdoiniciales.Visible = False
LblEdoinicial.Visible = True
TxtEdoinicial.Text = " "
TxtEdoinicial.Visible = True
```

```
LblEdofinal.Visible = False
TxtEdofinal.Text = " "
TxtEdofinal.Visible = False
LblEdop.Visible = False
TxtEdop.Text = " "
TxtEdop.Visible = False
LblEdoq.Visible = False
TxtEdoq.Text = " "
TxtEdoq.Visible = False
LblEdor.Visible = False
Txtedor.Text = " "
Txtedor.Visible = False
LblEdoSintaxis.Caption = "Y' = es la ecuación diferencial ordinaria Y'=f(t,y) y'=(t-y)/2"
& Chr(10) & "Puede usar las funciones con la variable t o y:
Ln(t);COS(t);SQRT(t);Log(t); Tan(t); PI(), el resto de funciones sigue la sintaxis de
Excel" & Chr(10) & _
"[a,b]= intervalo en el cual se busca la respuesta" & Chr(10) & "h= tamaño de paso" &
Chr(10) & " Y(a)= condición inicial, valor de la función Y a t=a"
End Sub

Private Sub Edoadams_Click()
'Seteamos el tipo de algoritmo
tipoalg = 5
   Edoabrir.Enabled = True
   Edoguardarcomo.Enabled = True
   FrmRecuperaTrabajo.ver = tipoalg
LblNomalg.Caption = "Método de Adams-Bashforth-Moulton"
'Presentamos los controles necesarios
LblEdofunción.Visible = True
TxtEdofunción.Text = " "
TxtEdofunción.Visible = True
LblEdonum.Visible = False
TxtEdonum.Visible = False
LblEdoa.Visible = True
TxtEdoa.Text = " "
TxtEdoa.Visible = True
LblEdob.Visible = True
TxtEdob.Text = " "
TxtEdob.Visible = True
LblEdoh.Visible = True
TxtEdoh.Text = " "
TxtEdoh.Visible = True
LblEdotitulo.Visible = False
LblEdotitulo2.Visible = False
fgEdosistema.Visible = False
fgEdoiniciales.Visible = False
```

```
LblEdoinicial.Visible = True
TxtEdoinicial.Text = " "
TxtEdoinicial.Visible = True
LblEdofinal.Visible = False
TxtEdofinal.Text = " "
TxtEdofinal.Visible = False
LblEdop.Visible = False
TxtEdop.Text = " "
TxtEdop.Visible = False
LblEdoq.Visible = False
TxtEdoq.Text = " "
TxtEdoq.Visible = False
LblEdor.Visible = False
Txtedor.Text = " "
Txtedor.Visible = False
fgEdoresultados.Clear
LblEdoSintaxis.Caption = "Y' = es la ecuación diferencial ordinaria Y'=f(t,y) y'=(t-y)/2"
& Chr(10) & "Puede usar las funciones con la variable t o y:
Ln(t);COS(t);SQRT(t);Log(t); Tan(t); PI(), el resto de funciones sigue la sintaxis de
Excel" & Chr(10) & _
"[a,b]= intervalo en el cual se busca la respuesta" & Chr(10) & "h= tamaño de paso" &
Chr(10) & " Y(a)= condición inicial, valor de la función Y a t=a"
End Sub

Private Sub Edomilene_Click()
'Seteamos el tipo de algoritmo
tipoalg = 6
   Edoabrir.Enabled = True
   Edoguardarcomo.Enabled = True
   FrmRecuperaTrabajo.ver = tipoalg
LblNomalg.Caption = "Método de Milene-Simpson"
'Presentamos los controles necesarios
LblEdofunción.Visible = True
TxtEdofunción.Text = " "
TxtEdofunción.Visible = True
LblEdonum.Visible = False
TxtEdonum.Visible = False
LblEdoa.Visible = True
TxtEdoa.Text = " "
TxtEdoa.Visible = True
LblEdob.Visible = True
TxtEdob.Text = " "
TxtEdob.Visible = True
LblEdoh.Visible = True
TxtEdoh.Text = " "
TxtEdoh.Visible = True
```

```
LblEdotitulo.Visible = False
LblEdotitulo2.Visible = False
fgEdosistema.Visible = False
fgEdoiniciales.Visible = False
LblEdoinicial.Visible = True
TxtEdoinicial.Text = " "
TxtEdoinicial.Visible = True
LblEdofinal.Visible = False
TxtEdofinal.Text = " "
TxtEdofinal.Visible = False
LblEdop.Visible = False
TxtEdop.Text = " "
TxtEdop.Visible = False
LblEdoq.Visible = False
TxtEdoq.Text = " "
TxtEdoq.Visible = False
LblEdor.Visible = False
Txtedor.Text = " "
Txtedor.Visible = False
fgEdoresultados.Clear
LblEdoSintaxis.Caption = "Y' = es la ecuación diferencial ordinaria Y'=f(t,y) y'=(t-y)/2"
& Chr(10) & "Puede usar las funciones con la variable t o y:
Ln(t);COS(t);SQRT(t);Log(t); Tan(t); PI(), el resto de funciones sigue la sintaxis de
Excel" & Chr(10) & _
"[a,b]= intervalo en el cual se busca la respuesta" & Chr(10) & "h= tamaño de paso" &
Chr(10) & " Y(a)= condición inicial, valor de la función Y a t=a"
End Sub

Private Sub Edosalir_Click()
Unload Me
End Sub

Private Sub Edosistemasrunge_Click()
'Seteamos el tipo de algoritmo
tipoalg = 7
   Edoabrir.Enabled = True
   Edoguardarcomo.Enabled = True
   FrmRecuperaTrabajo.ver = tipoalg
LblNomalg.Caption = "Método de Runge-Kutta de cuarto orden (SEDO)"
'Presentamos los controles necesarios
LblEdofunción.Visible = False
TxtEdofunción.Text = " "
TxtEdofunción.Visible = False
LblEdonum.Visible = True
TxtEdonum.Text = " "
TxtEdonum.Visible = True
```

```
LblEdoa.Visible = True
TxtEdoa.Text = " "
TxtEdoa.Visible = True
LblEdob.Visible = True
TxtEdob.Text = " "
TxtEdob.Visible = True
LblEdoh.Visible = True
TxtEdoh.Text = " "
TxtEdoh.Visible = True
LblEdotitulo.Visible = False
LblEdotitulo2.Visible = False
fgEdosistema.Clear
fgEdosistema.Visible = False
fgEdoiniciales.Clear
fgEdoiniciales.Visible = False
LblEdoinicial.Visible = False
TxtEdoinicial.Text = " "
TxtEdoinicial.Visible = False
LblEdofinal.Visible = False
TxtEdofinal.Text = " "
TxtEdofinal.Visible = False
LblEdop.Visible = False
TxtEdop.Text = " "
TxtEdop.Visible = False
LblEdoq.Visible = False
TxtEdoq.Text = " "
TxtEdoq.Visible = False
LblEdor.Visible = False
Txtedor.Text = " "
Txtedor.Visible = False
fgEdoresultados.Clear
LblEdoSintaxis.Caption = " Número de EDOs= número de ecuaciones diferenciales
ordinarias a resolver" & Chr(10) & "dyn/dt = es la ecuación diferencial ordinaria n-
esima dyn/dt=f(t,y1,y2,..,yn)" & Chr(10) & "Puede usar las funciones con la variable t
o y: Ln(t);COS(t);SQRT(t);Log(t); Tan(t); PI(), el resto de funciones sigue la sintaxis
de Excel" & Chr(10) & _
" y1(a)=,y2(a),..yn(a)= valores iniciales" & Chr(10) & "dy1/dt= y1+2*y2; dy2/dt=
3*y1+2*y2: con valores iniciales y1(0}=6; y2(0)=4 " & Chr(10) & "[a,b]= intervalo en el
cual se busca la respuesta" & Chr(10) & "h= tamaño de paso" & Chr(10) & " Yn(a)=
condición inicial, valor de la función Yn a t=a"
End Sub
Private Sub Edodisparo_Click()
'Seteamos el tipo de algoritmo
tipoalg = 8
   Edoabrir.Enabled = True
   Edoguardarcomo.Enabled = True
```

```
    FrmRecuperaTrabajo.ver = tipoalg
LblNomalg.Caption = "Método del Disparo Lineal"
'Presentamos los controles necesarios
LblEdofunción.Visible = False
TxtEdofunción.Text = " "
TxtEdofunción.Visible = False
LblEdonum.Visible = False
TxtEdonum.Visible = False
LblEdoa.Visible = True
TxtEdoa.Text = " "
TxtEdoa.Visible = True
LblEdob.Visible = True
TxtEdob.Text = " "
TxtEdob.Visible = True
LblEdoh.Visible = True
TxtEdoh.Text = " "
TxtEdoh.Visible = True
LblEdotitulo.Visible = False
LblEdotitulo2.Visible = False
fgEdosistema.Visible = False
fgEdoiniciales.Visible = False
LblEdoinicial.Visible = True
TxtEdoinicial.Text = " "
TxtEdoinicial.Visible = True
LblEdofinal.Visible = True
TxtEdofinal.Text = " "
TxtEdofinal.Visible = True
LblEdop.Visible = True
TxtEdop.Text = " "
TxtEdop.Visible = True
LblEdoq.Visible = True
TxtEdoq.Text = " "
TxtEdoq.Visible = True
LblEdor.Visible = True
Txtedor.Text = " "
Txtedor.Visible = True
TxtEdoEdit1.Text = " "
TxtEdoEdit1.Visible = False
TxtEdoEdit2.Text = " "
TxtEdoEdit2.Visible = False
fgEdoresultados.Clear
LblEdoSintaxis.Caption = "X""(t)= p(t) X'(t)+ q(t) X(t)+ r(t) es ecuación diferencial de
contorno lineal " & Chr(10) & "Puede usar las funciones : Ln(t);COS(t);SQRT(t);Log(t);
Tan(t); PI(), el resto de funciones sigue la sintaxis de Excel" & Chr(10) & _
"X""(t)= ((2*t)/(1+t^2)) X'(t) -(2/(1+t^2)) X(t)+ 1" & Chr(10) & "[a,b]= intervalo en el cual
se busca la respuesta" & Chr(10) & "h= tamaño de paso" & Chr(10) & " Y(a)=
```

```
condición inicial, valor de la función Y a t=a" & Chr(10) & " Y(b)= condición en la
frontera, valor de la función Y a t=b"
End Sub

Private Sub Form_Load()
   Edoabrir.Enabled = False
   Edoguardarcomo.Enabled = False
   Edoimprimir.Enabled = False
   VGForma = "EDO"
End Sub




Private Sub TxtEdoa_Change()
LTrim (TxtEdoa)
End Sub

Private Sub TxtEdoa_KeyPress(KeyAscii As Integer)
'filtra la entrada de datos que no sean números incluye el signo - y el punto
  If (KeyAscii < 48 Or KeyAscii > 57) Then
      If (KeyAscii = 45) Then
      Exit Sub
      End If
      If (KeyAscii <> 46) Then
         If (KeyAscii <> 8) Then
         KeyAscii = 0
         End If
      End If
   End If
  If KeyAscii = Asc(vbCr) Then ' Elimina los retornos para quitar los pitidos.
  KeyAscii = 0
  End If

End Sub

Private Sub TxtEdob_Change()
LTrim (TxtEdob)
End Sub

Private Sub TxtEdob_KeyPress(KeyAscii As Integer)
'filtra la entrada de datos que no sean números incluye el signo - y el punto
  If (KeyAscii < 48 Or KeyAscii > 57) Then
      If (KeyAscii = 45) Then
      Exit Sub
```

```
        End If
        If (KeyAscii <> 46) Then
           If (KeyAscii <> 8) Then
           KeyAscii = 0
           End If
        End If
     End If
    If KeyAscii = Asc(vbCr) Then ' Elimina los retornos para quitar los pitidos.
    KeyAscii = 0
    End If

End Sub

Private Sub TxtEdofinal_Change()
LTrim (TxtEdofinal)
End Sub

Private Sub TxtEdofinal_KeyPress(KeyAscii As Integer)
'filtra la entrada de datos que no sean números incluye el signo - y el punto
    If (KeyAscii < 48 Or KeyAscii > 57) Then
        If (KeyAscii = 45) Then
        Exit Sub
        End If
        If (KeyAscii <> 46) Then
           If (KeyAscii <> 8) Then
           KeyAscii = 0
           End If
        End If
     End If
    If KeyAscii = Asc(vbCr) Then ' Elimina los retornos para quitar los pitidos.
    KeyAscii = 0
    End If

End Sub

Private Sub TxtEdofunción_Change()
LTrim (TxtEdofunción)
End Sub



Private Sub TxtEdoh_Change()
LTrim (TxtEdoh)
End Sub

Private Sub TxtEdoh_KeyPress(KeyAscii As Integer)
```

```
'filtra la entrada de datos que no sean números incluye el signo - y el punto
  If (KeyAscii < 48 Or KeyAscii > 57) Then
      If (KeyAscii = 45) Then
      Exit Sub
      End If
      If (KeyAscii <> 46) Then
         If (KeyAscii <> 8) Then
         KeyAscii = 0
         End If
      End If
   End If
  If KeyAscii = Asc(vbCr) Then ' Elimina los retornos para quitar los pitidos.
  KeyAscii = 0
  End If

End Sub

Private Sub TxtEdoinicial_Change()
LTrim (TxtEdoinicial)
End Sub

Private Sub TxtEdoinicial_KeyPress(KeyAscii As Integer)
'filtra la entrada de datos que no sean números incluye el signo - y el punto
  If (KeyAscii < 48 Or KeyAscii > 57) Then
      If (KeyAscii = 45) Then
      Exit Sub
      End If
      If (KeyAscii <> 46) Then
         If (KeyAscii <> 8) Then
         KeyAscii = 0
         End If
      End If
   End If
  If KeyAscii = Asc(vbCr) Then ' Elimina los retornos para quitar los pitidos.
  KeyAscii = 0
  End If

End Sub

Private Sub txtEdonum_KeyPress(KeyAscii As Integer)


If (KeyAscii < 48 Or KeyAscii > 57) Then
If (KeyAscii <> 8) Then KeyAscii = 0

End If
```

```
End Sub

Private Sub txtEdonum_Change()
Dim mensaje As String
Dim i As Integer

LTrim (TxtEdonum)
If Val(TxtEdonum) > 50 Then
mensaje = MsgBox("Fuera del rango", 16, "Error en el ingreso")
TxtEdonum.Text = " "
Else
n = Val(TxtEdonum)
   If n = 0 Then
   n = 1
   End If
fgEdoiniciales.Cols = 2
fgEdoiniciales.ColWidth(0) = 800
fgEdoiniciales.ColWidth(1) = 1000
fgEdoiniciales.ColAlignmentFixed(0) = 4
fgEdoiniciales.ColAlignmentFixed(1) = 4

fgEdoiniciales.Rows = n + 1
fgEdoiniciales.FixedRows = 1
fgEdoiniciales.FixedCols = 1
fgEdoiniciales.TextArray(Fgi(0, 1)) = "Valores"
   For i = 1 To n
   fgEdoiniciales.TextArray(Fgi(i, 0)) = "y" & i & "(a)="
   Next i
fgEdoiniciales.Visible = True
LblEdotitulo2.Visible = True
'Seteamos fgEdosistema
fgEdosistema.Cols = 2
fgEdosistema.ColWidth(0) = 800
fgEdosistema.ColWidth(1) = 1500

fgEdosistema.ColAlignmentFixed(0) = 4
fgEdoiniciales.ColAlignmentFixed(1) = 4

fgEdosistema.Rows = n + 1
fgEdosistema.FixedRows = 1
fgEdosistema.FixedCols = 1
fgEdosistema.TextArray(Fgi(0, 1)) = "Ecuaciones"
   For i = 1 To n
   fgEdosistema.TextArray(Fgi(i, 0)) = "dy" & i & "/dt="
   Next i
```

```
fgEdosistema.Visible = True
LblEdotitulo.Visible = True
End If




End Sub

'Manejo de fgEdoiniciales  para ingreso de datos e incrementos respectivamente
Function Fgi(r As Integer, c As Integer) As Long
  Fgi = c + fgEdoiniciales.Cols * r
End Function

Sub fgEdoiniciales_KeyPress(KeyAscii As Integer)
  MSHFlexGridEdit fgEdoiniciales, TxtEdoEdit2, KeyAscii
End Sub

Sub fgEdoiniciales_DblClick()
  MSHFlexGridEdit fgEdoiniciales, TxtEdoEdit2, 32 ' Simula un espacio.
End Sub
Sub MSHFlexGridEdit(MSHFlexGrid As Control, _
Edt As Control, KeyAscii As Integer)

  ' Usar el carácter escrito.
  Select Case KeyAscii

  ' Un espacio significa modificar el texto actual.
  Case 0 To 32
    Edt = MSHFlexGrid
    Edt.SelStart = 1000

  ' Otro carácter reemplaza el texto actual.
  Case Else
    Edt = Chr(KeyAscii)
    Edt.SelStart = 1
  End Select

  ' Mostrar Edt en la posición correcta.
  Edt.Move MSHFlexGrid.Left + MSHFlexGrid.CellLeft, _
    MSHFlexGrid.Top + MSHFlexGrid.CellTop, _
    MSHFlexGrid.CellWidth - 8, _
    MSHFlexGrid.CellHeight - 8
  Edt.Visible = True
```

```
  ' Y hacer que funcione.
  Edt.SetFocus
End Sub

Sub TxTEdoEdit2_KeyPress(KeyAscii As Integer)
'filtra la entrada de datos que no sean números incluye el signo - y el punto
  If (KeyAscii < 48 Or KeyAscii > 57) Then
      If (KeyAscii = 45) Then
      Exit Sub
      End If
      If (KeyAscii <> 46) Then
         If (KeyAscii <> 8) Then
         KeyAscii = 0
         End If
      End If
   End If
  If KeyAscii = Asc(vbCr) Then ' Elimina los retornos para quitar los pitidos.
  KeyAscii = 0
  End If
End Sub

Sub TxTEdoEdit2_KeyDown(KeyCode As Integer, _
Shift As Integer)
  EditKeyCode1 fgEdoiniciales, TxtEdoEdit2, KeyCode, Shift
End Sub

Sub EditKeyCode1(MSHFlexGrid As Control, Edt As _
Control, KeyCode As Integer, Shift As Integer)

  ' Procesamiento del control de edición estándar.
  Select Case KeyCode

  Case 27   ' ESC: ocultar, devuelve el enfoque a          ' MSFlexGrid.
    Edt.Visible = False
    MSHFlexGrid.SetFocus

  Case 13   ' ENTRAR devuelve el enfoque a MSHFlexGrid.
    MSHFlexGrid.SetFocus

  Case 38     ' Arriba.
    MSHFlexGrid.SetFocus
    DoEvents
    If MSHFlexGrid.Row > MSHFlexGrid.FixedRows Then
      MSHFlexGrid.Row = MSHFlexGrid.Row - 1
    End If
```

```
  Case 40     ' Abajo.
    MSHFlexGrid.SetFocus
    DoEvents
    If MSHFlexGrid.Row < MSHFlexGrid.Rows - 1 Then
      MSHFlexGrid.Row = MSHFlexGrid.Row + 1
    End If
  End Select
End Sub

Sub fgEdoiniciales_GotFocus()
  If TxtEdoEdit2.Visible = False Then Exit Sub
  fgEdoiniciales = TxtEdoEdit2
  TxtEdoEdit2.Visible = False
End Sub

Sub fgEdoiniciales_LeaveCell()
  If TxtEdoEdit2.Visible = False Then Exit Sub
  fgEdoiniciales = TxtEdoEdit2
  TxtEdoEdit2.Visible = False
End Sub
'Manejo de fgEdosistema  para ingreso de datos e incrementos respectivamente
Function Fgi1(r As Integer, c As Integer) As Integer
  Fgi1 = c + fgEdosistema.Cols * r
End Function

Sub fgEdosistema_KeyPress(KeyAscii As Integer)
  MSHFlexGridEdit1 fgEdosistema, TxtEdoEdit1, KeyAscii
End Sub

Sub fgEdosistema_DblClick()
  MSHFlexGridEdit1 fgEdosistema, TxtEdoEdit1, 32 ' Simula un espacio.
End Sub

Sub MSHFlexGridEdit1(MSHFlexGrid As Control, _
Edt As Control, KeyAscii As Integer)

  ' Usar el carácter escrito.
  Select Case KeyAscii

  ' Un espacio significa modificar el texto actual.
  Case 0 To 32
    Edt = MSHFlexGrid
    Edt.SelStart = 1000

  ' Otro carácter reemplaza el texto actual.
```

```
  Case Else
    Edt = Chr(KeyAscii)
    Edt.SelStart = 1
  End Select

  ' Mostrar Edt en la posición correcta.
  Edt.Move MSHFlexGrid.Left + MSHFlexGrid.CellLeft, _
    MSHFlexGrid.Top + MSHFlexGrid.CellTop, _
    MSHFlexGrid.CellWidth - 8, _
    MSHFlexGrid.CellHeight - 8
  Edt.Visible = True

  ' Y hacer que funcione.
  Edt.SetFocus
End Sub

Sub TxTEdoEdit1_KeyPress(KeyAscii As Integer)
  ' Elimina los retornos para quitar los pitidos.
  If KeyAscii = Asc(vbCr) Then KeyAscii = 0

 End Sub

Sub TxTEdoEdit1_KeyDown(KeyCode As Integer, _
Shift As Integer)
  EditKeyCode fgEdosistema, TxtEdoEdit1, KeyCode, Shift
End Sub

Sub EditKeyCode(MSHFlexGrid As Control, Edt As _
Control, KeyCode As Integer, Shift As Integer)

  ' Procesamiento del control de edición estándar.
  Select Case KeyCode

  Case 27   ' ESC: ocultar, devuelve el enfoque a          ' MSFlexGrid.
    Edt.Visible = False
    MSHFlexGrid.SetFocus

  Case 13   ' ENTRAR devuelve el enfoque a MSHFlexGrid.
    MSHFlexGrid.SetFocus

  Case 38      ' Arriba.
    MSHFlexGrid.SetFocus
    DoEvents
    If MSHFlexGrid.Row > MSHFlexGrid.FixedRows Then
      MSHFlexGrid.Row = MSHFlexGrid.Row - 1
    End If
```

```
  Case 40     ' Abajo.
    MSHFlexGrid.SetFocus
    DoEvents
    If MSHFlexGrid.Row < MSHFlexGrid.Rows - 1 Then
      MSHFlexGrid.Row = MSHFlexGrid.Row + 1
    End If
  End Select
End Sub

Sub fgEdosistema_GotFocus()
  If TxtEdoEdit1.Visible = False Then Exit Sub
  fgEdosistema = TxtEdoEdit1
  TxtEdoEdit1.Visible = False
End Sub

Sub fgEdosistema_LeaveCell()
  If TxtEdoEdit1.Visible = False Then Exit Sub
  fgEdosistema = TxtEdoEdit1
  TxtEdoEdit1.Visible = False
End Sub
Private Sub SetearEdo()
Dim Pares() As String
Dim intervalo() As Double
Dim i As Integer
Dim tope As Integer

On Error GoTo err1
'copiamos los valores de FgEdosistema a la matriz pares
tope = fgEdosistema.Rows
ReDim Pares(0 To tope - 2) As String
For i = 1 To tope - 1
Pares(i - 1) = fgEdosistema.TextMatrix(i, 1)
Next i

'copiamos los valores de FgEdoiniciales a la matriz intervalo
tope = fgEdoiniciales.Rows
ReDim intervalo(0 To tope - 2) As Double
For i = 0 To tope - 2
intervalo(i) = Val(fgEdoiniciales.TextMatrix(i + 1, 1))
Next i


'Seteamos los atributos de la clase EDO
```

```
Edo.SetearEdo(EuHe, TxtEdofunción, CDbl(Val(TxtEdonum)), CDbl(Val(TxtEdoa)),
CDbl(Val(TxtEdob)), CDbl(Val(TxtEdoh)), CDbl(Val(TxtEdoinicial)),
CDbl(Val(TxtEdofinal)) _
, TxtEdop, TxtEdoq, Txtedor, Pares) = intervalo
Exit Sub
err1:
End Sub
'Manejo de presentación de resultados
Function Fgr(r As Integer, c As Integer) As Long
  Fgr = c + fgEdoresultados.Cols * r
End Function
Private Sub EdoPresentar()
Dim i As Integer
Dim j As Integer
Dim valor As Double
Dim limite As Integer
On Error GoTo err1
limite = ((Val(TxtEdob) - Val(TxtEdoa)) / Val(TxtEdoh)) + 1

Select Case tipoalg
Case Is = 5
fgEdoresultados.Cols = 3
fgEdoresultados.Rows = limite + 1
fgEdoresultados.FixedCols = 0
fgEdoresultados.FixedRows = 1
fgEdoresultados.TextArray(Fgr(0, 0)) = "t"
fgEdoresultados.TextArray(Fgr(0, 1)) = "Y"
fgEdoresultados.TextArray(Fgr(0, 2)) = "Predictor Pi"
fgEdoresultados.ColWidth(0) = 800
fgEdoresultados.ColWidth(1) = 800
fgEdoresultados.ColWidth(2) = 800

fgEdoresultados.ColAlignmentFixed(0) = 4
fgEdoresultados.ColAlignmentFixed(1) = 4
fgEdoresultados.ColAlignmentFixed(2) = 4


For i = 1 To limite
   For j = 0 To 2
   valor = Edo.solución(i - 1, j)
      If (valor <= 0 Or valor > 0.000000000000001) Then
      fgEdoresultados.TextMatrix(i, j) = valor
      Else
      fgEdoresultados.TextMatrix(i, j) = 0
      End If
  Next j
```

```
Next i
Case Is = 6
fgEdoresultados.Cols = 4
fgEdoresultados.Rows = limite + 1
fgEdoresultados.FixedCols = 0
fgEdoresultados.FixedRows = 1
fgEdoresultados.TextArray(Fgr(0, 0)) = "t"
fgEdoresultados.TextArray(Fgr(0, 1)) = "Y"
fgEdoresultados.TextArray(Fgr(0, 2)) = "Predictor Pi"
fgEdoresultados.TextArray(Fgr(0, 3)) = "Modificador mi"

fgEdoresultados.ColWidth(0) = 800
fgEdoresultados.ColWidth(1) = 800
fgEdoresultados.ColWidth(2) = 800
fgEdoresultados.ColWidth(3) = 800

fgEdoresultados.ColAlignmentFixed(0) = 4
fgEdoresultados.ColAlignmentFixed(1) = 4
fgEdoresultados.ColAlignmentFixed(2) = 4
fgEdoresultados.ColAlignmentFixed(3) = 4
'For i = 1 To limite
'fgEdoresultados.TextArray(Fgr(i, 0)) = i
'Next i
For i = 1 To limite
   For j = 0 To 3
   valor = Edo.solución(i - 1, j)
      If (valor <= 0 Or valor > 0.000000000000001) Then
      fgEdoresultados.TextMatrix(i, j) = valor
      Else
      fgEdoresultados.TextMatrix(i, j) = 0
      End If
  Next j
Next i
Case Is = 7
fgEdoresultados.Cols = Val(TxtEdonum) + 1
fgEdoresultados.Rows = limite + 1
fgEdoresultados.FixedCols = 0
fgEdoresultados.FixedRows = 1
fgEdoresultados.TextArray(Fgr(0, 0)) = "t"
fgEdoresultados.ColWidth(0) = 800
fgEdoresultados.ColAlignmentFixed(0) = 4
For i = 1 To fgEdoresultados.Cols - 1
fgEdoresultados.TextArray(Fgr(0, i)) = "y" & i
fgEdoresultados.ColWidth(i) = 2000
fgEdoresultados.ColAlignmentFixed(i) = 4
Next i
```

```
For i = 1 To limite
   For j = 0 To Val(TxtEdonum)
   valor = Edo.solución(i - 1, j)
      If (valor <= 0 Or valor > 0.000000000000001) Then
      fgEdoresultados.TextMatrix(i, j) = valor
      Else
      fgEdoresultados.TextMatrix(i, j) = 0
      End If
  Next j
Next i
Case Is = 8
fgEdoresultados.Cols = 5
fgEdoresultados.Rows = limite + 1
fgEdoresultados.FixedCols = 0
fgEdoresultados.FixedRows = 1
fgEdoresultados.TextArray(Fgr(0, 0)) = "t"
fgEdoresultados.TextArray(Fgr(0, 1)) = "u"
fgEdoresultados.TextArray(Fgr(0, 2)) = "v"
fgEdoresultados.TextArray(Fgr(0, 3)) = "w"
fgEdoresultados.TextArray(Fgr(0, 4)) = "x=uj+wj"
fgEdoresultados.ColWidth(0) = 800
fgEdoresultados.ColWidth(1) = 2500
fgEdoresultados.ColWidth(2) = 2500
fgEdoresultados.ColWidth(3) = 2500
fgEdoresultados.ColWidth(4) = 2500
fgEdoresultados.ColAlignmentFixed(0) = 4
fgEdoresultados.ColAlignmentFixed(1) = 4
fgEdoresultados.ColAlignmentFixed(2) = 4
fgEdoresultados.ColAlignmentFixed(3) = 4
fgEdoresultados.ColAlignmentFixed(4) = 4


For i = 1 To limite
   For j = 0 To 4
   valor = Edo.solución(i - 1, j)
      If (valor <= 0 Or valor > 0.000000000000001) Then
      fgEdoresultados.TextMatrix(i, j) = valor
      Else
      fgEdoresultados.TextMatrix(i, j) = 0
      End If
  Next j
Next i
```

```
Case Else
fgEdoresultados.Cols = 2
fgEdoresultados.Rows = limite + 1
fgEdoresultados.FixedCols = 0
fgEdoresultados.FixedRows = 1
fgEdoresultados.TextArray(Fgr(0, 0)) = "t"
fgEdoresultados.TextArray(Fgr(0, 1)) = "Y"
fgEdoresultados.ColWidth(0) = 2500
fgEdoresultados.ColWidth(1) = 2500
fgEdoresultados.ColAlignmentFixed(0) = 4
fgEdoresultados.ColAlignmentFixed(1) = 4


For i = 1 To limite
   For j = 0 To 1
   valor = Edo.solución(i - 1, j)
      If (valor <= 0 Or valor > 0.000000000000001) Then
      fgEdoresultados.TextMatrix(i, j) = valor
      Else
      fgEdoresultados.TextMatrix(i, j) = 0
      End If
  Next j
Next i
End Select
Exit Sub
err1:
End Sub

'******* Conexion a la BDD
Private Sub PLConexion(parBDD As Boolean)
Dim VTCad As String
Dim VTBase As Database
Dim VTcadena As String
   VTcadena = App.Path & "/Seguridad.mdb"
   Set VLConexion = New ADODB.Connection
   VTCad = "Provider=Microsoft.Jet.OLEDB.4.0;Data Source='" & VTcadena &
"';Persist Security Info=False"
   VLConexion.Open VTCad
End Sub

Private Sub PLGuardarBDD()
   Dim VTCad As String
   Dim VTReg As ADODB.Recordset
   Dim VTID As Integer
   Dim VB As Integer
   Dim VTNumero As String
```

```
On Error GoTo error1
   PLVerificaNombre Trim(VLNombre)
   If tipoalg >= 1 And tipoalg <= 6 Then
      If VLVerifica = True Then   'Si existe un nombre con el mismo nombre actualiza
         VB = MsgBox("El nombre de ese trabajo ya existe, desea reemplazarlo?",
vbYesNo, "Guardar Trabajo Como")
         If VB = 6 Then
               'VTCad = "update Integracion set " _
                     & " Fx= '" & Trim(TxtIntegrafunción.Text) & "', " _
                     & " Ia='" & TxtIntegraa.Text & "', " _
                     & " Ib='" & TxtIntegrab.Text & "' " _
                     & " where Nombre = '" & VLNombre & "'"
            VLConexion.Execute (VTCad)
         End If
      Else    'Guarda el trabajo
         VTID = PLRecuperaCodigo
         VTCad = "insert into EDO values " _
               & "(" & CInt(VTID) & ", " _
               & " '" & TxtEdoa.Text & "', '" & TxtEdob.Text & "', " _
               & " '" & TxtEdoh.Text & "', '" & 1 & "', " _
               & " '" & TxtEdoinicial.Text & "', '" & TxtEdofunción.Text & "', " _
               & " '" & Trim(VLNombre) & "', " & CInt(tipoalg) & ")"
               '& " '" & Trim(VLNombre) & "', " & CInt(tipoalg) & ", " & CInt(VGIDUsuario)
& ")"
         VLConexion.Execute (VTCad)
         MsgBox "Trabajo guardado exitosamente", vbInformation, "Guardar trabajo
como"
      End If

   ElseIf tipoalg = 7 Then
      If VLVerifica = True Then   'Si existe un nombre con el mismo nombre actualiza
         VB = MsgBox("El nombre de ese trabajo ya existe, desea reemplazarlo?",
vbYesNo, "Guardar Trabajo Como")
         If VB = 6 Then
               'VTCad = "update Integracion set " _
                     & " Fx= '" & Trim(TxtIntegrafunción.Text) & "', " _
                     & " Ia='" & TxtIntegraa.Text & "', " _
                     & " Ib='" & TxtIntegrab.Text & "' " _
                     & " where Nombre = '" & VLNombre & "'"
            VLConexion.Execute (VTCad)
         End If
      Else    'Guarda el trabajo
         VTID = PLRecuperaCodigo
         PLGuardaSistema CInt(TxtEdonum.Text)
         VTCad = "insert into EDO_SEDO values " _
               & "(" & CInt(VTID) & ", " _
```

```
            & " '" & TxtEdonum.Text & "', " _
            & " '" & TxtEdoa.Text & "', '" & TxtEdob.Text & "', " _
            & " '" & TxtEdoh.Text & "', '" & VLSistema & "', " _
            & " '" & VLValores & "', " _
            & " '" & Trim(VLNombre) & "', " & CInt(tipoalg) & ", " & 1 & ")"
            '& " '" & Trim(VLNombre) & "', " & CInt(tipoalg) & ", " & CInt(VGIDUsuario)
& ")"
        VLConexion.Execute (VTCad)
        MsgBox "Trabajo guardado exitosamente", vbInformation, "Guardar trabajo
como"
    End If

  ElseIf tipoalg = 8 Then
    If VLVerifica = True Then   'Si existe un nombre con el mismo nombre actualiza
      VB = MsgBox("El nombre de ese trabajo ya existe, desea reemplazarlo?",
vbYesNo, "Guardar Trabajo Como")
      If VB = 6 Then
          'VTCad = "update Integracion set " _
              & " Fx= '" & Trim(TxtIntegrafunción.Text) & "', " _
              & " Ia='" & TxtIntegraa.Text & "', " _
              & " Ib='" & TxtIntegrab.Text & "' " _
              & " where Nombre = '" & VLNombre & "'"
        VLConexion.Execute (VTCad)
      End If
    Else    'Guarda el trabajo
      VTID = PLRecuperaCodigo
      VTCad = "insert into EDO_Con values " _
          & "(" & CInt(VTID) & ", " _
          & " '" & TxtEdop.Text & "', '" & TxtEdoq.Text & "', " _
          & " '" & Txtedor.Text & "', '" & TxtEdoa.Text & "', " _
          & " '" & TxtEdob.Text & "', '" & TxtEdoh.Text & "', " _
          & " '" & TxtEdoinicial.Text & "', '" & TxtEdofinal.Text & "', " _
          & " '" & Trim(VLNombre) & "', " & CInt(tipoalg) & ", '" & 1 & "')"
          '& " '" & Trim(VLNombre) & "', " & CInt(tipoalg) & ", " & CInt(VGIDUsuario)
& ")"
      VLConexion.Execute (VTCad)
      MsgBox "Trabajo guardado exitosamente", vbInformation, "Guardar trabajo
como"
    End If

  End If
Exit Sub
error1:
  If err.Number <> 0 Then
    MsgBox "Se ha producido el siguiente error: " & err.Description & "." _
        & "Por favor consultar con el administrador", vbCritical
```

```
   End If
End Sub

Private Sub PLVerificaNombre(parNombre As String)
Dim VTReg As ADODB.Recordset
Dim VTCad As String
Dim VTTabla As String
On Error GoTo error1
   If tipoalg >= 1 And tipoalg <= 6 Then
      VTTabla = "EDO"
   ElseIf tipoalg = 7 Then
      VTTabla = "EDO_SEDO"
   ElseIf tipoalg = 8 Then
      VTTabla = "EDO_Con"
   End If
      VTCad = "select Nombre from " & VTTabla & " where Nombre = '" & parNombre
& "' and tipo = " & tipoalg
         Set VTReg = VLConexion.Execute(VTCad)
         With VTReg
            If Not VTReg.EOF Then
               .MoveFirst
               Do Until .EOF
                  If Trim(VTReg(0)) = parNombre Then
                     VLVerifica = True
                     Exit Do
                  Else
                     VLVerifica = False
                  End If
                  .MoveNext
               Loop
            Else
               VLVerifica = False
            End If
         End With
Exit Sub
error1:

End Sub
Private Function PLRecuperaCodigo() As Integer
   Dim VTCad As String
   Dim VTReg As ADODB.Recordset
   Dim VTTabla As String
On Error GoTo err1
   If tipoalg >= 1 And tipoalg <= 6 Then
      VTCad = "select max(IDEDP) from EDO"
   ElseIf tipoalg = 7 Then
```

```
      VTCad = "select max(IDEDO_SEDO) from EDO_SEDO"
   ElseIf tipoalg = 8 Then
      VTCad = "select max(IDEDO_Con) from EDO_Con"
   End If
      Set VTReg = VLConexion.Execute(VTCad)
      If Not IsNull(VTReg(0)) Then
         PLRecuperaCodigo = VTReg(0) + 1
      Else
         PLRecuperaCodigo = 1
      End If
Exit Function
err1:

End Function

Private Sub PLRecupera()
   Dim VTCad As String
   Dim VTReg As ADODB.Recordset
On Error GoTo err1
   If tipoalg >= 1 And tipoalg <= 6 Then
      VTCad = "select * from EDO where Tipo = " & CInt(tipoalg) & " and Nombre = '"
& VLNombreTrabajo & "'"
      Set VTReg = VLConexion.Execute(VTCad)
      If Not VTReg.EOF Then
         TxtEdoa.Text = Trim(VTReg(1))
         TxtEdob.Text = Trim(VTReg(2))
         TxtEdoh.Text = Trim(VTReg(3))
         TxtEdoinicial.Text = Trim(VTReg(5))
         TxtEdofunción.Text = Trim(VTReg(6))
      End If
   ElseIf tipoalg = 7 Then
      VTCad = "select * from EDO_SEDO where Tipo = " & CInt(tipoalg) & " and
Nombre = '" & VLNombreTrabajo & "'"
      Set VTReg = VLConexion.Execute(VTCad)
      If Not VTReg.EOF Then
         TxtEdonum.Text = Trim(VTReg(1))
         TxtEdoa.Text = Trim(VTReg(2))
         TxtEdob.Text = Trim(VTReg(3))
         TxtEdoh.Text = Trim(VTReg(4))
         PLRecuperaDatos CInt(VTReg(1)), Trim(VTReg(5)), Trim(VTReg(6))

      End If
   ElseIf tipoalg = 8 Then
      VTCad = "select * from EDO_Con where Tipo = " & CInt(tipoalg) & " and Nombre
= '" & VLNombreTrabajo & "'"
      Set VTReg = VLConexion.Execute(VTCad)
```

```
      If Not VTReg.EOF Then
         TxtEdop.Text = Trim(VTReg(1))
         TxtEdoq.Text = Trim(VTReg(2))
         Txtedor.Text = Trim(VTReg(3))
         TxtEdoa.Text = Trim(VTReg(4))
         TxtEdob.Text = Trim(VTReg(5))
         TxtEdoh.Text = Trim(VTReg(6))
         TxtEdoinicial.Text = Trim(VTReg(7))
         TxtEdofinal.Text = Trim(VTReg(8))
      End If

   End If
Exit Sub
err1:
   If err.Number <> 0 Then
      MsgBox "Se ha producido el siguiente error: " & err.Description & "." _
            & "Por favor consultar con el administrador", vbCritical
   End If
End Sub

Private Sub PLGuardaSistema(parNume As String)
Dim i, j As Integer
Dim Nume As Integer
On Error GoTo error1
   VLSistema = ""
   VLValores = ""
   Nume = CInt(parNume)
      For i = 1 To Nume
         For j = 1 To Nume - 1
            If fgEdosistema.TextMatrix(i, j) = "" Then
               VLSistema = VLSistema & "0"
            Else
               VLSistema = VLSistema & fgEdosistema.TextMatrix(i, j) & ";"
            End If
         Next j
      Next i
      'Para los valores
      For i = 1 To Nume
         For j = 1 To Nume - 1
            If fgEdoiniciales.TextMatrix(i, j) = "" Then
               VLValores = VLValores & "0"
            Else
               VLValores = VLValores & fgEdoiniciales.TextMatrix(i, j) & ";"
            End If
         Next j
      Next i
```

```
Exit Sub
error1:

End Sub

Private Sub PLRecuperaDatos(parNume As Integer, parSistema As String,
parValores As String)
Dim i, j, k As Integer
Dim VTCont As Integer
Dim VTPos, VTPos1 As Integer
Dim VTValor, VTValor1 As Integer
Dim VTLimite As Integer
Dim VTSis As String
On Error GoTo error1
   VTCont = 1
   VTSis = parSistema
      VTLimite = parNume
      For k = 1 To VTLimite
         VLSistemaMatriz(k) = 0
         VLValoresMatriz(k) = 0
      Next k

      For k = 1 To VTLimite
         VTPos = InStr(1, parSistema, ";")
         VTValor = Mid(parSistema, 1, VTPos - 1)
         parSistema = Mid(parSistema, VTPos + 1)
         VLSistemaMatriz(k) = VTValor
      Next k

      For k = 1 To VTLimite
         VTPos1 = InStr(1, parValores, ";")
         VTValor1 = Mid(parValores, 1, VTPos1 - 1)
         parValores = Mid(parValores, VTPos1 + 1)
         VLValoresMatriz(k) = VTValor1
      Next k

      For i = 1 To parNume
         For j = 1 To parNume - 1
            fgEdosistema.TextMatrix(i, j) = VLSistemaMatriz(VTCont)
            fgEdoiniciales.TextMatrix(i, j) = VLValoresMatriz(VTCont)
            VTCont = VTCont + 1
         Next j
      Next i

Exit Sub
```

```
error1:
   If err.Number <> 0 Then
      MsgBox "Se ha producido el siguiente error: " & err.Description & "." _
            & "Por favor consultar con el administrador", vbCritical
   End If
End Sub

Private Sub PLImprimir()
   PLCambio
      PrintForm
   PLOriginal
End Sub

Private Sub PLCambio()
   Me.BackColor = &HFFFFFF
   Me.BorderStyle = 0
   CmdEdocalcular.Visible = False
'   fgEdoresultados.BackColorBkg = &HFFFFFF
'   ShapeEdoresultados.Visible = False
'   fgEdoresultados.Top = 6000
'   fgEdoresultados.Width = 10000
End Sub

Private Sub PLOriginal()
   Me.BackColor = &H8000000F
   Me.BorderStyle = 2
   CmdEdocalcular.Visible = True
'   fgEdoresultados.BackColorBkg = &H8000000F
'   ShapeEdoresultados.Visible = True
'   fgEdoresultados.Top = 7200
'   fgEdoresultados.Width = 6855
End Sub

'Permite reemplazar los valores numéricos en la función de edo y evaluarla mediante
la
'llamada al objeto MscExcel
Private Function reemplazafedo(f As String, te As Double, ye As Double) As Double
  Dim nx
  Dim NF
  Dim nL
  Dim Ns
  Dim nr
  'Reemplaza t
  NF = f
  nx = te
  nL = Len(nx)
```

```
  Ns = InStr(1, nx, ",")
  nr = nL - Ns
  If Ns <> 0 Then


  nx = Mid(nx, 1, InStr(1, nx, ",") - 1) & "." & Mid(nx, InStr(1, nx, ",") + 1, Len(Mid(nx, 1,
nr)))
  End If

  While InStr(1, NF, "t") > 0
    NF = Mid(NF, 1, InStr(1, NF, "t") - 1) & nx & Mid(NF, InStr(1, NF, "t") + 1, Len(NF))
  Wend

'Reemplaza y

  nx = ye
  nL = Len(nx)
  Ns = InStr(1, nx, ",")
  nr = nL - Ns
  If Ns <> 0 Then


  nx = Mid(nx, 1, InStr(1, nx, ",") - 1) & "." & Mid(nx, InStr(1, nx, ",") + 1, Len(Mid(nx, 1,
nr)))
  End If

  While InStr(1, NF, "y") > 0
    NF = Mid(NF, 1, InStr(1, NF, "y") - 1) & nx & Mid(NF, InStr(1, NF, "y") + 1, Len(NF))
  Wend

Evaluarfedot.fa = UCase(NF)
Evaluarfedot.Tipox = True
Evaluarfedot.evaluarf
reemplazafedo = Evaluarfedot.Fdy
End Function
'Permite evaluar el sistema de EDO de n ecuaciones mediante la llamada al objeto
MscExcel
Private Function Evaluarsistema(f As String, valores() As Double) As Double
  Dim i As Integer
  Dim nx
  Dim NF
  Dim nL
  Dim Ns
  Dim nr
  Dim variables
  On Error GoTo err1
```

```
  'Reemplaza t
  NF = f
  nx = valores(0)
  nL = Len(nx)
  Ns = InStr(1, nx, ",")
  nr = nL - Ns
  If Ns <> 0 Then


  nx = Mid(nx, 1, InStr(1, nx, ",") - 1) & "." & Mid(nx, InStr(1, nx, ",") + 1, Len(Mid(nx, 1,
nr)))
  End If

  While InStr(1, NF, "t") > 0
    NF = Mid(NF, 1, InStr(1, NF, "t") - 1) & nx & Mid(NF, InStr(1, NF, "t") + 1, Len(NF))
  Wend
   'f = NF
  'Reemplaza el resto de variables
  For i = 1 To Val(TxtEdonum)

  nx = valores(i)
  nL = Len(nx)
  Ns = InStr(1, nx, ",")
  nr = nL - Ns
  variables = "y" & i
  If Ns <> 0 Then


  nx = Mid(nx, 1, InStr(1, nx, ",") - 1) & "." & Mid(nx, InStr(1, nx, ",") + 1, Len(Mid(nx, 1,
nr)))
  End If

  While InStr(1, NF, variables) > 0
    NF = Mid(NF, 1, InStr(1, NF, variables) - 1) & nx & Mid(NF, InStr(1, NF, variables)
+ 2, Len(NF))
  Wend
 Next i

Evaluarfedot.fa = UCase(NF)
Evaluarfedot.Tipox = True
Evaluarfedot.evaluarf
Evaluarsistema = Evaluarfedot.Fdy
Exit Function
err1:
End Function
```

```
Private Function Evaluart(f As String, te As Double) As Double
  Dim nx
  Dim NF
  Dim nL
  Dim Ns
  Dim nr
  On Error GoTo err1
  'Reemplaza t
  NF = f
  nx = te
  nL = Len(nx)
  Ns = InStr(1, nx, ",")
  nr = nL - Ns
  If Ns <> 0 Then


  nx = Mid(nx, 1, InStr(1, nx, ",") - 1) & "." & Mid(nx, InStr(1, nx, ",") + 1, Len(Mid(nx, 1,
nr)))
  End If

  While InStr(1, NF, "t") > 0
    NF = Mid(NF, 1, InStr(1, NF, "t") - 1) & nx & Mid(NF, InStr(1, NF, "t") + 1, Len(NF))
  Wend


Evaluarfedot.fa = UCase(NF)
Evaluarfedot.Tipox = True
Evaluarfedot.evaluarf
Evaluart = Evaluarfedot.Fdy
Exit Function
err1:

End Function

Private Sub Sincolor()
With Me
arribaSstab = .SSTab1.Top
izquierdaSstab = .SSTab1.Left
anchoSstab = .SSTab1.Width
altoSstab = .SSTab1.Height
colororg = .BackColor
.BackColor = &H80000009
.SSTab1.Top = .SSTab1.Top + 1000
.SSTab1.Left = .SSTab1.Left + 1000
.SSTab1.Width = .SSTab1.Width / 4
.SSTab1.Height = .SSTab1.Height / 4
```

```
.FraImprmir.BackColor = &H80000009
.fgEdoiniciales.BackColorBkg = &H80000009
.fgEdoiniciales.BackColorFixed = &H80000009
.fgEdoresultados.BackColorBkg = &H80000009
.fgEdoresultados.BackColorFixed = &H80000009
.fgEdosistema.BackColorBkg = &H80000009
.fgEdosistema.BackColorFixed = &H80000009
.LblEdoresultados.BackColor = &H80000009
.TxtEdoa.BorderStyle = 0
.TxtEdob.BorderStyle = 0
.TxtEdoEdit1.BorderStyle = 0
.TxtEdoEdit2.BorderStyle = 0
.TxtEdofinal.BorderStyle = 0
.TxtEdofunción.BorderStyle = 0
.TxtEdoh.BorderStyle = 0
.TxtEdoinicial.BorderStyle = 0
.TxtEdonum.BorderStyle = 0
.TxtEdop.BorderStyle = 0
.TxtEdoq.BorderStyle = 0
.Txtedor.BorderStyle = 0
.ShapeEdoresultados.BackColor = &H80000009
.LblNomalg.BackColor = &H80000009
.CmdEdocalcular.Visible = False
.LblEdoa.BackColor = &H80000009
.LblEdob.BackColor = &H80000009
.LblEdofinal.BackColor = &H80000009
.LblEdofunción.BackColor = &H80000009
.LblEdoh.BackColor = &H80000009
.LblEdoinicial.BackColor = &H80000009
.LblEdonum.BackColor = &H80000009
.LblEdop.BackColor = &H80000009
.LblEdoq.BackColor = &H80000009
.LblEdor.BackColor = &H80000009
.LblEdotitulo.BackColor = &H80000009
.LblEdotitulo2.BackColor = &H80000009
.fgEdoresultados.ScrollBars = 0
.fgEdoiniciales.ScrollBars = 0
.fgEdosistema.ScrollBars = 0
.FraImprmir.BorderStyle = 1
FraEdoSintaxis.Visible = False
End With
End Sub

Private Sub Concolor()
With Me
 .SSTab1.Top = arribaSstab
```

```
 .SSTab1.Left = izquierdaSstab
.SSTab1.Width = anchoSstab
.SSTab1.Height = altoSstab
 .BackColor = colororg
.FraImprmir.BackColor = colororg
.fgEdoiniciales.BackColorBkg = colororg
.fgEdoiniciales.BackColorFixed = colororg
.fgEdoresultados.BackColorBkg = colororg
.fgEdoresultados.BackColorFixed = colororg
.fgEdosistema.BackColorBkg = colororg
.fgEdosistema.BackColorFixed = colororg
.LblEdoresultados.BackColor = colororg
.TxtEdoa.BorderStyle = 1
.TxtEdob.BorderStyle = 1
.TxtEdoEdit1.BorderStyle = 1
.TxtEdoEdit2.BorderStyle = 1
.TxtEdofinal.BorderStyle = 1
.TxtEdofunción.BorderStyle = 1
.TxtEdoh.BorderStyle = 1
.TxtEdoinicial.BorderStyle = 1
.TxtEdonum.BorderStyle = 1
.TxtEdop.BorderStyle = 1
.TxtEdoq.BorderStyle = 1
.Txtedor.BorderStyle = 1
.ShapeEdoresultados.BackColor = colororg
.LblNomalg.BackColor = colororg
.CmdEdocalcular.Visible = True
.LblEdoa.BackColor = colororg
.LblEdob.BackColor = colororg
.LblEdofinal.BackColor = colororg
.LblEdofunción.BackColor = colororg
.LblEdoh.BackColor = colororg
.LblEdoinicial.BackColor = colororg
.LblEdonum.BackColor = colororg
.LblEdop.BackColor = colororg
.LblEdoq.BackColor = colororg
.LblEdor.BackColor = colororg
.LblEdotitulo.BackColor = colororg
.LblEdotitulo2.BackColor = colororg
.fgEdoiniciales.ScrollBars = 3
.fgEdoresultados.ScrollBars = 3
.fgEdosistema.ScrollBars = 3
.FraImprmir.BorderStyle = 0
End With
FraEdoSintaxis.Visible = True
```

```
'Se obtine el método de Euler modificado y el de Huen a través del método de
'Runge -Kutta de segundo orden de la forma y(t+h)=y(t)+Ahfo+Bhf1
'fo=f(t,y); f1=f(t+Ph,y+Qhfo)
Public Sub EulermodificadoyHeun()
Dim i As Integer
Dim j As Integer
Dim m As Double        'numero de pasos
Dim T() As Double      'Vector de los t
Dim mensaje As String
Dim Y() As Double      'Vector de respuestas
Dim ae As Double       'coeficiente para implementar metodos
Dim be As Double       'coeficiente para implementar metodos
Dim p As Double        'coeficiente para implementar metodos
Dim q As Double        'coeficiente para implementar metodos
Dim fo As Double       'coeficiente para implementar metodos
Dim f1 As Double       'coeficiente para implementar metodos

On Error GoTo err1
If EulerHeun = True Then
'Coeficientes para el método de Euler modificado
ae = 0
be = 1
p = 0.5
q = 0.5
Else
'Coeficientes para el método de Heun
ae = 0.5
be = 0.5
p = 1
q = 1
End If
'Implementamos el algoritmo general
m = (b - A) / h
ReDim Y(0 To m) As Double
ReDim T(0 To m) As Double
Y(0) = yo
For i = 0 To m
T(i) = A + i * h
Next i

   For i = 0 To m - 1       'Lazo para aplicar la formula general
   fo = reemplazafedo(fedo, T(i), Y(i))
   f1 = reemplazafedo(fedo, T(i) + p * h, Y(i) + q * h * fo)
   Y(i + 1) = Y(i) + ae * h * fo + be * h * f1
```

```
End Sub
```

Módulo de clase ClasEDO

```
Option Explicit
Dim Evaluarfedo As New McsExcel 'Crear una instancia la objeto MCsExcel para
evaluar fedo
Dim Resuledo() As Double 'matriz de resultados
Dim A As Double        'punto inical intervalo
Dim b As Double        'punto final intervalo
Dim Viniciales() As Double       'Vector de valores iniciales para el sistema
Dim plineal As String   ' Parámetro p(t) del método de disparo lineal
Dim qlineal As String   ' Parámetro q(t) del método de disparo lineal
Dim rlineal As String   ' Parámetro r(t) del método de disparo lineal
Dim Numedo As Double     'numero de ecuaciones EDO
Dim fedo As String     'y'=f(t,y) es f(t,y)
Dim fiedo() As String   'Matriz que contine las funciones del sistema EDO
Dim h As Double        'tamaño del paso
Dim yo As Double       'y inicial
Dim xa As Double       'valor en la frontera en el inicio del intervalo
Dim xb As Double       'valor en la frontera al final del intervalo
Dim EulerHeun As Boolean 'Indica que tipo de método  si es true Euler
modificado,false es Heun

Public Property Get solución() As Variant
On Error GoTo err1
solución = Resuledo
Exit Property
err1:
End Property

Public Property Let SetearEdo(Eh As Boolean, Fun As String, num As Double, ae As
Double, be As Double, hE As Double, Yini As Double _
, Yfin As Double, p As String, q As String, r As String, edos() As String, valoresh() As
Double)
On Error GoTo err1
fedo = Fun
Numedo = num
A = ae
b = be
h = hE
yo = Yini
xa = Yini
xb = Yfin
plineal = p
qlineal = q
rlineal = r
```

```
Viniciales = valoresh
fiedo = edos

Exit Property
err1:

End Property

Public Sub Euler()
Dim i As Integer
Dim j As Integer
Dim m As Double        'numero de pasos
Dim T() As Double      'Vector de los t
Dim mensaje As String
Dim Y() As Double      'Vector de respuestas
On Error GoTo err1
'Implementamos el algoritmo de Euler para varios h
m = (b - A) / h
ReDim Y(0 To m) As Double
ReDim T(0 To m) As Double
Y(0) = yo
For i = 0 To m
T(i) = A + i * h
Next i

   For i = 0 To m - 1       'Lazo para aplicar la formula de Euler
   Y(i + 1) = Y(i) + h * reemplazafedo(fedo, T(i), Y(i))
   Next i
   'Asignamos los resultados a la matriz de respuestas
   ReDim Resuledo(0 To m, 0 To 1)
   For i = 0 To m
   Resuledo(i, 1) = Y(i)
   Next i

'Asignamos los valores del tiempo calculado a la matriz de resultados
For i = 0 To m
Resuledo(i, 0) = T(i)
Next i
Exit Sub
err1:
If err.Number <> 0 Then
   mensaje = MsgBox("Parámetros del algoritmo incompletos", 48, "Error en el
ingreso")
   End If

End Sub
```

```
   Next i
   'Asignamos los resultados a la matriz de respuestas
   ReDim Resuledo(0 To m, 0 To 1)
   For i = 0 To m
   Resuledo(i, 1) = Y(i)
   Next i

'Asignamos los valores del tiempo calculado a la matriz de resultados
For i = 0 To m
Resuledo(i, 0) = T(i)
Next i
Exit Sub
err1:
If err.Number <> 0 Then
   mensaje = MsgBox("Parámetros del algoritmo incompletos", 48, "Error en el
ingreso")
   End If

End Sub

'Permite resolver EDO mediante el método de Runge-Kutta de cuato orden
Public Sub RKF4()
Dim i As Integer
Dim j As Integer
Dim m As Double        'numero de pasos
Dim T() As Double      'Vector de los t
Dim mensaje As String
Dim Y() As Double      'Vector de respuestas

Dim f1 As Double       'coeficiente para implementar metodos
Dim f2 As Double       'coeficiente para implementar metodos
Dim f3 As Double       'coeficiente para implementar metodos
Dim f4 As Double       'coeficiente para implementar metodos

On Error GoTo err1
'Implementamos el algoritmo general
m = (b - A) / h
ReDim Y(0 To m) As Double
ReDim T(0 To m) As Double
Y(0) = yo
For i = 0 To m
T(i) = A + i * h
Next i

   For i = 0 To m - 1      'Lazo para aplicar la formula general
   f1 = reemplazafedo(fedo, T(i), Y(i))
```

```
    f2 = reemplazafedo(fedo, T(i) + (h / 2), Y(i) + (h / 2) * f1)
    f3 = reemplazafedo(fedo, T(i) + (h / 2), Y(i) + (h / 2) * f2)
    f4 = reemplazafedo(fedo, T(i) + h, Y(i) + h * f3)
    Y(i + 1) = Y(i) + (h * (f1 + 2 * f2 + 2 * f3 + f4)) / 6
    Next i
    'Asignamos los resultados a la matriz de respuestas
    ReDim Resuledo(0 To m, 0 To 1)
    For i = 0 To m
    Resuledo(i, j + 1) = Y(i)
    Next i

'Asignamos los valores del tiempo calculado a la matriz de resultados
For i = 0 To m
Resuledo(i, 0) = T(i)
Next i
Exit Sub
err1:
If err.Number <> 0 Then
    mensaje = MsgBox("Parámetros del algoritmo incompletos", 48, "Error en el
ingreso")
    End If

End Sub
'Permite resolver una EDO usando el metodo predictor corrector de
'Adams-Bashforth-Moulton
Public Sub AdamsBashforthMoulton()
Dim i As Integer
Dim j As Integer
Dim m As Double        'numero de pasos
Dim T() As Double      'Vector de los t
Dim mensaje As String
Dim Y() As Double      'Vector de respuestas
Dim p() As Double      'Vector para los valores predictores
Dim f0 As Double       'coeficiente para implementar metodos
Dim f1 As Double       'coeficiente para implementar metodos
Dim f2 As Double       'coeficiente para implementar metodos
Dim f3 As Double       'coeficiente para implementar metodos
Dim f4 As Double       'coeficiente para implementar metodos
On Error GoTo err1
'Obtenemos los 3 primeros valores mediante RK4
m = (b - A) / h
ReDim Y(0 To m) As Double
ReDim T(0 To m) As Double
Y(0) = yo
For i = 0 To m
T(i) = A + i * h
```

```
Next i

   For i = 0 To 2       'Lazo para aplicar la formula general
   f1 = reemplazafedo(fedo, T(i), Y(i))
   f2 = reemplazafedo(fedo, T(i) + (h / 2), Y(i) + (h / 2) * f1)
   f3 = reemplazafedo(fedo, T(i) + (h / 2), Y(i) + (h / 2) * f2)
   f4 = reemplazafedo(fedo, T(i) + h, Y(i) + h * f3)
   Y(i + 1) = Y(i) + ((h * (f1 + 2 * f2 + 2 * f3 + f4)) / 6)
   Next i
'Implementamos el algoritmo de Adams-Bashforth-Moulton
ReDim p(0 To m) As Double
For i = 3 To m - 1
'Calculamos el predictor
f0 = reemplazafedo(fedo, T(i - 3), Y(i - 3))
f1 = reemplazafedo(fedo, T(i - 2), Y(i - 2))
f2 = reemplazafedo(fedo, T(i - 1), Y(i - 1))
f3 = reemplazafedo(fedo, T(i), Y(i))
p(i + 1) = Y(i) + ((h / 24) * (-9 * f0 + 37 * f1 - 59 * f2 + 55 * f3))
'Calculamos el corrector
f4 = reemplazafedo(fedo, T(i + 1), p(i + 1))
Y(i + 1) = Y(i) + ((h / 24) * (f1 - 5 * f2 + 19 * f3 + 9 * f4))
Next i
'Asignamos los resultados a la matriz de resultados
ReDim Resuledo(0 To m, 0 To 2)
For i = 0 To m
Resuledo(i, 0) = T(i)
Resuledo(i, 1) = Y(i)
Resuledo(i, 2) = p(i)
Next i
For i = 0 To 3
Resuledo(i, 2) = Resuledo(i, 1)
Resuledo(i, 1) = 0
Next
Exit Sub
err1:
'If err.Number <> 0 Then
   mensaje = MsgBox("Parámetros del algoritmo incompletos", 48, "Error en el
ingreso")
   'End If

End Sub

Public Sub MileneSimpson()
Dim i As Integer
Dim j As Integer
Dim m As Double        'numero de pasos
```

```
Dim T() As Double       'Vector de los t
Dim mensaje As String
Dim Y() As Double       'Vector de respuestas
Dim f0 As Double        'coeficiente para implementar metodos
Dim f1 As Double        'coeficiente para implementar metodos
Dim f2 As Double        'coeficiente para implementar metodos
Dim f3 As Double        'coeficiente para implementar metodos
Dim f4 As Double        'coeficiente para implementar metodos
Dim p() As Double       'Vector para los valores predictores
Dim Md() As Double       'Coeficiente del valor modificado
Dim pm As Double         'valor de p anterior
Dim ym As Double         'valor de y anterior
On Error GoTo err1
'Obtenemos los 3 primeros valores mediante RK4
m = (b - A) / h
ReDim Y(0 To m) As Double
ReDim T(0 To m) As Double
Y(0) = yo
For i = 0 To m
T(i) = A + i * h
Next i

   For i = 0 To 2       'Lazo para aplicar la formula general
   f1 = reemplazafedo(fedo, T(i), Y(i))
   f2 = reemplazafedo(fedo, T(i) + (h / 2), Y(i) + (h / 2) * f1)
   f3 = reemplazafedo(fedo, T(i) + (h / 2), Y(i) + (h / 2) * f2)
   f4 = reemplazafedo(fedo, T(i) + h, Y(i) + h * f3)
   Y(i + 1) = Y(i) + ((h * (f1 + 2 * f2 + 2 * f3 + f4)) / 6)
   Next i
'Implementamos el algoritmo de Milne-Simpson
ReDim p(0 To m) As Double
ReDim Md(0 To m) As Double
pm = 0
ym = 0

For i = 3 To m - 1
'Calculamos el predictor
f0 = reemplazafedo(fedo, T(i - 3), Y(i - 3))
f1 = reemplazafedo(fedo, T(i - 2), Y(i - 2))
f2 = reemplazafedo(fedo, T(i - 1), Y(i - 1))
f3 = reemplazafedo(fedo, T(i), Y(i))
p(i + 1) = Y(i - 3) + (4 * h * (2 * f1 - f2 + 2 * f3) / 3)
'Calculamos el valor modificado
Md(i + 1) = p(i + 1) + ((28 / 29) * (ym - pm))
'Calculamos el corrector
f4 = reemplazafedo(fedo, T(i + 1), Md(i + 1))
```

```
Y(i + 1) = Y(i - 1) + ((h / 3) * (f2 + 4 * f3 + f4))
'Asignamos los nuevos valores de pm y ym
pm = p(i + 1)
ym = Y(i + 1)
Next i
'Asignamos los resultados a la matriz de resultados
ReDim Resuledo(0 To m, 0 To 3)
For i = 0 To m
Resuledo(i, 0) = T(i)
Resuledo(i, 1) = Y(i)
Resuledo(i, 2) = p(i)
Resuledo(i, 3) = Md(i)
Next i
For i = 0 To 3
Resuledo(i, 2) = Resuledo(i, 1)
Resuledo(i, 1) = 0
Next

Exit Sub
err1:
'If err.Number <> 0 Then
   mensaje = MsgBox("Parámetros del algoritmo incompletos", 48, "Error en el
ingreso")
   'End If

End Sub

'Permite resolver un sistema de n EDO con condiciónes inicialesmediante RK4
Public Sub SistemasEDO()
Dim i As Integer
Dim j As Integer
Dim k As Integer
Dim m As Double        'numero de pasos
Dim T() As Double      'Vector de los t
Dim mensaje As String
Dim Y() As Double      'Vector de respuestas
Dim f() As Double      'Matriz de funciones EDO evaladas
Dim X() As Double      'Vector que contiene Xn los valores de la fnedo
                       ' en las que serán evaluadas


On Error GoTo err1
'Implementamos el algoritmo general
m = (b - A) / h
ReDim Y(0 To m, 0 To Numedo) As Double
ReDim T(0 To m) As Double
```

```
For i = 0 To m
T(i) = A + i * h
Next i
'Asignamos los valores iniciales y los de tiempo a la matriz Y
For i = 1 To Numedo
Y(0, i) = Viniciales(i - 1)
Next i
For i = 0 To m
Y(i, 0) = T(i)
Next i
ReDim f(1 To 4, 1 To Numedo) As Double

For i = 0 To m - 1

'Asignamos los valores iniciales al vector X para evaluar las EDO
'Para calcular los f1 de todos las varibles
ReDim X(0 To Numedo) As Double
For j = 0 To Numedo
X(j) = Y(i, j)
Next j
'Calculamos f1
For j = 1 To Numedo
f(1, j) = Evaluarsistema(fiedo(j - 1), X())
Next j
'Calculamos los nuevos valores para evaluar las edo
X(0) = X(0) + h / 2
k = 1
For j = 1 To Numedo
X(j) = Y(i, j) + (h * f(k, j)) / 2
Next j
'Calculamos f2
For j = 1 To Numedo
f(2, j) = Evaluarsistema(fiedo(j - 1), X())
Next j

'Calculamos los nuevos valores para evaluar las edo
k = 2
For j = 1 To Numedo
X(j) = Y(i, j) + (h * f(k, j)) / 2
Next j
'Calculamos f3
For j = 1 To Numedo
f(3, j) = Evaluarsistema(fiedo(j - 1), X())
Next j
'Calculamos los nuevos valores para evaluar las edo
X(0) = X(0) + h
```

```
k = 3
For j = 1 To Numedo
X(j) = Y(i, j) + h * f(k, j)
Next j
'Calculamos f4
For j = 1 To Numedo
f(4, j) = Evaluarsistema(fiedo(j - 1), X())
Next j
'Encontramos la nuevas aproximaciones  de la solución
For j = 1 To Numedo
Y(i + 1, j) = Y(i, j) + (h * (f(1, j) + 2 * f(2, j) + 2 * f(3, j) + f(4, j))) / 6
Next j
Next i

'Asignamos los resultados a la matriz de respuestas
ReDim Resuledo(0 To m, 0 To Numedo)
For i = 0 To m
   For j = 0 To Numedo
   Resuledo(i, j) = Y(i, j)
   Next j
Next i

Exit Sub
err1:
If err.Number <> 0 Then
   mensaje = MsgBox("Parámetros del algoritmo incompletos", 48, "Error en el
ingreso")
   End If

End Sub
Public Sub DisparoLineal()
Dim i As Integer
Dim j As Integer
Dim flag1   'Bandera que indica si el signo es + o - de q(t)
Dim flag2  'Bandera que indica si el signo es + o - de r(t)
Dim U() As Double     'Vector resultado del sistema2 EDO
Dim v() As Double     'Vector resultado del sistema2 EDO
Dim W() As Double      'vector de suma para el resultados
Dim XL() As Double    'Vector solución del sistema con condiciones en la frontera
Dim TL() As Double    'Vector de la variable t
Dim m As Double         'numero de intervalos
Dim funcionedo1 As String 'contiene la función del sistema1
Dim funcionedo2 As String 'contiene la función del sistema2
Dim mensaje As String

On Error GoTo err1
```

```
'Verificamos el signo de q(t)y r(t)
flag1 = InStr(1, LTrim(qlineal), "-", 1)
flag2 = InStr(1, LTrim(rlineal), "-", 1)

If flag1 <> 1 Then
flag1 = 0
End If

If flag2 <> 1 Then
flag2 = 0
End If


'Armamos el sistema de ecuaciones EDO 1
funcionedo1 = "(" & plineal & ")" & "*y2"
If flag1 = 0 Then
funcionedo1 = funcionedo1 & "+" & " ( " & qlineal & ")" & "*y1"
Else
funcionedo1 = funcionedo1 & "+1*" & "(" & qlineal & ")" & "*y1"
End If
If flag2 = 0 Then
funcionedo1 = funcionedo1 & " +" & rlineal
Else
funcionedo1 = funcionedo1 & rlineal
End If
ReDim fiedo(0 To 1) As String
fiedo(0) = "y2"
fiedo(1) = funcionedo1
'Resolvemos el primer sistema de valor inicial
ReDim Viniciales(0 To 1) As Double
Viniciales(0) = xa
Viniciales(1) = 0
Numedo = 2
SistemasEDO
'Asignamos la respuesta al vector u
m = (b - A) / h
ReDim U(0 To m)
For i = 0 To m
U(i) = Resuledo(i, 1)
Next i

'Armamos el sistema de ecuaciones EDO 2
funcionedo2 = "(" & plineal & ")" & "*y2"
If flag1 = 0 Then
funcionedo2 = funcionedo2 & "+" & "(" & qlineal & ")" & "*y1"
Else
```

```
funcionedo2 = funcionedo2 & "+1*" & " (" & qlineal & ")" & "*y1"
End If

ReDim fiedo(0 To 1) As String
fiedo(0) = "y2"
fiedo(1) = funcionedo2
'Resolvemos el primer sistema de valor inicial
ReDim Viniciales(0 To 1) As Double
Viniciales(0) = 0
Viniciales(1) = 1
Numedo = 2
SistemasEDO
'Asignamos la respuesta al vector v

ReDim v(0 To m)
For i = 0 To m
v(i) = Resuledo(i, 1)
Next i
'Obtenemos el vecto w
ReDim W(0 To m)
For i = 0 To m
W(i) = ((xb - U(m)) / v(m)) * v(i)
Next i

' obtenemos la respuesta del sistema de valor en la frontera
ReDim XL(0 To m)
For i = 0 To m
XL(i) = U(i) + W(i)
Next i
'Asignamos la respuesta a la matriz de resultados
ReDim TL(0 To m) As Double
For i = 0 To m
TL(i) = Resuledo(i, 0)
Next i
ReDim Resuledo(0 To m, 0 To 4) As Double
For i = 0 To m
Resuledo(i, 0) = TL(i)
Resuledo(i, 1) = U(i)
Resuledo(i, 2) = v(i)
Resuledo(i, 3) = W(i)
Resuledo(i, 4) = XL(i)
Next i
Exit Sub
err1:
'If err.Number <> 0 Then
```

```
    mensaje = MsgBox("Parámetros del algoritmo incompletos", 48, "Error en el
ingreso")
   ' End If


End Sub


'Permite reemplazar los valores numéricos en la función de edo y evaluarla mediante
la
'llamada al objeto MscExcel
Private Function reemplazafedo(f As String, te As Double, ye As Double) As Double
  Dim nx
  Dim NF
  Dim nL
  Dim Ns
  Dim nr
On Error GoTo err1
  'Reemplaza t
  NF = f
  nx = te
  nL = Len(nx)
  Ns = InStr(1, nx, ",")
  nr = nL - Ns
  If Ns <> 0 Then


  nx = Mid(nx, 1, InStr(1, nx, ",") - 1) & "." & Mid(nx, InStr(1, nx, ",") + 1, Len(Mid(nx, 1,
nr)))
  End If

  While InStr(1, NF, "t") > 0
    NF = Mid(NF, 1, InStr(1, NF, "t") - 1) & nx & Mid(NF, InStr(1, NF, "t") + 1, Len(NF))
  Wend

'Reemplaza y

  nx = ye
  nL = Len(nx)
  Ns = InStr(1, nx, ",")
  nr = nL - Ns
  If Ns <> 0 Then


  nx = Mid(nx, 1, InStr(1, nx, ",") - 1) & "." & Mid(nx, InStr(1, nx, ",") + 1, Len(Mid(nx, 1,
nr)))
```

```
 End If

 While InStr(1, NF, "y") > 0
   NF = Mid(NF, 1, InStr(1, NF, "y") - 1) & nx & Mid(NF, InStr(1, NF, "y") + 1, Len(NF))
 Wend

Evaluarfedo.fa = UCase(NF)
Evaluarfedo.Tipox = True
Evaluarfedo.evaluarf
reemplazafedo = Evaluarfedo.Fdy

Exit Function
err1:
End Function
'Permite evaluar el sistema de EDO de n ecuaciones mediante la llamada al objeto
MscExcel
Private Function Evaluarsistema(f As String, valores() As Double) As Double
 Dim i As Integer
 Dim nx
 Dim NF
 Dim nL
 Dim Ns
 Dim nr
 Dim variables
 On Error GoTo err1
 'Reemplaza t
 NF = f
 nx = valores(0)
 nL = Len(nx)
 Ns = InStr(1, nx, ",")
 nr = nL - Ns
 If Ns <> 0 Then


 nx = Mid(nx, 1, InStr(1, nx, ",") - 1) & "." & Mid(nx, InStr(1, nx, ",") + 1, Len(Mid(nx, 1,
nr)))
 End If

 While InStr(1, NF, "t") > 0
   NF = Mid(NF, 1, InStr(1, NF, "t") - 1) & nx & Mid(NF, InStr(1, NF, "t") + 1, Len(NF))
 Wend
  'f = NF
 'Reemplaza el resto de variables
 For i = 1 To Numedo

 nx = valores(i)
```

```
  nL = Len(nx)
  Ns = InStr(1, nx, ",")
  nr = nL - Ns
  variables = "y" & i
  If Ns <> 0 Then


  nx = Mid(nx, 1, InStr(1, nx, ",") - 1) & "." & Mid(nx, InStr(1, nx, ",") + 1, Len(Mid(nx, 1,
nr)))
  End If

  While InStr(1, NF, variables) > 0
    NF = Mid(NF, 1, InStr(1, NF, variables) - 1) & nx & Mid(NF, InStr(1, NF, variables)
+ 2, Len(NF))
  Wend
 Next i

Evaluarfedo.fa = UCase(NF)
Evaluarfedo.Tipox = True
Evaluarfedo.evaluarf
Evaluarsistema = Evaluarfedo.Fdy
Exit Function
err1:
End Function
```

Realizar una interpolación polinomial

Interfaz FrmInterpolación

```
Option Explicit
'En Microsoft TechNet puedes encontrar este artículo:
'HOWTO: Use HTML Help API in a Visual Basic 5.0 Application
'PSS ID Number: Q183434
'
'Aunque la definición de la Enumeración y la primera declaración
'es de las news
'
'Htmlhelp consts
Private Enum HH_COMMAND
   HH_DISPLAY_TOPIC = &H0
   HH_HELP_FINDER = &H0       ' WinHelp equivalent
   HH_DISPLAY_TOC = &H1       ' not currently implemented
   HH_DISPLAY_INDEX = &H2     ' not currently implemented
   HH_DISPLAY_SEARCH = &H3    ' not currently implemented
   HH_SET_WIN_TYPE = &H4
   HH_GET_WIN_TYPE = &H5
   HH_GET_WIN_HANDLE = &H6
```

```
    HH_GET_INFO_TYPES = &H7     ' not currently implemented
    HH_SET_INFO_TYPES = &H8     ' not currently implemented
    HH_SYNC = &H9
    HH_ADD_NAV_UI = &HA         ' not currently implemented
    HH_ADD_BUTTON = &HB         ' not currently implemented
    HH_GETBROWSER_APP = &HC     ' not currently implemented
    HH_KEYWORD_LOOKUP = &HD
    HH_DISPLAY_TEXT_POPUP = &HE ' display string resource id
                        ' or text in a popup window
    HH_HELP_CONTEXT = &HF       ' display mapped numeric value
                        ' in dwData
    HH_TP_HELP_CONTEXTMENU      ' Text pop-up help, similar to
                        ' WinHelp's HELP_CONTEXTMENU.
    HH_TP_HELP_WM_HELP = &H11   ' text pop-up help, similar to
                        ' WinHelp's HELP_WM_HELP.
    HH_CLOSE_ALL = &H12         ' close all windows opened directly
                        ' or indirectly by the caller
    HH_ALINK_LOOKUP = &H13      ' ALink version of HH_KEYWORD_LOOKUP
End Enum

'HtmlHelp api call
'NOTA: Si se usa esta forma, hay que indicar el último parámetro
'     con la palabra ByVal delante...
'Private Declare Function HtmlHelp Lib "hhctrl.ocx" Alias "HtmlHelpA" _
    (ByVal hwndCaller As Long, ByVal pszFile As String, _
    ByVal uCommand As HH_COMMAND, dwData As Any) As Long
'Con esta funciona perfectamente
Private Declare Function HtmlHelp Lib "hhctrl.ocx" Alias "HtmlHelpA" _
    (ByVal hwndCaller As Long, ByVal pszFile As String, _
    ByVal uCommand As HH_COMMAND, ByVal dwData As Long) As Long


Dim Evaluar As New McsExcel
Dim Interpolación As New ClasInterpolación
Dim Puntos() As Double
Dim derivadas() As Double
Dim IntpolinomioSpline() As String
Dim n As Integer
Dim grado As Integer
Dim polinomio
Dim punto As Double
Dim polievaluado As Double
Dim tipoalg As Integer
Dim mensaje As String
Dim aviso As Boolean
Dim arribaFraimprimir
```

```
Dim arribaFraResul
Dim anchoFraResul
Dim altoFraResul
Dim anchoFgresul
Dim altoFgresul
Dim arribaSstab
Dim izquierdaSstab
Dim anchoSstab
Dim altoSstab
Dim arribaFgitera
Dim izquierdaFgitera
Dim anchoFgitera
Dim altoFgitera
Dim anchocolumna
Dim VLabel As Boolean
Dim Vcmd As Boolean
Dim colororg
Private Acepta As Boolean
Private VLConexion As ADODB.Connection
Private VLRegistro As ADODB.Recordset
Private VLNombre As String
Private VLNombreTrabajo As String
Private VLVerifica As Boolean
Private VLBDD As Boolean
Private VLJacobiano As String
Private VLJacobianoMatriz(1 To 1000) As Double

Property Let Aceptar(parAceptar As Boolean)
   Acepta = parAceptar
End Property

Property Let Nombre(parNombre As String)
   VLNombre = parNombre
End Property

Property Let NombreTrabajo(parNombreTrabajo As String)
   VLNombreTrabajo = parNombreTrabajo
End Property


Private Sub Command1_Click()
   PLImprimir
End Sub

'Private Sub CheckPoli_Click()
'If CheckPoli.Value = 1 Then
```

```
'
'TxtPoli.Visible = True
"Else
'TxtPoli.Visible = False
'End If
'End Sub



Private Sub CmbPoli_Click()
Dim indice As Integer
indice = Val(CmbPoli.Text)
TxtPoli.Text = fgIntresultados.TextMatrix(indice, 1)
TxtPoli.Visible = True
End Sub

Private Sub CmdIntiteraciones_Click()
Label20.Visible = True
fgIntiteraciones.Visible = True
End Sub

Private Sub Form_Load()
   Intabrir.Enabled = False
   Intguardarcomo.Enabled = False
   Intimprimir.Enabled = False
   VGForma = "IP"
TxtPoli.Visible = False
End Sub

Private Sub Intabrir_Click()
   FrmRecuperaTrabajo.Show vbModal
   If Acepta = True Then
      PLConexion True
      PLRecupera
   End If
End Sub

Private Sub Intayuda_Click()
'Así se llamaría para mostrar un tópico de la ayuda
   Dim h As Long

   'h = HtmlHelp(Me.hWnd, "Milton.hlp", HH_DISPLAY_TOPIC, 0&)

   h = HtmlHelp(Me.hWnd, "Milton.hlp", HH_HELP_CONTEXT, 2050&)
End Sub
```

```
Private Sub Intergraficar_Click()
Frmescalas.Show
End Sub

Private Sub Intguardarcomo_Click()
   FrmNombre.Show vbModal
   If Acepta = True Then
      PLConexion True
      PLGuardarBDD
   End If
End Sub

Private Sub Intimprimir_Click()
Dim EndTime As Date
   Sincolor

      EndTime = DateAdd("s", 1 / 1E+16, Now)
      Do Until Now > EndTime
        DoEvents
      Loop
   'Form1.Command3_Click
   Set Form1.Picture1.Picture = CaptureClient(Me)
   Form1.Visible = True
   FrmPrincipalalgoritmos.Visible = False
   Concolor
End Sub

'SELECCION DEL TIPO DE ALGORITMO
Private Sub Intregresión_Click()
'Seteamos la variable tipoalg para indicar que tipo de algoritmos es
tipoalg = 1
   Intabrir.Enabled = True
   Intguardarcomo.Enabled = True
   Intimprimir.Enabled = False
   FrmRecuperaTrabajo.ver = tipoalg

' Borramos y presentamos todos los contorles necesarios en el formulario
LblInttitulo.Visible = False
LblIntnumpuntos.Visible = True
LblIntgrado.Visible = True
LblIntpunto.Visible = False
LblIntresultados.Visible = True
LblIntderivada1.Visible = False
LblIntderivada2.Visible = False
TxtIntderivada1.Visible = False
TxtIntderivada2.Visible = False
```

```
ShapeInt.Visible = True
TxtIntnumpuntos.Text = " "
TxtIntnumpuntos.Visible = True
TxtIntgrado.Text = " "
TxtIntgrado.Visible = True
TxtIntpunto.Text = " "
TxtIntpunto.Visible = False
FraIntevaluar.Visible = False
CmdIntevaluar.Visible = False
CmdIntiteraciones.Visible = False
CmdIntcalcular.Visible = False
CmdIntgraficar.Visible = False
Fg2int.Visible = False
fgIntresultados.Visible = True
fgIntiteraciones.Visible = False
Fg2int.Clear
fgIntiteraciones.Clear
fgIntresultados.Clear
Label20.Visible = False
TxtPoli.Visible = False
CmbPoli.Visible = False
LblNomalg.Caption = "Regresión Polinomial"
LblIntSintaxis.Caption = "Número de pares de datos= número natural,representa la
cantidad de pares de puntos a ingresar para ajustarlos a un polinomio" & Chr(10) & _
" Grado del polinomio= número nátural, que representa el grado del polinomio al cual
se ajustarán los pares de puntos ingresados"
End Sub

Private Sub Intsalir_Click()
Unload Me
End Sub

Private Sub Inttécnica_Click()
'Seteamos la variable tipoalg para indicar que tipo de algoritmos es
tipoalg = 2
   Intabrir.Enabled = True
   Intguardarcomo.Enabled = True
   FrmRecuperaTrabajo.ver = tipoalg
Intimprimir.Enabled = False
' Borramos y presentamos todos los contorles necesarios en el formulario
LblInttitulo.Visible = False
LblIntnumpuntos.Visible = False
LblIntgrado.Visible = True
LblIntpunto.Visible = False
LblIntresultados.Visible = True
LblIntderivada1.Visible = False
```

```
LblIntderivada2.Visible = False
TxtIntderivada1.Visible = False
TxtIntderivada2.Visible = False
ShapeInt.Visible = True
TxtIntnumpuntos.Text = " "
TxtIntnumpuntos.Visible = False
TxtIntgrado.Text = " "
TxtIntgrado.Visible = True
TxtIntpunto.Text = " "
TxtIntpunto.Visible = False
FraIntevaluar.Visible = False
CmdIntevaluar.Visible = False
CmdIntiteraciones.Visible = False
CmdIntcalcular.Visible = False
CmdIntgraficar.Visible = False
Fg2int.Visible = False
fgIntresultados.Visible = True
fgIntiteraciones.Visible = False
Fg2int.Clear
fgIntiteraciones.Clear
fgIntresultados.Clear
Label20.Visible = False
TxtPoli.Visible = False
CmbPoli.Visible = False
LblNomalg.Caption = "Técnica Matricial"
LblIntSintaxis.Caption = " Grado del polinomio= número nátural, que representa el
grado del polinomio al cual se ajustarán los pares de puntos ingresados"
End Sub
Private Sub Intlagrange_Click()
'Seteamos la variable tipoalg para indicar que tipo de algoritmos es
tipoalg = 3
   Intabrir.Enabled = True
   Intguardarcomo.Enabled = True
   FrmRecuperaTrabajo.ver = tipoalg
Intimprimir.Enabled = False
' Borramos y presentamos todos los contorles necesarios en el formulario
LblInttitulo.Visible = False
LblIntnumpuntos.Visible = False
LblIntgrado.Visible = True
LblIntpunto.Visible = True
LblIntresultados.Visible = True
LblIntderivada1.Visible = False
LblIntderivada2.Visible = False
TxtIntderivada1.Visible = False
TxtIntderivada2.Visible = False
ShapeInt.Visible = True
```

```
TxtIntnumpuntos.Text = " "
TxtIntnumpuntos.Visible = False
TxtIntgrado.Text = " "
TxtIntgrado.Visible = True
TxtIntpunto.Text = " "
TxtIntpunto.Visible = True
FraIntevaluar.Visible = True
CmdIntevaluar.Visible = False
CmdIntiteraciones.Visible = False
CmdIntcalcular.Visible = False
CmdIntgraficar.Visible = False
Fg2int.Visible = False
fgIntresultados.Visible = True
fgIntiteraciones.Visible = False
Fg2int.Clear
fgIntiteraciones.Clear
fgIntresultados.Clear
Label20.Visible = False
TxtPoli.Visible = False
CmbPoli.Visible = False
LblNomalg.Caption = "Interpolación de Lagrange"
LblIntSintaxis.Caption = " Grado del polinomio= número nátural, que representa el
grado del polinomio al cual se ajustarán los pares de puntos ingresados" & Chr(10) &
_
" Punto a evaluar= punto en el cual se evaluará el polinomio interpolador"
End Sub
Private Sub Intnewton_Click()
'Seteamos la variable tipoalg para indicar que tipo de algoritmos es
tipoalg = 4
   Intabrir.Enabled = True
   Intguardarcomo.Enabled = True
   FrmRecuperaTrabajo.ver = tipoalg
Intimprimir.Enabled = False
' Borramos y presentamos todos los contorles necesarios en el formulario
LblInttitulo.Visible = False
LblIntnumpuntos.Visible = False
LblIntgrado.Visible = True
LblIntpunto.Visible = True
LblIntresultados.Visible = True
LblIntderivada1.Visible = False
LblIntderivada2.Visible = False
TxtIntderivada1.Visible = False
TxtIntderivada2.Visible = False
ShapeInt.Visible = True
TxtIntnumpuntos.Text = " "
TxtIntnumpuntos.Visible = False
```

```
TxtIntgrado.Text = " "
TxtIntgrado.Visible = True
TxtIntpunto.Text = " "
TxtIntpunto.Visible = True
FraIntevaluar.Visible = True
CmdIntevaluar.Visible = False
CmdIntiteraciones.Visible = False
CmdIntcalcular.Visible = False
CmdIntgraficar.Visible = False
Fg2int.Visible = False
fgIntresultados.Visible = True
fgIntiteraciones.Visible = False
Fg2int.Clear
fgIntiteraciones.Clear
fgIntresultados.Clear
Label20.Visible = False
TxtPoli.Visible = False
CmbPoli.Visible = False
LblNomalg.Caption = "Interpolación de Newton"
LblIntSintaxis.Caption = " Grado del polinomio= número nátural, que representa el
grado del polinomio al cual se ajustarán los pares de puntos ingresados" & Chr(10) &
_
" Punto a evaluar= punto en el cual se evaluará el polinomio interpolador"
End Sub
Private Sub Inttrigonométrica_Click()
'Seteamos la variable tipoalg para indicar que tipo de algoritmos es
tipoalg = 5
   Intabrir.Enabled = True
   Intguardarcomo.Enabled = True
   FrmRecuperaTrabajo.ver = tipoalg
Intimprimir.Enabled = False
' Borramos y presentamos todos los contorles necesarios en el formulario
LblInttitulo.Visible = False
LblIntnumpuntos.Visible = True
LblIntgrado.Visible = True
LblIntpunto.Visible = False
LblIntresultados.Visible = True
LblIntderivada1.Visible = False
LblIntderivada2.Visible = False
TxtIntderivada1.Visible = False
TxtIntderivada2.Visible = False
ShapeInt.Visible = True
TxtIntnumpuntos.Text = " "
TxtIntnumpuntos.Visible = True
TxtIntgrado.Text = " "
TxtIntgrado.Visible = True
```

```
TxtIntpunto.Text = " "
TxtIntpunto.Visible = False
FraIntevaluar.Visible = False
CmdIntevaluar.Visible = False
CmdIntiteraciones.Visible = False
CmdIntcalcular.Visible = False
CmdIntgraficar.Visible = False
Fg2int.Visible = False
fgIntresultados.Visible = True
fgIntiteraciones.Visible = False
Fg2int.Clear
fgIntiteraciones.Clear
fgIntresultados.Clear
Label20.Visible = False
TxtPoli.Visible = False
CmbPoli.Visible = False
LblNomalg.Caption = "Interpolación Trigonométrica"
LblIntSintaxis.Caption = "Número de pares de datos= número natural,representa la
cantidad de pares de puntos a ingresar para ajustarlos a un polinomio" & Chr(10) & _
" Grado del polinomio= número nátural, que representa el grado del polinomio al cual
se ajustarán los pares de puntos ingresados" & Chr(10) & " Nota: Grado del
polinomio < Número de datos/2;valores equiespaciados en [-pi,pi]"
End Sub
Private Sub Intlineal_Click()
'Seteamos la variable tipoalg para indicar que tipo de algoritmos es
tipoalg = 6
   Intabrir.Enabled = True
   Intguardarcomo.Enabled = True
   FrmRecuperaTrabajo.ver = tipoalg
Intimprimir.Enabled = False
' Borramos y presentamos todos los contorles necesarios en el formulario
LblInttitulo.Visible = False
LblIntnumpuntos.Visible = True
LblIntgrado.Visible = False
LblIntpunto.Visible = False
LblIntresultados.Visible = True
LblIntderivada1.Visible = False
LblIntderivada2.Visible = False
TxtIntderivada1.Visible = False
TxtIntderivada2.Visible = False
ShapeInt.Visible = True
TxtIntnumpuntos.Text = " "
TxtIntnumpuntos.Visible = True
TxtIntgrado.Text = " "
TxtIntgrado.Visible = False
TxtIntpunto.Text = " "
```

```
TxtIntpunto.Visible = False
FraIntevaluar.Visible = False
CmdIntevaluar.Visible = False
CmdIntiteraciones.Visible = False
CmdIntcalcular.Visible = False
CmdIntgraficar.Visible = False
Fg2int.Visible = False
fgIntresultados.Visible = True
fgIntiteraciones.Visible = False
Fg2int.Clear
fgIntiteraciones.Clear
fgIntresultados.Clear
Label20.Visible = False
TxtPoli.Visible = False
CmbPoli.Visible = False
LblNomalg.Caption = "Interpolación Segmentaria Lineal"
LblIntSintaxis.Caption = "Número de pares de datos= número natural,representa la
cantidad de pares de puntos a ingresar para ajustarlos a un polinomio"
End Sub
Private Sub Intsujeta_Click()
'Seteamos la variable tipoalg para indicar que tipo de algoritmos es
tipoalg = 7
   Intabrir.Enabled = True
   Intguardarcomo.Enabled = True
   FrmRecuperaTrabajo.ver = tipoalg
Intimprimir.Enabled = False
' Borramos y presentamos todos los contorles necesarios en el formulario
LblInttitulo.Visible = False
LblIntnumpuntos.Visible = True
LblIntgrado.Visible = False
LblIntpunto.Visible = False
LblIntresultados.Visible = True
LblIntderivada1.Caption = "S' (Xo) ="
LblIntderivada2.Caption = "S' (Xn) ="
LblIntderivada1.Visible = True
LblIntderivada2.Visible = True
TxtIntderivada1.Text = " "
TxtIntderivada1.Visible = True
TxtIntderivada2.Text = " "
TxtIntderivada2.Visible = True
ShapeInt.Visible = True
TxtIntnumpuntos.Text = " "
TxtIntnumpuntos.Visible = True
TxtIntgrado.Text = " "
TxtIntgrado.Visible = False
TxtIntpunto.Text = " "
```

```
TxtIntpunto.Visible = False
FraIntevaluar.Visible = False
CmdIntevaluar.Visible = False
CmdIntiteraciones.Visible = False
CmdIntcalcular.Visible = False
CmdIntgraficar.Visible = False
Fg2int.Visible = False
fgIntresultados.Visible = True
fgIntiteraciones.Visible = False
Fg2int.Clear
fgIntiteraciones.Clear
fgIntresultados.Clear
Label20.Visible = False
TxtPoli.Visible = False
CmbPoli.Visible = False
LblNomalg.Caption = "Interpolación Segmentaria Lineal"
LblIntSintaxis.Caption = "Número de pares de datos= número natural,representa la
cantidad de pares de puntos a ingresar para ajustarlos a un polinomio"
End Sub
Private Sub Intsujeta_Click()
'Seteamos la variable tipoalg para indicar que tipo de algoritmos es
tipoalg = 7
   Intabrir.Enabled = True
   Intguardarcomo.Enabled = True
   FrmRecuperaTrabajo.ver = tipoalg
Intimprimir.Enabled = False
' Borramos y presentamos todos los contorles necesarios en el formulario
LblInttitulo.Visible = False
LblIntnumpuntos.Visible = True
LblIntgrado.Visible = False
LblIntpunto.Visible = False
LblIntresultados.Visible = True
LblIntderivada1.Caption = "S' (Xo) ="
LblIntderivada2.Caption = "S' (Xn) ="
LblIntderivada1.Visible = True
LblIntderivada2.Visible = True
TxtIntderivada1.Text = " "
TxtIntderivada1.Visible = True
TxtIntderivada2.Text = " "
TxtIntderivada2.Visible = True
ShapeInt.Visible = True
TxtIntnumpuntos.Text = " "
TxtIntnumpuntos.Visible = True
TxtIntgrado.Text = " "
TxtIntgrado.Visible = False
TxtIntpunto.Text = " "
```

```
TxtIntpunto.Visible = False
FraIntevaluar.Visible = False
CmdIntevaluar.Visible = False
CmdIntiteraciones.Visible = False
CmdIntcalcular.Visible = False
CmdIntgraficar.Visible = False
Fg2int.Visible = False
fgIntresultados.Visible = True
fgIntiteraciones.Visible = False
Fg2int.Clear
fgIntiteraciones.Clear
fgIntresultados.Clear
Label20.Visible = False
TxtPoli.Visible = False
CmbPoli.Visible = False
LblNomalg.Caption = "Interpolación segmentaria cúbica sujeta"
LblIntSintaxis.Caption = "Número de pares de datos= número natural,representa la
cantidad de pares de puntos a ingresar para ajustarlos a un polinomio" & Chr(10) & _
"S'(Xo)= valor de la primera derivada en la primera abscisa" & Chr(10) & "S'(Xn)=
valor de la primera derivada en la ultima abscisa"
End Sub
Private Sub Intnatural_Click()
'Seteamos la variable tipoalg para indicar que tipo de algoritmos es
tipoalg = 8
   Intabrir.Enabled = True
   Intguardarcomo.Enabled = True
   FrmRecuperaTrabajo.ver = tipoalg
Intimprimir.Enabled = False
' Borramos y presentamos todos los contorles necesarios en el formulario
LblInttitulo.Visible = False
LblIntnumpuntos.Visible = True
LblIntgrado.Visible = False
LblIntpunto.Visible = False
LblIntresultados.Visible = True
LblIntderivada1.Visible = False
LblIntderivada2.Visible = False
TxtIntderivada1.Visible = False
TxtIntderivada2.Visible = False
ShapeInt.Visible = True
TxtIntnumpuntos.Text = " "
TxtIntnumpuntos.Visible = True
TxtIntgrado.Text = " "
TxtIntgrado.Visible = False
TxtIntpunto.Text = " "
TxtIntpunto.Visible = False
FraIntevaluar.Visible = False
```

```
CmdIntevaluar.Visible = False
CmdIntiteraciones.Visible = False
CmdIntcalcular.Visible = False
CmdIntgraficar.Visible = False
Fg2int.Visible = False
fgIntresultados.Visible = True
fgIntiteraciones.Visible = False
Fg2int.Clear
fgIntiteraciones.Clear
fgIntresultados.Clear
Label20.Visible = False
LblNomalg.Caption = "Interpolación segmentaria cúbica natural"
LblIntSintaxis.Caption = "Número de pares de datos= número natural,representa la
cantidad de pares de puntos a ingresar para ajustarlos a un polinomio"
End Sub
Private Sub Intextrapolada_Click()
'Seteamos la variable tipoalg para indicar que tipo de algoritmos es
tipoalg = 9
   Intabrir.Enabled = True
   Intguardarcomo.Enabled = True
   FrmRecuperaTrabajo.ver = tipoalg
Intimprimir.Enabled = False
' Borramos y presentamos todos los contorles necesarios en el formulario
LblInttitulo.Visible = False
LblIntnumpuntos.Visible = True
LblIntgrado.Visible = False
LblIntpunto.Visible = False
LblIntresultados.Visible = True
LblIntderivada1.Visible = False
LblIntderivada2.Visible = False
TxtIntderivada1.Visible = False
TxtIntderivada2.Visible = False
ShapeInt.Visible = True
TxtIntnumpuntos.Text = " "
TxtIntnumpuntos.Visible = True
TxtIntgrado.Text = " "
TxtIntgrado.Visible = False
TxtIntpunto.Text = " "
TxtIntpunto.Visible = False
FraIntevaluar.Visible = False
CmdIntevaluar.Visible = False
CmdIntiteraciones.Visible = False
CmdIntcalcular.Visible = False
CmdIntgraficar.Visible = False
Fg2int.Visible = False
fgIntresultados.Visible = True
```

```
fgIntiteraciones.Visible = False
Fg2int.Clear
fgIntiteraciones.Clear
fgIntresultados.Clear
Label20.Visible = False
TxtPoli.Visible = False
CmbPoli.Visible = False
LblNomalg.Caption = "Interpolación segmentaria cúbica extrapolada"
LblIntSintaxis.Caption = "Número de pares de datos= número natural,representa la
cantidad de pares de puntos a ingresar para ajustarlos a un polinomio"
End Sub
Private Sub Intparabolica_Click()
'Seteamos la variable tipoalg para indicar que tipo de algoritmos es
tipoalg = 10
   Intabrir.Enabled = True
   Intguardarcomo.Enabled = True
   FrmRecuperaTrabajo.ver = tipoalg
Intimprimir.Enabled = False
' Borramos y presentamos todos los contorles necesarios en el formulario
LblInttitulo.Visible = False
LblIntnumpuntos.Visible = True
LblIntgrado.Visible = False
LblIntpunto.Visible = False
LblIntresultados.Visible = True
LblIntderivada1.Visible = False
LblIntderivada2.Visible = False
TxtIntderivada1.Visible = False
TxtIntderivada2.Visible = False
ShapeInt.Visible = True
TxtIntnumpuntos.Text = " "
TxtIntnumpuntos.Visible = True
TxtIntgrado.Text = " "
TxtIntgrado.Visible = False
TxtIntpunto.Text = " "
TxtIntpunto.Visible = False
FraIntevaluar.Visible = False
CmdIntevaluar.Visible = False
CmdIntiteraciones.Visible = False
CmdIntcalcular.Visible = False
CmdIntgraficar.Visible = False
Fg2int.Visible = False
fgIntresultados.Visible = True
fgIntiteraciones.Visible = False
Fg2int.Clear
fgIntiteraciones.Clear
fgIntresultados.Clear
```

```
Label20.Visible = False
TxtPoli.Visible = False
CmbPoli.Visible = False
LblNomalg.Caption = "Interpolación segmentaria cúbica con terminación parabólica"
LblIntSintaxis.Caption = "Número de pares de datos= número natural,representa la
cantidad de pares de puntos a ingresar para ajustarlos a un polinomio"
End Sub
Private Sub Intcurvatura_Click()
'Seteamos la variable tipoalg para indicar que tipo de algoritmos es
tipoalg = 11
    Intabrir.Enabled = True
    Intguardarcomo.Enabled = True
    FrmRecuperaTrabajo.ver = tipoalg
Intimprimir.Enabled = False
' Borramos y presentamos todos los contorles necesarios en el formulario
LblInttitulo.Visible = False
LblIntnumpuntos.Visible = True
LblIntgrado.Visible = False
LblIntpunto.Visible = False
LblIntresultados.Visible = True
LblIntderivada1.Caption = "S'" (Xo) ="
LblIntderivada2.Caption = "S'" (Xn) ="
LblIntderivada1.Visible = True
LblIntderivada2.Visible = True
TxtIntderivada1.Text = " "
TxtIntderivada1.Visible = True
TxtIntderivada2.Text = " "
TxtIntderivada2.Visible = True
ShapeInt.Visible = True
TxtIntnumpuntos.Text = " "
TxtIntnumpuntos.Visible = True
TxtIntgrado.Text = " "
TxtIntgrado.Visible = False
TxtIntpunto.Text = " "
TxtIntpunto.Visible = False
FraIntevaluar.Visible = False
CmdIntevaluar.Visible = False
CmdIntiteraciones.Visible = False
CmdIntcalcular.Visible = False
CmdIntgraficar.Visible = False
Fg2int.Visible = False
fgIntresultados.Visible = True
fgIntiteraciones.Visible = False
Fg2int.Clear
fgIntiteraciones.Clear
fgIntresultados.Clear
```

```
Label20.Visible = False
TxtPoli.Visible = False
CmbPoli.Visible = False
LblNomalg.Caption = "Interpolación segmentaria cúbica con curvatura en los
extremos"
LblIntSintaxis.Caption = "Número de pares de datos= número natural,representa la
cantidad de pares de puntos a ingresar para ajustarlos a un polinomio" & Chr(10) & _
"S"(Xo)= valor de la segunda derivada en la primera abscisa" & Chr(10) & "S"(Xn)=
valor de la segunda derivada en la ultima abscisa"
End Sub
Private Sub CmdIntcalcular_Click()
Dim i As Integer
On Error GoTo err1
Intimprimir.Enabled = True
Select Case (tipoalg)
Case Is = 1
If (TxtIntnumpuntos = "" Or TxtIntnumpuntos = " " Or TxtIntgrado = "" Or TxtIntgrado =
"") Then
mensaje = MsgBox("Parámetros del algoritmo incompletos", 48, "Error en el ingreso")
Exit Sub
End If

Enviar
Interpolación.RegresiónPolinomial
polinomio = Interpolación.intPolinomio
Intpresentar
fgIntresultados.TextMatrix(2, 1) = " "
fgIntresultados.Visible = True
Label20.Caption = "Tabla de ajuste"
CmdIntiteraciones.Caption = "Tabla de ajuste"
CmdIntiteraciones.Visible = True
   FraIntevaluar.Visible = True
   CmdIntevaluar.Visible = True
   TxtIntpunto.Visible = True
   LblIntpunto.Visible = True
Case Is = 2
 If (TxtIntgrado = "" Or TxtIntgrado = " ") Then
mensaje = MsgBox("Parámetros del algoritmo incompletos", 48, "Error en el ingreso")
Exit Sub
End If
Enviar
Interpolación.TécnicaMatricial
polinomio = Interpolación.intPolinomio
Intpresentar
fgIntresultados.TextMatrix(2, 1) = " "
fgIntresultados.Visible = True
```

```
Label20.Caption = "Sistema de Ecuaciones"
CmdIntiteraciones.Caption = "Sistema de Ecuaciones"
CmdIntiteraciones.Visible = True
   FraIntevaluar.Visible = True
   CmdIntevaluar.Visible = True
   TxtIntpunto.Visible = True
   LblIntpunto.Visible = True

Case Is = 3
 If (TxtIntgrado = "" Or TxtIntgrado = " " Or TxtIntpunto = "" Or TxtIntpunto = " ") Then
mensaje = MsgBox("Parámetros del algoritmo incompletos", 48, "Error en el ingreso")
Exit Sub
End If
Enviar
Interpolación.Lagrange
polievaluado = Interpolación.intptoevaluado
Intpresentar
fgIntresultados.Visible = True
'CmdIntiteraciones.Visible = True
Case Is = 4
 If (TxtIntgrado = "" Or TxtIntgrado = " " Or TxtIntpunto = "" Or TxtIntpunto = " ") Then
mensaje = MsgBox("Parámetros del algoritmo incompletos", 48, "Error en el ingreso")
Exit Sub
End If
Enviar
Interpolación.Newton
polievaluado = Interpolación.intptoevaluado
Intpresentar
fgIntresultados.Visible = True
'CmdIntiteraciones.Visible = True
Case Is = 5
 If (TxtIntnumpuntos = "" Or TxtIntnumpuntos = " " Or TxtIntgrado = "" Or TxtIntgrado
= " ") Then
mensaje = MsgBox("Parámetros del algoritmo incompletos", 48, "Error en el ingreso")
Exit Sub
End If
Enviar
Interpolación.Trigonométrica
polinomio = Interpolación.intPolinomio
Intpresentar
fgIntresultados.TextMatrix(2, 1) = " "
fgIntresultados.Visible = True
'CmdIntiteraciones.Visible = True
   FraIntevaluar.Visible = True
   CmdIntevaluar.Visible = True
   TxtIntpunto.Visible = True
```

```
    LblIntpunto.Visible = True
Case Is = 6
 If (TxtIntnumpuntos = "" Or TxtIntnumpuntos = " ") Then
mensaje = MsgBox("Parámetros del algoritmo incompletos", 48, "Error en el ingreso")
Exit Sub
End If
Enviar
Interpolación.SplineLineal
ReDim IntpolinomioSpline(0 To n - 2)
For i = 0 To n - 2
IntpolinomioSpline(i) = Interpolación.IntpolinomioSpline(i)
Next i
Intpresentar

fgIntresultados.Visible = True
'CmdIntiteraciones.Visible = True
    FraIntevaluar.Visible = True
    CmdIntevaluar.Visible = True
    TxtIntpunto.Visible = True
    LblIntpunto.Visible = True
Case Is = 7
If (TxtIntnumpuntos = "" Or TxtIntnumpuntos = " " Or TxtIntderivada1 = "" Or
TxtIntderivada1 = " " Or TxtIntderivada1 = "" Or TxtIntderivada1 = " ") Then
mensaje = MsgBox("Parámetros del algoritmo incompletos", 48, "Error en el ingreso")
Exit Sub
End If
Enviar
Interpolación.SplineSujeta
ReDim IntpolinomioSpline(0 To n - 2)
For i = 0 To n - 2
IntpolinomioSpline(i) = Interpolación.IntpolinomioSpline(i)
Next i
Intpresentar

fgIntresultados.Visible = True
'CmdIntiteraciones.Visible = True
    FraIntevaluar.Visible = True
    CmdIntevaluar.Visible = True
    TxtIntpunto.Visible = True
    LblIntpunto.Visible = True
Case Is = 8
If (TxtIntnumpuntos = "" Or TxtIntnumpuntos = " ") Then
mensaje = MsgBox("Parámetros del algoritmo incompletos", 48, "Error en el ingreso")
Exit Sub
End If
Enviar
```

```
Interpolación.SplineNatural
ReDim IntpolinomioSpline(0 To n - 2)
For i = 0 To n - 2
IntpolinomioSpline(i) = Interpolación.IntpolinomioSpline(i)
Next i
Intpresentar

fgIntresultados.Visible = True
'CmdIntiteraciones.Visible = True
   FraIntevaluar.Visible = True
   CmdIntevaluar.Visible = True
   TxtIntpunto.Visible = True
   LblIntpunto.Visible = True
Case Is = 9
If (TxtIntnumpuntos = "" Or TxtIntnumpuntos = " ") Then
mensaje = MsgBox("Parámetros del algoritmo incompletos", 48, "Error en el ingreso")
Exit Sub
End If
Enviar
Interpolación.SplineExtrapolada
ReDim IntpolinomioSpline(0 To n - 2)
For i = 0 To n - 2
IntpolinomioSpline(i) = Interpolación.IntpolinomioSpline(i)
Next i
Intpresentar

fgIntresultados.Visible = True
'CmdIntiteraciones.Visible = True
   FraIntevaluar.Visible = True
   CmdIntevaluar.Visible = True
   TxtIntpunto.Visible = True
   LblIntpunto.Visible = True
Case Is = 10
If (TxtIntnumpuntos = "" Or TxtIntnumpuntos = " ") Then
mensaje = MsgBox("Parámetros del algoritmo incompletos", 48, "Error en el ingreso")
Exit Sub
End If
Enviar
Interpolación.SplineTerminaciónparabolica
ReDim IntpolinomioSpline(0 To n - 2)
For i = 0 To n - 2
IntpolinomioSpline(i) = Interpolación.IntpolinomioSpline(i)
Next i
Intpresentar

fgIntresultados.Visible = True
```

```
'CmdIntiteraciones.Visible = True
   FraIntevaluar.Visible = True
   CmdIntevaluar.Visible = True
   TxtIntpunto.Visible = True
   LblIntpunto.Visible = True
Case Is = 11
If (TxtIntnumpuntos = "" Or TxtIntnumpuntos = " " Or TxtIntderivada1 = "" Or
TxtIntderivada1 = " " Or TxtIntderivada1 = "" Or TxtIntderivada1 = " ") Then
mensaje = MsgBox("Parámetros del algoritmo incompletos", 48, "Error en el ingreso")
Exit Sub
End If
Enviar
Interpolación.SplineCurvaturaenlosextremos
ReDim IntpolinomioSpline(0 To n - 2)
For i = 0 To n - 2
IntpolinomioSpline(i) = Interpolación.IntpolinomioSpline(i)
Next i
Intpresentar

fgIntresultados.Visible = True
'CmdIntiteraciones.Visible = True
   FraIntevaluar.Visible = True
   CmdIntevaluar.Visible = True
   TxtIntpunto.Visible = True
   LblIntpunto.Visible = True
End Select
If tipoalg <> 3 Then
   If tipoalg <> 4 Then
   CmbPoli.Visible = True
   End If
End If
Exit Sub

err1:
End Sub

Private Sub CmdIntevaluar_Click()
Dim i As Integer
Dim mensaje As String
Dim bander As Boolean
On Error GoTo err1
bander = False
If tipoalg < 6 Then
Evaluar.PtX = Val(TxtIntpunto.Text)
Evaluar.fa = fgIntresultados.TextMatrix(1, 1)
Evaluar.Tipox = False
```

```
Evaluar.evaluarf
fgIntresultados.TextMatrix(2, 1) = Evaluar.Fdy
Set Evaluar = Nothing
Else
For i = 1 To n - 1
   If Val(TxtIntpunto.Text) < Puntos(i, 0) Then
      bander = True
      Evaluar.PtX = Val(TxtIntpunto.Text)
      Evaluar.fa = fgIntresultados.TextMatrix(i, 1)
      Evaluar.Tipox = False
      Evaluar.evaluarf
      fgIntresultados.TextMatrix(i, 2) = Evaluar.Fdy
      Set Evaluar = Nothing
      Exit Sub
   Else
      If Val(TxtIntpunto.Text) = Puntos(i, 0) Then
      bander = True
      Evaluar.PtX = Val(TxtIntpunto.Text)
      Evaluar.fa = fgIntresultados.TextMatrix(i, 1)
      Evaluar.Tipox = False
      Evaluar.evaluarf
      fgIntresultados.TextMatrix(i, 2) = Evaluar.Fdy
      Evaluar.PtX = Val(TxtIntpunto.Text)
      Evaluar.fa = fgIntresultados.TextMatrix(i + 1, 1)
      Evaluar.Tipox = False
      Evaluar.evaluarf
      fgIntresultados.TextMatrix(i + 1, 2) = Evaluar.Fdy
      Set Evaluar = Nothing
      Exit Sub

      End If
   End If
Next i
If bander = False Then
mensaje = MsgBox("El punto a evaluar debe pertenecer a un intervalo valido de la
respuesta", 16, "Error en el ingreso")
End If
End If
Exit Sub
err1:
End Sub



Private Sub TxtIntderivada1_Change()
LTrim (TxtIntderivada1)
```

```
End Sub

Private Sub TxtIntderivada1_KeyPress(KeyAscii As Integer)
'filtra la entrada de datos que no sean números incluye el signo - y el punto
  If (KeyAscii < 48 Or KeyAscii > 57) Then
      If (KeyAscii = 45) Then
      Exit Sub
      End If
      If (KeyAscii <> 46) Then
         If (KeyAscii <> 8) Then
         KeyAscii = 0
         End If
      End If
   End If
  If KeyAscii = Asc(vbCr) Then ' Elimina los retornos para quitar los pitidos.
  KeyAscii = 0
  End If

End Sub


Private Sub TxtIntderivada2_Change()
LTrim (TxtIntderivada2)
End Sub

Private Sub TxtIntderivada2_KeyPress(KeyAscii As Integer)
'filtra la entrada de datos que no sean números incluye el signo - y el punto
  If (KeyAscii < 48 Or KeyAscii > 57) Then
      If (KeyAscii = 45) Then
      Exit Sub
      End If
      If (KeyAscii <> 46) Then
         If (KeyAscii <> 8) Then
         KeyAscii = 0
         End If
      End If
   End If
  If KeyAscii = Asc(vbCr) Then ' Elimina los retornos para quitar los pitidos.
  KeyAscii = 0
  End If

End Sub

Private Sub TxtIntgrado_Change()
On Error GoTo err1
LTrim (TxtIntgrado)
```

```
Select Case tipoalg
Case Is = 1
If Val(TxtIntgrado) > 100 Then
mensaje = MsgBox("Fuera del rango", 48, "Error en el ingreo")
Exit Sub
End If
Case Is = 2
If Val(TxtIntgrado) > 99 Then
mensaje = MsgBox("Fuera del rango", 48, "Error en el ingreo")
Exit Sub
End If
Case Is = 3
If Val(TxtIntgrado) > 99 Then
mensaje = MsgBox("Fuera del rango", 48, "Error en el ingreo")
Exit Sub
End If
Case Is = 4
If Val(TxtIntgrado) > 99 Then
mensaje = MsgBox("Fuera del rango", 48, "Error en el ingreo")
Exit Sub
End If
Case Is = 5
If (Val(TxtIntgrado) >= (Val(TxtIntnumpuntos) / 2)) Then
mensaje = MsgBox("Fuera del rango", 48, "Error en el ingreso")
Exit Sub
End If

End Select



If tipoalg <> 1 Then
If tipoalg >= 5 Then
Exit Sub
End If
n = Val(TxtIntgrado.Text) + 1
If n = 0 Then
n = 1
End If
   Fg2int.Cols = 3
   Fg2int.Rows = n + 1
   Fg2int.FixedCols = 1
   Fg2int.FixedRows = 1
   titulosenfg
   LblInttitulo.Visible = True
   Fg2int.Visible = True
```

```
    CmdIntcalcular.Visible = True

    LblIntresultados.Visible = True
    ShapeInt.Visible = True


End If
Exit Sub
err1:
End Sub

Private Sub TxtIntgrado_KeyPress(KeyAscii As Integer)
'filtra la entrada de datos que no sean números enteros y positivos
  If (KeyAscii < 48 Or KeyAscii > 57) Then

        If (KeyAscii <> 8) Then
        KeyAscii = 0
        End If

   End If
  If KeyAscii = Asc(vbCr) Then ' Elimina los retornos para quitar los pitidos.
  KeyAscii = 0
  End If

End Sub

Private Sub TxtIntnumpuntos_Change()
On Error GoTo err1
LTrim (TxtIntnumpuntos)
n = Val(TxtIntnumpuntos.Text)

If n > 100 Then
mensaje = MsgBox("Fuera del rango", 48, "Error en el ingreso")
Exit Sub
End If


If n = 0 Then
n = 1
End If
   Fg2int.Cols = 3
   Fg2int.Rows = n + 1
   Fg2int.FixedCols = 1
   Fg2int.FixedRows = 1
   titulosenfg
   LblInttitulo.Visible = True
```

```
    Fg2int.Visible = True
    CmdIntcalcular.Visible = True

    LblIntresultados.Visible = True
    ShapeInt.Visible = True
    If tipoalg < 6 Then
    LblIntgrado.Visible = True
    TxtIntgrado.Visible = True
    End If
Exit Sub
err1:
End Sub

Private Sub titulosenfg()
Dim i As Integer
On Error GoTo err1
Fg2int.ColWidth(0) = 500
Fg2int.ColWidth(1) = 1050
Fg2int.ColWidth(2) = 1050

For i = Fg2int.FixedRows To Fg2int.Rows - 1
Fg2int.TextArray(Fgi(i, 0)) = i
Next i
Fg2int.TextArray(Fgi(0, 2)) = "Y"
Fg2int.TextArray(Fgi(0, 1)) = "X"




  ' Inicializar el cuadro de edición (lo carga ahora).
  TxtEdit = ""
Exit Sub
err1:
End Sub
Function Fgit(r As Integer, c As Integer) As Integer
  Fgit = c + fgIntiteraciones.Cols * r
End Function
Function Fgi(r As Integer, c As Integer) As Integer
  Fgi = c + Fg2int.Cols * r
End Function
Function Fgr(r As Integer, c As Integer) As Integer
  Fgr = c + fgIntresultados.Cols * r
End Function
Sub Fg2int_KeyPress(KeyAscii As Integer)
  MSHFlexGridEdit Fg2int, TxtEdit, KeyAscii
```

```
End Sub

Sub Fg2int_DblClick()
  MSHFlexGridEdit Fg2int, TxtEdit, 32 ' Simula un espacio.
End Sub

Sub MSHFlexGridEdit(MSHFlexGrid As Control, _
Edt As Control, KeyAscii As Integer)

  ' Usar el carácter escrito.
  Select Case KeyAscii

  ' Un espacio significa modificar el texto actual.
  Case 0 To 32
    Edt = MSHFlexGrid
    Edt.SelStart = 1000

  ' Otro carácter reemplaza el texto actual.
  Case Else
    Edt = Chr(KeyAscii)
    Edt.SelStart = 1
  End Select

  ' Mostrar Edt en la posición correcta.
  Edt.Move MSHFlexGrid.Left + MSHFlexGrid.CellLeft, _
    MSHFlexGrid.Top + MSHFlexGrid.CellTop, _
    MSHFlexGrid.CellWidth - 8, _
    MSHFlexGrid.CellHeight - 8
  Edt.Visible = True

  ' Y hacer que funcione.
  Edt.SetFocus
End Sub

Sub txtEdit_KeyPress(KeyAscii As Integer)
  If tipoalg = 5 Then
  'filtra la entrada de datos que no sean números incluye el signo - y el punto
  If (KeyAscii < 48 Or KeyAscii > 57) Then
      If (KeyAscii = 45 Or KeyAscii = 40 Or KeyAscii = 47 Or KeyAscii = 41 Or
KeyAscii = 80 Or KeyAscii = 73) Then
      Exit Sub
      End If
      If (KeyAscii <> 46) Then
        If (KeyAscii <> 8) Then
        KeyAscii = 0
        End If
```

```
      End If
    End If
   If KeyAscii = Asc(vbCr) Then ' Elimina los retornos para quitar los pitidos.
   KeyAscii = 0
   End If
   End If
   If tipoalg <> 5 Then
  'filtra la entrada de datos que no sean números incluye el signo - y el punto
   If (KeyAscii < 48 Or KeyAscii > 57) Then
      If (KeyAscii = 45) Then
      Exit Sub
      End If
      If (KeyAscii <> 46) Then
         If (KeyAscii <> 8) Then
         KeyAscii = 0
         End If
      End If
    End If
   If KeyAscii = Asc(vbCr) Then ' Elimina los retornos para quitar los pitidos.
   KeyAscii = 0
   End If
Else
    If KeyAscii = Asc(vbCr) Then ' Elimina los retornos para quitar los pitidos.
   KeyAscii = 0
    End If
End If

End Sub

Sub txtEdit_KeyDown(KeyCode As Integer, _
Shift As Integer)
  EditKeyCode Fg2int, TxtEdit, KeyCode, Shift
End Sub

Sub EditKeyCode(MSHFlexGrid As Control, Edt As _
Control, KeyCode As Integer, Shift As Integer)

  ' Procesamiento del control de edición estándar.
  Select Case KeyCode

  Case 27   ' ESC: ocultar, devuelve el enfoque a          ' MSFlexGrid.
    Edt.Visible = False
    MSHFlexGrid.SetFocus

  Case 13   ' ENTRAR devuelve el enfoque a MSHFlexGrid.
    MSHFlexGrid.SetFocus
```

```
  Case 38     ' Arriba.
    MSHFlexGrid.SetFocus
    DoEvents
    If MSHFlexGrid.Row > MSHFlexGrid.FixedRows Then
      MSHFlexGrid.Row = MSHFlexGrid.Row - 1
    End If

  Case 40     ' Abajo.
    MSHFlexGrid.SetFocus
    DoEvents
    If MSHFlexGrid.Row < MSHFlexGrid.Rows - 1 Then
      MSHFlexGrid.Row = MSHFlexGrid.Row + 1
    End If
  End Select
End Sub

Sub Fg2int_GotFocus()
  If TxtEdit.Visible = False Then Exit Sub

  Fg2int = TxtEdit
  TxtEdit.Visible = False
End Sub

Sub Fg2int_LeaveCell()
  If TxtEdit.Visible = False Then Exit Sub

  Fg2int = TxtEdit
  TxtEdit.Visible = False
End Sub



Private Sub Enviar()
Dim i As Integer
Dim j As Integer

On Error GoTo err1

'Capturamos los valores de los texbox y del MSHFlexgrid, para asignarlos
'a las varibles generales del formulario para pasarlas al módulo ClasSEL



  ReDim Puntos(0 To n - 1, 0 To 1) As Double
  ReDim derivadas(0 To 1) As Double
```

```
    For i = 0 To n - 1
        For j = 0 To 1
        Evaluar.fa = Fg2int.TextMatrix(i + 1, j + 1)
        Evaluar.evaluarf
        Puntos(i, j) = Evaluar.Fdy
        'Puntos(i, j) = Val(Fg2int.TextMatrix(i + 1, j + 1))
        Next j
    Next i
    derivadas(0) = Val(TxtIntderivada1.Text)
    derivadas(1) = Val(TxtIntderivada2.Text)
    grado = Val(TxtIntgrado.Text)
    punto = Val(TxtIntpunto.Text)
    'Pasamos los valores al módulo
    Interpolación.Setearint(n, punto, grado, derivadas()) = Puntos()
Exit Sub
err1:
End Sub



Private Sub Intpresentar()
Dim i As Integer
Dim j As Integer
Dim n As Integer
On Error GoTo err1
TxtPoli.Text = " "
CmbPoli.Clear
CmbPoli.Text = "ver polinomio"
n = Val(TxtIntnumpuntos)
If tipoalg > 5 Then
For i = 1 To n - 1
CmbPoli.AddItem i
Next i
Else
CmbPoli.AddItem 1
End If
Select Case (tipoalg)

 Case Is = 1
  'Presentación de los resultados
  fgIntresultados.Cols = 2
  fgIntresultados.Rows = 3
  fgIntresultados.FixedCols = 1
  fgIntresultados.FixedRows = 1
  fgIntresultados.TextArray(Fgr(1, 0)) = "Polinomio="
  fgIntresultados.TextArray(Fgr(2, 0)) = "P(x)="
```

```
  fgIntresultados.TextArray(Fgr(1, 1)) = polinomio
  fgIntresultados.ColWidth(1) = 5000
  'Presentación de las iteraciones

  fgIntiteraciones.Cols = 7
  fgIntiteraciones.Rows = n + 1
  fgIntiteraciones.FixedCols = 0
  fgIntiteraciones.FixedRows = 1

fgIntiteraciones.TextArray(Fgr(0, 0)) = "Xi"
fgIntiteraciones.TextArray(Fgr(0, 1)) = "Yi"
fgIntiteraciones.TextArray(Fgr(0, 2)) = "P(Xi)"
fgIntiteraciones.TextArray(Fgr(0, 3)) = "Ei^2"
fgIntiteraciones.TextArray(Fgr(0, 4)) = "(P(Xi)-Ymedia)^2"
fgIntiteraciones.TextArray(Fgr(0, 5)) = "(Yi-Ymedia)^2"
fgIntiteraciones.TextArray(Fgr(0, 6)) = "r^2"

For i = 0 To 6
fgIntiteraciones.ColWidth(i) = 800
fgIntiteraciones.ColAlignmentFixed(i) = 4
Next i




  For i = 0 To n - 1
   For j = 0 To 6
    fgIntiteraciones.TextMatrix(i + 1, j) = Interpolación.Regresaiteraciones(i, j)
   Next j
Next i
Case Is = 2
  'Presentación de los resultados
  fgIntresultados.Cols = 2
  fgIntresultados.Rows = 3
  fgIntresultados.FixedCols = 1
  fgIntresultados.FixedRows = 1
  fgIntresultados.TextArray(Fgr(1, 0)) = "Polinomio="
  fgIntresultados.TextArray(Fgr(2, 0)) = "P(x)="
  fgIntresultados.TextArray(Fgr(1, 1)) = polinomio
  fgIntresultados.ColWidth(1) = 5000
'Presentación de las iteraciones
  n = Val(TxtIntgrado)
  fgIntiteraciones.Cols = n + 2
  fgIntiteraciones.Rows = n + 2
  fgIntiteraciones.FixedCols = 0
```

```
  fgIntiteraciones.FixedRows = 1
For i = 0 To n
fgIntiteraciones.TextArray(Fgr(0, i)) = "X" & i + 1
Next i
fgIntiteraciones.TextArray(Fgr(0, n + 1)) = "b"


For i = 0 To n + 1
fgIntiteraciones.ColWidth(i) = 800
fgIntiteraciones.ColAlignmentFixed(i) = 4
Next i


  For i = 0 To n
   For j = 0 To n + 1
    fgIntiteraciones.TextMatrix(i + 1, j) = Interpolación.Regresaiteraciones(i, j)
   Next j
Next i

Case Is = 3
  'Presentación de los resultados
  fgIntresultados.Cols = 2
  fgIntresultados.Rows = 2
  fgIntresultados.FixedCols = 1
  fgIntresultados.FixedRows = 1
  fgIntresultados.TextArray(Fgr(1, 0)) = "P(x)="
  fgIntresultados.TextArray(Fgr(1, 1)) = polievaluado
  fgIntresultados.ColWidth(1) = 2500
Case Is = 4
  'Presentación de los resultados
  fgIntresultados.Cols = 2
  fgIntresultados.Rows = 2
  fgIntresultados.FixedCols = 1
  fgIntresultados.FixedRows = 1
  fgIntresultados.TextArray(Fgr(1, 0)) = "P(x)="
  fgIntresultados.TextArray(Fgr(1, 1)) = polievaluado
  fgIntresultados.ColWidth(1) = 2500
Case Is = 5
  'Presentación de los resultados
  fgIntresultados.Cols = 2
  fgIntresultados.Rows = 3
  fgIntresultados.FixedCols = 1
  fgIntresultados.FixedRows = 1
  fgIntresultados.TextArray(Fgr(1, 0)) = "Polinomio="
  fgIntresultados.TextArray(Fgr(2, 0)) = "P(x)="
  fgIntresultados.TextArray(Fgr(1, 1)) = polinomio
```

```
    fgIntresultados.ColWidth(1) = 5000

Case Is = 6
    'Presentación de los resultados
    If n >= 2 Then
    fgIntresultados.Cols = 3
    fgIntresultados.Rows = n
    fgIntresultados.FixedCols = 1
    fgIntresultados.FixedRows = 1
    fgIntresultados.TextArray(Fgr(0, 0)) = "Intervalo"
    fgIntresultados.TextArray(Fgr(0, 1)) = "Polinomio"
    fgIntresultados.TextArray(Fgr(0, 2)) = "P(x)"
    For i = 1 To n - 1
    fgIntresultados.TextArray(Fgr(i, 0)) = "[" & Puntos(i - 1, 0) & "," & Puntos(i, 0) & "]"
    fgIntresultados.TextArray(Fgr(i, 1)) = IntpolinomioSpline(i - 1)
    Next i

     End If

Case Is = 7
    'Presentación de los resultados
    If n >= 2 Then
    fgIntresultados.Cols = 3
    fgIntresultados.Rows = n
    fgIntresultados.FixedCols = 1
    fgIntresultados.FixedRows = 1
    fgIntresultados.TextArray(Fgr(0, 0)) = "Intervalo"
    fgIntresultados.TextArray(Fgr(0, 1)) = "Polinomio"
    fgIntresultados.TextArray(Fgr(0, 2)) = "P(x)"
    For i = 1 To n - 1
    fgIntresultados.TextArray(Fgr(i, 0)) = "[" & Puntos(i - 1, 0) & "," & Puntos(i, 0) & "]"
    fgIntresultados.TextArray(Fgr(i, 1)) = IntpolinomioSpline(i - 1)
    Next i

     End If
Case Is = 8
    'Presentación de los resultados
    If n >= 2 Then
    fgIntresultados.Cols = 3
    fgIntresultados.Rows = n
    fgIntresultados.FixedCols = 1
    fgIntresultados.FixedRows = 1
    fgIntresultados.TextArray(Fgr(0, 0)) = "Intervalo"
    fgIntresultados.TextArray(Fgr(0, 1)) = "Polinomio"
    fgIntresultados.TextArray(Fgr(0, 2)) = "P(x)"
    For i = 1 To n - 1
```

```
    fgIntresultados.TextArray(Fgr(i, 0)) = "[" & Puntos(i - 1, 0) & "," & Puntos(i, 0) & "]"
    fgIntresultados.TextArray(Fgr(i, 1)) = IntpolinomioSpline(i - 1)
    Next i

     End If
  Case Is = 9
    'Presentación de los resultados
    If n >= 2 Then
    fgIntresultados.Cols = 3
    fgIntresultados.Rows = n
    fgIntresultados.FixedCols = 1
    fgIntresultados.FixedRows = 1
    fgIntresultados.TextArray(Fgr(0, 0)) = "Intervalo"
    fgIntresultados.TextArray(Fgr(0, 1)) = "Polinomio"
    fgIntresultados.TextArray(Fgr(0, 2)) = "P(x)"
    For i = 1 To n - 1
    fgIntresultados.TextArray(Fgr(i, 0)) = "[" & Puntos(i - 1, 0) & "," & Puntos(i, 0) & "]"
    fgIntresultados.TextArray(Fgr(i, 1)) = IntpolinomioSpline(i - 1)
    Next i

     End If

  Case Is = 10
    'Presentación de los resultados
    If n >= 2 Then
    fgIntresultados.Cols = 3
    fgIntresultados.Rows = n
    fgIntresultados.FixedCols = 1
    fgIntresultados.FixedRows = 1
    fgIntresultados.TextArray(Fgr(0, 0)) = "Intervalo"
    fgIntresultados.TextArray(Fgr(0, 1)) = "Polinomio"
    fgIntresultados.TextArray(Fgr(0, 2)) = "P(x)"
    For i = 1 To n - 1
    fgIntresultados.TextArray(Fgr(i, 0)) = "[" & Puntos(i - 1, 0) & "," & Puntos(i, 0) & "]"
    fgIntresultados.TextArray(Fgr(i, 1)) = IntpolinomioSpline(i - 1)
    Next i

     End If
  Case Is = 11
    'Presentación de los resultados
    If n >= 2 Then
    fgIntresultados.Cols = 3
    fgIntresultados.Rows = n
    fgIntresultados.FixedCols = 1
    fgIntresultados.FixedRows = 1
    fgIntresultados.TextArray(Fgr(0, 0)) = "Intervalo"
```

```
    fgIntresultados.TextArray(Fgr(0, 1)) = "Polinomio"
    fgIntresultados.TextArray(Fgr(0, 2)) = "P(x)"
    For i = 1 To n - 1
    fgIntresultados.TextArray(Fgr(i, 0)) = "[" & Puntos(i - 1, 0) & "," & Puntos(i, 0) & "]"
    fgIntresultados.TextArray(Fgr(i, 1)) = IntpolinomioSpline(i - 1)
    Next i

     End If
  Case Else
  "Presentación de los resultados
    ' fgSELresultados.Cols = 2
    ' fgSELresultados.Rows = n + 1
    ' fgSELresultados.FixedCols = 1
    ' fgSELresultados.FixedRows = 1
    ' For i = fgSELresultados.FixedRows To fgSELresultados.Rows - 1
     '   fgSELresultados.TextArray(Fgr(i, 0)) = "X" & i
     'Next i

     ' fgSELresultados.TextArray(Fgi(0, 1)) = "Solución"

     ' For i = 0 To n - 1
     ' fgSELresultados.TextMatrix(i + 1, 1) = XR(i)
     ' Next i

     'Presentación de las iteraciones

     'fgSELiteraciones.Cols = n + 2
    ' fgSELiteraciones.Rows = itermax + 1
    ' fgSELiteraciones.FixedCols = 1
    ' fgSELiteraciones.FixedRows = 1
    ' For i = fgSELiteraciones.FixedRows To fgSELiteraciones.Rows - 1
    '    fgSELiteraciones.TextArray(Fgit(i, 0)) = i
    ' Next i
    ' For i = fgSELiteraciones.FixedCols To fgSELiteraciones.Cols - 1
       ' If i = fgSELiteraciones.Cols - 1 Then
        'fgSELiteraciones.TextArray(Fgit(0, i)) = "Error"
       ' Else
       'fgSELiteraciones.TextArray(Fgit(0, i)) = "X" & i
       ' End If
    ' Next i
  ' For i = 0 To itermax - 1
        ' For j = 0 To n
        ' fgSELiteraciones.TextMatrix(i + 1, j + 1) = R(i, j, 0)
        ' Next j
    ' Next i
```

```
End Select
Exit Sub
err1:
End Sub

'Conexion a la BDD
Private Sub PLConexion(parBDD As Boolean)
Dim VTCad As String
Dim VTBase As Database
Dim VTcadena As String
   VTcadena = App.Path & "/Seguridad.mdb"
   Set VLConexion = New ADODB.Connection
   If parBDD = False Then
      VTCad = "Provider=SQLOLEDB.1;Integrated Security=SSPI;Persist Security
Info=False;Initial Catalog=Seguridad"
   Else
      VTCad = "Provider=Microsoft.Jet.OLEDB.4.0;Data Source='" & VTcadena &
"';Persist Security Info=False"
   End If
   VLConexion.Open VTCad
End Sub


Private Sub PLGuardarBDD()
   Dim VTCad As String
   Dim VTReg As ADODB.Recordset
   Dim VTID As Integer
   Dim VB As Integer
On Error GoTo error1
   PLVerificaNombre Trim(VLNombre)
      If VLVerifica = True Then   'Si existe un nombre con el mismo nombre actualiza
         VB = MsgBox("El nombre de ese trabajo ya existe, desea reemplazarlo?",
vbYesNo, "Guardar Trabajo Como")
         If VB = 6 Then
            PLGuardaDatos
            If tipoalg = 1 Then
               VTCad = "update IP set " _
                     & " Nume= '" & Trim(TxtIntgrado.Text) & "', " _
                     & " jacobiano='" & VLJacobiano & "' " _
                     & " where Nombre = '" & VLNombre & "'"
            End If
            VLConexion.Execute (VTCad)
         End If
      Else    'Guarda el trabajo
         VTID = PLRecuperaCodigo
         PLGuardaDatos
```

```
        VTCad = "insert into IP values " _
              & "(" & CInt(VTID) & ", " _
              & " '" & TxtIntnumpuntos.Text & "', '" & TxtIntderivada1.Text & "', " _
              & " '" & TxtIntderivada2.Text & "', '" & TxtIntgrado.Text & "', " _
              & " '" & VLJacobiano & "', " _
              & " '" & Trim(VLNombre) & "', " & 1 & ", " & CInt(tipoalg) & ")"
              '& " '" & Trim(VLNombre) & "', " & CInt(VGIDUsuario) & ", " & CInt(tipoalg
) & ")"
        VLConexion.Execute (VTCad)
        MsgBox "Trabajo guardado exitosamente", vbInformation, "Guardar trabajo
como"
      End If

Exit Sub
error1:
   'If err <> 0 Then
      'MsgBox "Se ha producido el siguiente error: " & err.Description & "." _
            & "Por favor consultar con el administrador", vbCritical
'   End If
End Sub

Private Sub PLVerificaNombre(parNombre As String)
Dim VTReg As ADODB.Recordset
Dim VTCad As String
On Error GoTo error1
   VTCad = "select Nombre from IP where Nombre = '" & parNombre & "' and tipo = "
& tipoalg
      Set VTReg = VLConexion.Execute(VTCad)
      With VTReg
         If Not VTReg.EOF Then
            .MoveFirst
            Do Until .EOF
               If Trim(VTReg(0)) = parNombre Then
                  VLVerifica = True
                  Exit Do
               Else
                  VLVerifica = False
               End If
               .MoveNext
            Loop
         Else
            VLVerifica = False
         End If
      End With
Exit Sub
error1:
```

```
End Sub
Private Function PLRecuperaCodigo() As Integer
   Dim VTCad As String
   Dim VTReg As ADODB.Recordset
On Error GoTo err1
      VTCad = "select max(IDIP) from IP"
      Set VTReg = VLConexion.Execute(VTCad)
      If Not IsNull(VTReg(0)) Then
         PLRecuperaCodigo = VTReg(0) + 1
      Else
         PLRecuperaCodigo = 1
      End If
Exit Function
err1:

End Function


Private Sub PLGuardaDatos()
Dim i, j As Integer
Dim Nume As Integer
On Error GoTo error1
   VLJacobiano = ""
   'Cuando matriz(2)=4 celdas
   If tipoalg = 1 Or tipoalg = 5 Or (tipoalg >= 7) Then
      Nume = CInt(TxtIntnumpuntos.Text)
      For i = 1 To Nume
         For j = 1 To 2
            VLJacobiano = VLJacobiano & Fg2int.TextMatrix(i, j) & ";"
         Next j
      Next i
   ElseIf tipoalg = 2 Or tipoalg = 3 Or tipoalg = 4 Or tipoalg = 6 Then
      Nume = CInt(TxtIntgrado.Text)
      For i = 1 To Nume + 1
         For j = 1 To 2
            VLJacobiano = VLJacobiano & Fg2int.TextMatrix(i, j) & ";"
         Next j
      Next i
   End If
'VLJacobiano
Exit Sub
error1:

End Sub
```

```
Private Sub PLRecupera()
   Dim VTCad As String
   Dim VTReg As ADODB.Recordset
On Error GoTo err1
   '******** APROXIMACIONES SUCESIVAS
      VTCad = "select * from IP where Nombre = '" & VLNombreTrabajo & "'"
      Set VTReg = VLConexion.Execute(VTCad)
      If Not VTReg.EOF Then
         TxtIntnumpuntos.Text = Trim(VTReg(1))
         TxtIntderivada1.Text = Trim(VTReg(2))
         TxtIntderivada2.Text = Trim(VTReg(3))
         TxtIntgrado.Text = Trim(VTReg(4))
         If tipoalg = 1 Or tipoalg = 5 Or (tipoalg >= 7) Then
            PLRecuperaDatos CInt(TxtIntnumpuntos.Text), Trim(VTReg(5))
         ElseIf tipoalg = 2 Or tipoalg = 3 Or tipoalg = 4 Or tipoalg = 6 Then
            PLRecuperaDatos CInt(TxtIntgrado.Text), Trim(VTReg(5))
         Else
         End If
      End If
Exit Sub
err1:
   'If err.Number <> 0 Then
    '  MsgBox "Se ha producido el siguiente error: " & err.Description & "." _
            & "Por favor consultar con el administrador", vbCritical
   'End If

End Sub

Private Sub PLRecuperaDatos(parNume As Integer, parJacobiano As String)
Dim i, j, k As Integer
Dim VTCont As Integer
Dim VTPos As Integer
Dim VTValor As Double
Dim VTLimite As Integer
Dim VTJacob As String
On Error GoTo error1
   VTCont = 1
   VTJacob = parJacobiano
   If tipoalg = 1 Or tipoalg = 5 Or (tipoalg >= 7) Then
      VTLimite = parNume + parNume
      For k = 1 To VTLimite
         VLJacobianoMatriz(k) = 0
      Next k

      For k = 1 To VTLimite
         VTPos = InStr(1, parJacobiano, ";")
```

```
            VTValor = Mid(parJacobiano, 1, VTPos - 1)
            parJacobiano = Mid(parJacobiano, VTPos + 1)
            VLJacobianoMatriz(k) = VTValor
        Next k
        For i = 1 To parNume
            For j = 1 To 2
                Fg2int.TextMatrix(i, j) = VLJacobianoMatriz(VTCont)
                VTCont = VTCont + 1
            Next j
        Next i
    ElseIf tipoalg = 2 Or tipoalg = 3 Or tipoalg = 4 Or tipoalg = 6 Then
        VTLimite = (parNume * parNume) + parNume
        For k = 1 To VTLimite
            VLJacobianoMatriz(k) = 0
        Next k

        For k = 1 To VTLimite
            VTPos = InStr(1, parJacobiano, ";")
            VTValor = Mid(parJacobiano, 1, VTPos - 1)
            parJacobiano = Mid(parJacobiano, VTPos + 1)
            VLJacobianoMatriz(k) = VTValor
        Next k
        For i = 1 To parNume + 1
            For j = 1 To 2
                Fg2int.TextMatrix(i, j) = VLJacobianoMatriz(VTCont)
                VTCont = VTCont + 1
            Next j
        Next i
    End If
Exit Sub
error1:
    'If err.Number <> 0 Then
     '  MsgBox "Se ha producido el siguiente error: " & err.Description & "." _
            & "Por favor consultar con el administrador", vbCritical
    'End If

End Sub

Private Sub PLImprimir()
    PLCambio
        PrintForm
    PLOriginal
End Sub

Private Sub PLCambio()
    Me.BackColor = &HFFFFFF
```

```
    Me.BorderStyle = 0
    CmdIntcalcular.Visible = False
    CmdIntgraficar.Visible = False
End Sub

Private Sub PLOriginal()
    Me.BackColor = &H8000000F
    Me.BorderStyle = 2
    CmdIntcalcular.Visible = True
    CmdIntgraficar.Visible = True

End Sub



Private Sub TxtIntnumpuntos_KeyPress(KeyAscii As Integer)
'filtra la entrada de datos que no sean números enteros y positivos
  If (KeyAscii < 48 Or KeyAscii > 57) Then

          If (KeyAscii <> 8) Then
          KeyAscii = 0
          End If

    End If
  If KeyAscii = Asc(vbCr) Then ' Elimina los retornos para quitar los pitidos.
  KeyAscii = 0
  End If

End Sub

Private Sub TxtIntpunto_Change()
LTrim (TxtIntpunto)
End Sub

Private Sub TxtIntpunto_KeyPress(KeyAscii As Integer)
'filtra la entrada de datos que no sean números incluye el signo - y el punto
  If (KeyAscii < 48 Or KeyAscii > 57) Then
      If (KeyAscii = 45) Then
      Exit Sub
      End If
      If (KeyAscii <> 46) Then
         If (KeyAscii <> 8) Then
         KeyAscii = 0
         End If
      End If
   End If
```

```
  If KeyAscii = Asc(vbCr) Then ' Elimina los retornos para quitar los pitidos.
  KeyAscii = 0
  End If

End Sub

Private Sub Sincolor()
With Me
arribaFraimprimir = .FraImprimir.Top
arribaFraResul = .FraResul.Top
anchoFraResul = .FraResul.Width
altoFraResul = .FraResul.Height
arribaSstab = .SSTab1.Top
izquierdaSstab = .SSTab1.Left
anchoSstab = .SSTab1.Width
altoSstab = .SSTab1.Height
'arribaFgresul = .fgIntresultados.Top
anchoFgresul = .fgIntresultados.Width
altoFgresul = .fgIntresultados.Height
arribaFgitera = .fgIntiteraciones.Top
izquierdaFgitera = .fgIntiteraciones.Left
anchoFgitera = .fgIntiteraciones.Width
altoFgitera = .fgIntiteraciones.Height
anchocolumna = .fgIntresultados.ColWidth(1)
colororg = .BackColor
VLabel = .Label20.Visible
Vcmd = .CmdIntevaluar.Visible
.FraImprimir.Top = 0
.SSTab1.Top = .SSTab1.Top + 1000
.SSTab1.Left = .SSTab1.Left + 1000
.SSTab1.Width = .SSTab1.Width / 4
.SSTab1.Height = .SSTab1.Height / 4
.FraResul.Top = .FraImprimir.Top + .FraImprimir.Height
.FraResul.Width = .FraResul.Width + 7000
.FraResul.Height = .FraResul.Height + 50
.fgIntresultados.Height = .fgIntresultados.Height + 50
.fgIntresultados.Width = .fgIntresultados.Width + 7000
.fgIntiteraciones.Top = .FraResul.Top + .FraResul.Height
.fgIntiteraciones.Height = .fgIntiteraciones.Height + 50
.fgIntiteraciones.Width = .FraResul.Width
.fgIntiteraciones.Left = .FraImprimir.Left
.BackColor = &H80000009
.FraImprimir.BackColor = &H80000009
.FraPara.BackColor = &H80000009
.FraIntevaluar.BackColor = &H80000009
.CmdIntevaluar.Visible = False
```

```
.LblIntderivada1.BackColor = &H80000009
.LblIntderivada2.BackColor = &H80000009
.LblIntgrado.BackColor = &H80000009
.LblIntnumpuntos.BackColor = &H80000009
.LblInttitulo.Visible = False
.TxtEdit.BorderStyle = 0
.TxtIntderivada1.BorderStyle = 0
.TxtIntderivada2.BorderStyle = 0
.TxtIntgrado.BorderStyle = 0
.TxtIntnumpuntos.BorderStyle = 0
.TxtIntpunto.BorderStyle = 0
.TxtPoli.Visible = False
.FraResul.BackColor = &H80000009
.Fg2int.BackColorBkg = &H80000009
.Fg2int.BackColorFixed = &H80000009
.fgIntresultados.BackColorBkg = &H80000009
.fgIntresultados.BackColorFixed = &H80000009
.fgIntiteraciones.BackColorBkg = &H80000009
.fgIntiteraciones.BackColorFixed = &H80000009
.Label20.Visible = False
.CmdIntcalcular.Visible = False
.CmdIntiteraciones.Visible = False
.CmbPoli.Visible = False
.FraPara.BorderStyle = 1
.Fg2int.ScrollBars = 0
.fgIntiteraciones.ScrollBars = 0
.fgIntresultados.ScrollBars = 0
.ShapeInt.Visible = False
.LblNomalg.BackColor = &H80000009
.LblIntpunto.BackColor = &H80000009
.fgIntresultados.ColWidth(1) = .fgIntresultados.ColWidth(1) + 5000
FraIntSintaxis.Visible = False
End With


End Sub

Private Sub Concolor()
With Me
 .FraImprimir.Top = arribaFraimprimir
 .FraResul.Top = arribaFraResul
 .FraResul.Width = anchoFraResul
 .FraResul.Height = altoFraResul
 .SSTab1.Top = arribaSstab
 .SSTab1.Left = izquierdaSstab
 .SSTab1.Width = anchoSstab
```

```
 .SSTab1.Height = altoSstab
 '.fgIntresultados.Top = arribaFgresul
 .fgIntresultados.Width = anchoFgresul
 .fgIntresultados.Height = altoFgresul
 .fgIntiteraciones.Top = arribaFgitera
 .fgIntiteraciones.Left = izquierdaFgitera
 .fgIntiteraciones.Width = anchoFgitera
 .fgIntiteraciones.Height = altoFgitera
 .BackColor = colororg
.FraImprimir.BackColor = colororg
.FraPara.BackColor = colororg
.FraIntevaluar.BackColor = colororg
If Vcmd = True Then
.CmdIntevaluar.Visible = True
End If
.LblIntderivada1.BackColor = colororg
.LblIntderivada2.BackColor = colororg
.LblIntgrado.BackColor = colororg
.LblIntnumpuntos.BackColor = colororg
.LblInttitulo.Visible = True
.TxtEdit.BorderStyle = 1
.TxtIntderivada1.BorderStyle = 1
.TxtIntderivada2.BorderStyle = 1
.TxtIntgrado.BorderStyle = 1
.TxtIntnumpuntos.BorderStyle = 1
.TxtIntpunto.BorderStyle = 1
.TxtPoli.Visible = True
.FraResul.BackColor = colororg
.Fg2int.BackColorBkg = colororg
.Fg2int.BackColorFixed = colororg
.fgIntresultados.BackColorBkg = colororg
.fgIntresultados.BackColorFixed = colororg
.fgIntiteraciones.BackColorBkg = colororg
.fgIntiteraciones.BackColorFixed = colororg
If VLabel = True Then
.Label20.Visible = False
End If
.CmdIntcalcular.Visible = True
.CmdIntiteraciones.Visible = True
.CmbPoli.Visible = True
.FraPara.BorderStyle = 0
.Fg2int.ScrollBars = 3
.fgIntiteraciones.ScrollBars = 3
.fgIntresultados.ScrollBars = 3
.ShapeInt.Visible = True
.LblNomalg.BackColor = colororg
```

```
.LblIntpunto.BackColor = colororg
.fgIntresultados.ColWidth(1) = anchocolumna
FraIntSintaxis.Visible = True
End With

End Sub
```

Módulo de clase ClasInterpolación

```
Option Explicit
Dim eval As New McsExcel
Dim SEL As New ClasSEL
Dim A() As Double
Dim b() As Double
Dim f() As Double
Dim h() As Double
Dim d() As Double
Dim U() As Double
Dim s() As Double
Dim m() As Double
Dim Resulint() As Double
Dim derivadasS() As Double
Dim Ns As Integer
Dim polinomioSpline() As String
Dim ordendd  As Integer
Dim err As Boolean
Dim tol As Double
Dim itermax As Integer
Dim numecua As Integer
Dim mensaje As String
Dim numpares As Integer
Dim Pares() As Double
Dim gradopoli As Integer
Dim polinomio As String
Dim coefpoli() As Double
Dim ptoaevaluar As Double
Dim ptoevaluado As Double


Public Property Let Setearint(Np As Integer, punto As Double, g As Integer, Deri() As
Double, p() As Double)
On Error GoTo err1
numpares = Np
Pares = p
gradopoli = g
derivadasS = Deri
ptoaevaluar = punto
```

```
Exit Property
err1:
End Property

Public Property Get intptoevaluado() As Variant
On Error GoTo err1
intptoevaluado = ptoevaluado
Exit Property
err1:
End Property
Public Property Get intPolinomio() As Variant
On Error GoTo err1
intPolinomio = polinomio
Exit Property
err1:
End Property
Public Property Get IntpolinomioSpline() As Variant
On Error GoTo err1
IntpolinomioSpline = polinomioSpline
Exit Property
err1:
End Property
Public Property Get Returndiferenciasdivididas() As Variant
'Returndifereniasdivididas = f
End Property

Public Property Let Setearparesdatos(grado As Integer, numdatos As Integer,
Puntos() As Double)
On Error GoTo err1
ordendd = grado
Pares = Puntos
numpares = numdatos
Exit Property
err1:
End Property
Public Property Get Regresaiteraciones() As Variant
On Error GoTo err1
Regresaiteraciones = Resulint
Exit Property
err1:
End Property


Public Sub RegresiónPolinomial()
Dim i As Integer
Dim j As Integer
```

```
Dim k As Integer
Dim L As Integer
Dim sum As Double
Dim tmp As Double
Dim Ypro As Double
Dim Pi As Double
Dim Ei As Double
Dim PXY As Double
Dim YiY As Double
Dim r As Double
On Error GoTo err1
'Comprobamos que el número de pares de datos sea mayor que el grado
'del polinomio a hallar
If numpares < gradopoli Then
mensaje = MsgBox("El grado del polinomio debe ser menor o igual al número de
pares de datos", 48, "Error en el ingreso")
Exit Sub
Else
'Rediminsionamos las matrices a y b para armar el sistema
'de ecuaciones a resolver
ReDim A(0 To gradopoli, 0 To gradopoli) As Double
ReDim b(0 To gradopoli) As Double
ReDim Resulint(0 To numpares - 1, 0 To 6)
'Implementamos el algoritmo
For i = 1 To gradopoli + 1
   For j = 1 To i
   k = i + j - 2
   sum = 0
      For L = 0 To numpares - 1
      sum = sum + (Pares(L, 0)) ^ k
      Next L
   A(i - 1, j - 1) = sum
   A(j - 1, i - 1) = sum
   Next j
   sum = 0
   For L = 0 To numpares - 1
   tmp = Pares(L, 1) * (Pares(L, 0)) ^ (i - 1)
   sum = sum + tmp
   Next L
b(i - 1) = sum
Next i
err = False
tol = 0.000001
itermax = 500
numecua = gradopoli + 1
'Pasamos los valores al módulo ClasSEI para resolver el sistema de ecuaciones
```

```
SEL.setearSEL(numecua, tol, itermax, err, A()) = b()
SEL.EliGaussPivotación
ReDim coefpoli(0 To numecua - 1)
For i = 0 To numecua - 1
coefpoli(i) = SEL.ReturSELX(i)
Next i
'Armamos el polinomio para la presentación al usuario
polinomio = coefpoli(0)
If coefpoli(1) < 0 Then
polinomio = polinomio & coefpoli(1) & "*" & "X"
Else
polinomio = polinomio & "+" & coefpoli(1) & "*" & "X"
End If
If gradopoli >= 2 Then
i = 2
Do
   If coefpoli(i) < 0 Then
   polinomio = polinomio & coefpoli(i) & "*" & "X^" & i
   Else
   polinomio = polinomio & "+" & coefpoli(i) & "*" & "X^" & i
   End If
i = i + 1
Loop Until i > gradopoli
End If
End If
'Formamos la tabla de ajuste
'Y promedio
sum = 0
For i = 0 To numpares - 1
sum = sum + Pares(i, 1)
Next i
Ypro = sum / numpares


For i = 0 To numpares - 1
Resulint(i, 0) = Pares(i, 0)
Resulint(i, 1) = Pares(i, 1)
Resulint(i, 2) = Peval(polinomio, Pares(i, 0))
Resulint(i, 3) = (Resulint(i, 1) - Resulint(i, 2)) ^ 2
Resulint(i, 4) = (Resulint(i, 2) - Ypro) ^ 2
Resulint(i, 5) = (Resulint(i, 1) - Ypro) ^ 2
Next i
sum = 0
For i = 0 To numpares - 1
sum = sum + Resulint(i, 4)
Next i
```

```
PXY = sum
sum = 0
For i = 0 To numpares - 1
sum = sum + Resulint(i, 5)
Next i
YiY = sum
Resulint(0, 6) = PXY / YiY
Exit Sub
err1:
'If err.Number <> 0 Then
   'mensaje = MsgBox("Parámetros del algoritmo incompletos", 48, "Error en el
ingreso")
   'End If



End Sub



Public Sub TécnicaMatricial()
Dim i As Integer
Dim j As Integer
On Error GoTo err1
'Comprobamos que el número de pares de datos sea mayor que el grado
'del polinomio a hallar
If numpares < gradopoli Then
mensaje = MsgBox("El grado del polinomio debe ser menor o igual al número de
pares de datos", 48, "Error en el ingreso")
Exit Sub
Else
'Rediminsionamos las matrices a y b para armar el sistema
'de ecuaciones a resolver
ReDim A(0 To gradopoli, 0 To gradopoli) As Double
ReDim b(0 To gradopoli) As Double
ReDim Resulint(0 To gradopoli, 0 To gradopoli + 1) As Double
'Implementamos el algoritmo
For j = 0 To gradopoli
   For i = 0 To gradopoli
   If j = 0 Then
   A(i, j) = 1
   Else

   A(i, j) = (Pares(i, 0)) ^ j
   End If
   Next i
```

```
Next j
For i = 0 To gradopoli
b(i) = Pares(i, 1)
Next i

'Guardamos para mostrar el Sistema

For j = 0 To gradopoli
   For i = 0 To gradopoli
   Resulint(i, j) = A(i, j)
   Next i
Next j
For i = 0 To gradopoli
Resulint(i, gradopoli + 1) = b(i)
Next i



err = False
tol = 0.000001
itermax = 500
numecua = gradopoli + 1
'Pasamos los valores al módulo ClasSEI para resolver el sistema de ecuaciones
SEL.setearSEL(numecua, tol, itermax, err, A()) = b()
SEL.EliGaussPivotación
ReDim coefpoli(0 To numecua - 1)
For i = 0 To numecua - 1
coefpoli(i) = SEL.ReturSELX(i)
Next i
'Armamos el polinomio para la presentación al usuario
polinomio = coefpoli(0)
If coefpoli(1) < 0 Then
polinomio = polinomio & coefpoli(1) & "*" & "X"
Else
polinomio = polinomio & "+" & coefpoli(1) & "*" & "X"
End If
If gradopoli >= 2 Then
i = 2
Do
   If coefpoli(i) < 0 Then
   polinomio = polinomio & coefpoli(i) & "*" & "X^" & i
   Else
   polinomio = polinomio & "+" & coefpoli(i) & "*" & "X ^" & i
   End If
i = i + 1
```

```
Loop Until i > gradopoli
End If
End If
Exit Sub
err1:
'If err.Number <> 0 Then
   mensaje = MsgBox("Parámetros del algoritmo incompletos", 48, "Error en el
ingreso")
   'End If

End Sub

Public Sub Lagrange()
Dim i As Integer
Dim k As Integer
Dim sum As Double
Dim Lparcial As Double
Dim Lparcial2 As Double
Dim L As Double
Dim mensaje As String

On Error GoTo err1
'Comprobamos que el número de pares de datos sea mayor que el grado
'del polinomio a hallar
If numpares < gradopoli Then
mensaje = MsgBox("El grado del polinomio debe ser menor o igual al número de
pares de datos", 48, "Error en el ingreso")
Exit Sub
Else
sum = 0
For k = 0 To gradopoli
   L = 1
   For i = 0 To gradopoli
   If i <> k Then
   Lparcial = ptoaevaluar - Pares(i, 0)
   Lparcial2 = Pares(k, 0) - Pares(i, 0)
      If Lparcial2 = 0 Then
      mensaje = MsgBox("No se puede obtener el polinomio", 16, "Fracaso")
      Exit Sub
      End If
   L = L * (Lparcial / Lparcial2)
   End If
   Next i
   sum = sum + (L * Pares(k, 1))
Next k
ptoevaluado = sum
```

```
End If
Exit Sub
err1:
'If err.Number <> 0 Then
   mensaje = MsgBox("Parámetros del algoritmo incompletos", 48, "Error en el
ingreso")
 '  End If

End Sub

Public Sub Newton()
Dim i As Integer
Dim j As Integer
Dim producto As Double
Dim producto2 As Double
Dim productof As Double
Dim sum As Double
Dim mensaje As String
On Error GoTo err1
If gradopoli >= numpares Then
mensaje = MsgBox("El grado del polinomio debe ser menor al número de pares de
datos", 16, "Fracaso")
Exit Sub
Else
ordendd = gradopoli
diferenciasdivididas
sum = Pares(0, 1)
For j = 1 To gradopoli
producto = 1
   For i = 0 To j - 1
   producto2 = ptoaevaluar - Pares(i, 0)
   producto = producto * producto2
   Next i
productof = f(0, j) * producto
sum = sum + productof
Next j
ptoevaluado = sum
End If
Exit Sub
err1:
'If err.Number <> 0 Then
   mensaje = MsgBox("Parámetros del algoritmo incompletos", 48, "Error en el
ingreso")
 '  End If
```

```
End Sub

Public Sub Trigonométrica()
Dim i As Integer
Dim j As Integer
Dim sum As Double
Dim parcial As Double
Dim mensaje As String
On Error GoTo err1
sum = (numpares - 1) / 2
If gradopoli < sum Then
ReDim A(0 To gradopoli)
ReDim b(1 To gradopoli)

' Calculamos el coeficiente ao del polinómio trigonométrico
sum = 0
For i = 0 To numpares - 1
sum = sum + Pares(i, 1)
Next i
A(0) = (sum * 2) / (numpares)
' Calculamos los demás coeficientes del polinómio trigonométrico
For j = 1 To gradopoli
   sum = 0
   For i = 0 To numpares - 1
   parcial = j * Pares(i, 0)
   sum = sum + (Pares(i, 1) * Cos(parcial))
   Next i
A(j) = (2 / (numpares - 1)) * sum
 sum = 0
   For i = 0 To numpares - 1
   parcial = j * Pares(i, 0)
   sum = sum + (Pares(i, 1) * Sin(parcial))
   Next i
b(j) = (2 / (numpares)) * sum
Next j
'Reducimos a cero los coeficientes que sean menores que 1E-15
If A(0) < 0.000000001 Then
A(0) = 0
End If

For j = 1 To gradopoli
If Abs(A(j)) < 0.000000001 Then
A(j) = 0
End If
If Abs(b(j)) < 0.000000001 Then
b(j) = 0
```

```
End If
Next j

'Construimos el polinomio trigonométrico para su presentación al usuario
polinomio = A(0) / 2

For j = 1 To gradopoli
If A(j) < 0 Then
polinomio = polinomio & A(j) & "*" & "COS(" & j & "*X)"
Else
If A(j) > 0 Then
polinomio = polinomio & "+" & A(j) & "*" & "COS(" & j & "*X)"
End If
End If
If b(j) < 0 Then
polinomio = polinomio & b(j) & "*" & "SIN(" & j & "X)"
Else
If b(j) > 0 Then
polinomio = polinomio & "+" & b(j) & "*" & "SIN(" & j & "*X)"
End If
End If
Next j
Else
mensaje = MsgBox("El grado del polinomio trigonométrico debe ser menor que la
mitad del número de pares ingresados", 16, "Error en el ingreso")
End If
Exit Sub
err1:
'If err.Number <> 0 Then
   mensaje = MsgBox("Parámetros del algoritmo incompletos", 48, "Error en el
ingreso")
'  End If

End Sub

Public Sub SplineLineal()
Dim i  As Integer
On Error GoTo err1
Ns = numpares - 1
Calculohdu
ReDim polinomioSpline(0 To Ns - 1) As String
'Armamos el polinómio para la presentación al usuario
For i = 0 To Ns - 1
   If d(i) > 0 Then
      If Pares(i, 0) > 0 Then
      polinomioSpline(i) = Pares(i, 1) & "+" & d(i) & "*(X-" & Pares(i, 0) & ")"
```

```
      Else
         If Pares(i, 0) = 0 Then
         polinomioSpline(i) = d(i) & "*(X)"
         Else
         polinomioSpline(i) = Pares(i, 1) & "+" & d(i) & "*(X+" & Abs(Pares(i, 0)) & ")"
         End If
      End If
   Else
      If d(i) = 0 Then
      polinomioSpline(i) = Pares(i, 1)
      End If
      If Pares(i, 0) > 0 Then
      polinomioSpline(i) = Pares(i, 1) & d(i) & "*(X-" & Pares(i, 0) & ")"
      Else
         If Pares(i, 0) = 0 Then
         polinomioSpline(i) = d(i) & "*(X)"
         Else
         polinomioSpline(i) = Pares(i, 1) & d(i) & "*(X+" & Abs(Pares(i, 0)) & ")"
         End If
      End If
   End If
 Next i
Exit Sub
err1:
'If err.Number <> 0 Then
   mensaje = MsgBox("Parámetros del algoritmo incompletos", 48, "Error en el
ingreso")
 '  End If

End Sub
Public Sub SplineSujeta()
Dim i As Integer
Dim j As Integer
Dim n As Integer
Dim Atmp() As Double
Dim Btmp() As Double
Dim mensaje As String
On Error GoTo err1
Ns = numpares - 1
n = Ns
Select Case Ns
Case Is = 0
   mensaje = MsgBox(" Debe ingresar al menos dos puntos para obtener el
polinómio", 16, "Error en el ingreso")
   Exit Sub
Case Is = 1
```

```
    Calculohdu
    ReDim A(0 To 1, 0 To 1) As Double
    ReDim b(0 To 1) As Double
    A(0, 0) = 1
    A(0, 1) = 0.5
    A(1, 0) = 0.5
    A(1, 1) = 1
    b(0) = (3 / h(0)) * (d(0) - derivadasS(0))
    b(1) = (3 / h(0)) * (derivadasS(1) - d(0))

    err = False
    tol = 0.000001
    itermax = 500
    numecua = 2
    'Pasamos los valores al módulo ClasSEl para resolver el sistema de ecuaciones
    SEL.setearSEL(numecua, tol, itermax, err, A()) = b()
    SEL.EliGaussPivotación
    ReDim m(0 To 1)
    For i = 0 To 1
    m(i) = SEL.ReturSELX(i)
    Next i
    FormadorpolinomioSpline
Case Is = 2
    Calculohdu
    ReDim A(0 To 2, 0 To 2) As Double
    ReDim b(0 To 2) As Double
    A(0, 0) = 1
    A(0, 1) = 0.5
    A(0, 2) = 0
    A(1, 0) = 0
    A(1, 1) = 0.5
    A(1, 2) = 1
    A(2, 0) = 0
    A(2, 1) = ((3 / 2) * h(0)) + (2 * h(1))
    A(2, 2) = h(1)
    b(0) = (3 / h(0)) * (d(0) - derivadasS(0))
    b(1) = (3 / h(1)) * (derivadasS(1) - d(1))
    b(2) = U(1) - (3 * (d(0) - derivadasS(0)))
    err = False
    tol = 0.000001
    itermax = 500
    numecua = 3
    'Pasamos los valores al módulo ClasSEl para resolver el sistema de ecuaciones
    SEL.setearSEL(numecua, tol, itermax, err, A()) = b()
    SEL.EliGaussPivotación
    ReDim m(0 To 2)
```

```
    For i = 0 To 2
    m(i) = SEL.ReturSELX(i)
    Next i
    FormadorpolinomioSpline
Case Else
    Calculohdu
    ReDim Atmp(1 To n - 1, 1 To n - 1) As Double
    ReDim Btmp(1 To n - 1) As Double

    Atmp(1, 1) = ((3 / 2) * h(0)) + (2 * h(1))
    Atmp(1, 2) = h(1)
    Btmp(1) = U(1) - (3 * (d(0) - derivadasS(0)))

    For i = 2 To n - 2
    Atmp(i, i - 1) = h(i - 1)
    Atmp(i, i) = 2 * (h(i - 1) + h(i))
    Atmp(i, i + 1) = h(i)
    Btmp(i) = U(i)
    Next i
    Atmp(n - 1, n - 2) = h(n - 2)
    Atmp(n - 1, n - 1) = (2 * h(n - 2)) + ((3 / 2) * h(n - 1))
    Btmp(n - 1) = U(n - 1) - (3 * (derivadasS(1) - d(n - 1)))
    ReDim A(0 To n - 2, 0 To n - 2) As Double
    ReDim b(0 To n - 2) As Double
    For i = 1 To n - 1
        For j = 1 To n - 1
        A(i - 1, j - 1) = Atmp(i, j)
        Next j
    Next i
    For i = 1 To n - 1
    b(i - 1) = Btmp(i)
    Next i
    err = False
    tol = 0.000001
    itermax = 500
    numecua = n - 1
    'Pasamos los valores al módulo ClasSEI para resolver el sistema de ecuaciones
    SEL.setearSEL(numecua, tol, itermax, err, A()) = b()
    SEL.EliGaussPivotación
    ReDim m(0 To n)
    For i = 0 To n - 2
    m(i + 1) = SEL.ReturSELX(i)
    Next i
    m(0) = ((3 / h(0)) * (d(0) - derivadasS(0))) - (m(1) / 2)
    m(n) = ((3 / h(n - 1)) * (derivadasS(1) - d(n - 1))) - (m(n - 1) / 2)
```

```
    FormadorpolinomioSpline
 End Select
Exit Sub
err1:
'If err.Number <> 0 Then
    mensaje = MsgBox("Parámetros del algoritmo incompletos", 48, "Error en el
ingreso")
 '   End If

End Sub

Public Sub SplineNatural()
Dim i As Integer
Dim j As Integer
Dim n As Integer
Dim Atmp() As Double
Dim Btmp() As Double
Dim mensaje As String
On Error GoTo err1
Ns = numpares - 1
n = Ns
Select Case Ns
Case Is = 0
    mensaje = MsgBox(" Debe ingresar al menos dos puntos para obtener el
polinómio", 16, "Error en el ingreso")
    Exit Sub
Case Is = 1
    Calculohdu
    ReDim m(0 To 1)
    m(0) = 0
    m(1) = 0
    FormadorpolinomioSpline
Case Is = 2
    Calculohdu

    ReDim m(0 To 2)
    m(0) = 0
    m(1) = 0
    m(2) = U(1) / h(1)
    FormadorpolinomioSpline
Case Else
    Calculohdu
    ReDim Atmp(1 To n - 1, 1 To n - 1) As Double
    ReDim Btmp(1 To n - 1) As Double

    Atmp(1, 1) = 2 * (h(0) + h(1))
```

```
    Atmp(1, 2) = h(1)
    Btmp(1) = U(1)

    For i = 2 To n - 2
    Atmp(i, i - 1) = h(i - 1)
    Atmp(i, i) = 2 * (h(i - 1) + h(i))
    Atmp(i, i + 1) = h(i)
    Btmp(i) = U(i)
    Next i
    Atmp(n - 1, n - 2) = h(n - 2)
    Atmp(n - 1, n - 1) = 2 * (h(n - 2) + h(n - 1))
    Btmp(n - 1) = U(n - 1)
    ReDim A(0 To n - 2, 0 To n - 2) As Double
    ReDim b(0 To n - 2) As Double
    For i = 1 To n - 1
       For j = 1 To n - 1
       A(i - 1, j - 1) = Atmp(i, j)
       Next j
    Next i
    For i = 1 To n - 1
    b(i - 1) = Btmp(i)
    Next i
    err = False
    tol = 0.000001
    itermax = 500
    numecua = n - 1
    'Pasamos los valores al módulo ClasSEI para resolver el sistema de ecuaciones
    SEL.setearSEL(numecua, tol, itermax, err, A()) = b()
    SEL.EliGaussPivotación
    ReDim m(0 To n)
    For i = 0 To n - 2
    m(i + 1) = SEL.ReturSELX(i)
    Next i
    m(0) = 0
    m(n) = 0

    FormadorpolinomioSpline
End Select
Exit Sub
err1:
'If err.Number <> 0 Then
    mensaje = MsgBox("Parámetros del algoritmo incompletos", 48, "Error en el
ingreso")
    'End If

End Sub
```

```
Public Sub SplineExtrapolada()
Dim i As Integer
Dim j As Integer
Dim n As Integer
Dim Atmp() As Double
Dim Btmp() As Double
Dim mensaje As String
On Error GoTo err1
Ns = numpares - 1
n = Ns
Select Case Ns
Case Is = 0
   mensaje = MsgBox(" Debe ingresar al menos dos puntos para obtener el
polinómio", 16, "Error en el ingreso")
   Exit Sub
Case Is = 1
   mensaje = MsgBox(" Debe ingresar al menos dos puntos para obtener el
polinómio", 16, "Error en el ingreso")
   Exit Sub
Case Is = 2
   Calculohdu
   ReDim A(0 To 2, 0 To 2) As Double
   ReDim b(0 To 2) As Double
   A(0, 1) = (3 * h(0)) + (2 * h(1)) + ((h(0) ^ 2) / h(1))
   A(0, 2) = h(1) - ((h(0) ^ 2) / h(1))
   A(0, 0) = 0
   A(1, 0) = 1
   A(1, 1) = -1 - (h(0) / h(1))
   A(1, 2) = h(0) / h(1)
   A(2, 0) = h(1) / h(0)
   A(2, 1) = -1 - (h(1) / h(0))
   A(2, 2) = 1
   b(0) = U(1)
   b(1) = 0
   b(2) = 0
   err = False
   tol = 0.000001
   itermax = 500
   numecua = 3
   'Pasamos los valores al módulo ClasSEI para resolver el sistema de ecuaciones
   SEL.setearSEL(numecua, tol, itermax, err, A()) = b()
   SEL.EliGaussPivotación
   ReDim m(0 To 2)
   For i = 0 To 2
   m(i) = SEL.ReturSELX(i)
```

```
    Next i
    FormadorpolinomioSpline
  Case Else
    Calculohdu
    ReDim Atmp(1 To n - 1, 1 To n - 1) As Double
    ReDim Btmp(1 To n - 1) As Double

    Atmp(1, 1) = 3 * h(0) + 2 * h(1) + ((h(0) ^ 2) / h(1))
    Atmp(1, 2) = h(1) - ((h(0) ^ 2) / h(1))
    Btmp(1) = U(1)

    For i = 2 To n - 2
    Atmp(i, i - 1) = h(i - 1)
    Atmp(i, i) = 2 * (h(i - 1) + h(i))
    Atmp(i, i + 1) = h(i)
    Btmp(i) = U(i)
    Next i
    Atmp(n - 1, n - 2) = h(n - 2) - (((h(n - 1)) ^ 2) / h(n - 2))
    Atmp(n - 1, n - 1) = 2 * h(n - 2) + 3 * h(n - 1) + ((h(n - 1)) ^ 2 / h(n - 2))
    Btmp(n - 1) = U(n - 1)
    ReDim A(0 To n - 2, 0 To n - 2) As Double
    ReDim b(0 To n - 2) As Double
    For i = 1 To n - 1
       For j = 1 To n - 1
       A(i - 1, j - 1) = Atmp(i, j)
       Next j
    Next i
    For i = 1 To n - 1
    b(i - 1) = Btmp(i)
    Next i
    err = False
    tol = 0.000001
    itermax = 500
    numecua = n - 1
    'Pasamos los valores al módulo ClasSEl para resolver el sistema de ecuaciones
    SEL.setearSEL(numecua, tol, itermax, err, A()) = b()
    SEL.EliGaussPivotación
    ReDim m(0 To n)
    For i = 0 To n - 2
    m(i + 1) = SEL.ReturSELX(i)
    Next i
    m(0) = m(1) - ((h(0) * (m(2) - m(1))) / h(1))
    m(n) = m(n - 1) + ((h(n - 1) * (m(n - 1) - m(n - 2))) / h(n - 2))

   FormadorpolinomioSpline
End Select
```

```
Exit Sub
err1:
'If err.Number <> 0 Then
   mensaje = MsgBox("Parámetros del algoritmo incompletos", 48, "Error en el
ingreso")
 '  End If

End Sub

Public Sub SplineTerminaciónparabolica()
Dim i As Integer
Dim j As Integer
Dim n As Integer
Dim Atmp() As Double
Dim Btmp() As Double
Dim mensaje As String
On Error GoTo err1
Ns = numpares - 1
n = Ns
Select Case Ns
Case Is = 0
   mensaje = MsgBox(" Debe ingresar al menos dos puntos para obtener el
polinómio", 16, "Error en el ingreso")
   Exit Sub
Case Is = 1
   mensaje = MsgBox(" Debe ingresar al menos dos puntos para obtener el
polinómio", 16, "Error en el ingreso")
   Exit Sub
Case Is = 2
   Calculohdu
   ReDim m(0 To 2)
   m(1) = U(1) / (3 * (h(0) + h(1)))
   m(0) = m(1)
   m(2) = m(1)
   FormadorpolinomioSpline
Case Else
   Calculohdu
   ReDim Atmp(1 To n - 1, 1 To n - 1) As Double
   ReDim Btmp(1 To n - 1) As Double

   Atmp(1, 1) = 3 * h(0) + 2 * h(1)
   Atmp(1, 2) = h(1)
   Btmp(1) = U(1)

   For i = 2 To n - 2
   Atmp(i, i - 1) = h(i - 1)
```

```
    Atmp(i, i) = 2 * (h(i - 1) + h(i))
    Atmp(i, i + 1) = h(i)
    Btmp(i) = U(i)
    Next i
    Atmp(n - 1, n - 2) = h(n - 2)
    Atmp(n - 1, n - 1) = 2 * h(n - 2) + 3 * h(n - 1)
    Btmp(n - 1) = U(n - 1)
    ReDim A(0 To n - 2, 0 To n - 2) As Double
    ReDim b(0 To n - 2) As Double
    For i = 1 To n - 1
       For j = 1 To n - 1
       A(i - 1, j - 1) = Atmp(i, j)
       Next j
    Next i
    For i = 1 To n - 1
    b(i - 1) = Btmp(i)
    Next i
    err = False
    tol = 0.000001
    itermax = 500
    numecua = n - 1
    'Pasamos los valores al módulo ClasSEI para resolver el sistema de ecuaciones
    SEL.setearSEL(numecua, tol, itermax, err, A()) = b()
    SEL.EliGaussPivotación
    ReDim m(0 To n)
    For i = 0 To n - 2
    m(i + 1) = SEL.ReturSELX(i)
    Next i
    m(0) = m(1)
    m(n) = m(n - 1)

    FormadorpolinomioSpline
End Select
Exit Sub
err1:
'If err.Number <> 0 Then
    mensaje = MsgBox("Parámetros del algoritmo incompletos", 48, "Error en el
ingreso")
 '  End If

End Sub

Public Sub SplineCurvaturaenlosextremos()
Dim i As Integer
Dim j As Integer
Dim n As Integer
```

```
Dim Atmp() As Double
Dim Btmp() As Double
Dim mensaje As String
On Error GoTo err1
Ns = numpares - 1
n = Ns
Select Case Ns
Case Is = 0
   mensaje = MsgBox(" Debe ingresar al menos dos puntos para obtener el
polinómio", 16, "Error en el ingreso")
   Exit Sub
Case Is = 1
   Calculohdu
   ReDim m(0 To 1)
   m(0) = derivadasS(0)
   m(1) = derivadasS(1)
   FormadorpolinomioSpline
Case Is = 2
   Calculohdu
   ReDim m(0 To 2)
   m(0) = derivadasS(0)
   m(2) = derivadasS(1)
   m(1) = (U(1) - h(0) * derivadasS(0) - h(1) * m(2)) / (2 * (h(0) + h(1)))
   FormadorpolinomioSpline
Case Else
   Calculohdu
   ReDim Atmp(1 To n - 1, 1 To n - 1) As Double
   ReDim Btmp(1 To n - 1) As Double

   Atmp(1, 1) = 2 * (h(0) + h(1))
   Atmp(1, 2) = h(1)
   Btmp(1) = U(1) - h(0) * derivadasS(0)

   For i = 2 To n - 2
   Atmp(i, i - 1) = h(i - 1)
   Atmp(i, i) = 2 * (h(i - 1) + h(i))
   Atmp(i, i + 1) = h(i)
   Btmp(i) = U(i)
   Next i
   Atmp(n - 1, n - 2) = h(n - 2)
   Atmp(n - 1, n - 1) = 2 * (h(n - 2) + h(n - 1))
   Btmp(n - 1) = U(n - 1) - h(n - 1) * derivadasS(1)
   ReDim A(0 To n - 2, 0 To n - 2) As Double
   ReDim b(0 To n - 2) As Double
   For i = 1 To n - 1
      For j = 1 To n - 1
```

```
        A(i - 1, j - 1) = Atmp(i, j)
        Next j
    Next i
    For i = 1 To n - 1
    b(i - 1) = Btmp(i)
    Next i
    err = False
    tol = 0.000001
    itermax = 500
    numecua = n - 1
    'Pasamos los valores al módulo ClasSEl para resolver el sistema de ecuaciones
    SEL.setearSEL(numecua, tol, itermax, err, A()) = b()
    SEL.EliGaussPivotación
    ReDim m(0 To n)
    For i = 0 To n - 2
    m(i + 1) = SEL.ReturSELX(i)
    Next i
    m(0) = derivadasS(0)
    m(n) = derivadasS(1)

    FormadorpolinomioSpline
End Select
Exit Sub
err1:
'If err.Number <> 0 Then
    mensaje = MsgBox("Parámetros del algoritmo incompletos", 48, "Error en el
ingreso")
 '   End If

End Sub

Public Sub diferenciasdivididas()
Dim i As Integer
Dim j As Integer
Dim mensaje As String
Dim fparcial As Double
Dim fparcial2 As Double

On Error GoTo err1
ReDim f(0 To numpares - 1, 0 To ordendd) As Double

If ordendd <= numpares - 1 Then
For i = 1 To numpares - 1
f(i, 0) = Pares(i, 1)
Next i
For j = 1 To ordendd
```

```
    For i = 0 To numpares - j - 1
    fparcial = f(i + 1, j - 1) - f(i, j - 1)
    fparcial2 = Pares(i + j, 0) - Pares(i, 0)
    If fparcial2 = 0 Then
    mensaje = MsgBox("No es posible obtener las diferencias divididas", 16,
"Fracaso")
    End If
    f(i, j) = fparcial / fparcial2
    Next i
Next j
Else
mensaje = MsgBox("El orden de las diferencias divididas debe ser menor al número
de pares de datos", 16, "Fracaso")
End If
Exit Sub
err1:

End Sub

Private Sub Calculohdu()
Dim i As Integer

Dim mensaje As String

On Error GoTo err1
ReDim h(0 To Ns - 1) As Double
ReDim d(0 To Ns - 1) As Double
ReDim U(1 To Ns - 1) As Double

For i = 0 To Ns - 1
h(i) = Pares(i + 1, 0) - Pares(i, 0)
If h(i) = 0 Then
mensaje = MsgBox("No es posible encontrar el polinomio interpolante a trozos", 16,
"Error en el ingreso")
Exit Sub
End If
Next i

For i = 0 To Ns - 1
d(i) = (Pares(i + 1, 1) - Pares(i, 1)) / h(i)
Next i

For i = 1 To Ns - 1
U(i) = 6 * (d(i) - d(i - 1))
Next i
Exit Sub
```

```
err1:
End Sub


Private Sub FormadorpolinomioSpline()
Dim i As Integer
Dim j As Integer
On Error GoTo err1
ReDim s(0 To 3, 0 To Ns - 1) As Double

For j = 0 To Ns - 1
   s(0, j) = Pares(j, 1)
   s(1, j) = d(j) - ((h(j) * ((2 * m(j)) + m(j + 1))) / 6)
   s(2, j) = m(j) / 2
   s(3, j) = (m(j + 1) - m(j)) / (6 * h(j))
Next j

'Armamos el primer polinómio
ReDim polinomioSpline(0 To Ns - 1) As String
If s(0, 0) <> 0 Then
polinomioSpline(0) = s(0, 0)
End If
If s(1, 0) <> 0 Then
   If s(1, 0) > 0 Then
      If s(0, 0) = 0 Then
      polinomioSpline(0) = polinomioSpline(0) & s(1, 0) & "*X"
      Else
      polinomioSpline(0) = polinomioSpline(0) & "+" & s(1, 0) & "*X"
      End If
   Else
   polinomioSpline(0) = polinomioSpline(0) & s(1, 0) & "*X"
   End If
End If
If s(2, 0) <> 0 Then
   If s(2, 0) > 0 Then
      If s(1, 0) = 0 And s(0, 0) = 0 Then
      polinomioSpline(0) = polinomioSpline(0) & s(2, 0) & "*X^2"
      Else
      polinomioSpline(0) = polinomioSpline(0) & "+" & s(2, 0) & "*X^2"
      End If
   Else
   polinomioSpline(0) = polinomioSpline(0) & s(2, 0) & "*X^2"
   End If
End If
If s(3, 0) <> 0 Then
   If s(3, 0) > 0 Then
```

```
      If s(2, 0) = 0 And s(1, 0) = 0 And s(0, 0) = 0 Then
      polinomioSpline(0) = polinomioSpline(0) & s(3, 0) & "*X^3"
      Else
      polinomioSpline(0) = polinomioSpline(0) & "+" & s(3, 0) & "*X^3"
      End If
   Else
   polinomioSpline(0) = polinomioSpline(0) & s(3, 0) & "*X^3"
   End If
End If

'Armamos los demás polinomios
For j = 1 To Ns - 1
If s(0, j) <> 0 Then
polinomioSpline(j) = s(0, j)
End If
If s(1, j) <> 0 Then
   If s(1, j) > 0 Then
      If Pares(j, 0) = 0 Then
         If s(0, j) = 0 Then
         polinomioSpline(j) = polinomioSpline(j) & s(1, j) & "*(X)"
         Else
         polinomioSpline(j) = polinomioSpline(j) & "+" & s(1, j) & "*(X)"
         End If
      Else
         If Pares(j, 0) > 0 Then
            If s(0, j) = 0 Then
            polinomioSpline(j) = polinomioSpline(j) & s(1, j) & "*(X-" & Pares(j, 0) & ")"
            Else
            polinomioSpline(j) = polinomioSpline(j) & "+" & s(1, j) & "*(X-" & Pares(j, 0)
& ")"
            End If
         Else
            If s(0, j) = 0 Then
            polinomioSpline(j) = polinomioSpline(j) & s(1, j) & "*(X+" & Abs(Pares(j, 0))
& ")"
            Else
            polinomioSpline(j) = polinomioSpline(j) & "+" & s(1, j) & "*(X+" &
Abs(Pares(j, 0)) & ")"
            End If
         End If
      End If
   Else
      If Pares(j, 0) = 0 Then
      polinomioSpline(j) = polinomioSpline(j) & s(1, j) & "*(X)"
      Else
         If Pares(j, 0) > 0 Then
```

```
            polinomioSpline(j) = polinomioSpline(j) & s(1, j) & "*(X-" & Pares(j, 0) & ")"
            Else
            polinomioSpline(j) = polinomioSpline(j) & s(1, j) & "*(X+" & Abs(Pares(j, 0)) &
")"
            End If
        End If

    End If
End If
If s(2, j) <> 0 Then
    If s(2, j) > 0 Then
    If Pares(j, 0) = 0 Then
            If s(1, j) = 0 And s(0, j) = 0 Then
            polinomioSpline(j) = polinomioSpline(j) & s(2, j) & "*X^2"
            Else
            polinomioSpline(j) = polinomioSpline(j) & "+" & s(2, j) & "*X^2"
            End If
        Else
            If Pares(j, 0) > 0 Then
                If s(1, j) = 0 And s(0, j) = 0 Then
                polinomioSpline(j) = polinomioSpline(j) & s(2, j) & "*(X-" & Pares(j, 0) & ")^2"
                Else
                polinomioSpline(j) = polinomioSpline(j) & "+" & s(2, j) & "*(X-" & Pares(j, 0)
& ")^2"
                End If
            Else
                If s(1, j) = 0 And s(0, j) = 0 Then
                polinomioSpline(j) = polinomioSpline(j) & s(2, j) & "*(X+" & Abs(Pares(j, 0))
& ")^2"
                Else
                polinomioSpline(j) = polinomioSpline(j) & "+" & s(2, j) & "*(X+" &
Abs(Pares(j, 0)) & ")^2"
                End If
            End If
        End If

    Else
    If Pares(j, 0) = 0 Then
        polinomioSpline(j) = polinomioSpline(j) & s(2, j) & "*X^2"
        Else
            If Pares(j, 0) > 0 Then
            polinomioSpline(j) = polinomioSpline(j) & s(2, j) & "*(X-" & Pares(j, 0) & ")^2"
            Else
            polinomioSpline(j) = polinomioSpline(j) & s(2, j) & "*(X+" & Abs(Pares(j, 0)) &
")^2"
            End If
```

```
      End If

   End If
End If
If s(3, j) <> 0 Then
   If s(3, j) > 0 Then
   If Pares(j, 0) = 0 Then
         If s(2, j) = 0 And s(1, j) = 0 And s(0, j) = 0 Then
         polinomioSpline(j) = polinomioSpline(j) & s(3, j) & "*X^3"
         Else
         polinomioSpline(j) = polinomioSpline(j) & "+" & s(3, j) & "*X^3"
         End If
      Else
         If Pares(j, 0) > 0 Then
            If s(2, j) = 0 And s(1, j) = 0 And s(0, j) = 0 Then
            polinomioSpline(j) = polinomioSpline(j) & s(3, j) & "*(X-" & Pares(j, 0) & ")^3"
            Else
            polinomioSpline(j) = polinomioSpline(j) & "+" & s(3, j) & "*(X-" & Pares(j, 0)
& ")^3"
            End If
         Else
            If s(2, j) = 0 And s(1, j) = 0 And s(0, j) = 0 Then
            polinomioSpline(j) = polinomioSpline(j) & s(3, j) & "*(X+" & Abs(Pares(j, 0))
& ")^3"
            Else
            polinomioSpline(j) = polinomioSpline(j) & "+" & s(3, j) & "*(X+" &
Abs(Pares(j, 0)) & ")^3"
            End If
         End If
      End If

   Else
   If Pares(j, 0) = 0 Then
      polinomioSpline(j) = polinomioSpline(j) & s(3, j) & "*X^3"
      Else
         If Pares(j, 0) > 0 Then
         polinomioSpline(j) = polinomioSpline(j) & s(3, j) & "*(X-" & Pares(j, 0) & ")^3"
         Else
         polinomioSpline(j) = polinomioSpline(j) & s(3, j) & "*(X+" & Abs(Pares(j, 0)) &
")^3"
         End If
      End If

   End If
End If
Next j
```

```
Exit Sub
err1:
End Sub



Private Function Peval(f As String, X As Double) As Double
On Error GoTo err1
eval.Tipox = False
eval.fa = f
eval.PtX = X
eval.evaluarf
Peval = eval.Fdy
Exit Function
err1:
End Function
```

CODIGO DEL CLUSTER (6)
Encontrar raíces de funciones no lineales

Interfaz FrmRFNL

```
Option Explicit
'En Microsoft TechNet puedes encontrar este artículo:
'HOWTO: Use HTML Help API in a Visual Basic 5.0 Application
'PSS ID Number: Q183434
'
'Aunque la definición de la Enumeración y la primera declaración
'es de las news
'
'Htmlhelp consts
Private Enum HH_COMMAND
   HH_DISPLAY_TOPIC = &H0
   HH_HELP_FINDER = &H0       ' WinHelp equivalent
   HH_DISPLAY_TOC = &H1       ' not currently implemented
   HH_DISPLAY_INDEX = &H2     ' not currently implemented
   HH_DISPLAY_SEARCH = &H3    ' not currently implemented
   HH_SET_WIN_TYPE = &H4
   HH_GET_WIN_TYPE = &H5
   HH_GET_WIN_HANDLE = &H6
   HH_GET_INFO_TYPES = &H7    ' not currently implemented
   HH_SET_INFO_TYPES = &H8    ' not currently implemented
   HH_SYNC = &H9
   HH_ADD_NAV_UI = &HA        ' not currently implemented
   HH_ADD_BUTTON = &HB        ' not currently implemented
```

```
    HH_GETBROWSER_APP = &HC     ' not currently implemented
    HH_KEYWORD_LOOKUP = &HD
    HH_DISPLAY_TEXT_POPUP = &HE ' display string resource id
                        ' or text in a popup window
    HH_HELP_CONTEXT = &HF       ' display mapped numeric value
                        ' in dwData
    HH_TP_HELP_CONTEXTMENU      ' Text pop-up help, similar to
                        ' WinHelp's HELP_CONTEXTMENU.
    HH_TP_HELP_WM_HELP = &H11   ' text pop-up help, similar to
                        ' WinHelp's HELP_WM_HELP.
    HH_CLOSE_ALL = &H12         ' close all windows opened directly
                        ' or indirectly by the caller
    HH_ALINK_LOOKUP = &H13      ' ALink version of HH_KEYWORD_LOOKUP
End Enum

'HtmlHelp api call
'NOTA: Si se usa esta forma, hay que indicar el último parámetro
'     con la palabra ByVal delante...
'Private Declare Function HtmlHelp Lib "hhctrl.ocx" Alias "HtmlHelpA" _
    (ByVal hwndCaller As Long, ByVal pszFile As String, _
    ByVal uCommand As HH_COMMAND, dwData As Any) As Long
'Con esta funciona perfectamente
Private Declare Function HtmlHelp Lib "hhctrl.ocx" Alias "HtmlHelpA" _
    (ByVal hwndCaller As Long, ByVal pszFile As String, _
    ByVal uCommand As HH_COMMAND, ByVal dwData As Long) As Long

Dim exval As New McsExcel
Dim MG As New Graficar
Dim XR As Double
Dim Yr As Double
Dim XT As Double
Dim Yt As Double
Dim Xmtmp As Double
Dim Xmatmp As Double
Dim Ymtmp As Double
Dim Ymatmp As Double
Dim n As Integer
Dim tipoalg As Integer
Dim datosgrafico() As Double
Dim raíz As New RFNL
Dim altoSstab
Dim anchoSstab
Dim arribaSstab
Dim izquierdaSstab
Dim altoResul
Dim anchoResul
```

```
Dim arribaResul
Dim izquierdaResul
Dim anchoFgresul
Dim altoFgresul
Dim altoitera
Dim anchoitera
Dim arribaltera
Dim izquierdaltera
Dim arribaP
Dim colororg
Public borrar As Boolean
Private Acepta As Boolean
Private VLConexion As ADODB.Connection
Private VLRegistro As ADODB.Recordset
Private VLNombre As String
Private VLNombreTrabajo As String
Private VLVerifica As Boolean
Private VLBDD As Boolean

Property Let Aceptar(parAceptar As Boolean)
   Acepta = parAceptar
End Property

Property Let Nombre(parNombre As String)
   VLNombre = parNombre
End Property

Property Let NombreTrabajo(parNombreTrabajo As String)
   VLNombreTrabajo = parNombreTrabajo
End Property


Private Sub ArchivoAbrir_Click()
   FrmRecuperaTrabajo.Show vbModal
   If Acepta = True Then
      PLConexion True
      PLRecupera
   End If
End Sub

Private Sub ArchivoGuardarcomo_Click()
   FrmNombre.Show vbModal
   If Acepta = True Then
      PLConexion True
      PLGuardarBDD
   End If
```

```
End Sub

Private Sub ArchivoImprimir_Click()
   Dim EndTime As Date
   Sincolor

      EndTime = DateAdd("s", 0.1, Now)
      Do Until Now > EndTime
        DoEvents
      Loop
   'Form1.Command3_Click
   Set Form1.Picture1.Picture = CaptureClient(Me)
   Form1.Visible = True
   FrmPrincipalalgoritmos.Visible = False
   Concolor
End Sub

Private Sub Form_Load()
   ArchivoAbrir.Enabled = False
   ArchivoGuardarcomo.Enabled = False
   ArchivoImprimir.Enabled = False
   VGForma = "RFNL"
End Sub




Private Sub MnuAyuda_Click()
'Así se llamaría para mostrar un tópico de la ayuda
   Dim h As Long

   'h = HtmlHelp(Me.hWnd, "Milton.hlp", HH_DISPLAY_TOPIC, 0&)

   h = HtmlHelp(Me.hWnd, "Milton.hlp", HH_HELP_CONTEXT, 2020&)
End Sub

Private Sub Rfnlaproxinmaciones_Click()
'Seteamos el tipo de algoritmo elejido en este caso aproximaciones sucesivas
tipoalg = 1
'Enviando el tipo de algoritmo elejido a la forma de guardar el nombre
   ArchivoAbrir.Enabled = True
   ArchivoGuardarcomo.Enabled = True
   FrmRecuperaTrabajo.ver = tipoalg
'Presentamos los controles necesarios para el funcionamiento del algoritmo
LblRfnltitulo.Visible = True
```

```
LblRfnlfunción.Visible = True:  lbl1.Visible = True
TxtRfnlfunción.Text = " "
TxtRfnlfunción.Visible = True:  txt1.Visible = True
LblRfnlderivada.Visible = False:    lbl2.Visible = False
TxtRfnlderivada.Text = " "
TxtRfnlderivada.Visible = False:    txt2.Visible = False
LblRfnla.Caption = "Xo="
LblRfnla.Visible = True:    lbl3.Visible = True
TxtRfnla.Text = " "
TxtRfnla.Visible = True:    txt3.Visible = True
LblRfnlb.Caption = "h="
LblRfnlb.Visible = True:    lbl4.Visible = True
TxtRfnlb.Text = " "
TxtRfnlb.Visible = True:    txt4.Visible = True
FraRfnlitera.Visible = True
LblRfnliteraciones.Visible = True:  lbl5.Visible = True
TxtRfnliteraciones.Text = " "
TxtRfnliteraciones.Visible = True:  txt5.Visible = True
LblRfnltolerancia.Visible = False:  lbl6.Visible = False
TxtRfnltolerancia.Text = " "
TxtRfnltolerancia.Visible = False:  txt6.Visible = False
FraRfnlerror.Visible = False
OptRfnlex.Visible = False
OptRfnlef.Visible = False
ShaRfnlresultados.Visible = True
LblRfnlresultados.Visible = True
fgRfnlresultados.Clear
fgRfnlresultados.Visible = True
fgRfnliteraciones.Clear
fgRfnliteraciones.Visible = False
CmdRfnlcalcular.Visible = True
CmdRfnliteraciones.Visible = False
CmdRfnlgraficar.Visible = False
Label1.Visible = False
LblNomalg.Caption = "Aproximaciones Sucesivas"
LblRfnlSintaxis.Caption = "f(x)= función para la cual se obtendrá el intervalo en el que
se encuentra la raíz  ejemplo:exp(-1*X^2)*(SIN(X)/7)" & Chr(10) & "Puede usar las
funciones: Ln(X);COS(X);SQRT(X);Log(X); Tan(X); PI(), el resto de funciones sigue
la sintaxis de Excel" & Chr(10) & _
"Xo= Punto en el cual se inicia la búsqueda de la raíz" & Chr(10) & "h= incremento
que se realiza al punto inical de búsqueda" & Chr(10) & "Iteraciones máximas=
número natural, representa el número de iteraciones que realizará el algoritmo"
LblAviso.Visible = False
End Sub

Private Sub RfnlbisecciÓn_Click()
```

```
'Seteamos el tipo de algoritmo elejido en este caso bisección
tipoalg = 2
   ArchivoAbrir.Enabled = True
   ArchivoGuardarcomo.Enabled = True
   FrmRecuperaTrabajo.ver = tipoalg
'Presentamos los controles necesarios para el funcionamiento del algoritmo
LblRfnltitulo.Visible = True
LblRfnlfunción.Visible = True:  lbl1.Visible = True
TxtRfnlfunción.Text = " "
TxtRfnlfunción.Visible = True:  txt1.Visible = True
LblRfnlderivada.Visible = False:    lbl2.Visible = False
TxtRfnlderivada.Text = " "
TxtRfnlderivada.Visible = False:    txt2.Visible = False
LblRfnla.Caption = "a="
LblRfnla.Visible = True:    lbl3.Visible = True
TxtRfnla.Text = " "
TxtRfnla.Visible = True:    txt3.Visible = True
LblRfnlb.Caption = "b="
LblRfnlb.Visible = True:    lbl4.Visible = True
TxtRfnlb.Text = " "
TxtRfnlb.Visible = True:    txt4.Visible = True
FraRfnlitera.Visible = True
LblRfnliteraciones.Visible = True:  lbl5.Visible = True
TxtRfnliteraciones.Text = " "
TxtRfnliteraciones.Visible = True:  txt5.Visible = True
LblRfnltolerancia.Visible = True:   lbl6.Visible = True
TxtRfnltolerancia.Text = " "
TxtRfnltolerancia.Visible = True:   txt6.Visible = True
FraRfnlerror.Visible = True
OptRfnlex.Visible = True
OptRfnlef.Visible = True
ShaRfnlresultados.Visible = True
LblRfnlresultados.Visible = True
fgRfnlresultados.Clear
fgRfnlresultados.Visible = True
fgRfnliteraciones.Clear
fgRfnliteraciones.Visible = False
CmdRfnlcalcular.Visible = True
CmdRfnliteraciones.Visible = False
CmdRfnlgraficar.Visible = False
Label1.Visible = False
LblNomalg.Caption = "Algoritmo de Bisección"
LblRfnlSintaxis.Caption = "f(x)= función para la cual se obtendrá la raíz ejemplo:exp(-
1*X^2)*(SIN(X)/7)" & Chr(10) & "Puede usar las funciones:
Ln(X);COS(X);SQRT(X);Log(X); Tan(X); PI(), el resto de funciones sigue la sintaxis
de Excel" & Chr(10) & _
```

```
"[a,b] intervalo en el que se encuentra la raíz" & Chr(10) & "Tolerancia= 0.001" &
Chr(10) & "Iteraciones máximas= número natural, representa el número de
iteraciones que realizará el algoritmo" & Chr(10) & "Er=error relativo; Ef= error en la
función"
LblAviso.Visible = False
End Sub

Private Sub RfnlfalsaposiciónClick()
'Seteamos el tipo de algoritmo elejido en este caso falsa posición
tipoalg = 3
   ArchivoAbrir.Enabled = True
   ArchivoGuardarcomo.Enabled = True
   FrmRecuperaTrabajo.ver = tipoalg
'Presentamos los controles necesarios para el funcionamiento del algoritmo
LblRfnltitulo.Visible = True
LblRfnlfunción.Visible = True:  lbl1.Visible = True
TxtRfnlfunción.Text = " "
TxtRfnlfunción.Visible = True:     txt1.Visible = True
LblRfnlderivada.Visible = False:   lbl2.Visible = False
TxtRfnlderivada.Text = " "
TxtRfnlderivada.Visible = False:   txt2.Visible = False
LblRfnla.Caption = "a="
LblRfnla.Visible = True:   lbl3.Visible = True
TxtRfnla.Text = " "
TxtRfnla.Visible = True:   txt3.Visible = True
LblRfnlb.Caption = "b="
LblRfnlb.Visible = True:   lbl4.Visible = True
TxtRfnlb.Text = " "
TxtRfnlb.Visible = True:   txt4.Visible = True
FraRfnlitera.Visible = True
LblRfnliteraciones.Visible = True:  lbl5.Visible = True
TxtRfnliteraciones.Text = " "
TxtRfnliteraciones.Visible = True:  txt5.Visible = True
LblRfnltolerancia.Visible = True:   lbl6.Visible = True
TxtRfnltolerancia.Text = " "
TxtRfnltolerancia.Visible = True:   txt6.Visible = True
FraRfnlerror.Visible = True
OptRfnlex.Visible = True
OptRfnlef.Visible = True
ShaRfnlresultados.Visible = True
LblRfnlresultados.Visible = True
fgRfnlresultados.Clear
fgRfnlresultados.Visible = True
fgRfnliteraciones.Clear
fgRfnliteraciones.Visible = False
CmdRfnlcalcular.Visible = True
```

```
CmdRfnIiteraciones.Visible = False
CmdRfnIgraficar.Visible = False
Label1.Visible = False
LblNomalg.Caption = "Algoritmo de Falsa Posición"
LblRfnlSintaxis.Caption = "f(x)= función para la cual se obtendrá la raíz ejemplo:exp(-
1*X^2)*(SIN(X)/7)" & Chr(10) & "Puede usar las funciones:
Ln(X);COS(X);SQRT(X);Log(X); Tan(X); PI(), el resto de funciones sigue la sintaxis
de Excel" & Chr(10) & _
"[a,b] intervalo en el que se encuentra la raíz" & Chr(10) & "Tolerancia= 0.001" &
Chr(10) & "Iteraciones máximas= número natural, representa el número de
iteraciones que realizará el algoritmo" & Chr(10) & "Er=error relativo; Ef= error en la
función"
LblAviso.Visible = False
End Sub



Private Sub RfnIgraficar_Click()
Frmescalas.Show
End Sub

Private Sub RfnIprimerorde_Click()
'Seteamos el tipo de algoritmo elejido en este caso falsa posición
tipoalg = 4
   ArchivoAbrir.Enabled = True
   ArchivoGuardarcomo.Enabled = True
   FrmRecuperaTrabajo.ver = tipoalg
   lbl1.Visible = False:   lbl2.Visible = True
   lbl3.Visible = True:   lbl4.Visible = False
   lbl5.Visible = True:   lbl6.Visible = True
   txt1.Visible = False:   txt2.Visible = True
   txt3.Visible = True:   txt4.Visible = False
   txt5.Visible = True:   txt6.Visible = True
'Presentamos los controles necesarios para el funcionamiento del algoritmo
LblRfnltitulo.Visible = True
LblRfnlfunción.Visible = False
TxtRfnlfunción.Text = " "
TxtRfnlfunción.Visible = False
LblRfnlderivada.Caption = "g(x)="
LblRfnlderivada.Visible = True
TxtRfnlderivada.Text = " "
TxtRfnlderivada.Visible = True
LblRfnla.Caption = "Xo="
LblRfnla.Visible = True
TxtRfnla.Text = " "
TxtRfnla.Visible = True
```

```
LblRfnlb.Caption = "b="
LblRfnlb.Visible = False
TxtRfnlb.Text = " "
TxtRfnlb.Visible = False
FraRfnlitera.Visible = True
LblRfnliteraciones.Visible = True
TxtRfnliteraciones.Text = " "
TxtRfnliteraciones.Visible = True
LblRfnltolerancia.Visible = True
TxtRfnltolerancia.Text = " "
TxtRfnltolerancia.Visible = True
FraRfnlerror.Visible = True
OptRfnlex.Visible = True
OptRfnlef.Visible = True
ShaRfnlresultados.Visible = True
LblRfnlresultados.Visible = True
fgRfnlresultados.Clear
fgRfnlresultados.Visible = True
fgRfnliteraciones.Clear
fgRfnliteraciones.Visible = False
CmdRfnlcalcular.Visible = True
CmdRfnliteraciones.Visible = False
CmdRfnlgraficar.Visible = False
Label1.Visible = False
LblNomalg.Caption = "Algoritmo de Punto Fijo"
LblRfnlSintaxis.Caption = "g(x)= función a la cual se le aplicará la iteración de punto
fijo para hallar la raíz de la función f(x) ejemplo:exp(-1*X^2)*(SIN(X)/7)" & Chr(10) &
"Puede usar las funciones: Ln(X);COS(X);SQRT(X);Log(X); Tan(X); PI(), el resto de
funciones sigue la sintaxis de Excel" & Chr(10) & _
"Xo= punto inicial para empezar la iteración de punto fijo" & Chr(10) & "Tolerancia=
0.001" & Chr(10) & "Iteraciones máximas= número natural, representa el número de
iteraciones que realizará el algoritmo" & Chr(10) & "Er=error relativo; Ef= error en la
función"
LblAviso.Caption = "Nota: Ingrese g(x) si y solo sí cumple con la condición de
convergencia de primer orden es decir; g(x) de ser definida de manera que cumpla
con la condición g' (x) diferente de cero y que en el intervalo [a,b] donde tiene la raíz
satisfaga la condición " & Chr(124) & " g' (x) " & Chr(124) & "<1"
LblAviso.Visible = True
End Sub

Private Sub Rfnlraphson_Click()
'Seteamos el tipo de algoritmo elejido en este caso bisección
tipoalg = 5
   ArchivoAbrir.Enabled = True
   ArchivoGuardarcomo.Enabled = True
   FrmRecuperaTrabajo.ver = tipoalg
```

```
    lbl1.Visible = True:   lbl2.Visible = True
    lbl3.Visible = True:   lbl4.Visible = False
    lbl5.Visible = True:   lbl6.Visible = True
    txt1.Visible = True:   txt2.Visible = True
    txt3.Visible = True:   txt4.Visible = False
    txt5.Visible = True:   txt6.Visible = True

'Presentamos los controles necesarios para el funcionamiento del algoritmo
LblRfnltitulo.Visible = True
LblRfnlfunción.Visible = True
TxtRfnlfunción.Text = " "
TxtRfnlfunción.Visible = True
LblRfnlderivada.Caption = "f'(x)="
LblRfnlderivada.Visible = True
TxtRfnlderivada.Text = " "
TxtRfnlderivada.Visible = True
LblRfnla.Caption = "Xo="
LblRfnla.Visible = True
TxtRfnla.Text = " "
TxtRfnla.Visible = True
LblRfnlb.Caption = "b="
LblRfnlb.Visible = False
TxtRfnlb.Text = " "
TxtRfnlb.Visible = False
FraRfnlitera.Visible = True
LblRfnliteraciones.Visible = True
TxtRfnliteraciones.Text = " "
TxtRfnliteraciones.Visible = True
LblRfnltolerancia.Visible = True
TxtRfnltolerancia.Text = " "
TxtRfnltolerancia.Visible = True
FraRfnlerror.Visible = True
OptRfnlex.Visible = True
OptRfnlef.Visible = True
ShaRfnlresultados.Visible = True
LblRfnlresultados.Visible = True
fgRfnlresultados.Clear
fgRfnlresultados.Visible = True
fgRfnliteraciones.Clear
fgRfnliteraciones.Visible = False
CmdRfnlcalcular.Visible = True
CmdRfnliteraciones.Visible = False
CmdRfnlgraficar.Visible = False
Label1.Visible = False
LblNomalg.Caption = "Algoritmo de Newton-Rapshon"
```

```
LblRfnlSintaxis.Caption = "f(x)= función de cual se encontrará la raíz; f'(x)= primera
derivada de la función f(x)  ejemplo:exp(-1*X^2)*(SIN(X)/7)" & Chr(10) & "Puede usar
las funciones: Ln(X);COS(X);SQRT(X);Log(X); Tan(X); PI(), el resto de funciones
sigue la sintaxis de Excel" & Chr(10) & _
"Xo= punto inicial cercano a la raíz" & Chr(10) & "Tolerancia= 0.001" & Chr(10) &
"Iteraciones máximas= número natural, representa el número de iteraciones que
realizará el algoritmo" & Chr(10) & "Er=error relativo; Ef= error en la función"
LblAviso.Visible = False
End Sub

Private Sub RfnlSalir_Click()
Unload Me
End Sub

Private Sub RfnlSecante_Click()
'Seteamos el tipo de algoritmo elejido en este caso bisección
tipoalg = 6
   ArchivoAbrir.Enabled = True
   ArchivoGuardarcomo.Enabled = True
   FrmRecuperaTrabajo.ver = tipoalg
   lbl1.Visible = True:   lbl2.Visible = False
   lbl3.Visible = True:   lbl4.Visible = True
   lbl5.Visible = True:   lbl6.Visible = True
   txt1.Visible = True:   txt2.Visible = False
   txt3.Visible = True:   txt4.Visible = True
   txt5.Visible = True:   txt6.Visible = True
'Presentamos los controles necesarios para el funcionamiento del algoritmo
LblRfnlTitulo.Visible = True
LblRfnlFunción.Visible = True
TxtRfnlFunción.Text = " "
TxtRfnlFunción.Visible = True
LblRfnlderivada.Visible = False
TxtRfnlderivada.Text = " "
TxtRfnlderivada.Visible = False
LblRfnla.Caption = "Xo="
LblRfnla.Visible = True
TxtRfnla.Text = " "
TxtRfnla.Visible = True
LblRfnlb.Caption = "X1="
LblRfnlb.Visible = True
TxtRfnlb.Text = " "
TxtRfnlb.Visible = True
FraRfnlitera.Visible = True
LblRfnliteraciones.Visible = True
TxtRfnliteraciones.Text = " "
TxtRfnliteraciones.Visible = True
```

```
LblRfnltolerancia.Visible = True
TxtRfnltolerancia.Text = " "
TxtRfnltolerancia.Visible = True
FraRfnlerror.Visible = True
OptRfnlex.Visible = True
OptRfnlef.Visible = True
ShaRfnlresultados.Visible = True
LblRfnlresultados.Visible = True
fgRfnlresultados.Clear
fgRfnlresultados.Visible = True
fgRfnliteraciones.Clear
fgRfnliteraciones.Visible = False
CmdRfnlcalcular.Visible = True
CmdRfnliteraciones.Visible = False
CmdRfnlgraficar.Visible = False
Label1.Visible = False
LblNomalg.Caption = "Algoritmo de la Secante"
LblRfnlSintaxis.Caption = "f(x)= función de cual se encontrará la raíz ejemplo:exp(-
1*X^2)*(SIN(X)/7)" & Chr(10) & "Puede usar las funciones:
Ln(X);COS(X);SQRT(X);Log(X); Tan(X); PI(), el resto de funciones sigue la sintaxis
de Excel" & Chr(10) & _
"Xo y X1  puntos iniciales cercanos a la raíz" & Chr(10) & "Tolerancia= 0.001" &
Chr(10) & "Iteraciones máximas= número natural, representa el número de
iteraciones que realizará el algoritmo" & Chr(10) & "Er=error relativo; Ef= error en la
función"
LblAviso.Visible = False
End Sub
Private Sub CmdRfnlcalcular_Click()
Dim mensaje As String
Dim verifica As Double

Select Case tipoalg
Case Is = 1
'manejo del error en el ingreso de la función


If (TxtRfnlfunción = "" Or TxtRfnlfunción = " " Or TxtRfnla = "" Or TxtRfnla = " " Or
TxtRfnlb = "" Or TxtRfnlb = " " Or _
TxtRfnliteraciones = "" Or TxtRfnliteraciones = " ") Then
mensaje = MsgBox("Parámetros del algoritmo incompletos", 48, "Error en el ingreso")
Exit Sub
End If
exval.Tipox = False
exval.fa = TxtRfnlfunción
exval.PtX = 500
exval.evaluarf
```

```
verifica = exval.Fdy
   If verifica = 8121975 Then
      mensaje = MsgBox("Debe ingresar la función de acuerdo a la sintaxis
especificada", 16, "Error en el ingreso")
      Exit Sub
   End If

Enviar
raíz.AproximacionesSucesivas
If borrar = True Then
fgRfnlresultados.Clear
fgRfnliteraciones.Clear
borrar = False
Else
Presentar
End If
CmdRfnliteraciones.Visible = True
Case Is = 2
'manejo del error en el ingreso de la función
If (TxtRfnlfunción = "" Or TxtRfnlfunción = " " Or TxtRfnla = "" Or TxtRfnla = " " Or
TxtRfnlb = "" Or TxtRfnlb = " " Or _
TxtRfnliteraciones = "" Or TxtRfnliteraciones = " " Or TxtRfnltolerancia = "" Or
TxtRfnltolerancia = " ") Then
mensaje = MsgBox("Parámetros del algoritmo incompletos", 48, "Error en el ingreso")
Exit Sub
End If
exval.Tipox = False
exval.fa = TxtRfnlfunción
exval.PtX = 500
exval.evaluarf
verifica = exval.Fdy
   If verifica = 8121975 Then
      mensaje = MsgBox("Debe ingresar la función de acuerdo a la sintaxis
especificada", 16, "Error en el ingreso")
      Exit Sub
   End If
Enviar
raíz.Bisección
If borrar = True Then
fgRfnlresultados.Clear
fgRfnliteraciones.Clear
borrar = False
Else
Presentar
End If
CmdRfnliteraciones.Visible = True
```

```
Case Is = 3
'manejo del error en el ingreso de la función
If (TxtRfnlfunción = "" Or TxtRfnlfunción = " " Or TxtRfnla = "" Or TxtRfnla = " " Or
TxtRfnlb = "" Or TxtRfnlb = " " Or _
TxtRfnliteraciones = "" Or TxtRfnliteraciones = " " Or TxtRfnltolerancia = "" Or
TxtRfnltolerancia = " ") Then
mensaje = MsgBox("Parámetros del algoritmo incompletos", 48, "Error en el ingreso")
Exit Sub
End If
exval.Tipox = False
exval.fa = TxtRfnlfunción
exval.PtX = 500
exval.evaluarf
verifica = exval.Fdy
   If verifica = 8121975 Then
      mensaje = MsgBox("Debe ingresar la función de acuerdo a la sintaxis
especificada", 16, "Error en el ingreso")
      Exit Sub
   End If
Enviar
raíz.FalsePosición
If borrar = True Then
fgRfnlresultados.Clear
fgRfnliteraciones.Clear
borrar = False
Else
Presentar
End If
CmdRfnliteraciones.Visible = True
Case Is = 4
'manejo del error en el ingreso de la función
If (TxtRfnlderivada = "" Or TxtRfnlderivada = " " Or TxtRfnla = "" Or TxtRfnla = " " Or _
TxtRfnliteraciones = "" Or TxtRfnliteraciones = " " Or TxtRfnltolerancia = "" Or
TxtRfnltolerancia = " ") Then
mensaje = MsgBox("Parámetros del algoritmo incompletos", 48, "Error en el ingreso")
Exit Sub
End If
exval.Tipox = False
exval.fa = TxtRfnlderivada
exval.PtX = 500
exval.evaluarf
verifica = exval.Fdy
   If verifica = 8121975 Then
      mensaje = MsgBox("Debe ingresar la función de acuerdo a la sintaxis
especificada", 16, "Error en el ingreso")
      Exit Sub
```

```
   End If

Enviar
raíz.Primerorden
If borrar = True Then
fgRfnIresultados.Clear
fgRfnIiteraciones.Clear
borrar = False
Else
Presentar
End If
CmdRfnIiteraciones.Visible = True
Case Is = 5
'manejo del error en el ingreso de la función
'manejo del error en el ingreso de la función
If (TxtRfnIfunción = "" Or TxtRfnIfunción = " " Or TxtRfnIderivada = "" Or
TxtRfnIderivada = " " Or TxtRfnIa = "" Or TxtRfnIa = " " Or _
TxtRfnIiteraciones = "" Or TxtRfnIiteraciones = " " Or TxtRfnItolerancia = "" Or
TxtRfnItolerancia = " ") Then
mensaje = MsgBox("Parámetros del algoritmo incompletos", 48, "Error en el ingreso")
Exit Sub
End If
exval.Tipox = False
exval.fa = TxtRfnIderivada
exval.PtX = 500
exval.evaluarf
verifica = exval.Fdy
   If verifica = 8121975 Then
      mensaje = MsgBox("Debe ingresar la derivada de acuerdo a la sintaxis
especificada", 16, "Error en el ingreso")
      Exit Sub
   End If
exval.Tipox = False
exval.fa = TxtRfnIfunción
exval.PtX = 500
exval.evaluarf
verifica = exval.Fdy
   If verifica = 8121975 Then
      mensaje = MsgBox("Debe ingresar la función de acuerdo a la sintaxis
especificada", 16, "Error en el ingreso")
      Exit Sub
   End If
Enviar
raíz.NewtonRaphson
If borrar = True Then
fgRfnIresultados.Clear
```

```
fgRfnIiteraciones.Clear
borrar = False
Else
Presentar
End If
CmdRfnIiteraciones.Visible = True
Case Else
If (TxtRfnIfunción = "" Or TxtRfnIfunción = " " Or TxtRfnIa = "" Or TxtRfnIa = " " Or
TxtRfnIb = "" Or TxtRfnIb = " " Or _
TxtRfnIiteraciones = "" Or TxtRfnIiteraciones = " " Or TxtRfnItolerancia = "" Or
TxtRfnItolerancia = " ") Then
mensaje = MsgBox("Parámetros del algoritmo incompletos", 48, "Error en el ingreso")
Exit Sub
End If
exval.Tipox = False
exval.fa = TxtRfnIfunción
exval.PtX = 500
exval.evaluarf
verifica = exval.Fdy
   If verifica = 8121975 Then
      mensaje = MsgBox("Debe ingresar la función de acuerdo a la sintaxis
especificada", 16, "Error en el ingreso")
      Exit Sub
   End If
Enviar
raíz.Secante
If borrar = True Then
fgRfnIresultados.Clear
fgRfnIiteraciones.Clear
borrar = False
Else
Presentar
End If
CmdRfnIiteraciones.Visible = True
End Select
PLPasaValores
End Sub
Private Sub CmdRfnIiteraciones_Click()
Label1.Visible = True
fgRfnIiteraciones.Visible = True
End Sub
'Envia datos de los algoritmos
Private Sub Enviar()
Dim fsal As String
Dim gsal As String
Dim asal As Double
```

```
Dim bsal As Double
Dim tolsal As Double
Dim iterasal As Integer
Dim exsal As Boolean
On Error GoTo err1
'Enviamos los datos al módulo RFNL para que ejecute el algoritmo requerido
fsal = TxtRfnlfunción.Text
gsal = TxtRfnlderivada.Text
asal = Val(TxtRfnla.Text)
bsal = Val(TxtRfnlb.Text)
tolsal = Val(TxtRfnltolerancia.Text)
iterasal = Val(TxtRfnliteraciones.Text)
exsal = OptRfnlex.Value
raíz.SeteraRfnl(fsal, gsal, asal, bsal, iterasal, tolsal) = exsal
Exit Sub
err1:
   If err.Number <> 0 Then
      MsgBox "Se ha producido el siguiente error: " & err.Description & "." _
           & "Por favor consultar con el administrador", vbCritical
   End If
End Sub
Function Fgit(r As Integer, c As Integer) As Integer
  Fgit = c + fgRfnliteraciones.Cols * r
End Function

Function Fgr(r As Integer, c As Integer) As Integer
  Fgr = c + fgRfnlresultados.Cols * r
End Function
'Presenta datos de los algoritmos
Public Sub Presentar()
Dim i As Integer
Dim j As Integer
Dim valor As Double
On Error GoTo err1
Select Case tipoalg
Case Is = 1
'Presentación de los resultados
  fgRfnlresultados.Cols = 2
  fgRfnlresultados.Rows = 1
  fgRfnlresultados.FixedCols = 1
  fgRfnlresultados.TextArray(Fgr(0, 0)) = "Intervalo="
  fgRfnlresultados.ColWidth(1) = 5000
  valor = raíz.regresarraíz
  fgRfnlresultados.TextMatrix(0, 1) = "[" & TxtRfnla.Text & "," & valor & "]"

'Presentación de iteraciones
```

```
    fgRfnIiteraciones.Cols = 5
    fgRfnIiteraciones.Rows = Val(TxtRfnIiteraciones.Text) + 1
    fgRfnIiteraciones.FixedRows = 1
    fgRfnIiteraciones.FixedCols = 1
    fgRfnIiteraciones.TextArray(Fgit(0, 0)) = "Iteraciones"
    fgRfnIiteraciones.TextArray(Fgit(0, 1)) = "a"
    fgRfnIiteraciones.TextArray(Fgit(0, 2)) = "fa"
    fgRfnIiteraciones.TextArray(Fgit(0, 3)) = "b"
    fgRfnIiteraciones.TextArray(Fgit(0, 4)) = "fb"

    For i = 0 To 4
    fgRfnIiteraciones.ColAlignmentFixed(i) = 4
    fgRfnIresultados.ColWidth(i) = 2500
    Next i
    For i = 1 To Val(TxtRfnIiteraciones.Text)
    fgRfnIiteraciones.TextArray(Fgit(i, 0)) = i
    Next i

    For i = 0 To Val(TxtRfnIiteraciones.Text) - 1
        For j = 0 To 3
        fgRfnIiteraciones.TextMatrix(i + 1, j + 1) = raíz.regresariteraciones(i, j)
        Next j
     Next i
  Case Is = 2
  'Presentación de los resultados
    fgRfnIresultados.Cols = 2
    fgRfnIresultados.Rows = 1
    fgRfnIresultados.FixedCols = 1
    fgRfnIresultados.TextArray(Fgr(0, 0)) = "Raíz estimada="
    fgRfnIresultados.ColWidth(1) = 8000
    valor = raíz.regresarraíz
    fgRfnIresultados.TextMatrix(0, 1) = valor

  'Presentación de iteraciones
    fgRfnIiteraciones.Cols = 6
    fgRfnIiteraciones.Rows = Val(TxtRfnIiteraciones.Text) + 1
    fgRfnIiteraciones.FixedRows = 1
    fgRfnIiteraciones.FixedCols = 1
    For i = 0 To 5
    fgRfnIiteraciones.ColAlignmentFixed(i) = 4
    fgRfnIresultados.ColWidth(i) = 2500
    Next i
    For i = 0 To Val(TxtRfnIiteraciones.Text)
    fgRfnIiteraciones.TextArray(Fgit(i, 0)) = i
    Next i
```

```
  fgRfnIteraciones.TextArray(Fgit(0, 0)) = "Iteraciones"
  fgRfnIteraciones.TextArray(Fgit(0, 1)) = "a"
  fgRfnIteraciones.TextArray(Fgit(0, 2)) = "b"
  fgRfnIteraciones.TextArray(Fgit(0, 3)) = "Xi"
  fgRfnIteraciones.TextArray(Fgit(0, 4)) = "fXi"
  fgRfnIteraciones.TextArray(Fgit(0, 5)) = "Error"

  For i = 0 To Val(TxtRfnIteraciones.Text) - 1
     For j = 0 To 4
     fgRfnIteraciones.TextMatrix(i + 1, j + 1) = raíz.regresariteraciones(i, j)
     Next j
   Next i


Case Is = 3
'Presentación de los resultados
  fgRfnIresultados.Cols = 2
  fgRfnIresultados.Rows = 1
  fgRfnIresultados.FixedCols = 1
  fgRfnIresultados.TextArray(Fgr(0, 0)) = "Raíz estimada="
  fgRfnIresultados.ColWidth(1) = 8000
  valor = raíz.regresarraíz
  fgRfnIresultados.TextMatrix(0, 1) = valor

'Presentación de iteraciones
  fgRfnIteraciones.Cols = 6
  fgRfnIteraciones.Rows = Val(TxtRfnIteraciones.Text) + 1
  fgRfnIteraciones.FixedRows = 1
  fgRfnIteraciones.FixedCols = 1
  For i = 0 To 5
  fgRfnIteraciones.ColAlignmentFixed(i) = 4
  fgRfnIresultados.ColWidth(i) = 2500
  Next i
  For i = 0 To Val(TxtRfnIteraciones.Text)
  fgRfnIteraciones.TextArray(Fgit(i, 0)) = i
  Next i

  fgRfnIteraciones.TextArray(Fgit(0, 0)) = "Iteraciones"
  fgRfnIteraciones.TextArray(Fgit(0, 1)) = "a"
  fgRfnIteraciones.TextArray(Fgit(0, 2)) = "b"
  fgRfnIteraciones.TextArray(Fgit(0, 3)) = "Xi"
  fgRfnIteraciones.TextArray(Fgit(0, 4)) = "fXi"
  fgRfnIteraciones.TextArray(Fgit(0, 5)) = "Error"

  For i = 0 To Val(TxtRfnIteraciones.Text) - 1
     For j = 0 To 4
```

```
        fgRfnliteraciones.TextMatrix(i + 1, j + 1) = raíz.regresariteraciones(i, j)
        Next j
    Next i
Case Is = 4
'Presentación de los resultados
  fgRfnlresultados.Cols = 2
  fgRfnlresultados.Rows = 1
  fgRfnlresultados.FixedCols = 1
  fgRfnlresultados.TextArray(Fgr(0, 0)) = "Raíz estimada="
  fgRfnlresultados.ColWidth(1) = 8000
  valor = raiz.regresarraiz
  fgRfnlresultados.TextMatrix(0, 1) = valor

'Presentación de iteraciones
  fgRfnliteraciones.Cols = 4
  fgRfnliteraciones.Rows = Val(TxtRfnliteraciones.Text) + 1
  fgRfnliteraciones.FixedRows = 1
  fgRfnliteraciones.FixedCols = 1
  fgRfnlresultados.ColWidth(0) = 2500
  For i = 1 To 2
  fgRfnliteraciones.ColAlignmentFixed(i) = 4
  fgRfnlresultados.ColWidth(i) = 3500
  Next i
  For i = 0 To Val(TxtRfnliteraciones.Text)
  fgRfnliteraciones.TextArray(Fgit(i, 0)) = i
  Next i

  fgRfnliteraciones.TextArray(Fgit(0, 0)) = "Iteraciones"
  fgRfnliteraciones.TextArray(Fgit(0, 1)) = "Xi-1"
  fgRfnliteraciones.TextArray(Fgit(0, 2)) = "Xi"
  fgRfnliteraciones.TextArray(Fgit(0, 3)) = "Error"


  For i = 0 To Val(TxtRfnliteraciones.Text) - 1
      For j = 0 To 2
      fgRfnliteraciones.TextMatrix(i + 1, j + 1) = raíz.regresariteraciones(i, j)
      Next j
    Next i
Case Is = 5
'Presentación de los resultados
  fgRfnlresultados.Cols = 2
  fgRfnlresultados.Rows = 1
  fgRfnlresultados.FixedCols = 1
  fgRfnlresultados.TextArray(Fgr(0, 0)) = "Raíz estimada="
  fgRfnlresultados.ColWidth(1) = 8000
  valor = raiz.regresarraiz
```

```
    fgRfnIresultados.TextMatrix(0, 1) = valor

'Presentación de iteraciones
    fgRfnIiteraciones.Cols = 5
    fgRfnIiteraciones.Rows = Val(TxtRfnIiteraciones.Text) + 1
    fgRfnIiteraciones.FixedRows = 1
    fgRfnIiteraciones.FixedCols = 1
    For i = 0 To 4
    fgRfnIiteraciones.ColAlignmentFixed(i) = 4
    fgRfnIresultados.ColWidth(i) = 2500
    Next i
    For i = 0 To Val(TxtRfnIiteraciones.Text)
    fgRfnIiteraciones.TextArray(Fgit(i, 0)) = i
    Next i

    fgRfnIiteraciones.TextArray(Fgit(0, 0)) = "Iteraciones"
    fgRfnIiteraciones.TextArray(Fgit(0, 1)) = "Xi-1"
    fgRfnIiteraciones.TextArray(Fgit(0, 2)) = "Xi"
    fgRfnIiteraciones.TextArray(Fgit(0, 3)) = "f(Xi)"
    fgRfnIiteraciones.TextArray(Fgit(0, 4)) = "Error"


    For i = 0 To Val(TxtRfnIiteraciones.Text) - 1
        For j = 0 To 3
        fgRfnIiteraciones.TextMatrix(i + 1, j + 1) = raíz.regresariteraciones(i, j)
        Next j
     Next i
Case Else
'Presentación de los resultados
    fgRfnIresultados.Cols = 2
    fgRfnIresultados.Rows = 1
    fgRfnIresultados.FixedCols = 1
    fgRfnIresultados.TextArray(Fgr(0, 0)) = "Raíz estimada="
    fgRfnIresultados.ColWidth(1) = 8000
    valor = raíz.regresarraíz
    fgRfnIresultados.TextMatrix(0, 1) = valor

'Presentación de iteraciones
    fgRfnIiteraciones.Cols = 6
    fgRfnIiteraciones.Rows = Val(TxtRfnIiteraciones.Text) + 1
    fgRfnIiteraciones.FixedRows = 1
    fgRfnIiteraciones.FixedCols = 1
    For i = 0 To 5
    fgRfnIiteraciones.ColAlignmentFixed(i) = 4
    fgRfnIresultados.ColWidth(i) = 2500
    Next i
```

```
   For i = 0 To Val(TxtRfnliteraciones.Text)
   fgRfnliteraciones.TextArray(Fgit(i, 0)) = i
   Next i

   fgRfnliteraciones.TextArray(Fgit(0, 0)) = "Iteraciones"
   fgRfnliteraciones.TextArray(Fgit(0, 1)) = "X-1"
   fgRfnliteraciones.TextArray(Fgit(0, 2)) = "Xo"
   fgRfnliteraciones.TextArray(Fgit(0, 3)) = "X1"
   fgRfnliteraciones.TextArray(Fgit(0, 4)) = "f(x)"
   fgRfnliteraciones.TextArray(Fgit(0, 5)) = "Error"

   For i = 0 To Val(TxtRfnliteraciones.Text) - 1
      For j = 0 To 4
      fgRfnliteraciones.TextMatrix(i + 1, j + 1) = raíz.regresariteraciones(i, j)
      Next j
    Next i
End Select
Exit Sub
err1:

End Sub

'Conexion a la BDD
Private Sub PLConexion(parBDD As Boolean)
Dim VTCad As String
Dim VTBase As Database
Dim VTcadena As String
   VTcadena = App.Path & "/Seguridad.mdb"
   Set VLConexion = New ADODB.Connection
   If parBDD = False Then
      VTCad = "Provider=SQLOLEDB.1;Integrated Security=SSPI;Persist Security
Info=False;Initial Catalog=Seguridad"
   Else
      VTCad = "Provider=Microsoft.Jet.OLEDB.4.0;Data Source='" & VTcadena &
"';Persist Security Info=False"
   End If
   VLConexion.Open VTCad
End Sub

Private Sub PLGuardarBDD()
   Dim VTCad As String
   Dim VTReg As ADODB.Recordset
   Dim VTID As Integer
   Dim VB As Integer
On Error GoTo err1
   PLVerificaNombre Trim(VLNombre)
```

```
   '******* APROXIMACIONES SUCESIVAS, MIUB
   If tipoalg = 1 Then
      If VLVerifica = True Then   'Si existe un nombre con el mismo nombre actualiza
         VB = MsgBox("El nombre de ese trabajo ya existe, desea reemplazarlo?",
vbYesNo, "Guardar Trabajo Como")
         If VB = 6 Then
            VTCad = "update aproximaciones set " _
                  & " fx= '" & Trim(TxtRfnlfunción.Text) & "', Xo='" & Trim(TxtRfnla.Text)
& "', " _
                  & " h='" & Trim(TxtRfnlb.Text) & "', Itera='" &
Trim(TxtRfnliteraciones.Text) & "' " _
                  & " where Nombre = '" & VLNombre & "'"
            VLConexion.Execute (VTCad)
         End If
      Else    'Guarda el trabajo
         VTID = PLRecuperaCodigo
         VTCad = "insert into aproximaciones values " _
               & "(" & CInt(VTID) & ", " _
               & " '" & TxtRfnlfunción.Text & "', '" & TxtRfnla.Text & "', " _
               & " '" & TxtRfnlb.Text & "', '" & TxtRfnliteraciones.Text & "', " _
               & " '" & Trim(VLNombre) & "', " & 1 & ", " & CInt(tipoalg) & ")"
               '& " '" & Trim(VLNombre) & "', " & CInt(VGIDUsuario) & ", " & CInt(tipoalg
) & ")"
         VLConexion.Execute (VTCad)
         MsgBox "Trabajo guardado exitosamente", vbInformation, "Guardar trabajo
como"
      End If
   '******* MIU
   ElseIf tipoalg = 2 Or tipoalg = 3 Then
      If VLVerifica = True Then   'Si existe un nombre con el mismo nombre actualiza
         VB = MsgBox("El nombre de ese trabajo ya existe, desea reemplazarlo?",
vbYesNo, "Guardar Trabajo Como")
         If VB = 6 Then
            VTCad = "update MIU set " _
                  & " fx= '" & Trim(TxtRfnlfunción.Text) & "', a='" & Trim(TxtRfnla.Text) &
"', " _
                  & " b='" & Trim(TxtRfnlb.Text) & "', Imax='" &
Trim(TxtRfnliteraciones.Text) & "' " _
                  & " Tol='" & Trim(TxtRfnltolerancia.Text) & "' " _
                  & " where Nombre = '" & VLNombre & "' and tipo = " & tipoalg
            VLConexion.Execute (VTCad)
         End If
      Else    'Guarda el trabajo
         VTID = PLRecuperaCodigo
         VTCad = "insert into MIU values " _
               & "(" & CInt(VTID) & ", " _
```

```
            & " '" & TxtRfnlfunción.Text & "', '" & TxtRfnla.Text & "', " _
            & " '" & TxtRfnlb.Text & "', '" & TxtRfnliteraciones.Text & "', " _
            & " '" & TxtRfnltolerancia.Text & "', " _
            & " '" & Trim(VLNombre) & "', " & 1 & ", " & CInt(tipoalg) & ")"
            '& " '" & Trim(VLNombre) & "', " & CInt(VGIDUsuario) & ", " & CInt(tipoalg
) & ")"
        VLConexion.Execute (VTCad)
        MsgBox "Trabajo guardado exitosamente", vbInformation, "Guardar trabajo
como"
      End If
   '******* MIPF
   ElseIf tipoalg = 4 Or tipoalg = 5 Or tipoalg = 6 Then
      If VLVerifica = True Then   'Si existe un nombre con el mismo nombre actualiza
        VB = MsgBox("El nombre de ese trabajo ya existe, desea reemplazarlo?",
vbYesNo, "Guardar Trabajo Como")
        If VB = 6 Then
          If tipoalg = 4 Then
            VTCad = "update MIPF set " _
                & " gx_fxseg='" & Trim(TxtRfnlderivada.Text) & "', " _
                & " Xo='" & Trim(TxtRfnla.Text) & "', " _
                & " Imax='" & Trim(TxtRfnliteraciones.Text) & "', " _
                & " Tol='" & Trim(TxtRfnltolerancia.Text) & "' " _
                & " where Nombre = '" & VLNombre & "' and tipo = " & tipoalg
          ElseIf tipoalg = 5 Then
            VTCad = "update MIPF set " _
                & " fxprim= '" & Trim(TxtRfnlfunción.Text) & "', gx_fxseg= '" &
Trim(TxtRfnlderivada.Text) & "', " _
                & " Xo='" & Trim(TxtRfnla.Text) & "', " _
                & " Imax='" & Trim(TxtRfnliteraciones.Text) & "', " _
                & " Tol='" & Trim(TxtRfnltolerancia.Text) & "' " _
                & " where Nombre = '" & VLNombre & "' and tipo = " & tipoalg
          ElseIf tipoalg = 6 Then
            VTCad = "update MIPF set " _
                & " fxprim= '" & Trim(TxtRfnlfunción.Text) & "', Xo='" &
Trim(TxtRfnla.Text) & "', " _
                & " X1='" & Trim(TxtRfnlb.Text) & "', Imax='" &
Trim(TxtRfnliteraciones.Text) & "', " _
                & " Tol='" & Trim(TxtRfnltolerancia.Text) & "' " _
                & " where Nombre = '" & VLNombre & "' and tipo = " & tipoalg
          End If
          VLConexion.Execute (VTCad)
        End If
      Else   'Guarda el trabajo
        VTID = PLRecuperaCodigo
        VTCad = "insert into MIPF values " _
            & "(" & CInt(VTID) & ", " _
```

```
                & " '" & TxtRfnlfunción.Text & "', '" & TxtRfnlderivada.Text & "', " _
                & "'" & TxtRfnla.Text & "', " _
                & "'" & TxtRfnliteraciones.Text & "', '" & TxtRfnltolerancia.Text & "', " _
                & "'" & TxtRfnlb.Text & "', " _
                & "'" & Trim(VLNombre) & "', " & 1 & ", " & CInt(tipoalg) & ")"
                '& " '" & Trim(VLNombre) & "', " & CInt(VGIDUsuario) & ", " & CInt(tipoalg
) & ")"
          VLConexion.Execute (VTCad)
          MsgBox "Trabajo guardado exitosamente", vbInformation, "Guardar trabajo
como"
       End If

   End If



Exit Sub
err1:
   If err.Number <> 0 Then
      MsgBox "Se ha producido el siguiente error: " & err.Description & "." _
            & "Por favor consultar con el administrador", vbCritical
   End If
End Sub

Private Sub PLVerificaNombre(parNombre As String)
Dim VTReg As ADODB.Recordset
Dim VTCad As String
   If tipoalg = 1 Then
      VTCad = "select Nombre from aproximaciones where Nombre = '" & parNombre
& "' and tipo = " & tipoalg
      Set VTReg = VLConexion.Execute(VTCad)
      With VTReg
         If Not VTReg.EOF Then
            .MoveFirst
            Do Until .EOF
               If Trim(VTReg(0)) = parNombre Then
                  VLVerifica = True
                  Exit Do
               Else
                  VLVerifica = False
               End If
               .MoveNext
            Loop
         Else
            VLVerifica = False
         End If
```

```
        End With
    '****** MIU
    ElseIf tipoalg = 2 Or tipoalg = 3 Then
        VTCad = "select Nombre from MIU where Nombre = '" & parNombre & "' and
tipo = " & tipoalg
        Set VTReg = VLConexion.Execute(VTCad)
        With VTReg
            If Not VTReg.EOF Then
                .MoveFirst
                Do Until .EOF
                    If Trim(VTReg(0)) = parNombre Then
                        VLVerifica = True
                        Exit Do
                    Else
                        VLVerifica = False
                    End If
                    .MoveNext
                Loop
            Else
                VLVerifica = False
            End If
        End With
    '****** MIPF
    ElseIf tipoalg = 4 Or tipoalg = 5 Or tipoalg = 6 Then
        VTCad = "select Nombre from MIPF where Nombre = '" & parNombre & "' and
tipo = " & tipoalg
        Set VTReg = VLConexion.Execute(VTCad)
        With VTReg
            If Not VTReg.EOF Then
                .MoveFirst
                Do Until .EOF
                    If Trim(VTReg(0)) = parNombre Then
                        VLVerifica = True
                        Exit Do
                    Else
                        VLVerifica = False
                    End If
                    .MoveNext
                Loop
            Else
                VLVerifica = False
            End If
        End With
    End If

End Sub
```

```
Private Function PLRecuperaCodigo() As Integer
   Dim VTCad As String
   Dim VTReg As ADODB.Recordset
On Error GoTo err1
   '******** APROXIMACIONES
   If tipoalg = 1 Then
      VTCad = "select max(IDA) from Aproximaciones where tipo = " & tipoalg
      Set VTReg = VLConexion.Execute(VTCad)
      If Not IsNull(VTReg(0)) Then
         PLRecuperaCodigo = VTReg(0) + 1
      Else
         PLRecuperaCodigo = 1
      End If
   '******** MIU
   ElseIf tipoalg = 2 Or tipoalg = 3 Then
      VTCad = "select max(IDMIU) from MIU"
      Set VTReg = VLConexion.Execute(VTCad)
      If Not IsNull(VTReg(0)) Then
         PLRecuperaCodigo = VTReg(0) + 1
      Else
         PLRecuperaCodigo = 1
      End If
   '******** MIPF
   ElseIf tipoalg = 4 Or tipoalg = 5 Or tipoalg = 6 Then
      VTCad = "select max(IDMIPF) from MIPF"
      Set VTReg = VLConexion.Execute(VTCad)
      If Not IsNull(VTReg(0)) Then
         PLRecuperaCodigo = VTReg(0) + 1
      Else
         PLRecuperaCodigo = 1
      End If
   End If

Exit Function
err1:

End Function

Private Sub PLRecupera()
   Dim VTCad As String
   Dim VTReg As ADODB.Recordset
On Error GoTo err1
   '******** APROXIMACIONES SUCESIVAS
   If tipoalg = 1 Then
      VTCad = "select * from Aproximaciones where Nombre = '" & VLNombreTrabajo
& "'"
```

```
      Set VTReg = VLConexion.Execute(VTCad)
      If Not VTReg.EOF Then
         TxtRfnlfunción.Text = Trim(VTReg(1))
         TxtRfnla.Text = Trim(VTReg(2))
         TxtRfnlb.Text = Trim(VTReg(3))
         TxtRfnliteraciones.Text = Trim(VTReg(4))
      End If
   '******** MIU
   ElseIf tipoalg = 2 Or tipoalg = 3 Then
      VTCad = "select * from MIU where Nombre = '" & VLNombreTrabajo & "'"
      Set VTReg = VLConexion.Execute(VTCad)
      If Not VTReg.EOF Then
         TxtRfnlfunción.Text = Trim(VTReg(1))
         TxtRfnla.Text = Trim(VTReg(2))
         TxtRfnlb.Text = Trim(VTReg(3))
         TxtRfnliteraciones.Text = Trim(VTReg(4))
         TxtRfnltolerancia.Text = Trim(VTReg(5))
      End If
   '******** MIPF
   ElseIf tipoalg = 4 Or tipoalg = 5 Or tipoalg = 6 Then
      VTCad = "select * from MIPF where Nombre = '" & VLNombreTrabajo & "'"
      Set VTReg = VLConexion.Execute(VTCad)
      If Not VTReg.EOF Then
         TxtRfnlfunción.Text = Trim(VTReg(1))
         TxtRfnlderivada.Text = Trim(VTReg(2))
         TxtRfnla.Text = Trim(VTReg(3))
         TxtRfnlb.Text = Trim(VTReg(6))
         TxtRfnliteraciones.Text = Trim(VTReg(4))
         TxtRfnltolerancia.Text = Trim(VTReg(5))
      End If
   End If

Exit Sub
err1:

End Sub

Private Sub PLMascaraImprime()
Dim i, j As Integer
   Me.BackColor = &HFFFFFF:   Me.BorderStyle = 0
   lblprin.Caption = Me.Caption
   fraMascara.Visible = True
   MnuArchivo.Visible = False: MnuAyuda.Visible = False
   Rfnlalgotirmos.Visible = False: Rfnlopciones.Visible = False
   SSTab1.BackColor = &HFFFFFF
   SSTab1.Visible = False
```

```
    txt1.BorderStyle = 0:   txt2.BorderStyle = 0
    txt3.BorderStyle = 0:   txt4.BorderStyle = 0
    txt5.BorderStyle = 0:   txt6.BorderStyle = 0
    fgRfnlresultadosl.Visible = True
    lbl7.Visible = True
    LblRfnlresultados.Visible = False
    fgRfnlresultados.Visible = False
    'For i = 0 To 1
       For j = 0 To 1
          fgRfnlresultadosl.TextMatrix(0, j) = fgRfnlresultados.TextMatrix(0, j)
       Next j
    'Next i
    CmdRfnlcalcular.Visible = False
    CmdRfnliteraciones.Visible = False
    ShaRfnlresultados.Visible = False
    If tipoalg = 3 Or tipoalg = 4 Or tipoalg = 5 Or tipoalg = 6 Then
       CmdRfnlgraficar.Visible = False
    End If
End Sub

Private Sub PLMascaraOriginal()
Dim j As Integer
    fraMascara.Visible = False: Me.BorderStyle = 0
    MnuArchivo.Visible = True: MnuAyuda.Visible = True
    Rfnlalgotirmos.Visible = True: Rfnlopciones.Visible = True
    Me.BackColor = &H8000000A
    SSTab1.BackColor = &H8000000A
    SSTab1.Visible = True
    txt1.BorderStyle = 1:   txt2.BorderStyle = 1
    txt3.BorderStyle = 1:   txt4.BorderStyle = 1
    txt5.BorderStyle = 1:   txt6.BorderStyle = 1
    fgRfnlresultadosl.Visible = False
    lbl7.Visible = True
    LblRfnlresultados.Visible = True
    fgRfnlresultados.Visible = True
       For j = 0 To 1
          fgRfnlresultadosl.TextMatrix(0, j) = ""
       Next j
    CmdRfnlcalcular.Visible = True
    CmdRfnliteraciones.Visible = True
    ShaRfnlresultados.Visible = True
    If tipoalg = 3 Or tipoalg = 4 Or tipoalg = 5 Or tipoalg = 6 Then
       CmdRfnlgraficar.Visible = True
    End If
End Sub
```

```
Private Sub PLPasaValores()
ArchivoImprimir.Enabled = True
   txt1.Text = TxtRfnlfunción.Text
   txt2.Text = TxtRfnlderivada.Text
   txt3.Text = TxtRfnla.Text
   txt4.Text = TxtRfnlb.Text
   txt5.Text = TxtRfnliteraciones.Text
   txt6.Text = TxtRfnltolerancia.Text
End Sub


Private Sub TxtRfnla_KeyPress(KeyAscii As Integer)
'filtra la entrada de datos que no sean números incluye el signo - y el punto
  If (KeyAscii < 48 Or KeyAscii > 57) Then
      If (KeyAscii = 45) Then
      Exit Sub
      End If
      If (KeyAscii <> 46) Then
         If (KeyAscii <> 8) Then
         KeyAscii = 0
         End If
      End If
   End If
  If KeyAscii = Asc(vbCr) Then ' Elimina los retornos para quitar los pitidos.
  KeyAscii = 0
  End If
End Sub


Private Sub TxtRfnlb_KeyPress(KeyAscii As Integer)
'filtra la entrada de datos que no sean números incluye el signo - y el punto
  If (KeyAscii < 48 Or KeyAscii > 57) Then
      If (KeyAscii = 45) Then
      Exit Sub
      End If
      If (KeyAscii <> 46) Then
         If (KeyAscii <> 8) Then
         KeyAscii = 0
         End If
      End If
   End If
  If KeyAscii = Asc(vbCr) Then ' Elimina los retornos para quitar los pitidos.
  KeyAscii = 0
  End If
```

```
End Sub



Private Sub TxtRfnliteraciones_Change()
Dim mensaje As String
If Val(TxtRfnliteraciones.Text) > 32767 Then
mensaje = MsgBox("Fuera del rango", 48, "Error en el ingreso")
End If
End Sub

Private Sub TxtRfnliteraciones_KeyPress(KeyAscii As Integer)
'filtra la entrada de datos que no sean números enteros y positivos
  If (KeyAscii < 48 Or KeyAscii > 57) Then

        If (KeyAscii <> 8) Then
        KeyAscii = 0
        End If

   End If
  If KeyAscii = Asc(vbCr) Then ' Elimina los retornos para quitar los pitidos.
  KeyAscii = 0
  End If
End Sub

Private Sub TxtRfnltolerancia_Change()
Dim mensaje As String
If (Val(TxtRfnltolerancia) < -32768 Or Val(TxtRfnltolerancia) > 32767) Then
mensaje = MsgBox("Fuera del rango", 48, "Error en el ingreso")
End If
End Sub

Private Sub TxtRfnltolerancia_KeyPress(KeyAscii As Integer)
'filtra la entrada de datos que no sean números no incluye signo - y si el punto
  If (KeyAscii < 48 Or KeyAscii > 57) Then
      If (KeyAscii <> 46) Then
         If (KeyAscii <> 8) Then
         KeyAscii = 0
         End If
      End If
   End If
  If KeyAscii = Asc(vbCr) Then ' Elimina los retornos para quitar los pitidos.
  KeyAscii = 0
  End If
End Sub
```

```
Private Sub Sincolor()
altoSstab = SSTab1.Height
anchoSstab = SSTab1.Width
arribaSstab = SSTab1.Top
izquierdaSstab = SSTab1.Left
colororg = Me.BackColor
SSTab1.Height = SSTab1.Height / 4
SSTab1.Width = SSTab1.Width / 4
SSTab1.Top = SSTab1.Top + 800
SSTab1.Left = SSTab1.Left + 500
altoResul = FraResul.Height
anchoResul = FraResul.Width
arribaResul = FraResul.Top
izquierdaResul = FraResul.Left
anchoFgresul = Me.fgRfnlresultados.Width
altoFgresul = Me.fgRfnlresultados.Height
altoitera = Me.fgRfnliteraciones.Height
anchoitera = Me.fgRfnliteraciones.Width
arribaItera = Me.fgRfnliteraciones.Top
izquierdaItera = Me.fgRfnliteraciones.Left
arribaP = Me.FraP.Top
Me.fgRfnliteraciones.ScrollBars = 0
Me.FraP.Top = 0
Me.FraResul.Top = Me.FraP.Height + 1
Me.FraResul.Left = Me.FraP.Left
Me.FraResul.Height = Me.FraResul.Height + 500
Me.FraResul.Width = Me.FraResul.Width + 7000
Me.fgRfnlresultados.Height = Me.fgRfnlresultados.Height + 600
Me.fgRfnlresultados.Width = Me.fgRfnlresultados.Width + 7000
Me.FraResul.BackColor = &H80000009
Me.fgRfnliteraciones.Top = Me.FraResul.Top + Me.FraResul.Height + 1
Me.fgRfnliteraciones.Width = Me.FraResul.Width
Me.fgRfnliteraciones.Left = Me.FraP.Left
Me.fgRfnliteraciones.Height = Me.fgRfnliteraciones.Height + 100
LblNomalg.BackColor = &H80000009
Me.BackColor = &H80000009
Me.LblRfnla.BackColor = &H80000009
Me.LblRfnlb.BackColor = &H80000009
Me.LblRfnlderivada.BackColor = &H80000009
Me.LblRfnlfunción.BackColor = &H80000009
Me.LblRfnliteraciones.BackColor = &H80000009
Me.LblRfnlresultados.BackColor = &H80000009
Me.LblRfnltitulo.BackColor = &H80000009
Me.LblRfnltolerancia.BackColor = &H80000009
Me.FraRfnlerror.BackColor = &H80000009
Me.FraRfnlerror.BorderStyle = 0
```

```
Me.FraRfnlimprimir.BackColor = &H80000009
Me.FraRfnlitera.BackColor = &H80000009
Me.FraRfnlitera.BorderStyle = 0
Me.FraP.BackColor = &H80000009
Me.ShaRfnlresultados.Visible = False
Me.TxtRfnla.BorderStyle = 0
Me.TxtRfnlb.BorderStyle = 0
Me.TxtRfnlderivada.BorderStyle = 0
Me.TxtRfnlfunción.BorderStyle = 0
Me.TxtRfnliteraciones.BorderStyle = 0
Me.TxtRfnltolerancia.BorderStyle = 0
Me.fgRfnlresultados.BackColorBkg = &H80000009
Me.fgRfnlresultados.BackColorFixed = &H80000009
'Me.fgRfnlresultados.BorderStyle = 0
Me.fgRfnliteraciones.BackColorBkg = &H80000009
Me.fgRfnliteraciones.BackColorFixed = &H80000009
'Me.fgRfnliteraciones.BorderStyle = 0
Me.CmdRfnlcalcular.Visible = False
Me.CmdRfnliteraciones.Visible = False
Me.BorderStyle = 0
Me.Label1.Visible = False
Me.FraRfnlimprimir.BorderStyle = 1
Me.LblRfnltitulo.Visible = False
Me.OptRfnlef.BackColor = &H80000009
Me.OptRfnlex.BackColor = &H80000009
FraRfnlSintaxis.Visible = False
LblAviso.Visible = False
End Sub

Private Sub Concolor()
SSTab1.Height = altoSstab
SSTab1.Width = anchoSstab
SSTab1.Top = arribaSstab
SSTab1.Left = izquierdaSstab
Me.FraResul.Height = altoResul
Me.FraResul.Width = anchoResul
Me.FraResul.Top = arribaResul
Me.FraResul.Left = izquierdaResul
Me.fgRfnlresultados.Width = anchoFgresul
Me.fgRfnlresultados.Height = altoFgresul
Me.fgRfnliteraciones.Top = arribaItera
Me.fgRfnliteraciones.Width = anchoitera
Me.fgRfnliteraciones.Height = altoitera
Me.fgRfnliteraciones.Left = izquierdaItera
Me.fgRfnliteraciones.ScrollBars = 3
Me.FraP.Top = arribaP
```

```
Me.BackColor = colororg
Me.LblRfnla.BackColor = colororg
Me.LblRfnlb.BackColor = colororg
Me.LblRfnlderivada.BackColor = colororg
Me.LblRfnlfunción.BackColor = colororg
Me.LblRfnliteraciones.BackColor = colororg
Me.LblRfnlresultados.BackColor = colororg
Me.LblRfnltitulo.BackColor = colororg
Me.LblRfnltolerancia.BackColor = colororg
Me.FraRfnlerror.BackColor = colororg
Me.FraRfnlerror.BorderStyle = 1
Me.FraRfnlimprimir.BackColor = colororg
Me.FraResul.BackColor = colororg
Me.FraRfnlitera.BackColor = colororg
Me.FraRfnlitera.BorderStyle = 1
Me.FraP.BackColor = colororg
Me.ShaRfnlresultados.Visible = True
Me.TxtRfnla.BorderStyle = 1
Me.TxtRfnlb.BorderStyle = 1
Me.TxtRfnlderivada.BorderStyle = 1
Me.TxtRfnlfunción.BorderStyle = 1
Me.TxtRfnliteraciones.BorderStyle = 1
Me.TxtRfnltolerancia.BorderStyle = 1
Me.fgRfnlresultados.BackColorBkg = colororg
Me.fgRfnlresultados.BackColorFixed = colororg
Me.fgRfnlresultados.BorderStyle = 1
Me.fgRfnliteraciones.BackColorBkg = colororg
Me.fgRfnliteraciones.BackColorFixed = colororg
Me.fgRfnliteraciones.BorderStyle = 1
Me.CmdRfnlcalcular.Visible = True
Me.CmdRfnliteraciones.Visible = True
Me.BorderStyle = 2
Me.Label1.Visible = True
Me.FraRfnlimprimir.BorderStyle = 0
Me.LblRfnltitulo.Visible = True
Me.OptRfnlef.BackColor = colororg
Me.OptRfnlex.BackColor = colororg
LblNomalg.BackColor = colororg
FraRfnlSintaxis.Visible = True
If tipoalg = 4 Then
LblAviso.Visible = True
End If
End Sub
```

Módulo de clase RFNL

```
Option Explicit
Dim Evaluar As New McsExcel
Dim f As String
Dim g As String
Dim A As Double
Dim b As Double
Dim iteramax As Integer
Dim tol As Double
Dim ex As Boolean
Dim raíz As Double
Dim Resultadorfnl() As Double


Public Property Let SeteraRfnl(Fun As String, gx As String, ai As Double, bi As
Double, itera As Integer _
, toler As Double, err As Boolean)

f = Fun
g = gx
A = ai
b = bi
iteramax = itera
tol = toler
ex = err
End Property
Public Property Get regresarraíz() As Variant
regresarraíz = raíz
End Property
Public Property Get regresariteraciones() As Variant
regresariteraciones = Resultadorfnl
End Property


Public Sub AproximacionesSucesivas()
Dim i As Integer
Dim k As Integer
Dim punto As Double
Dim fa As Double
Dim fpunto As Double
Dim mensaje As String
On Error GoTo err1
ReDim Resultadorfnl(0 To iteramax - 1, 0 To 3) As Double
'Evaluamos la función en el punto inicial a
Evaluar.Tipox = False
Evaluar.PtX = A
```

```
Evaluar.fa = f
Evaluar.evaluarf
fa = Evaluar.Fdy
If fa = 0 Then
mensaje = MsgBox("El valor inicial es una raíz de la función", 48, "Raiz de la
Función")
Exit Sub
Else
 For k = 1 To iteramax
 punto = A + k * b
 Evaluar.Tipox = False
 Evaluar.PtX = punto
 Evaluar.fa = f
 Evaluar.evaluarf
 fpunto = Evaluar.Fdy
 'Guardamos las evaluaciones en la matriz de resultados
 Resultadorfnl(k - 1, 0) = A
 Resultadorfnl(k - 1, 1) = fa
 Resultadorfnl(k - 1, 2) = punto
 Resultadorfnl(k - 1, 3) = fpunto
 'comprobamos si cambia de signo la función en el intervalo
 If fa * fpunto <= 0 Then
 If fa * fpunto = 0 Then
 mensaje = MsgBox("El limite superior del intervalo es una raíz de la función", 48,
"Raíz")
 End If
 raíz = punto
 Exit Sub
 End If
 Next k
 mensaje = MsgBox("No se ha encontrado el intervalo donde esta la raíz,las
iteraciones se han superado; Sugerencia: incremente el número de iteraciones", 48,
"Iteraciónes ")
FrmRfnl.borrar = True
End If
Exit Sub
err1:
If err.Number <> 0 Then
   mensaje = MsgBox("Parámetros del algoritmo incompletos", 48, "Error en el
ingreso")
   End If
End Sub

Public Sub Bisección()
Dim i As Integer
Dim j As Integer
```

```
Dim k As Integer
Dim prueba As Double    'valor del error para compararlo
Dim mensaje As String
Dim xm As Double
On Error GoTo err1
If A = b Then
mensaje = MsgBox("Debe ingresar un intervalo", 16, "Error en el ingreso")
Exit Sub
End If
ReDim Resultadorfnl(0 To iteramax - 1, 0 To 4) As Double
k = 1
Do
xm = (A + b) / 2
If (fevalx(A) * fevalx(xm)) = 0 Then
'mensaje = MsgBox("Se ha encontrado la raíz exacta", 48, "Exito")
Resultadorfnl(k - 1, 0) = A
Resultadorfnl(k - 1, 1) = b
Resultadorfnl(k - 1, 2) = xm
Resultadorfnl(k - 1, 3) = fevalx(xm)
Resultadorfnl(k - 1, 4) = 0
Exit Sub
Else
  If (fevalx(A) * fevalx(xm)) < 0 Then
   If ex = True Then
   prueba = Abs((xm - A) / xm)
   Resultadorfnl(k - 1, 4) = prueba
   Else
   prueba = Abs(fevalx(xm))
   Resultadorfnl(k - 1, 4) = prueba
   End If
   Resultadorfnl(k - 1, 0) = A
   Resultadorfnl(k - 1, 1) = b
   Resultadorfnl(k - 1, 2) = xm
   Resultadorfnl(k - 1, 3) = fevalx(xm)
   b = xm
  Else
   If ex = True Then
   prueba = Abs((xm - A) / xm)
   Resultadorfnl(k - 1, 4) = prueba
   Else
   prueba = Abs(fevalx(xm))
   Resultadorfnl(k - 1, 4) = prueba
   End If
   Resultadorfnl(k - 1, 0) = A
   Resultadorfnl(k - 1, 1) = b
   Resultadorfnl(k - 1, 2) = xm
```

```
   Resultadorfnl(k - 1, 3) = fevalx(xm)
   A = xm
  End If
End If
k = k + 1
Loop Until (k > iteramax Or prueba <= tol)
raíz = Resultadorfnl(k - 2, 2)
If k > iteramax Then
mensaje = MsgBox("Número de iteraciones máximas superado, no fue posible
encontrar la respuesta; Sugerencia: incremente el número de iteraciones", 48,
"Iteraciones")
FrmRfnl.borrar = True
End If
Exit Sub
err1:
If err.Number <> 0 Then
   mensaje = MsgBox("Parámetros del algoritmo incompletos", 48, "Error en el
ingreso")
   End If
End Sub

Public Sub FalsePosición()
Dim i As Integer
Dim j As Integer
Dim k As Integer
Dim prueba As Double    'valor del error para compararlo
Dim mensaje As String
Dim xm As Double

On Error GoTo err1
If A = b Then
mensaje = MsgBox("Debe ingresar un intervalo", 16, "Error en el ingreso")
Exit Sub
End If
ReDim Resultadorfnl(0 To iteramax - 1, 0 To 4) As Double
k = 1
Do
xm = (A * fevalx(b) - b * fevalx(A)) / (fevalx(b) - fevalx(A))
If (fevalx(A) * fevalx(xm)) = 0 Then
mensaje = MsgBox("Se ha encontrado la raíz exacta", 48, "Exito")
Resultadorfnl(k - 1, 0) = A
Resultadorfnl(k - 1, 1) = b
Resultadorfnl(k - 1, 2) = xm
Resultadorfnl(k - 1, 3) = fevalx(xm)
Resultadorfnl(k - 1, 4) = 0
Exit Sub
```

```
Else
  If (fevalx(A) * fevalx(xm)) < 0 Then
   If ex = True Then
   prueba = Abs((xm - A) / xm)
   Resultadorfnl(k - 1, 4) = prueba
   Else
   prueba = Abs(fevalx(xm))
   Resultadorfnl(k - 1, 4) = prueba
   End If
   Resultadorfnl(k - 1, 0) = A
   Resultadorfnl(k - 1, 1) = b
   Resultadorfnl(k - 1, 2) = xm
   Resultadorfnl(k - 1, 3) = fevalx(xm)
   b = xm
  Else
   If ex = True Then
   prueba = Abs((xm - A) / xm)
   Resultadorfnl(k - 1, 4) = prueba
   Else
   prueba = Abs(fevalx(xm))
   Resultadorfnl(k - 1, 4) = prueba
   End If
   Resultadorfnl(k - 1, 0) = A
   Resultadorfnl(k - 1, 1) = b
   Resultadorfnl(k - 1, 2) = xm
   Resultadorfnl(k - 1, 3) = fevalx(xm)
   A = xm
  End If
End If
k = k + 1
Loop Until (k > iteramax Or prueba <= tol)
raíz = Resultadorfnl(k - 2, 2)

If k > iteramax Then
mensaje = MsgBox("Número de iteraciones máximas superado, no fue posible
encontrar la respuesta; Sugerencia: incremente el número de iteraciones", 48,
"Iteraciones")
FrmRfnl.borrar = True
End If

Exit Sub
err1:
If err.Number <> 0 Then
   mensaje = MsgBox("Parámetros del algoritmo incompletos", 48, "Error en el
ingreso")
   End If
```

```
End Sub

Public Sub Primerorden()

Dim k As Integer
Dim prueba As Double    'valor del error para compararlo
Dim mensaje As String
Dim xm As Double

On Error GoTo err1
ReDim Resultadorfnl(0 To iteramax - 1, 0 To 3) As Double
k = 1
'Inicia el algoritmo
Do
xm = gevalx(A)
'calculamos el error
   If ex = True Then
   prueba = Abs((xm - A) / xm)
   Resultadorfnl(k - 1, 3) = prueba
   Else
   prueba = Abs(gevalx(xm))
   Resultadorfnl(k - 1, 3) = prueba
   End If
Resultadorfnl(k - 1, 0) = A
Resultadorfnl(k - 1, 1) = xm
Resultadorfnl(k - 1, 2) = prueba
A = xm
k = k + 1
Loop Until (k > iteramax Or prueba <= tol)
raíz = Resultadorfnl(k - 2, 1)
If k > iteramax Then
mensaje = MsgBox("Número de iteraciones máximas superado, no fue posible
encontrar la respuesta; Sugerencia incremente el número de iteraciones", 48,
"Iteraciones")
FrmRfnl.borrar = True
End If
Exit Sub
err1:
If err.Number <> 0 Then
   mensaje = MsgBox("Parámetros del algoritmo incompletos", 48, "Error en el
ingreso")
   End If
End Sub

Public Sub NewtonRaphson()
Dim k As Integer
```

```
Dim prueba As Double   'valor del error para compararlo
Dim mensaje As String
Dim xm As Double

On Error GoTo err1
ReDim Resultadorfnl(0 To iteramax - 1, 0 To 3) As Double
k = 1
'Inicia el algoritmo
Do

xm = A - (fevalx(A) / gevalx(A))
'calculamos el error
   If ex = True Then
   prueba = Abs((xm - A) / xm)
   Resultadorfnl(k - 1, 3) = prueba
   Else
   prueba = Abs(fevalx(xm))
   Resultadorfnl(k - 1, 3) = prueba
   End If
Resultadorfnl(k - 1, 0) = A
Resultadorfnl(k - 1, 1) = xm
Resultadorfnl(k - 1, 2) = fevalx(xm)
A = xm
k = k + 1
Loop Until (k > iteramax Or prueba <= tol)
raíz = Resultadorfnl(k - 2, 1)

If k > iteramax Then
mensaje = MsgBox("Número de iteraciones máximas superado, no fue posible
encontrar la respuesta; Sugerencia : incremente el número de iteraciones", 48,
"Iteraciones")
FrmRfnl.borrar = True
End If
Exit Sub
err1:
If err.Number <> 0 Then
   mensaje = MsgBox("Parámetros del algoritmo incompletos", 48, "Error en el
ingreso")
   End If
End Sub

Public Sub Secante()
Dim k As Integer
Dim prueba As Double   'valor del error para compararlo
Dim mensaje As String
Dim xm As Double
```

```
On Error GoTo err1
ReDim Resultadorfnl(0 To iteramax - 1, 0 To 4) As Double
k = 1
'Inicia el algoritmo
Do

xm = b - ((fevalx(b) * (b - A)) / (fevalx(b) - fevalx(A)))
'calculamos el error
   If ex = True Then
   prueba = Abs((xm - b) / xm)
   Resultadorfnl(k - 1, 4) = prueba
   Else
   prueba = Abs(fevalx(xm))
   Resultadorfnl(k - 1, 4) = prueba
   End If
Resultadorfnl(k - 1, 0) = A
Resultadorfnl(k - 1, 1) = b
Resultadorfnl(k - 1, 2) = xm
Resultadorfnl(k - 1, 3) = fevalx(xm)
A = b
b = xm
k = k + 1
Loop Until (k > iteramax Or prueba <= tol)
raíz = Resultadorfnl(k - 2, 2)
If k > iteramax Then
mensaje = MsgBox("Número de iteraciones máximas superado, no fue posible
encontrar la respuesta; Sugerencia: incremente el número de iteraciones", 48,
"Iteraciones")
FrmRfnl.borrar = True
End If
Exit Sub
err1:
If err.Number <> 0 Then
   mensaje = MsgBox("Parámetros del algoritmo incompletos", 48, "Error en el
ingreso")
   End If
End Sub
' Funciones que manejan la evaluación de funciones ingresadas por teclado
'con ayuda de MscExcel
Private Function fevalx(X As Double) As Double
On Error GoTo err1
Evaluar.Tipox = False
Evaluar.PtX = X
Evaluar.fa = f
Evaluar.evaluarf
```

```
fevalx = Evaluar.Fdy
Exit Function
err1:
End Function
Private Function gevalx(X As Double) As Double
On Error GoTo err1
Evaluar.Tipox = False
Evaluar.PtX = X
Evaluar.fa = g
Evaluar.evaluarf
gevalx = Evaluar.Fdy
Exit Function
err1:

End Function
```

Calcular la integral de una función

Interfaz Integración

```
Option Explicit
'En Microsoft TechNet puedes encontrar este artículo:
'HOWTO: Use HTML Help API in a Visual Basic 5.0 Application
'PSS ID Number: Q183434
'
'Aunque la definición de la Enumeración y la primera declaración
'es de las news
'
'Htmlhelp consts
Private Enum HH_COMMAND
   HH_DISPLAY_TOPIC = &H0
   HH_HELP_FINDER = &H0       ' WinHelp equivalent
   HH_DISPLAY_TOC = &H1       ' not currently implemented
   HH_DISPLAY_INDEX = &H2     ' not currently implemented
   HH_DISPLAY_SEARCH = &H3    ' not currently implemented
   HH_SET_WIN_TYPE = &H4
   HH_GET_WIN_TYPE = &H5
   HH_GET_WIN_HANDLE = &H6
   HH_GET_INFO_TYPES = &H7    ' not currently implemented
   HH_SET_INFO_TYPES = &H8    ' not currently implemented
   HH_SYNC = &H9
   HH_ADD_NAV_UI = &HA        ' not currently implemented
   HH_ADD_BUTTON = &HB        ' not currently implemented
   HH_GETBROWSER_APP = &HC    ' not currently implemented
   HH_KEYWORD_LOOKUP = &HD
   HH_DISPLAY_TEXT_POPUP = &HE ' display string resource id
                               ' or text in a popup window
```

```
    HH_HELP_CONTEXT = &HF       ' display mapped numeric value
                                ' in dwData
    HH_TP_HELP_CONTEXTMENU      ' Text pop-up help, similar to
                                ' WinHelp's HELP_CONTEXTMENU.
    HH_TP_HELP_WM_HELP = &H11   ' text pop-up help, similar to
                                ' WinHelp's HELP_WM_HELP.
    HH_CLOSE_ALL = &H12         ' close all windows opened directly
                                ' or indirectly by the caller
    HH_ALINK_LOOKUP = &H13      ' ALink version of HH_KEYWORD_LOOKUP
End Enum

'HtmlHelp api call
'NOTA: Si se usa esta forma, hay que indicar el último parámetro
'     con la palabra ByVal delante...
'Private Declare Function HtmlHelp Lib "hhctrl.ocx" Alias "HtmlHelpA" _
    (ByVal hwndCaller As Long, ByVal pszFile As String, _
    ByVal uCommand As HH_COMMAND, dwData As Any) As Long
'Con esta funciona perfectamente
Private Declare Function HtmlHelp Lib "hhctrl.ocx" Alias "HtmlHelpA" _
    (ByVal hwndCaller As Long, ByVal pszFile As String, _
    ByVal uCommand As HH_COMMAND, ByVal dwData As Long) As Long

Dim exval As New McsExcel
Dim integración As New ClasIntegración
Dim tipoalg As Integer
Dim n As Integer
Dim titulos() As String
Dim arribaSstab
Dim izquierdaSstab
Dim anchoSstab
Dim altoSstab
Dim colororg
Private Acepta As Boolean
Private VLConexion As ADODB.Connection
Private VLRegistro As ADODB.Recordset
Private VLNombre As String
Private VLNombreTrabajo As String
Private VLVerifica As Boolean
Private VLBDD As Boolean
Private VLJacobiano As String
Private VLJacobianoMatriz(1 To 1000) As Double
Private VLIndice As Integer
Private VLRec As Boolean

Property Let Aceptar(parAceptar As Boolean)
    Acepta = parAceptar
```

```
End Property

Property Let Nombre(parNombre As String)
   VLNombre = parNombre
End Property

Property Let NombreTrabajo(parNombreTrabajo As String)
   VLNombreTrabajo = parNombreTrabajo
End Property


Private Sub CmdIntegracalcular_Click()
Dim mensaje As String
Dim verifica As Double


Select Case tipoalg

Case Is = 1
   If OptIntegrafunción.Value = True Then
      If (TxtIntegrafunción = "" Or TxtIntegrafunción = " " Or TxtIntegraa = "" Or
TxtIntegraa = " " Or TxtIntegrab = "" Or TxtIntegrab = " ") Then
         mensaje = MsgBox("Parámetros del algoritmo incompletos", 48, "Error en la
función")
         Exit Sub
      End If
   Else
      If (TxtIntegraEdit1 = "" Or TxtIntegraEdit1 = " ") Then
         mensaje = MsgBox("Parámetros del algoritmo incompletos", 48, "Error en la
función")
         Exit Sub
      End If
   End If
   exval.Tipox = False
   exval.fa = TxtIntegrafunción
   exval.PtX = 500
   exval.evaluarf
   verifica = exval.Fdy
   If verifica = 8121975 Then
   mensaje = MsgBox("Debe ingresar la función de acuerdo a la sintaxis
especificada", 16, "Error en el ingreso")
   Exit Sub
   End If

   n = Val(CmbIntegraN.Text)
   SetearIntegración
```

```
    If OptIntegrafunción = True Then
       integración.NewtonCotesfunción
       Else
       integración.NewtonCotesdatos
    End If
    IntegraPresentar
Case Is = 2

       If (TxtIntegrafunción = "" Or TxtIntegrafunción = " " Or TxtIntegraa = "" Or
TxtIntegraa = " " Or TxtIntegrab = "" Or TxtIntegrab = " " Or TxtIntegraN = "" Or
TxtIntegraN = " ") Then
          mensaje = MsgBox("Parámetros del algoritmo incompletos", 48, "Error en la
función")
          Exit Sub
       End If

    exval.Tipox = False
    exval.fa = TxtIntegrafunción
    exval.PtX = 500
    exval.evaluarf
    verifica = exval.Fdy
    If verifica = 8121975 Then
    mensaje = MsgBox("Debe ingresar la función de acuerdo a la sintaxis
especificada", 16, "Error en el ingreso")
    Exit Sub
    End If


    n = Val(TxtIntegraN)
    SetearIntegración
    integración.ReglaExtendidaTrapecio
    IntegraPresentar
Case Is = 3

       If (TxtIntegrafunción = "" Or TxtIntegrafunción = " " Or TxtIntegraa = "" Or
TxtIntegraa = " " Or TxtIntegrab = "" Or TxtIntegrab = " " Or TxtIntegraN = "" Or
TxtIntegraN = " ") Then
          mensaje = MsgBox("Parámetros del algoritmo incompletos", 48, "Error en la
función")
          Exit Sub
       End If

    exval.Tipox = False
    exval.fa = TxtIntegrafunción
    exval.PtX = 500
    exval.evaluarf
```

```
    verifica = exval.Fdy
    If verifica = 8121975 Then
    mensaje = MsgBox("Debe ingresar la función de acuerdo a la sintaxis
especificada", 16, "Error en el ingreso")
    Exit Sub
    End If

    n = Val(TxtIntegraN)
    SetearIntegración
    integración.ReglaExtendidaSimpson
    IntegraPresentar
Case Is = 4

        If (TxtIntegrafunción = "" Or TxtIntegrafunción = " " Or TxtIntegraa = "" Or
TxtIntegraa = " " Or TxtIntegrab = "" Or TxtIntegrab = " ") Then
            mensaje = MsgBox("Parámetros del algoritmo incompletos", 48, "Error en la
función")
            Exit Sub
        End If

    exval.Tipox = False
    exval.fa = TxtIntegrafunción
    exval.PtX = 500
    exval.evaluarf
    verifica = exval.Fdy
    If verifica = 8121975 Then
    mensaje = MsgBox("Debe ingresar la función de acuerdo a la sintaxis
especificada", 16, "Error en el ingreso")
    Exit Sub
    End If
    n = 1
    SetearIntegración
    integración.IntegraciónGaussiana
    IntegraPresentar
End Select

Integraimprimir.Enabled = True
End Sub

Private Sub Command1_Click()
    PLImprimir
End Sub

Private Sub Form_Load()
    Integraabrir.Enabled = False
    Integraguardarcomo.Enabled = False
```

```
    Integraimprimir.Enabled = False
    VGForma = "Integracion"
End Sub

Private Sub Integraabrir_Click()
    FrmRecuperaTrabajo.Show vbModal
    If Acepta = True Then
        PLConexion True
        PLRecupera
    End If
End Sub

Private Sub Integraayuda_Click()
'Así se llamaría para mostrar un tópico de la ayuda
    Dim h As Long

    'h = HtmlHelp(Me.hWnd, "Milton.hlp", HH_DISPLAY_TOPIC, 0&)

    h = HtmlHelp(Me.hWnd, "Milton.hlp", HH_HELP_CONTEXT, 2070&)
End Sub

Private Sub integracerrada_Click()
'seteamos los titulos a mostrar cuando se hace click en el CmbIntegraN
ReDim titulos(0 To 5) As String
titulos(0) = "Regla del Trapecio"
titulos(1) = "Regla de 1/3 de Simpson"
titulos(2) = " Regla de 3/8 de Simpson"
titulos(3) = "Regla de Boole"
titulos(4) = "Formula de orden superior"
titulos(5) = "Formula de orden superior"
'Seteamos el tipo de algoritmo elegido
tipoalg = 1
    Integraabrir.Enabled = False
    Integraguardarcomo.Enabled = False
    FrmRecuperaTrabajo.ver = tipoalg
Integraimprimir.Enabled = False
'Presentamos los controles necesarios
FraIntegratipo.Visible = True
OptIntegrafunción.Value = False
OptIntegrafunción.Visible = True
OptIntegradatos.Value = False
OptIntegradatos.Visible = True
LblIntegrafunción.Visible = False
TxtIntegrafunción.Text = " "
TxtIntegrafunción.Visible = False
LblIntegraN.Visible = False
```

```
TxtIntegraN.Text = " "
TxtIntegraN.Visible = False
CmbIntegraN.Visible = False
FraIntegraintervalo.Visible = False
LblIntegraa.Visible = False
TxtIntegraa.Text = " "
TxtIntegrab.Visible = False
LblIntegrab.Visible = False
TxtIntegrab.Text = " "
TxtIntegrab.Visible = False
CheckIntegraerror.Visible = False
FraIntegraerror.Visible = False
LblIntegrareal.Visible = False
TxtIntegrareal.Text = " "
TxtIntegrareal.Visible = False
LblIntegratitulo.Visible = False
LblIntegratitulo2.Visible = False
fgIntegradatos.Clear
fgIntegradatos.Visible = False
fgIntegraresultados.Clear
LblNomalg.Caption = "Integración de Newton-Cotes"
LblInteSintaxis.Caption = " Función= integración de una función; Datos= integración
de datos" & Chr(10) & "f(x)=función de la cual se obtendrá la integral exp(-
1*X^2)*(SIN(X)/7)" & Chr(10) & "Puede usar las funciones:
Ln(X);COS(X);SQRT(X);Log(X); Tan(X); PI(), el resto de funciones sigue la sintaxis
de Excel" & Chr(10) & "[a,b]= intervalo de integración" & Chr(10) & _
"Número de intervalos= representa el número de intervalos usados para la
integración n= 1,2,..,6; si n=1 Regla del trapecio, n=2 Regla de 1/3 de Simpson, etc"
& Chr(10) & "Calcular error= permite calcular el error absoluto si se conoce el valor
de la integral real"
End Sub

Private Sub Integraficar_Click()
Frmescalas.Show
End Sub

Private Sub Integraguardarcomo_Click()
   FrmNombre.Show vbModal
   If Acepta = True Then
      PLConexion True
      PLGuardarBDD
   End If
End Sub

Private Sub Integraimprimir_Click()
Dim EndTime As Date
```

```
    Sincolor

        EndTime = DateAdd("s", 1 / 1E+16, Now)
        Do Until Now > EndTime
          DoEvents
        Loop
    'Form1.Command3_Click
    Set Form1.Picture1.Picture = CaptureClient(Me)
    Form1.Visible = True
    FrmPrincipalalgoritmos.Visible = False
    Concolor
End Sub

Private Sub Integrasalir_Click()
Unload Me
End Sub

Private Sub Integratrapecio_Click()
'Seteamos el tipo de algoritmo elegido
tipoalg = 2
    Integraabrir.Enabled = True
    Integraguardarcomo.Enabled = True
    FrmRecuperaTrabajo.ver = tipoalg
Integraimprimir.Enabled = False
'Presentamos los controles necesarios
FraIntegratipo.Visible = False
OptIntegrafunción.Value = False
OptIntegrafunción.Visible = False
OptIntegradatos.Value = False
OptIntegradatos.Visible = False
LblIntegrafunción.Visible = True
TxtIntegrafunción.Text = " "
TxtIntegrafunción.Visible = True
LblIntegraN.Visible = True
TxtIntegraN.Text = " "
TxtIntegraN.Visible = True
CmbIntegraN.Visible = False
FraIntegraintervalo.Visible = True
LblIntegraa.Visible = True
TxtIntegraa.Text = " "
TxtIntegraa.Visible = True
LblIntegrab.Visible = True
TxtIntegrab.Text = " "
TxtIntegrab.Visible = True
LblIntegratitulo.Visible = False
CheckIntegraerror.Value = 0
```

```
CheckIntegraerror.Visible = True
FraIntegraerror.Visible = False
LblIntegrareal.Visible = False
TxtIntegrareal.Text = " "
TxtIntegrareal.Visible = False
LblIntegratitulo2.Caption = "Número de segmentos"
LblIntegratitulo2.Visible = True
fgIntegradatos.Clear
fgIntegradatos.Visible = False
fgIntegraresultados.Clear
LblNomalg.Caption = "Regla extendida del Trapecio"
LblInteSintaxis.Caption = "f(x)=función de la cual se obtendrá la integral exp(-
1*X^2)*(SIN(X)/7)" & Chr(10) & "Puede usar las funciones:
Ln(X);COS(X);SQRT(X);Log(X); Tan(X); PI(), el resto de funciones sigue la sintaxis
de Excel" & Chr(10) & "[a,b]= intervalo de integración" & Chr(10) & _
"N= Número de segmentos usados para la integración múltiple" & Chr(10) & "Calcular
error= permite calcular el error absoluto si se conoce el valor de la integral real"
End Sub
Private Sub Integrasimpson_Click()
'Seteamos el tipo de algoritmo elegido
tipoalg = 3
   Integraabrir.Enabled = True
   Integraguardarcomo.Enabled = True
   FrmRecuperaTrabajo.ver = tipoalg
Integraimprimir.Enabled = False
'Presentamos los controles necesarios
FraIntegratipo.Visible = False
OptIntegrafunción.Value = False
OptIntegrafunción.Visible = False
OptIntegradatos.Value = False
OptIntegradatos.Visible = False
LblIntegrafunción.Visible = True
TxtIntegrafunción.Text = " "
TxtIntegrafunción.Visible = True
LblIntegraN.Visible = True
TxtIntegraN.Text = " "
TxtIntegraN.Visible = True
CmbIntegraN.Visible = False
FraIntegraintervalo.Visible = True
LblIntegraa.Visible = True
TxtIntegraa.Text = " "
TxtIntegraa.Visible = True
LblIntegrab.Visible = True
TxtIntegrab.Text = " "
TxtIntegrab.Visible = True
CheckIntegraerror.Value = 0
```

```
CheckIntegraerror.Visible = True
FraIntegraerror.Visible = False
LblIntegrareal.Visible = False
TxtIntegrareal.Text = " "
TxtIntegrareal.Visible = False
LblIntegratitulo.Visible = False
LblIntegratitulo2.Caption = "Número de segmentos"
LblIntegratitulo2.Visible = True
fgIntegradatos.Clear
fgIntegradatos.Visible = False
fgIntegraresultados.Clear
LblNomalg.Caption = "Regla extendida de 1/3 de Simpson"
LblInteSintaxis.Caption = "f(x)=función de la cual se obtendrá la integral exp(-
1*X^2)*(SIN(X)/7)" & Chr(10) & "Puede usar las funciones:
Ln(X);COS(X);SQRT(X);Log(X); Tan(X); PI(), el resto de funciones sigue la sintaxis
de Excel" & Chr(10) & "[a,b]= intervalo de integración" & Chr(10) & _
"N= Número de segmentos usados para la integración múltiple" & Chr(10) & "Calcular
error= permite calcular el error absoluto si se conoce el valor de la integral real"
End Sub
Private Sub Integragaussiana_Click()
Dim i As Integer
'Seteamos el tipo de algoritmo elegido
tipoalg = 4
   Integraabrir.Enabled = True
   Integraguardarcomo.Enabled = True
   FrmRecuperaTrabajo.ver = tipoalg
Integraimprimir.Enabled = False
'Presentamos los controles necesarios
FraIntegratipo.Visible = False
OptIntegrafunción.Value = False
OptIntegrafunción.Visible = False
OptIntegradatos.Value = False
OptIntegradatos.Visible = False
LblIntegrafunción.Visible = True
TxtIntegrafunción.Text = " "
TxtIntegrafunción.Visible = True
LblIntegraN.Visible = False
TxtIntegraN.Text = " "
TxtIntegraN.Visible = False
CmbIntegraN.Visible = False
FraIntegraintervalo.Visible = True
LblIntegraa.Visible = True
TxtIntegraa.Text = " "
TxtIntegraa.Visible = True
LblIntegrab.Visible = True
TxtIntegrab.Text = " "
```

```
TxtIntegrab.Visible = True
CheckIntegraerror.Value = 0
CheckIntegraerror.Visible = True
FraIntegraerror.Visible = False
LblIntegrareal.Visible = False
TxtIntegrareal.Text = " "
TxtIntegrareal.Visible = False
LblIntegratitulo.Visible = False
LblIntegratitulo2.Visible = False
fgIntegradatos.Clear
fgIntegradatos.Visible = False
fgIntegraresultados.Clear
LblNomalg.Caption = "Integración  Gaussiana"
LblInteSintaxis.Caption = "f(x)=función de la cual se obtendrá la integral exp(-
1*X^2)*(SIN(X)/7)" & Chr(10) & "Puede usar las funciones:
Ln(X);COS(X);SQRT(X);Log(X); Tan(X); PI(), el resto de funciones sigue la sintaxis
de Excel" & Chr(10) & "[a,b]= intervalo de integración" & Chr(10) & "Calcular error=
permite calcular el error absoluto si se conoce el valor de la integral real"
End Sub


Private Sub OptIntegradatos_Click()
Dim i As Integer
Select Case tipoalg
Case Is = 1
FraIntegratipo.Visible = True
OptIntegrafunción.Visible = True
OptIntegradatos.Visible = True
LblIntegrafunción.Visible = False
TxtIntegrafunción.Text = " "
TxtIntegrafunción.Visible = False
LblIntegraN.Visible = False
TxtIntegraN.Text = " "
TxtIntegraN.Visible = False
CmbIntegraN.Clear
CmbIntegraN.Text = "Número de intervalos"
For i = 1 To 6
CmbIntegraN.AddItem i
Next i
CmbIntegraN.Visible = True
FraIntegraintervalo.Visible = False
LblIntegraa.Visible = False
TxtIntegraa.Text = " "
TxtIntegrab.Visible = False
LblIntegrab.Visible = False
```

```
TxtIntegrab.Text = " "
TxtIntegrab.Visible = False
CheckIntegraerror.Value = 0
CheckIntegraerror.Visible = True
LblIntegratitulo.Visible = False
fgIntegradatos.Clear
fgIntegradatos.Visible = False
End Select

   Integraabrir.Enabled = True
   Integraguardarcomo.Enabled = True
   FrmRecuperaTrabajo.ver = tipoalg

End Sub

Private Sub OptIntegrafunción_Click()
Select Case tipoalg
Case Is = 1
FraIntegratipo.Visible = True
OptIntegrafunción.Visible = True
OptIntegradatos.Visible = True
LblIntegrafunción.Visible = True
TxtIntegrafunción.Text = " "
TxtIntegrafunción.Visible = True
LblIntegraN.Visible = False
TxtIntegraN.Text = " "
TxtIntegraN.Visible = False
CmbIntegraN.Visible = False
FraIntegraintervalo.Visible = True
LblIntegraa.Visible = True
TxtIntegraa.Text = " "
TxtIntegraa.Visible = True
LblIntegrab.Visible = True
TxtIntegrab.Text = " "
TxtIntegrab.Visible = True
CheckIntegraerror.Value = 0
CheckIntegraerror.Visible = True
LblIntegratitulo.Visible = False
LblIntegratitulo2.Visible = False
fgIntegradatos.Clear
fgIntegradatos.Visible = False


   Integraabrir.Enabled = True
   Integraguardarcomo.Enabled = True
   FrmRecuperaTrabajo.ver = tipoalg
```

```
End Select

End Sub

Private Sub TxtIntegraa_Change()
LTrim (TxtIntegraa)
End Sub

Private Sub TxtIntegraa_KeyPress(KeyAscii As Integer)
    'filtra la entrada de datos que no sean números
  If (KeyAscii < 48 Or KeyAscii > 57) Then
      If (KeyAscii <> 46) Then
         If (KeyAscii <> 45) Then
            If (KeyAscii <> 8) Then
            KeyAscii = 0
            End If
         End If
      End If
   End If
  If KeyAscii = Asc(vbCr) Then ' Elimina los retornos para quitar los pitidos.
  KeyAscii = 0
  End If
End Sub

Private Sub TxtIntegrab_Change()
LTrim (TxtIntegrab)
End Sub

Private Sub TxtIntegraEdit1_Change()
LTrim (TxtIntegraEdit1)
End Sub

'Filtra  el ingreso para la integral real
Private Sub TxtIntegrareal_KeyPress(KeyAscii As Integer)
    'filtra la entrada de datos que no sean números
  If (KeyAscii < 48 Or KeyAscii > 57) Then
      If (KeyAscii <> 46) Then
         If (KeyAscii <> 45) Then
            If (KeyAscii <> 8) Then
            KeyAscii = 0
            End If
         End If
      End If
   End If
  If KeyAscii = Asc(vbCr) Then ' Elimina los retornos para quitar los pitidos.
```

```
  KeyAscii = 0
  End If
End Sub

'Manejo del  fgIntegradatos
Function Fgi(r As Integer, c As Integer) As Integer
  Fgi = c + fgIntegradatos.Cols * r
End Function
Sub fgIntegradatos_KeyPress(KeyAscii As Integer)
  MSHFlexGridEdit fgIntegradatos, TxtIntegraEdit1, KeyAscii
End Sub

Sub fgIntegradatos_DblClick()
  MSHFlexGridEdit fgIntegradatos, TxtIntegraEdit1, 32 ' Simula un espacio.
End Sub
Sub MSHFlexGridEdit(MSHFlexGrid As Control, _
Edt As Control, KeyAscii As Integer)

  ' Usar el carácter escrito.
  Select Case KeyAscii

  ' Un espacio significa modificar el texto actual.
  Case 0 To 32
    Edt = MSHFlexGrid
    Edt.SelStart = 1000

  ' Otro carácter reemplaza el texto actual.
  Case Else
    Edt = Chr(KeyAscii)
    Edt.SelStart = 1
  End Select

  ' Mostrar Edt en la posición correcta.
  Edt.Move MSHFlexGrid.Left + MSHFlexGrid.CellLeft, _
    MSHFlexGrid.Top + MSHFlexGrid.CellTop, _
    MSHFlexGrid.CellWidth - 8, _
    MSHFlexGrid.CellHeight - 8
  Edt.Visible = True

  ' Y hacer que funcione.
  Edt.SetFocus
End Sub
Sub TxtIntegraEdit1_KeyPress(KeyAscii As Integer)
    'filtra la entrada de datos que no sean números
  If (KeyAscii < 48 Or KeyAscii > 57) Then
      If (KeyAscii <> 46) Then
```

```
        If (KeyAscii <> 45) Then
          If (KeyAscii <> 8) Then
          KeyAscii = 0
          End If
        End If
      End If
    End If
   If KeyAscii = Asc(vbCr) Then ' Elimina los retornos para quitar los pitidos.
   KeyAscii = 0
   End If
End Sub

Sub TxtIntegraEdit1_KeyDown(KeyCode As Integer, _
Shift As Integer)
  EditKeyCode fgIntegradatos, TxtIntegraEdit1, KeyCode, Shift
End Sub

Sub EditKeyCode(MSHFlexGrid As Control, Edt As _
Control, KeyCode As Integer, Shift As Integer)

  ' Procesamiento del control de edición estándar.
  Select Case KeyCode

  Case 27   ' ESC: ocultar, devuelve el enfoque a      ' MSFlexGrid.
    Edt.Visible = False
    MSHFlexGrid.SetFocus

  Case 13   ' ENTRAR devuelve el enfoque a MSHFlexGrid.
    MSHFlexGrid.SetFocus

  Case 38     ' Arriba.
    MSHFlexGrid.SetFocus
    DoEvents
    If MSHFlexGrid.Row > MSHFlexGrid.FixedRows Then
      MSHFlexGrid.Row = MSHFlexGrid.Row - 1
    End If

  Case 40     ' Abajo.
    MSHFlexGrid.SetFocus
    DoEvents
    If MSHFlexGrid.Row < MSHFlexGrid.Rows - 1 Then
      MSHFlexGrid.Row = MSHFlexGrid.Row + 1
    End If
  End Select
End Sub
Sub fgIntegradatos_GotFocus()
```

```
    If TxtIntegraEdit1.Visible = False Then Exit Sub
    fgIntegradatos = TxtIntegraEdit1
    TxtIntegraEdit1.Visible = False
End Sub

Sub fgIntegradatos_LeaveCell()
    If TxtIntegraEdit1.Visible = False Then Exit Sub
    fgIntegradatos = TxtIntegraEdit1
    TxtIntegraEdit1.Visible = False
End Sub
'Manejo de la presentación de controles para el cálculo del error
Private Sub CheckIntegraerror_Click()
If CheckIntegraerror.Value = 1 Then
FraIntegraerror.Visible = True
LblIntegrareal.Visible = True
TxtIntegrareal.Text = " "
TxtIntegrareal.Visible = True
Else
FraIntegraerror.Visible = False
LblIntegrareal.Visible = False
TxtIntegrareal.Text = " "
TxtIntegrareal.Visible = False
End If
End Sub

'Manejo de presentación del número de datos en fgIntegradatos
Private Sub CmbIntegraN_Click()
Dim nu As Integer
If tipoalg = 1 Then
If VLRec = True Then
    LblIntegratitulo2.Caption = titulos(VLIndice)
Else
    LblIntegratitulo2.Caption = titulos(CmbIntegraN.ListIndex)
End If
LblIntegratitulo2.Visible = True
fgIntegradatos.Clear
fgIntegradatos.Visible = True
If VLRec = True Then
    nu = VLIndice + 3
    n = VLIndice + 1
    VLRec = False
Else
    nu = CmbIntegraN.ListIndex + 3
    n = CmbIntegraN.ListIndex + 1
End If
fgIntegradatos.Cols = 3
```

```
fgIntegradatos.ColWidth(0) = 500
fgIntegradatos.ColAlignmentFixed(0) = 4
fgIntegradatos.ColAlignmentFixed(1) = 4
fgIntegradatos.ColAlignmentFixed(2) = 4
fgIntegradatos.Rows = nu
fgIntegradatos.FixedRows = 1
fgIntegradatos.FixedCols = 1
fgIntegradatos.TextArray(Fgi(0, 1)) = "X"
fgIntegradatos.TextArray(Fgi(0, 2)) = "Y"
fgIntegradatos.TextArray(Fgi(1, 0)) = "a="
fgIntegradatos.TextArray(Fgi(nu - 1, 0)) = "b="
fgIntegradatos.Visible = True
LblIntegratitulo.Visible = True

End If

End Sub
Private Sub TxtIntegraN_Change()
Dim mensaje As String
Dim i As Integer
If Val(TxtIntegraN) > 32767 Then
mensaje = MsgBox("Fuera del rango", 16, "Error en el ingreso")
TxtIntegraN.Text = " "
End If
n = Val(TxtIntegraN)
End Sub

Private Sub TxtIntegraN_KeyPress(KeyAscii As Integer)
Dim mensaje As String
If (KeyAscii < 49 Or KeyAscii > 54) Then
If (KeyAscii <> 8) Then KeyAscii = 0
End If

End Sub

Private Sub SetearIntegración()
Dim i As Integer
Dim j As Integer
Dim datos() As Double
Dim tope As Integer
On Error GoTo err1
tope = fgIntegradatos.Rows - 1
ReDim datos(0 To (tope - 1), 1)
Select Case tipoalg
Case Is = 1
If OptIntegrafunción = False Then
```

```
For i = 0 To tope - 1
   For j = 0 To 1
   datos(i, j) = fgIntegradatos.TextMatrix(i + 1, j + 1)
   Next j
Next i
End If
End Select
integración.Setearintegra(TxtIntegrafunción, n, CDbl(Val(TxtIntegraa)),
CDbl(Val(TxtIntegrab)), CDbl(Val(TxtIntegrareal)) _
, OptIntegrafunción.Value, OptIntegradatos.Value, CheckIntegraerror.Value) = datos
Exit Sub
err1:
End Sub

'Manejo de presentación de resultados
Function Fgr(r As Integer, c As Integer) As Long
  Fgr = c + fgIntegraresultados.Cols * r
End Function
Private Sub IntegraPresentar()
Dim i As Integer
Dim j As Integer
Dim valor As Double
On Error GoTo err1
'Seleccionamos el número de filas a mostrar en el fgIntegraresultados
Select Case tipoalg
Case Is = 1
If OptIntegrafunción = True Then
n = 6
Else
n = 1
End If
Case Is = 2
n = 1
Case Is = 3
n = 1
Case Is = 4
n = 7
End Select
If tipoalg <> 4 Then
'Damos formato al fgIntegraresultados
fgIntegraresultados.Cols = 4
fgIntegraresultados.Rows = n + 1
fgIntegraresultados.FixedCols = 0
fgIntegraresultados.FixedRows = 1
fgIntegraresultados.TextArray(Fgr(0, 0)) = "N"
fgIntegraresultados.TextArray(Fgr(0, 1)) = "h"
```

```
fgIntegraresultados.TextArray(Fgr(0, 2)) = "Integral"
fgIntegraresultados.TextArray(Fgr(0, 3)) = "Error"
fgIntegraresultados.ColWidth(0) = 500
fgIntegraresultados.ColWidth(1) = 2000
fgIntegraresultados.ColWidth(2) = 2000
fgIntegraresultados.ColWidth(3) = 2000
fgIntegraresultados.ColAlignmentFixed(0) = 4
fgIntegraresultados.ColAlignmentFixed(1) = 4
fgIntegraresultados.ColAlignmentFixed(2) = 4
fgIntegraresultados.ColAlignmentFixed(3) = 4
'Cargamos los resultados
For i = 1 To n
   For j = 0 To 3
   valor = integración.Resultadointegración(i - 1, j)
      If (valor <= 0 Or valor > 0.000000000000001) Then
      fgIntegraresultados.TextMatrix(i, j) = valor
      Else
      fgIntegraresultados.TextMatrix(i, j) = 0
      End If
  Next j
Next i

Else
'Damos formato al fgIntegraresultados
fgIntegraresultados.Cols = 3
fgIntegraresultados.Rows = n + 1
fgIntegraresultados.FixedCols = 0
fgIntegraresultados.FixedRows = 1
fgIntegraresultados.TextArray(Fgr(0, 0)) = "N"
fgIntegraresultados.TextArray(Fgr(0, 1)) = "Integral"
fgIntegraresultados.TextArray(Fgr(0, 2)) = "Error"
fgIntegraresultados.ColWidth(0) = 500
fgIntegraresultados.ColWidth(1) = 2000
fgIntegraresultados.ColWidth(2) = 2000
fgIntegraresultados.ColAlignmentFixed(0) = 4
fgIntegraresultados.ColAlignmentFixed(1) = 4
fgIntegraresultados.ColAlignmentFixed(2) = 4

'Cargamos los resultados
For i = 1 To n
   For j = 0 To 2
   valor = integración.Resultadointegración(i - 1, j)
      If (valor <= 0 Or valor > 0.000000000000001) Then
      fgIntegraresultados.TextMatrix(i, j) = valor
      Else
      fgIntegraresultados.TextMatrix(i, j) = 0
```

```
      End If
   Next j
Next i

End If
Exit Sub
err1:
End Sub

'****** Conexion a la BDD
Private Sub PLConexion(parBDD As Boolean)
Dim VTCad As String
Dim VTBase As Database
Dim VTcadena As String
   VTcadena = App.Path & "/Seguridad.mdb"
   Set VLConexion = New ADODB.Connection
   If parBDD = False Then
      VTCad = "Provider=SQLOLEDB.1;Integrated Security=SSPI;Persist Security
Info=False;Initial Catalog=Seguridad"
   Else
      VTCad = "Provider=Microsoft.Jet.OLEDB.4.0;Data Source='" & VTcadena &
"';Persist Security Info=False"
   End If
   VLConexion.Open VTCad
End Sub

Private Sub PLGuardarBDD()
   Dim VTCad As String
   Dim VTReg As ADODB.Recordset
   Dim VTID As Integer
   Dim VB As Integer
   Dim VTNumero As String
On Error GoTo error1
   PLVerificaNombre Trim(VLNombre)
   If tipoalg = 1 Then
      If VLVerifica = True Then   'Si existe un nombre con el mismo nombre actualiza
         VB = MsgBox("El nombre de ese trabajo ya existe, desea reemplazarlo?",
vbYesNo, "Guardar Trabajo Como")
         If VB = 6 Then
            If OptIntegrafunción = True Then
               VTCad = "update Integracion set " _
                     & " Fx= '" & Trim(TxtIntegrafunción.Text) & "', " _
                     & " Ia='" & TxtIntegraa.Text & "', " _
                     & " Ib='" & TxtIntegrab.Text & "' " _
                     & " where Nombre = '" & VLNombre & "'"
            End If
```

```
            If OptIntegradatos = True Then
               PLGuardaDatos CmbIntegraN
               VTCad = "update Integracion set " _
                     & " Fx= '" & Trim(TxtIntegrafunción.Text) & "', " _
                     & " Datos='" & VLJacobiano & "' " _
                     & " where Nombre = '" & VLNombre & "'"
            End If
            VLConexion.Execute (VTCad)
         End If
      Else    'Guarda el trabajo
         VTID = PLRecuperaCodigo
         If OptIntegrafunción = True Then
            VTNumero = ""
            VLJacobiano = ""
         End If
         If OptIntegradatos = True Then
            VTNumero = CmbIntegraN.Text
            PLGuardaDatos CmbIntegraN
         End If
         VTCad = "insert into Integracion values " _
               & "(" & CInt(VTID) & ", " _
               & " '" & TxtIntegrafunción.Text & "', '" & VTNumero & "', " _
               & " '" & VLJacobiano & "', '" & TxtIntegraa.Text & "', " _
               & " '" & TxtIntegrab.Text & "', " _
               & " '" & Trim(VLNombre) & "', " & CInt(tipoalg) & ", " & 1 & ")"
               '& " '" & Trim(VLNombre) & "', " & CInt(tipoalg) & ", " & CInt(VGIDUsuario)
& ")"
         VLConexion.Execute (VTCad)
         MsgBox "Trabajo guardado exitosamente", vbInformation, "Guardar trabajo
como"
      End If
   Else
      If VLVerifica = True Then   'Si existe un nombre con el mismo nombre actualiza
         VB = MsgBox("El nombre de ese trabajo ya existe, desea reemplazarlo?",
vbYesNo, "Guardar Trabajo Como")
         If VB = 6 Then
            'PLGuardaJacobiano
            If tipoalg = 2 Or tipoalg = 3 Then
               VTCad = "update Integracion set " _
                     & " Fx= '" & Trim(TxtIntegrafunción.Text) & "', " _
                     & " Numi='" & TxtIntegraN.Text & "', " _
                     & " Ia='" & TxtIntegraa.Text & "', " _
                     & " Ib='" & TxtIntegrab.Text & "' " _
                     & " where Nombre = '" & VLNombre & "'"
            ElseIf tipoalg = 4 Then
               VTCad = "update Integracion set " _
```

```
                    & " Fx= '" & Trim(TxtIntegrafunción.Text) & "', " _
                    & " Ia='" & TxtIntegraa.Text & "', " _
                    & " Ib='" & TxtIntegrab.Text & "' " _
                    & " where Nombre = '" & VLNombre & "'"
            End If
            VLConexion.Execute (VTCad)
        End If
    Else    'Guarda el trabajo
        VTID = PLRecuperaCodigo
        VLJacobiano = ""
        VTCad = "insert into Integracion values " _
            & "(" & CInt(VTID) & ", " _
            & " '" & TxtIntegrafunción.Text & "', '" & TxtIntegraN.Text & "', " _
            & " '" & VLJacobiano & "', '" & TxtIntegraa.Text & "', " _
            & " '" & TxtIntegrab.Text & "', " _
            & " '" & Trim(VLNombre) & "', " & CInt(tipoalg) & ", " & 1 & ")"
            '& " '" & Trim(VLNombre) & "', " & CInt(tipoalg) & ", " & CInt(VGIDUsuario)
& ")"
        VLConexion.Execute (VTCad)
        MsgBox "Trabajo guardado exitosamente", vbInformation, "Guardar trabajo
como"
    End If
  End If

Exit Sub
error1:
  If err.Number <> 0 Then
    MsgBox "Se ha producido el siguiente error: " & err.Description & "." _
        & "Por favor consultar con el administrador", vbCritical
  End If
End Sub

Private Sub PLVerificaNombre(parNombre As String)
Dim VTReg As ADODB.Recordset
Dim VTCad As String
On Error GoTo error1
  VTCad = "select Nombre from Integracion where Nombre = '" & parNombre & "'
and tipo = " & tipoalg
    Set VTReg = VLConexion.Execute(VTCad)
    With VTReg
      If Not VTReg.EOF Then
        .MoveFirst
        Do Until .EOF
          If Trim(VTReg(0)) = parNombre Then
            VLVerifica = True
            Exit Do
```

```
                Else
                    VLVerifica = False
                End If
                .MoveNext
            Loop
        Else
            VLVerifica = False
        End If
    End With
Exit Sub
error1:

End Sub
Private Function PLRecuperaCodigo() As Integer
    Dim VTCad As String
    Dim VTReg As ADODB.Recordset
On Error GoTo err1
    VTCad = "select max(IDIntegracion) from Integracion"
    Set VTReg = VLConexion.Execute(VTCad)
    If Not IsNull(VTReg(0)) Then
        PLRecuperaCodigo = VTReg(0) + 1
    Else
        PLRecuperaCodigo = 1
    End If
Exit Function
err1:

End Function

Private Sub PLGuardaDatos(parNume As String)
Dim i, j As Integer
Dim Nume As Integer
On Error GoTo error1
    VLJacobiano = ""
    Nume = CInt(parNume)
    If tipoalg = 1 Then
        For i = 1 To Nume + 1
            For j = 1 To Nume
                VLJacobiano = VLJacobiano & fgIntegradatos.TextMatrix(i, j) & ";"
            Next j
        Next i
    End If
'VLJacobiano
Exit Sub
error1:
```

```
End Sub

Private Sub PLRecupera()
   Dim VTCad As String
   Dim VTReg As ADODB.Recordset
On Error GoTo err1

      VTCad = "select * from Integracion where Nombre = '" & VLNombreTrabajo & "'"
      Set VTReg = VLConexion.Execute(VTCad)
      If Not VTReg.EOF Then
         TxtIntegrafunción.Text = Trim(VTReg(1))
         TxtIntegraa.Text = Trim(VTReg(4))
         TxtIntegrab.Text = Trim(VTReg(5))
         If tipoalg = 1 And OptIntegradatos.Value = True Then
            VLIndice = CInt(VTReg(2)) - 1
            VLRec = True
            CmbIntegraN.Text = CInt(VTReg(2))
            CmbIntegraN_Click
            PLRecuperaDatos CInt(VTReg(2)), Trim(VTReg(3))
         End If
         If OptIntegrafunción.Value = True Then

         End If
         TxtIntegraN.Text = Trim(VTReg(2))
      End If
Exit Sub
err1:
   If err.Number <> 0 Then
      MsgBox "Se ha producido el siguiente error: " & err.Description & "." _
            & "Por favor consultar con el administrador", vbCritical
   End If
End Sub


Private Sub PLRecuperaDatos(parNume As Integer, parJacobiano As String)
Dim i, j, k As Integer
Dim VTCont As Integer
Dim VTPos As Integer
Dim VTValor As Double
Dim VTLimite As Integer
Dim VTJacob As String
On Error GoTo error1
   VTCont = 1
   VTJacob = parJacobiano
      VTLimite = (parNume + parNume) + 2
      For k = 1 To VTLimite
```

```
        VLJacobianoMatriz(k) = 0
    Next k

    For k = 1 To VTLimite
        VTPos = InStr(1, parJacobiano, ";")
        VTValor = Mid(parJacobiano, 1, VTPos - 1)
        parJacobiano = Mid(parJacobiano, VTPos + 1)
        VLJacobianoMatriz(k) = VTValor
    Next k
    For i = 1 To parNume + 1
        For j = 1 To parNume
            fgIntegradatos.TextMatrix(i, j) = VLJacobianoMatriz(VTCont)
            VTCont = VTCont + 1
        Next j
    Next i

Exit Sub
error1:
    If err.Number <> 0 Then
        MsgBox "Se ha producido el siguiente error: " & err.Description & "." _
            & "Por favor consultar con el administrador", vbCritical
    End If
End Sub

Private Sub PLImprimir()
    PLCambio
        PrintForm
    PLOriginal
End Sub

Private Sub PLCambio()
    Me.BackColor = &HFFFFFF
    Me.BorderStyle = 0

End Sub

Private Sub PLOriginal()
    Me.BackColor = &H8000000F
    Me.BorderStyle = 2

End Sub


Private Sub Sincolor()
With Me
arribaSstab = .SSTab1.Top
```

```
izquierdaSstab = .SSTab1.Left
anchoSstab = .SSTab1.Width
altoSstab = .SSTab1.Height
colororg = .BackColor
.SSTab1.Top = .SSTab1.Top + 1000
.SSTab1.Left = .SSTab1.Left + 1000
.SSTab1.Width = .SSTab1.Width / 4
.SSTab1.Height = .SSTab1.Height / 4
.BackColor = &H80000009
.FraImprimir.BorderStyle = 1
.FraImprimir.BackColor = &H80000009
.fgIntegradatos.BackColorBkg = &H80000009
.fgIntegradatos.BackColorFixed = &H80000009
.fgIntegraresultados.BackColorBkg = &H80000009
.fgIntegraresultados.BackColorFixed = &H80000009

.TxtIntegraa.BorderStyle = 0
.TxtIntegrab.BorderStyle = 0
.TxtIntegraEdit1.BorderStyle = 0
.TxtIntegrafunción.BorderStyle = 0
.TxtIntegraN.BorderStyle = 0
.TxtIntegrareal.BorderStyle = 0
.LblIntegraa.BackColor = &H80000009
.LblIntegrab.BackColor = &H80000009
.LblIntegrafunción.BackColor = &H80000009
.LblIntegraN.BackColor = &H80000009
.LblIntegrareal.BackColor = &H80000009
.LblIntegraresultados.BackColor = &H80000009
.LblIntegratitulo.BackColor = &H80000009
.LblIntegratitulo2.BackColor = &H80000009
.LblNomalg.BackColor = &H80000009
.OptIntegradatos.BackColor = &H80000009
.OptIntegrafunción.BackColor = &H80000009
.FraIntegraerror.BackColor = &H80000009
.FraIntegraintervalo.BackColor = &H80000009
.FraIntegratipo.BackColor = &H80000009
.CheckIntegraerror.BackColor = &H80000009
.CmdIntegracalcular.Visible = False
.ShapeIntegraresultados.Visible = False
FraInteSintaxis.Visible = False
End With
End Sub

Private Sub Concolor()
With Me
 .SSTab1.Top = arribaSstab
```

```
 .SSTab1.Left = izquierdaSstab
 .SSTab1.Width = anchoSstab
 .SSTab1.Height = altoSstab
.BackColor = colororg

.BackColor = colororg
.FraImprimir.BorderStyle = 0
.FraImprimir.BackColor = colororg
.fgIntegradatos.BackColorBkg = colororg
.fgIntegradatos.BackColorFixed = colororg
.fgIntegraresultados.BackColorBkg = colororg
.fgIntegraresultados.BackColorFixed = colororg

.TxtIntegraa.BorderStyle = 1
.TxtIntegrab.BorderStyle = 1
.TxtIntegraEdit1.BorderStyle = 1
.TxtIntegrafunción.BorderStyle = 1
.TxtIntegraN.BorderStyle = 1
.TxtIntegrareal.BorderStyle = 1
.LblIntegraa.BackColor = colororg
.LblIntegrab.BackColor = colororg
.LblIntegrafunción.BackColor = colororg
.LblIntegraN.BackColor = colororg
.LblIntegrareal.BackColor = colororg
.LblIntegraresultados.BackColor = colororg
.LblIntegratitulo.BackColor = colororg
.LblIntegratitulo2.BackColor = colororg
.LblNomalg.BackColor = colororg
.OptIntegradatos.BackColor = colororg
.OptIntegrafunción.BackColor = colororg
.FraIntegraerror.BackColor = colororg
.FraIntegraintervalo.BackColor = colororg
.FraIntegratipo.BackColor = colororg
.CheckIntegraerror.BackColor = colororg
.CmdIntegracalcular.Visible = True
.ShapeIntegraresultados.Visible = True
FraInteSintaxis.Visible = True
End With
End Sub
```

Módulo de clase ClasIntegración

```
Option Explicit
Dim evaluarf As New McsExcel
Dim Constantes() As Double
Dim ParesDatos() As Double  'pares de datos
Dim n As Integer            'número de segmentos
```

```
Dim integralreal As Double  'integral real
Dim función As String       'función
Dim Integraldefunción As Boolean
Dim Integraldedatos As Boolean
Dim Flagerror As Boolean    'indica si hay que calcular el error
Dim A As Double             'límite inicial del intervalo
Dim b As Double             'límite final del intervalo
Dim Resulintegración() As Double 'matriz de Resultados


Public Property Get Resultadointegración() As Variant
On Error GoTo err1
Resultadointegración = Resulintegración
Exit Property
err1:
End Property

Public Property Let Setearintegra(Fun As String, num As Integer, ai As Double, bi As
Double _
, Ireal As Double, Ifunción As Boolean, Idatos As Boolean, Flag As Boolean, Pares()
As Double)
On Error GoTo err1
función = Fun
n = num
integralreal = Ireal
Integraldefunción = Ifunción
Integraldedatos = Idatos
A = ai
b = bi
Flagerror = Flag
ParesDatos = Pares
Exit Property
err1:
End Property
'Permite encontrar la integrar de datos con hasta 6 intervalos
Public Sub NewtonCotesdatos()
Dim i As Integer
Dim j As Integer
Dim h As Double
Dim mensaje As String
Dim Wf As Double
Dim sum As Double
Dim alfa As Double
On Error GoTo err1
'Llenamos la matriz de constantes para las formulas
'Cerradas de Newton-Cotes
```

```
ReDim Constantes(1 To 6, 0 To 7) As Double
Constantes(1, 0) = 1 / 2
Constantes(1, 1) = 1
Constantes(1, 2) = 1
Constantes(2, 0) = 1 / 3
Constantes(2, 1) = 1
Constantes(2, 2) = 4
Constantes(2, 3) = 1
Constantes(3, 0) = 3 / 8
Constantes(3, 1) = 1
Constantes(3, 2) = 3
Constantes(3, 3) = 3
Constantes(3, 4) = 1
Constantes(4, 0) = 2 / 45
Constantes(4, 1) = 7
Constantes(4, 2) = 32
Constantes(4, 3) = 12
Constantes(4, 4) = 32
Constantes(4, 5) = 7
Constantes(5, 0) = 5 / 288
Constantes(5, 1) = 19
Constantes(5, 2) = 75
Constantes(5, 3) = 50
Constantes(5, 4) = 50
Constantes(5, 5) = 75
Constantes(5, 6) = 19
Constantes(6, 0) = 1 / 140
Constantes(6, 1) = 41
Constantes(6, 2) = 216
Constantes(6, 3) = 27
Constantes(6, 4) = 272
Constantes(6, 5) = 27
Constantes(6, 6) = 216
Constantes(6, 7) = 41
'Implementamos el algoritmo
If n <= 0 Then
mensaje = MsgBox(" El valor de N debe estar entre 1 y 6", 16, "Error en el ingreso")
Exit Sub
End If
If n > 6 Then
mensaje = MsgBox(" El valor de N debe estar entre 1 y 6", 16, "Error en el ingreso")
Exit Sub
End If
A = ParesDatos(0, 0)
b = ParesDatos(n, 0)
alfa = Constantes(n, 0)
```

```
h = (b - A) / n
sum = 0
For i = 0 To n
Wf = Constantes(n, i + 1) * ParesDatos(i, 1)
sum = sum + Wf
Next i
ReDim Resulintegración(0, 0 To 3)
Resulintegración(0, 2) = alfa * h * sum
'Calculamos el error
If Flagerror = True Then
Resulintegración(0, 3) = integralreal - Resulintegración(0, 2)
End If
Resulintegración(0, 1) = h
Resulintegración(0, 0) = n
Exit Sub
err1:
If err.Number <> 0 Then
   mensaje = MsgBox("Parámetros del algoritmo incompletos", 48, "Error en el
ingreso")
   End If


End Sub

'Permite encontrar la integrar para funciones con hasta 6 intervalos
Public Sub NewtonCotesfunción()

Dim i As Integer
Dim j As Integer
Dim h As Double
Dim mensaje As String
Dim Wf As Double
Dim sum As Double
Dim alfa As Double
On Error GoTo err1
'Implementamos el algoritmo
If A = b Then
mensaje = MsgBox("Debe ingresar un intervalo, a debe ser diferente de b", 16, "Error
en el ingreso")
Exit Sub
End If
ReDim Constantes(1 To 6, 0 To 7) As Double
Constantes(1, 0) = 1 / 2
Constantes(1, 1) = 1
Constantes(1, 2) = 1
Constantes(2, 0) = 1 / 3
```

```
Constantes(2, 1) = 1
Constantes(2, 2) = 4
Constantes(2, 3) = 1
Constantes(3, 0) = 3 / 8
Constantes(3, 1) = 1
Constantes(3, 2) = 3
Constantes(3, 3) = 3
Constantes(3, 4) = 1
Constantes(4, 0) = 2 / 45
Constantes(4, 1) = 7
Constantes(4, 2) = 32
Constantes(4, 3) = 12
Constantes(4, 4) = 32
Constantes(4, 5) = 7
Constantes(5, 0) = 5 / 288
Constantes(5, 1) = 19
Constantes(5, 2) = 75
Constantes(5, 3) = 50
Constantes(5, 4) = 50
Constantes(5, 5) = 75
Constantes(5, 6) = 19
Constantes(6, 0) = 1 / 140
Constantes(6, 1) = 41
Constantes(6, 2) = 216
Constantes(6, 3) = 27
Constantes(6, 4) = 272
Constantes(6, 5) = 27
Constantes(6, 6) = 216
Constantes(6, 7) = 41
ReDim Resulintegración(0 To 5, 0 To 3)
For n = 1 To 6
   alfa = Constantes(n, 0)
   h = (b - A) / n
   sum = 0
      For i = 0 To n
      Wf = Constantes(n, i + 1) * feval(A + h * i)
      sum = sum + Wf
      Next i

   Resulintegración(n - 1, 2) = alfa * h * sum
   'Calculamos el error
      If Flagerror = True Then
      Resulintegración(n - 1, 3) = integralreal - Resulintegración(0, 0)
      End If
   Resulintegración(n - 1, 1) = h
   Resulintegración(n - 1, 0) = n
```

```
Next n
Exit Sub
err1:
If err.Number <> 0 Then
   mensaje = MsgBox("Parámetros del algoritmo incompletos", 48, "Error en el
ingreso")
   End If

End Sub

Public Sub IntegraciónGaussiana()
Dim i As Integer
Dim j As Integer
Dim h As Double
Dim mensaje As String
Dim Wf As Double
Dim sum As Double
Dim alfa As Double
Dim Constantes() As Double
Dim parcial As Double
Dim W() As Double
Dim z() As Double
Dim funcionevaluada() As Double
On Error GoTo err1
'Llenamos las matriz de constantes para la integración Gaussiana
ReDim Constantes(1 To 8, 2 To 8) As Double
Constantes(1, 2) = 0.577350269
Constantes(2, 2) = -0.577350269
Constantes(1, 3) = 0
Constantes(2, 3) = 0.774596669
Constantes(3, 3) = -0.774596669
Constantes(1, 4) = 0.339981043
Constantes(2, 4) = -0.339981043
Constantes(3, 4) = 0.861136312
Constantes(4, 4) = -0.861136312
Constantes(1, 5) = 0
Constantes(2, 5) = 0.53846931
Constantes(3, 5) = -0.53846931
Constantes(4, 5) = 0.906179846
Constantes(5, 5) = -0.906179846
Constantes(1, 6) = 0.238619186
Constantes(2, 6) = -0.238619186
Constantes(3, 6) = 0.661209387
Constantes(4, 6) = -0.661209387
Constantes(5, 6) = 0.932469514
Constantes(6, 6) = -0.932469514
```

```
Constantes(1, 7) = 0
Constantes(2, 7) = 0.4058451514
Constantes(3, 7) = -0.4058451514
Constantes(4, 7) = 0.7415311856
Constantes(5, 7) = -0.7415311856
Constantes(6, 7) = 0.9491079123
Constantes(7, 7) = -0.9491079123
Constantes(1, 8) = 0.1834346425
Constantes(2, 8) = -0.1834346425
Constantes(3, 8) = 0.5255324099
Constantes(4, 8) = -0.5255324099
Constantes(5, 8) = 0.7966664774
Constantes(6, 8) = -0.7966664774
Constantes(7, 8) = 0.9602898565
Constantes(8, 8) = -0.9602898565

ReDim W(1 To 8, 2 To 8) As Double
W(1, 2) = 1
W(2, 2) = 1
W(1, 3) = 0.888888889
W(2, 3) = 0.555555556
W(3, 3) = 0.555555556
W(1, 4) = 0.652145155
W(2, 4) = 0.652145155
W(3, 4) = 0.347854845
W(4, 4) = 0.347854845
W(1, 5) = 0.568888889
W(2, 5) = 0.47862867
W(3, 5) = 0.47862867
W(4, 5) = 0.236926885
W(5, 5) = 0.236926885
W(1, 6) = 0.467913935
W(2, 6) = 0.467913935
W(3, 6) = 0.360761573
W(4, 6) = 0.360761573
W(5, 6) = 0.171324492
W(6, 6) = 0.171324492
W(1, 7) = 0.4179591837
W(2, 7) = 0.3818300505
W(3, 7) = 0.3818300505
W(4, 7) = 0.2797053915
W(5, 7) = 0.2797053915
W(6, 7) = 0.1294849662
W(7, 7) = 0.1294849662
W(1, 8) = 0.3626837834
W(2, 8) = 0.3626837834
```

```
W(3, 8) = 0.3137066459
W(4, 8) = 0.3137066459
W(5, 8) = 0.2223810345
W(6, 8) = 0.2223810345
W(7, 8) = 0.1012285363
W(8, 8) = 0.1012285363
'Implementamos el algoritmo
If n <= 0 Then
mensaje = MsgBox(" El valor de N debe estar entre 1 y 8", 16, "Error en el ingreso")
Exit Sub
End If
If n > 8 Then
mensaje = MsgBox(" El valor de N debe estar entre 1 y 8", 16, "Error en el ingreso")
Exit Sub
End If
ReDim Resulintegración(0 To 6, 0 To 2) As Double

'Calculamos los nuevos valores al realizar el cambio de varibles para evaluar la
función
ReDim z(1 To 8, 2 To 8) As Double
For j = 2 To 8
   For i = 1 To 8
   z(i, j) = (A + b + ((b - A) * Constantes(i, j))) / 2
   Next i
Next j
'Implementamos el algoritmo
For j = 2 To 8
ReDim funciónevaluada(1 To j) As Double
'Evaluamos la función en los nuevos valores
For i = 1 To j
funciónevaluada(i) = feval(z(i, j))
Next i
'Calculamos la integral
sum = 0
For i = 1 To j
parcial = W(i, j) * funciónevaluada(i)
sum = sum + parcial
Next i
Resulintegración(j - 2, 1) = ((b - A) * sum) / 2
'Calculamos el error
If Flagerror = True Then
Resulintegración(j - 2, 2) = integralreal - Resulintegración(j - 2, 1)
End If
Resulintegración(j - 2, 0) = j
Next j
Exit Sub
```

```
err1:
If err.Number <> 0 Then
   mensaje = MsgBox("Parámetros del algoritmo incompletos", 48, "Error en el
ingreso")
   End If

End Sub



'Permite obtener la integral para datos mediante el uso de la regla compuesta
trapezoidal
Public Sub ReglaExtendidaTrapecio()
Dim i As Integer
Dim j As Integer
Dim mensaje As String
Dim h As Double
Dim sum As Double
Dim parcial As Double
On Error GoTo err1
If n <= 0 Then
mensaje = MsgBox("Debe ingresar un número de intervalos valido, el cual deebe ser
mayor que cero", 16, "Error en el ingreso")
Exit Sub
End If

If A = b Then
mensaje = MsgBox("Debe ingresar un intervalo, a debe ser diferente de b", 16, "Error
en el ingreso")
Exit Sub
End If
h = (b - A) / n
'Implementamos el algoritmo
sum = 0

For i = 1 To n - 1
sum = sum + feval(A + h * i)
Next i
parcial = feval(A) + 2 * sum + feval(b)
'Calculamos la integral y la guardamos
ReDim Resulintegración(0, 3) As Double
Resulintegración(0, 2) = (h * parcial) / 2
'Calculamos el error
If Flagerror = True Then
Resulintegración(0, 3) = integralreal - Resulintegración(0, 2)
End If
```

```
Resulintegración(0, 0) = n
Resulintegración(0, 1) = h
Exit Sub
err1:
If err.Number <> 0 Then
   mensaje = MsgBox("Parámetros del algoritmo incompletos", 48, "Error en el
ingreso")
   End If

End Sub
'Permite obtener la integral para datos mediante el uso de la regla compuesta  1/3 de
Simpson
Public Sub ReglaExtendidaSimpson()
Dim i As Integer
Dim j As Integer
Dim k As Integer
Dim mensaje As String
Dim h As Double
Dim sumimpar As Double
Dim sumpar As Double
Dim impar As Boolean
Dim parcial As Double


On Error GoTo err1
If n <= 0 Then
mensaje = MsgBox("Debe ingresar un número de intervalos valido, el cual deebe ser
mayor que cero", 16, "Error en el ingreso")
Exit Sub
End If
'Evaluamos la función para obtener los nodos necesarios

If A = b Then
mensaje = MsgBox("Debe ingresar un intervalo, a debe ser diferente de b", 16, "Error
en el ingreso")
Exit Sub
End If
h = (b - A) / n
'Implementamos el algoritmo
impar = True
sumimpar = 0
sumpar = 0
For i = 1 To n - 1
   If impar = True Then
   'Suma Impar
   sumimpar = sumimpar + feval(A + h * i)
```

```
    impar = False
    Else
    'Suma par
    sumpar = sumpar + feval(A + h * i)
    impar = True
    End If
Next i
'Obtenemos el resultado parcial
parcial = feval(A) + 4 * sumimpar + 2 * sumpar + feval(b)
'Calculamos la integral y la guardamos
ReDim Resulintegración(0, 3) As Double
Resulintegración(0, 2) = (h * parcial) / 3
'Calculamos el error
If Flagerror = True Then
Resulintegración(0, 3) = integralreal - Resulintegración(0, 2)
End If
Resulintegración(0, 0) = n
Resulintegración(0, 1) = h
Exit Sub
err1:
If err.Number <> 0 Then
   mensaje = MsgBox("Parámetros del algoritmo incompletos", 48, "Error en el
ingreso")
   End If

End Sub


Public Sub NewtonCotes()
Dim mensaje As String
On Error GoTo err1
If Integraldefunción = True Then
NewtonCotesfunción
End If
If Integraldedatos = True Then
NewtonCotesdatos
End If

Exit Sub
err1:
If err.Number <> 0 Then
   mensaje = MsgBox("Parámetros del algoritmo incompletos", 48, "Error en el
ingreso")
   End If
```

```
End Sub
Private Function feval(X As Double) As Double
On Error GoTo err1
evaluarf.Tipox = False
evaluarf.fa = función
evaluarf.PtX = X
evaluarf.evaluarf
feval = evaluarf.Fdy
Exit Function
err1:
End Function
```

Impresión Módule 1

```
Option Explicit

'-------------------------------------------------------------------
    ''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
    '
    ' Visual Basic 4.0 16/32 Capture Routines
    '
    ' This module contains several routines for capturing windows into a
    ' picture.  All the routines work on both 16 and 32 bit Windows
    ' platforms.
    ' The routines also have palette support.
    '
    ' CreateBitmapPicture - Creates a picture object from a bitmap and
    ' palette
    ' CaptureWindow - Captures any window given a window handle
    ' CaptureActiveWindow - Captures the active window on the desktop
    ' CaptureForm - Captures the entire form
    ' CaptureClient - Captures the client area of a form
    ' CaptureScreen - Captures the entire screen
    ' PrintPictureToFitPage - prints any picture as big as possible on
    ' the page
    '
    ' NOTES
    '    - No error trapping is included in these routines
    ''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
    '

    Option Base 0

    Private Type PALETTEENTRY
      peRed As Byte
      peGreen As Byte
      peBlue As Byte
```

```
  peFlags As Byte
End Type

Private Type LOGPALETTE
  palVersion As Integer
  palNumEntries As Integer
  palPalEntry(255) As PALETTEENTRY  ' Enough for 256 colors
End Type

Private Type GUID
  Data1 As Long
  Data2 As Integer
  Data3 As Integer
  Data4(7) As Byte
End Type

#If Win32 Then

  Private Const RASTERCAPS As Long = 38
  Private Const RC_PALETTE As Long = &H100
  Private Const SIZEPALETTE As Long = 104

  Private Type RECT
    Left As Long
    Top As Long
    Right As Long
    Bottom As Long
  End Type

  Private Declare Function CreateCompatibleDC Lib "GDI32" ( _
    ByVal hDC As Long) As Long
  Private Declare Function CreateCompatibleBitmap Lib "GDI32" ( _
    ByVal hDC As Long, ByVal nWidth As Long, _
    ByVal nHeight As Long) As Long
  Private Declare Function GetDeviceCaps Lib "GDI32" ( _
    ByVal hDC As Long, ByVal iCapabilitiy As Long) As Long
  Private Declare Function GetSystemPaletteEntries Lib "GDI32" ( _
    ByVal hDC As Long, ByVal wStartIndex As Long, _
    ByVal wNumEntries As Long, lpPaletteEntries As PALETTEENTRY) _
    As Long
  Private Declare Function CreatePalette Lib "GDI32" ( _
    lpLogPalette As LOGPALETTE) As Long
  Private Declare Function SelectObject Lib "GDI32" ( _
    ByVal hDC As Long, ByVal hObject As Long) As Long
  Private Declare Function BitBlt Lib "GDI32" ( _
    ByVal hDCDest As Long, ByVal XDest As Long, _
```

```
    ByVal YDest As Long, ByVal nWidth As Long, _
    ByVal nHeight As Long, ByVal hDCSrc As Long, _
    ByVal XSrc As Long, ByVal YSrc As Long, ByVal dwRop As Long) _
    As Long
  Private Declare Function DeleteDC Lib "GDI32" ( _
    ByVal hDC As Long) As Long
  Private Declare Function GetForegroundWindow Lib "USER32" () _
    As Long
  Private Declare Function SelectPalette Lib "GDI32" ( _
    ByVal hDC As Long, ByVal hPalette As Long, _
    ByVal bForceBackground As Long) As Long
  Private Declare Function RealizePalette Lib "GDI32" ( _
    ByVal hDC As Long) As Long
  Private Declare Function GetWindowDC Lib "USER32" ( _
    ByVal hWnd As Long) As Long
  Private Declare Function GetDC Lib "USER32" ( _
    ByVal hWnd As Long) As Long
  Private Declare Function GetWindowRect Lib "USER32" ( _
    ByVal hWnd As Long, lpRect As RECT) As Long
  Private Declare Function ReleaseDC Lib "USER32" ( _
    ByVal hWnd As Long, ByVal hDC As Long) As Long
  Private Declare Function GetDesktopWindow Lib "USER32" () As Long

  Private Type PicBmp
    Size As Long
    Type As Long
    hBmp As Long
    hPal As Long
    Reserved As Long
  End Type

  Private Declare Function OleCreatePictureIndirect _
    Lib "olepro32.dll" (PicDesc As PicBmp, RefIID As GUID, _
    ByVal fPictureOwnsHandle As Long, IPic As IPicture) As Long

#ElseIf Win16 Then

  Private Const RASTERCAPS As Integer = 38
  Private Const RC_PALETTE As Integer = &H100
  Private Const SIZEPALETTE As Integer = 104

  Private Type RECT
    Left As Integer
    Top As Integer
    Right As Integer
    Bottom As Integer
```

```
End Type

Private Declare Function CreateCompatibleDC Lib "GDI" ( _
  ByVal hDC As Integer) As Integer
Private Declare Function CreateCompatibleBitmap Lib "GDI" ( _
  ByVal hDC As Integer, ByVal nWidth As Integer, _
  ByVal nHeight As Integer) As Integer
Private Declare Function GetDeviceCaps Lib "GDI" ( _
  ByVal hDC As Integer, ByVal iCapabilitiy As Integer) As Integer
Private Declare Function GetSystemPaletteEntries Lib "GDI" ( _
  ByVal hDC As Integer, ByVal wStartIndex As Integer, _
  ByVal wNumEntries As Integer, _
  lpPaletteEntries As PALETTEENTRY) As Integer
Private Declare Function CreatePalette Lib "GDI" ( _
  lpLogPalette As LOGPALETTE) As Integer
Private Declare Function SelectObject Lib "GDI" ( _
  ByVal hDC As Integer, ByVal hObject As Integer) As Integer
Private Declare Function BitBlt Lib "GDI" ( _
  ByVal hDCDest As Integer, ByVal XDest As Integer, _
  ByVal YDest As Integer, ByVal nWidth As Integer, _
  ByVal nHeight As Integer, ByVal hDCSrc As Integer, _
  ByVal XSrc As Integer, ByVal YSrc As Integer, _
  ByVal dwRop As Long) As Integer
Private Declare Function DeleteDC Lib "GDI" ( _
  ByVal hDC As Integer) As Integer
Private Declare Function GetForegroundWindow Lib "USER" _
  Alias "GetActiveWindow" () As Integer
Private Declare Function SelectPalette Lib "USER" ( _
  ByVal hDC As Integer, ByVal hPalette As Integer, ByVal _
  bForceBackground As Integer) As Integer
Private Declare Function RealizePalette Lib "USER" ( _
  ByVal hDC As Integer) As Integer
Private Declare Function GetWindowDC Lib "USER" ( _
  ByVal hWnd As Integer) As Integer
Private Declare Function GetDC Lib "USER" ( _
  ByVal hWnd As Integer) As Integer
Private Declare Function GetWindowRect Lib "USER" ( _
  ByVal hWnd As Integer, lpRect As RECT) As Integer
Private Declare Function ReleaseDC Lib "USER" ( _
  ByVal hWnd As Integer, ByVal hDC As Integer) As Integer
Private Declare Function GetDesktopWindow Lib "USER" () As Integer

Private Type PicBmp
  Size As Integer
  Type As Integer
  hBmp As Integer
```

```
    hPal As Integer
    Reserved As Integer
  End Type

  Private Declare Function OleCreatePictureIndirect _
    Lib "oc25.dll" (PictDesc As PicBmp, RefIID As GUID, _
    ByVal fPictureOwnsHandle As Integer, IPic As IPicture) _
    As Integer

#End If

''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
'
' CreateBitmapPicture
'   - Creates a bitmap type Picture object from a bitmap and palette
'
' hBmp
'   - Handle to a bitmap
'
' hPal
'   - Handle to a Palette
'   - Can be null if the bitmap doesn't use a palette
'
' Returns
'   - Returns a Picture object containing the bitmap
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
'
#If Win32 Then
  Public Function CreateBitmapPicture(ByVal hBmp As Long, _
    ByVal hPal As Long) As Picture

    Dim r As Long
#ElseIf Win16 Then
  Public Function CreateBitmapPicture(ByVal hBmp As Integer, _
    ByVal hPal As Integer) As Picture

    Dim r As Integer
#End If
  Dim Pic As PicBmp
  ' IPicture requires a reference to "Standard OLE Types"
  Dim IPic As IPicture
  Dim IID_IDispatch As GUID

  ' Fill in with IDispatch Interface ID
  With IID_IDispatch
    .Data1 = &H20400
```

```
    .Data4(0) = &HC0
    .Data4(7) = &H46
  End With

  ' Fill Pic with necessary parts
  With Pic
    .Size = Len(Pic)          ' Length of structure
    .Type = vbPicTypeBitmap   ' Type of Picture (bitmap)
    .hBmp = hBmp              ' Handle to bitmap
    .hPal = hPal              ' Handle to palette (may be null)
  End With

  ' Create Picture object
  r = OleCreatePictureIndirect(Pic, IID_IDispatch, 1, IPic)

  ' Return the new Picture object
  Set CreateBitmapPicture = IPic
End Function

''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
'
' CaptureWindow
'    - Captures any portion of a window
'
' hWndSrc
'    - Handle to the window to be captured
'
' Client
'    - If True CaptureWindow captures from the client area of the
'      window
'    - If False CaptureWindow captures from the entire window
'
' LeftSrc, TopSrc, WidthSrc, HeightSrc
'    - Specify the portion of the window to capture
'    - Dimensions need to be specified in pixels
'
' Returns
'    - Returns a Picture object containing a bitmap of the specified
'      portion of the window that was captured
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
''''''
'
#If Win32 Then
  Public Function CaptureWindow(ByVal hWndSrc As Long, _
    ByVal Client As Boolean, ByVal LeftSrc As Long, _
    ByVal TopSrc As Long, ByVal WidthSrc As Long, _
```

```
    ByVal HeightSrc As Long) As Picture

    Dim hDCMemory As Long
    Dim hBmp As Long
    Dim hBmpPrev As Long
    Dim r As Long
    Dim hDCSrc As Long
    Dim hPal As Long
    Dim hPalPrev As Long
    Dim RasterCapsScrn As Long
    Dim HasPaletteScrn As Long
    Dim PaletteSizeScrn As Long
#ElseIf Win16 Then
  Public Function CaptureWindow(ByVal hWndSrc As Integer, _
    ByVal Client As Boolean, ByVal LeftSrc As Integer, _
    ByVal TopSrc As Integer, ByVal WidthSrc As Long, _
    ByVal HeightSrc As Long) As Picture

    Dim hDCMemory As Integer
    Dim hBmp As Integer
    Dim hBmpPrev As Integer
    Dim r As Integer
    Dim hDCSrc As Integer
    Dim hPal As Integer
    Dim hPalPrev As Integer
    Dim RasterCapsScrn As Integer
    Dim HasPaletteScrn As Integer
    Dim PaletteSizeScrn As Integer
#End If
  Dim LogPal As LOGPALETTE

  ' Depending on the value of Client get the proper device context
  If Client Then
    hDCSrc = GetDC(hWndSrc) ' Get device context for client area
  Else
    hDCSrc = GetWindowDC(hWndSrc) ' Get device context for entire
                                  ' window
  End If

  ' Create a memory device context for the copy process
  hDCMemory = CreateCompatibleDC(hDCSrc)
  ' Create a bitmap and place it in the memory DC
  hBmp = CreateCompatibleBitmap(hDCSrc, WidthSrc, HeightSrc)
  hBmpPrev = SelectObject(hDCMemory, hBmp)

  ' Get screen properties
```

```
  RasterCapsScrn = GetDeviceCaps(hDCSrc, RASTERCAPS) ' Raster
                                          'capabilities
  HasPaletteScrn = RasterCapsScrn And RC_PALETTE      ' Palette
                                           'support
  PaletteSizeScrn = GetDeviceCaps(hDCSrc, SIZEPALETTE) ' Size of
                                           ' palette

  ' If the screen has a palette make a copy and realize it
  If HasPaletteScrn And (PaletteSizeScrn = 256) Then
    ' Create a copy of the system palette
    LogPal.palVersion = &H300
    LogPal.palNumEntries = 256
    r = GetSystemPaletteEntries(hDCSrc, 0, 256, _
       LogPal.palPalEntry(0))
    hPal = CreatePalette(LogPal)
    ' Select the new palette into the memory DC and realize it
    hPalPrev = SelectPalette(hDCMemory, hPal, 0)
    r = RealizePalette(hDCMemory)
  End If

  ' Copy the on-screen image into the memory DC
  r = BitBlt(hDCMemory, 0, 0, WidthSrc, HeightSrc, hDCSrc, _
    LeftSrc, TopSrc, vbSrcCopy)

' Remove the new copy of the  on-screen image
  hBmp = SelectObject(hDCMemory, hBmpPrev)

  ' If the screen has a palette get back the palette that was
  ' selected in previously
  If HasPaletteScrn And (PaletteSizeScrn = 256) Then
    hPal = SelectPalette(hDCMemory, hPalPrev, 0)
  End If

  ' Release the device context resources back to the system
  r = DeleteDC(hDCMemory)
  r = ReleaseDC(hWndSrc, hDCSrc)

  ' Call CreateBitmapPicture to create a picture object from the
  ' bitmap and palette handles.  Then return the resulting picture
  ' object.
  Set CaptureWindow = CreateBitmapPicture(hBmp, hPal)
End Function

''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
'
' CaptureScreen
```

```
'    - Captures the entire screen
'
' Returns
'    - Returns a Picture object containing a bitmap of the screen
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
'
Public Function CaptureScreen() As Picture
  #If Win32 Then
    Dim hWndScreen As Long
  #ElseIf Win16 Then
    Dim hWndScreen As Integer
  #End If

  ' Get a handle to the desktop window
  hWndScreen = GetDesktopWindow()

  ' Call CaptureWindow to capture the entire desktop give the handle
  ' and return the resulting Picture object

  Set CaptureScreen = CaptureWindow(hWndScreen, False, 0, 0, _
    Screen.Width \ Screen.TwipsPerPixelX, _
    Screen.Height \ Screen.TwipsPerPixelY)
End Function

''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
'
' CaptureForm
'    - Captures an entire form including title bar and border
'
' frmSrc
'    - The Form object to capture
'
' Returns
'    - Returns a Picture object containing a bitmap of the entire
'      form
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
'
Public Function CaptureForm(frmSrc As Form) As Picture
  ' Call CaptureWindow to capture the entire form given it's window
  ' handle and then return the resulting Picture object
  Set CaptureForm = CaptureWindow(frmSrc.hWnd, False, 0, 0, _
    frmSrc.ScaleX(frmSrc.Width, vbTwips, vbPixels), _
    frmSrc.ScaleY(frmSrc.Height, vbTwips, vbPixels))
End Function

''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
```

```
'
' CaptureClient
'    - Captures the client area of a form
'
' frmSrc
'    - The Form object to capture
'
' Returns
'    - Returns a Picture object containing a bitmap of the form's
' client area
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
'
Public Function CaptureClient(frmSrc As Form) As Picture
   ' Call CaptureWindow to capture the client area of the form given
   ' it's window handle and return the resulting Picture object
   Set CaptureClient = CaptureWindow(frmSrc.hWnd, True, 0, 0, _
      frmSrc.ScaleX(frmSrc.ScaleWidth, frmSrc.ScaleMode, vbPixels), _
      frmSrc.ScaleY(frmSrc.ScaleHeight, frmSrc.ScaleMode, vbPixels))
End Function

''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
'
' CaptureActiveWindow
'    - Captures the currently active window on the screen
'
' Returns
'    - Returns a Picture object containing a bitmap of the active
'      window
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
'
Public Function CaptureActiveWindow() As Picture
  #If Win32 Then
     Dim hWndActive As Long
     Dim r As Long
  #ElseIf Win16 Then
     Dim hWndActive As Integer
     Dim r As Integer
  #End If
  Dim RectActive As RECT

  ' Get a handle to the active/foreground window
  hWndActive = GetForegroundWindow()

  ' Get the dimensions of the window
  r = GetWindowRect(hWndActive, RectActive)
```

```
  ' Call CaptureWindow to capture the active window given it's
' handle and return the Resulting Picture object
Set CaptureActiveWindow = CaptureWindow(hWndActive, False, 0, 0, _
    RectActive.Right - RectActive.Left, _
    RectActive.Bottom - RectActive.Top)
End Function

''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
'
' PrintPictureToFitPage
'   - Prints a Picture object as big as possible
'
' Prn
'   - Destination Printer object
'
' Pic
'   - Source Picture object
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
'
Public Sub PrintPictureToFitPage(Prn As Printer, Pic As Picture)
  Const vbHiMetric As Integer = 8
  Dim PicRatio As Double
  Dim PrnWidth As Double
  Dim PrnHeight As Double
  Dim PrnRatio As Double
  Dim PrnPicWidth As Double
  Dim PrnPicHeight As Double

  ' Determine if picture should be printed in landscape or portrait
  ' and set the orientation
  If Pic.Height >= Pic.Width Then
    Prn.Orientation = vbPRORPortrait   ' Taller than wide
  Else
    Prn.Orientation = vbPRORLandscape  ' Wider than tall
  End If

  ' Calculate device independent Width to Height ratio for picture
  PicRatio = Pic.Width / Pic.Height

  ' Calculate the dimentions of the printable area in HiMetric
  PrnWidth = Prn.ScaleX(Prn.ScaleWidth, Prn.ScaleMode, vbHiMetric)
  PrnHeight = Prn.ScaleY(Prn.ScaleHeight, Prn.ScaleMode, vbHiMetric)
  ' Calculate device independent Width to Height ratio for printer
  PrnRatio = PrnWidth / PrnHeight

  ' Scale the output to the printable area
```

```
  If PicRatio >= PrnRatio Then
    ' Scale picture to fit full width of printable area
    PrnPicWidth = Prn.ScaleX(PrnWidth, vbHiMetric, Prn.ScaleMode)
    PrnPicHeight = Prn.ScaleY(PrnWidth / PicRatio, vbHiMetric, _
      Prn.ScaleMode)
  Else
    ' Scale picture to fit full height of printable area
    PrnPicHeight = Prn.ScaleY(PrnHeight, vbHiMetric, Prn.ScaleMode)
    PrnPicWidth = Prn.ScaleX(PrnHeight * PicRatio, vbHiMetric, _
      Prn.ScaleMode)
  End If

  ' Print the picture using the PaintPicture method
  Prn.PaintPicture Pic, 0, 0, PrnPicWidth, PrnPicHeight
End Sub
'-------------------------------------------------------------
```

MANUAL DE USUARIO

INTRODUCCIÓN

El Paquete de Software para la Enseñanza-Aprendizaje de Métodos Numéricos, enseña la materia de Métodos Numéricos a los usuarios que deseen aprender la misma de una manera interactiva.

La materia de Métodos Numéricos pretende resolver mediante operaciones matemáticas sencillas problemas que abarcan matemáticas más complicada como el caso de Ecuaciones Diferenciales Ordinarias o Parciales, Integración, Derivación, Sistemas de Ecuaciones Lineales y no Lineales, etc.

Para tal efecto el Paquete de Software Didáctico para la Enseñanza-Aprendizaje de Métodos Numéricos contiene cuatro módulos:

- Módulo de Graficación
- Módulo de Contenido Teórico
- Módulo de Algoritmos
- Módulo de Evaluación.

Cada uno abarca diferentes aspectos relacionados al ámbito del aprendizaje de esta materia, permitiendo una enseñanza integral de la misma. El Paquete de Software Didáctico para la Enseñanza-Aprendizaje de Métodos Numéricos es el resultado de un Proyecto de Titulación para la carrera de Electrónica y Telecomunicaciones de la Escuela Politécnica Nacional.

PROCESO DE INSTALACIÓN

REQUERIMIENTOS DE LA APLICACIÓN

Para que el correcto funcionamiento del Paquete de Software Didáctico para la Enseñanza-Aprendizaje de Métodos Numéricos será necesario contar con el siguiente equipo:

Requerimientos del software:

Sistema Operativo	Windows 98 versión B/XP/NT
Software adicional	MsExcel, Acces 97

Requerimientos del Hardware:

CPU	Procesador 486 mínimo
Monitor	SVGA
Disco Duro	1GB mínimo
RAM	256 MB
CDROM	52X
FLOPY	3 1/2
Impresora	Matricial/inyección a tinta
Teclado	101/102 Teclas
Ratón	

PASOS HA SERGUIR PARA LA INSTALACIÓN

Para que le Paquete de Software Didáctico para la Enseñanza-Aprendizaje de Métodos Numéricos quede listo para ser utilizado, debe realizar su instalación, la cual se realiza de la siguiente forma:

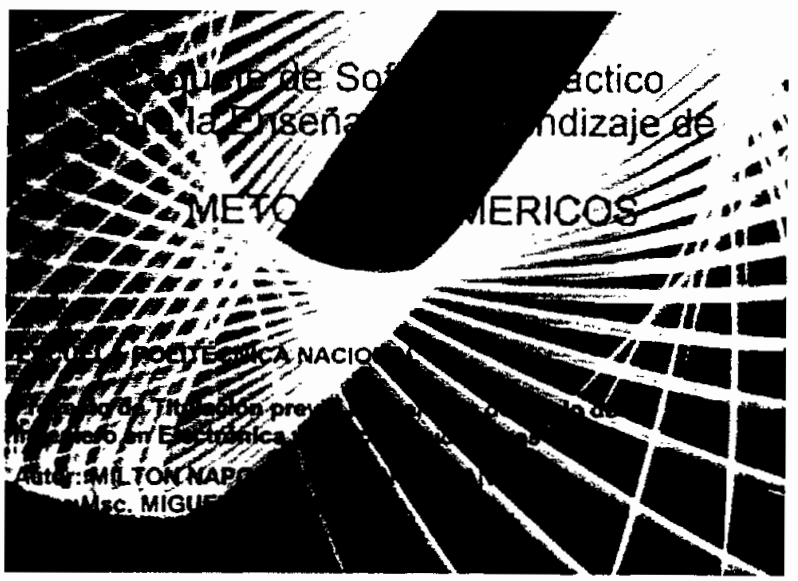
- Introduzca el CD en donde corresponda y averiguar la letra del dispositivo del CD (generalmente D o E)
- A partir de que cierre la compuerta del dispositivo de CD, la máquina le ira indicando que hacer. En caso de que no sea así proceda como sigue:
  - Oprima el Icono de "Mi PC".
  - Oprima con el ratón el icono del dispositivo de CD y escoja el archivo instalar (Setup).

MÓDULOS DEL SOFTWARE

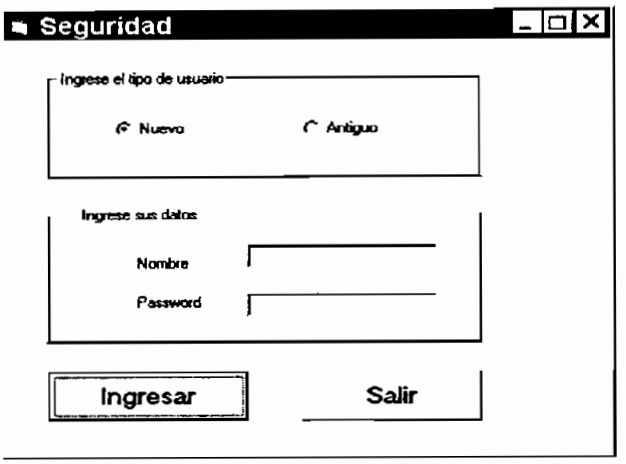
El Paquete de Software Didáctico para la Enseñanza-Aprendizaje de Métodos Numéricos contiene cuatro módulos:

- Módulo de Contenido Teórico
- Módulo de Graficación
- Módulo de Algoritmos
- Módulo de Evaluación

Al ingresar al Paquete de Software Didáctico para la Enseñanza-Aprendizaje de Métodos Numéricos se presentara una pantalla de inicio como la siguiente:

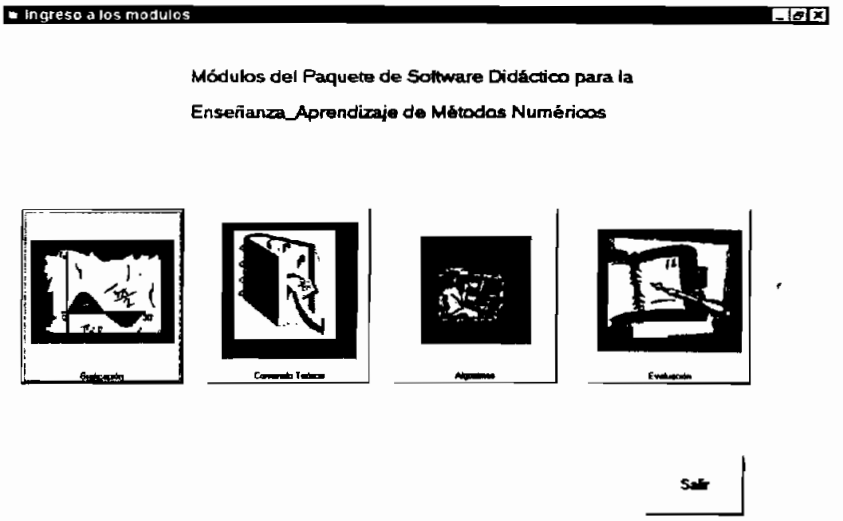


luego la de Ingreso de usuario



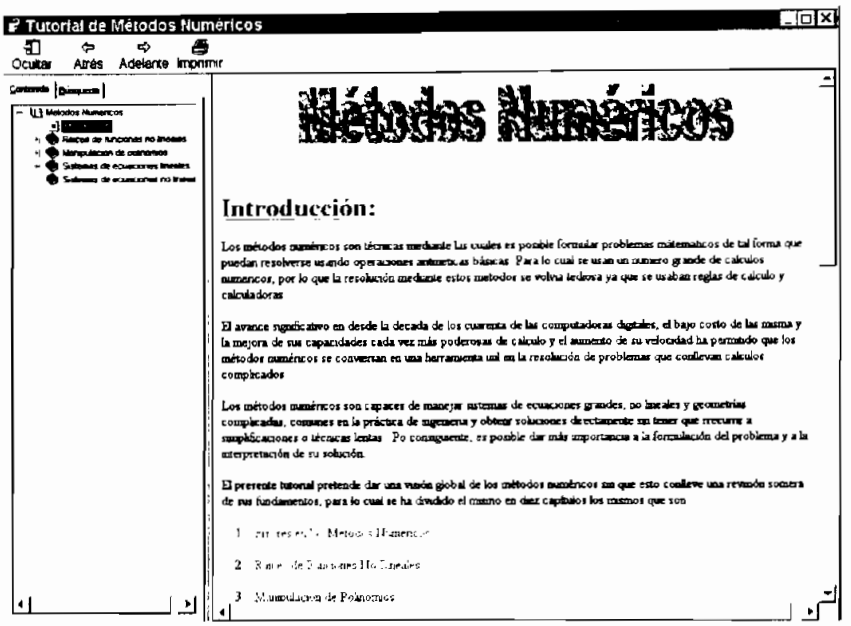
En la cual debe ingresar el tipo de usuario, si es nuevo o antiguo y su password, esta pantalla no es una seguridad en si ya que si ha olvidado su password puede ingresar como usuario nuevo sin ninguna dificultad. El objetivo de esta pantalla es el de recabar información suya para su posterior utilización en el módulo de evaluación.

Sí ha ingresado correctamente los datos pedidos se presentara la pantalla de los módulos que componen la aplicación:



MÓDULO DE CONTENIDO TEÓRICO

Al elegir esta se presentara la siguiente pantalla:



El módulo de contenido teórico esta constituido de un tutorial referente a la materia de Métodos Numéricos.

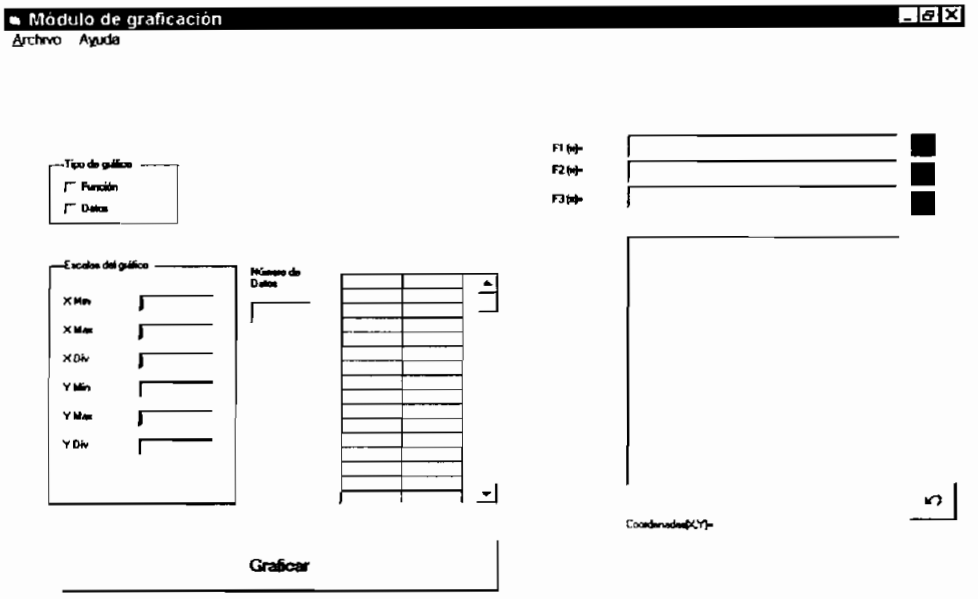
En ella se presentan los temas que son tratados en este software como son:

- Teoría de errores
- Raíces de funciones no lineales
- Manipulación de polinomios
- Sistemas de Ecuaciones lineales y no lineales
- Interpolación
- Derivación Numérica
- Integración Numérica
- Ecuaciones diferenciales ordinarias
- Ecuaciones diferenciales parciales

Cada una explicada mediante ejercicios resueltos con este software. Este módulo permite imprimir su contenido, realizar consultas mediante la pestaña de búsqueda y navegar entre las paginas mediante la pestaña de Contenido.

MÓDULO DE GRAFICACIÓN

Sí usted ha elegido esta opción se le presentara la siguiente pantalla:



El módulo de graficación permite realizar graficar de cualquier tipo de función con un máximo de tres funciones de acuerdo a la sintaxis establecida para tal efecto. También permite la graficación de datos tabulados hasta un máximo de 32767 pares de datos.

Se puede elegir entre graficar funciones o datos mediante los botones de selección, si gráfica ambos a la vez se presentaran los valores que se encuentre dentro del rango de las escalas que ha ingresado para el efecto.

Se puede observar las coordenadas del gráfico colocando el ratón sobre cualquier punto de la gráfica, también permite la ampliación de cualquier parte del gráfico solamente señalando mediante el ratón un recuadro de izquierda a derecha sobre la zona deseada con el botón izquierdo del ratón pulsado.

El Módulo de Graficación cuenta también con un botón de deshacer que permite restablecer solamente el gráfico anterior.

Es necesario introducir las funciones a graficar usando la sintaxis siguiente o de lo contrario de lo contrario se producirá un gráfico erróneo de la función introducida o no se presentara el gráfico.

Usar como variable la letra X mayúscula, si la X multiplica a un número como en el caso de 2X anteponer a la X el signo de multiplicación 2*X, o si se tiene que asignar el signo de – a la variable X se deberá poner –1*X.
Para elevar a una potencia a la variable:

X^2 X^2.

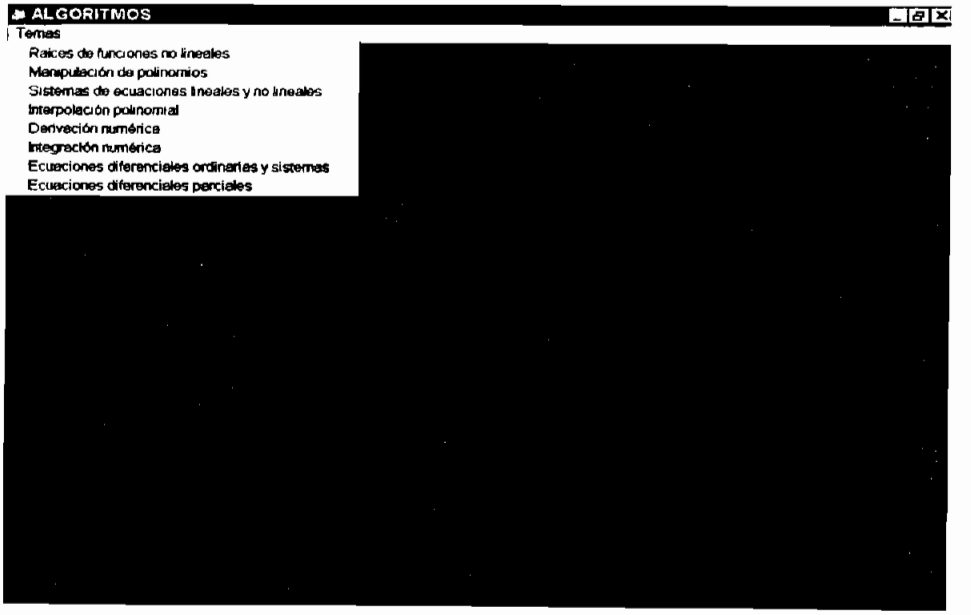
Para pi se debe introducir PI(), el cuadro siguiente resume las funciones y su sintaxis.

Función	**Sintaxis**
Seno (X)	SIN(X)
Coseno (X)	COS(X)
Tan(X)	Tan(X)
RAIZ(X)	SQRT(X)
LN(x)	Ln(X)
LOG(X)	Lg(x)
EXP(X)	exp(X)

El resto de funciones sigue el estándar de ingreso de funciones en las hojas de calculo de la aplicación MsExel ya que esta es usada para evaluar las funciones ingresadas a graficar.

MÓDULOS DE ALGORITMOS

Sí su elección fue está se le mostrara la pantalla principal del módulo algoritmos como la siguiente:



En este módulo se pueden resolver problemas relacionados con los temas del tutorial de métodos numéricos, mediante el ingreso adecuado de los parámetros de cada algoritmo que se desee usar.

El módulo de algoritmos cuenta con los siguientes temas de algoritmos:

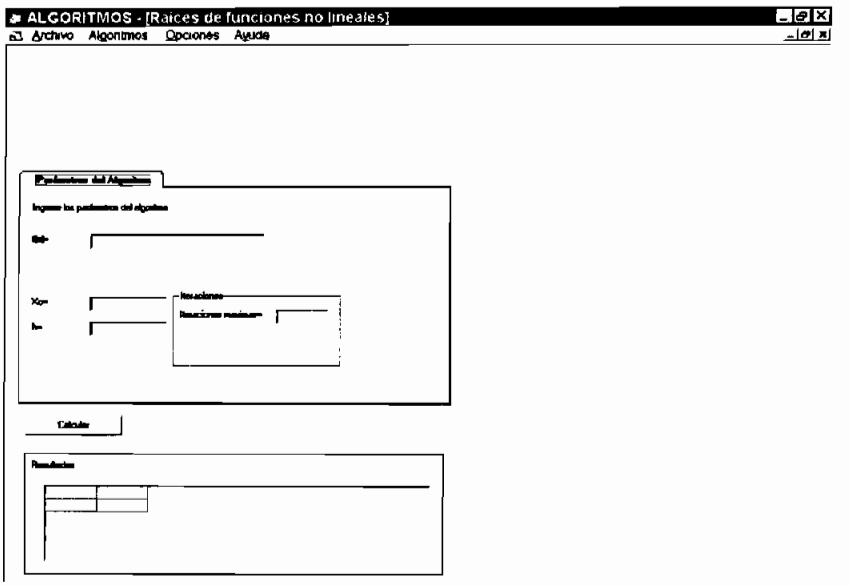
- Raíces de funciones no lineales
- Manipulación de polinomios
- Sistemas de ecuaciones lineales y no lineales
- Interpolación

- Derivación numérica
- Integración numérica
- Ecuaciones diferenciales ordinarias
- Ecuaciones diferenciales parciales

En cada tema se implementan los algoritmos más usados. También cuenta con el acceso al módulo de graficación mediante el botón opciones de la barra de menú.

RAÍCES DE FUNCIONES NO LINEALES

Al elegir esta opción se mostrará una pantalla como la siguiente:



En este se implementan los siguientes algoritmos:

- Aproximaciones sucesivas
- Bisección
- Falsa posición
- Iterativo de punto fijo(primer orden)
- Newton-Raphson
- Secante

Cada uno de ellos presenta los datos necesarios de entrada para la implementación del algoritmo como sigue:

f(x)= función de la cual se encontrará la raíz, debe tener la misma sintaxis que la del ingreso de funciones en el módulo de graficación.

f'(X)= derivada de f(x)

Xo = punto inicial para la búsqueda de la raíz

X1 = punto inicial para la búsqueda de la raíz

h = incremento para la búsqueda de la raíz

a = Punto inicial del intervalo que contiene la raíz

b = Punto final del intervalo que contiene la raíz.

Iteraciones máximas = número de iteraciones máximas que se realizarán

Tolerancia = error que debe tener la respuesta, ingrese en el formato 0.001

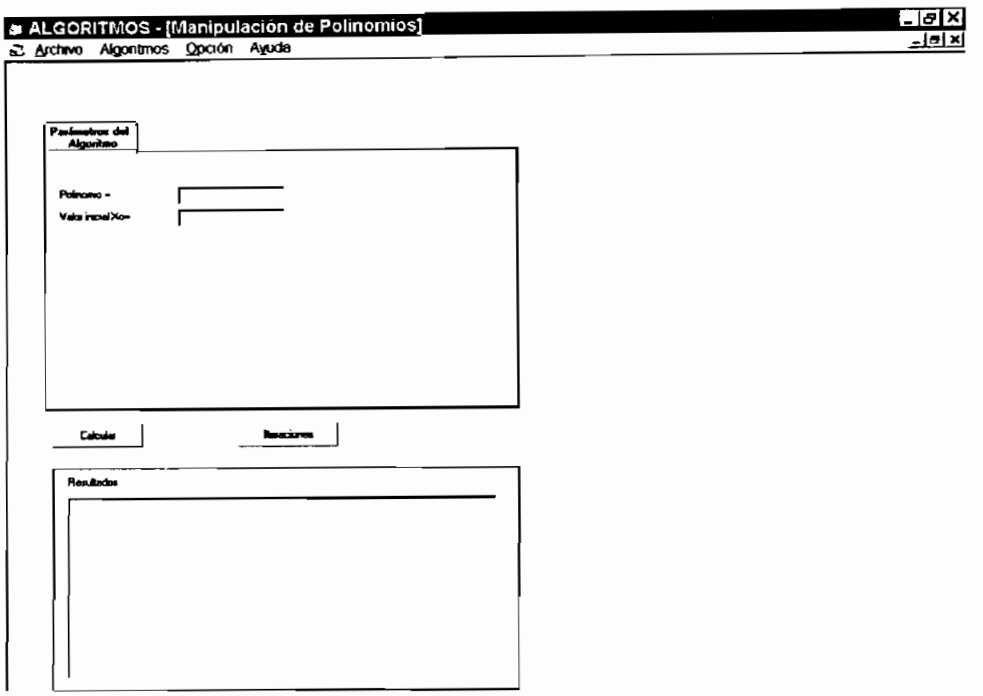
Ex= error relativo en X

Ef= error absoluto en la función

El algoritmo se ejecuta cuando se presiona en el botón calcular obteniéndose los resultados correctos siempre y cuando los datos ingresados estén de acuerdo al algoritmo elegido.

MANIPULACIÓN DE POLINOMIOS

Al elegir esta opción se presentará la siguiente pantalla:



En este se implementan los siguientes algoritmos:

- Horner
- Newton-Horner
- Newton-Bairstow
- Aplicación de Newton-Bairstow

El algoritmo de Horner permite evaluar un polinomio y sus derivadas en un argumento real; el de Newton-Horner permite encontrar las raíces reales de un polinomio de grado mayor o igual a tres; el Newton-Baristow permite obtener las derivadas reales y /o complejas de un polinomio de grado mayor o igual a tres y la aplicación de Newton-Bairstow permite evaluar un polinomio en un argumento complejo.

Polinomio = polinomio del cual se encontrará la raíz, debe tener la misma sintaxis que la del ingreso de funciones en el módulo de graficación.

Xo = punto inicial para la búsqueda de la raíz, debe ser un número dentro del anillo complejo del polinomio normalizado.

Valor r = valor real inicial

Valor s = valor imaginario inicial

Valor real Re = Valor real del número imaginario en el que se evaluará el polinomio.

Valor imaginario Im= Valor imaginario del número imaginario en el que se evaluará el polinomio

Iteraciones máximas = número de iteraciones máximas que se realizarán

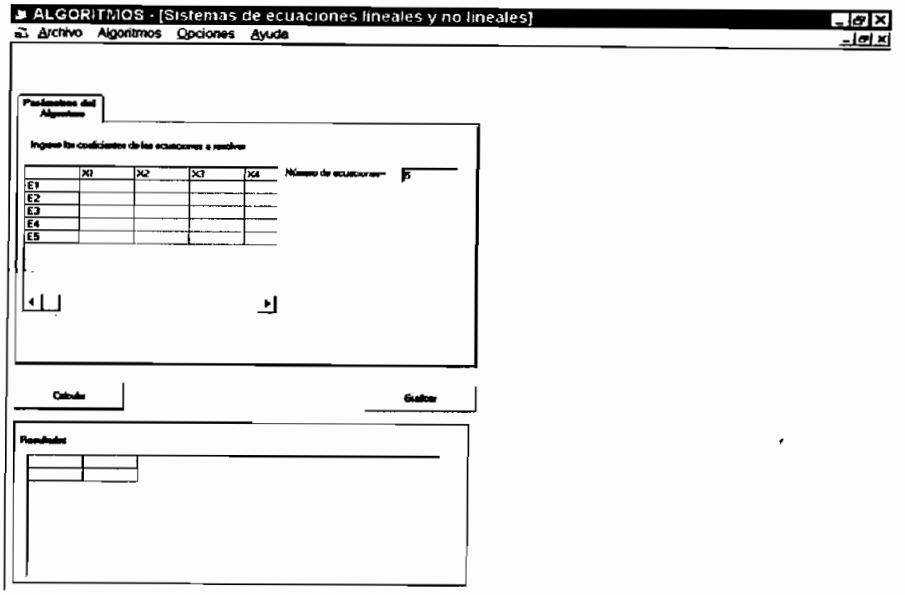
Tolerancia = error que debe tener la respuesta, ingrese en el formato 0.001

Ea= error absoluto

Ep= error absoluto en el polinomio.

SISTEMAS DE ECUACIONES LINEALES Y NO LINEALES

La pantalla que se presenta es:



En este se implementan los siguientes algoritmos:

- Eliminación Gaussiana
- Gauss Jordan
- Factorización
- Jacobi
- Gauss-Seidel
- Jacobi (Sistemas de ecuaciones no lineales)
- Gauss-Seidel (Sistemas de ecuaciones no lineales)
- Newton (Sistemas de ecuaciones no lineales)

Los métodos de eliminación gaussiana, Gauss Jordan y factorización son métodos directo de solución, en cambio los de métodos de Jacobi, Gauss-Seidel y Newton son iterativos. El significado de los parámetros de entrada es:

Número de ecuaciones = número de ecuaciones simultáneas a ser resueltas

Iteraciones máximas = número de iteraciones máximas que se realizarán

Tolerancia = error que debe tener la respuesta, ingrese en el formato 0.001

Ea= error absoluto.

Er= error relativo.

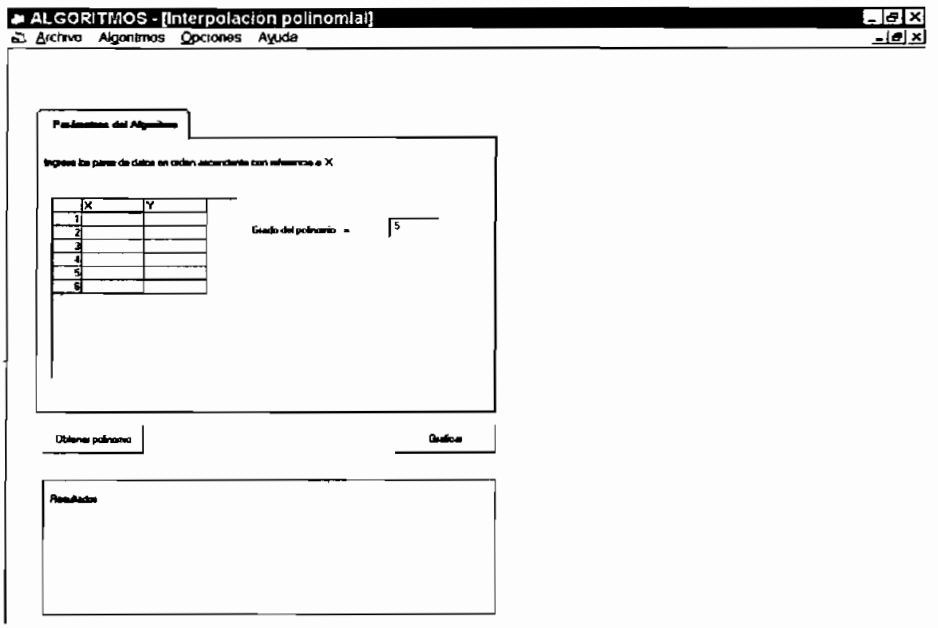
Para los métodos iterativos se debe ingresar además de los coeficientes de las ecuaciones el vector inicial, donde esta indicado, Para los métodos iterativos de sistemas de ecuaciones no lineales se debe ingresar las funciones despejadas para cada Xi con el vector inicial donde se indica.

Para el caso del método de Newton se debe ingresar las funciones del Jacobiano, las funciones del sistema y el vector inicial donde se indica.

Las funciones del sistema a resolver, debe tener la misma sintaxis que la del ingreso de funciones en el módulo de graficación.

INTERPOLACIÓN

La pantalla presentada para este caso será



En este se implementan los siguientes algoritmos:

- Regresión Polinomial
- Técnica Matricial
- Polinomio de Lagrange
- Polinomio de Newton
- Interpolación Trigonométrica
- Interpolación Segmentaria (Spline)

Estos algoritmos permiten ajustar pares de datos o funciones tabuladas complejas a polinomios o funciones trigonométricas. El significado de los parámetros de entrada es:

Número de pares de datos = número de pares de datos

Grado del polinomio = grado del polinomio al que se desea ajustar los datos.

Punto a evaluar = punto en el cual se desea evaluar el polinomio encontrado.

S'(Xo) = primera derivada en el punto inicial de los pares de datos

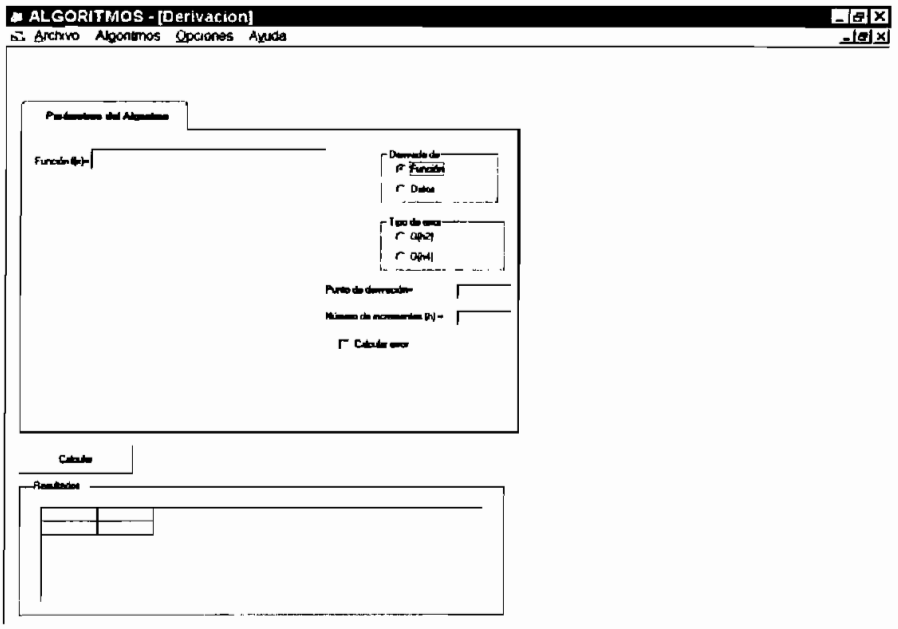
S'(Xn) = primera derivada en el punto final de los pares de datos

S''(Xo) = segunda derivada en el punto inicial de los pares de datos

S''(Xn) = segunda derivada en el punto final de los pares de datos.

DERIVACIÓN NUMÉRICA

Si ha elegido esta opción se presentará:



En este se implementan los siguientes algoritmos:

- Diferencias centradas
- Diferencias progresivas
- Diferencias regresivas

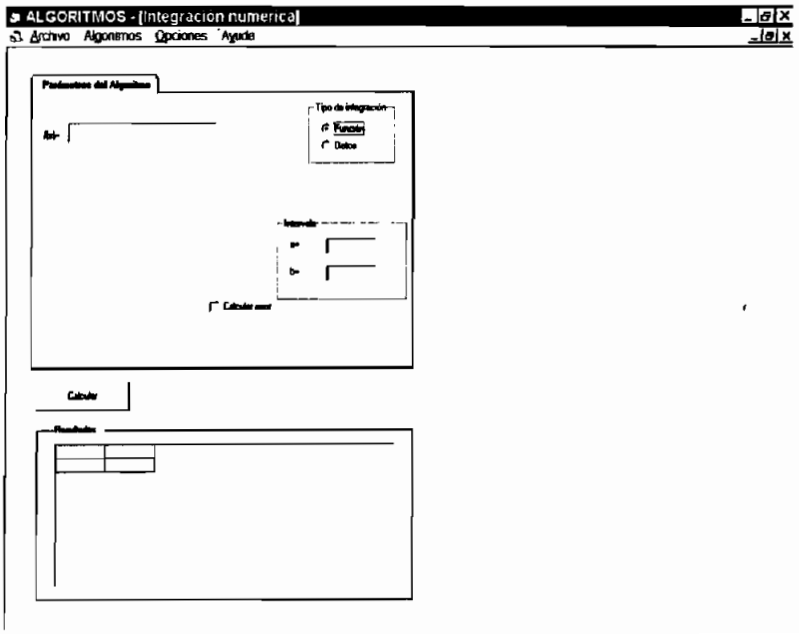
Estos algoritmos permiten encontrar derivadas tanto de datos como de funciones para lo cual los datos deben estar separados uniformemente. El significado de los parámetros de entrada es:

f(X) = función de la cual se obtendrá la derivada, La función a derivar, debe tener la misma sintaxis que la del ingreso de funciones en el módulo de graficación.
O(h),O(h2),O(h4) = error que se desea al realizar la integración
Punto de derivación = punto en el cual se encontrara la derivada
h = incremento con el cual sé hallara la derivada; Este debe ser la separación que tiene los datos, los que deben encontrarse separados uniformemente.
Calcular error = opción que permite el calculo del error siempre y cuando se conozca el valor real de la derivada.
Valor real = Valor real de la derivada.
Debe tomarse en cuenta que cuando de deriva datos se presentan casilleros donde deben ser introducidos así que se tiene la siguiente nomenclatura para esto:

f-3,f-2,f-1,fo,f1,f2,f3 = indican f(x)+h, por ejemplo si se desea obtener la primera derivada mediante diferencias centradas con h =0,1 en X=2, f-1 sería 1.9 y f1 2.1.

INTEGRACIÓN NUMÉRICA

La pantalla para este caso será:



En este se implementan los siguientes algoritmos:

- Newton-Cotes cerrada
- Regla Extendida del Trapecio
- Regla Extendida de 1/3 de Simpson
- Gaussiana

Estos algoritmos permiten calcular la integral de una función o de datos. El significado de los parámetros de entrada es:

f(X) = función de la cual se obtendrá la integral, La función a integrar, debe tener la misma sintaxis que la del ingreso de funciones en el módulo de graficación.

a = Punto inicial del intervalo donde se integrará

b = Punto final del intervalo donde se integrará

Número de intervalos = número de intervalos en el que se desea integrar

Número de segmentos = número de segmentos donde se aplicará las reglas extendidas.

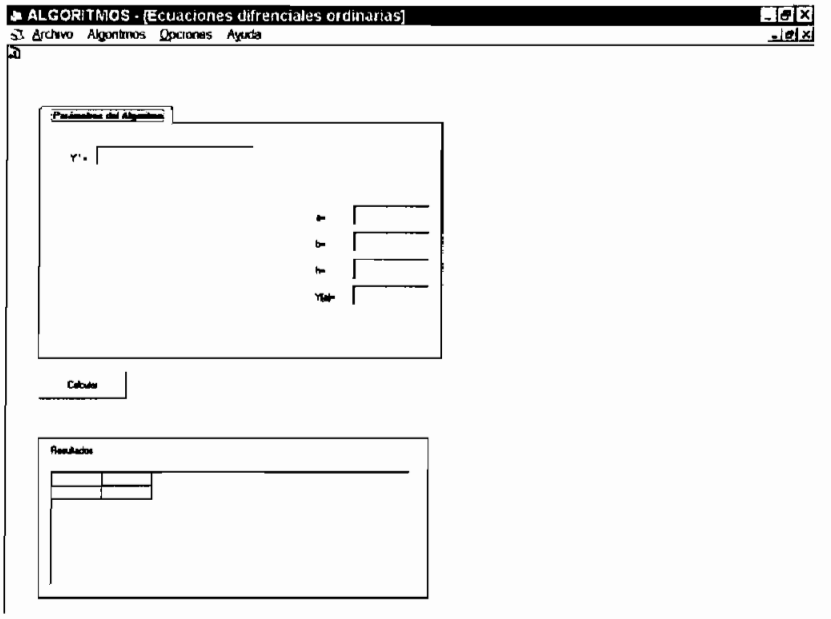
Calcular error = opción que permite el calculo del error siempre y cuando se conozca el valor real de la derivada.

Valor real = Valor real de la derivada.

Debe tomarse en consideración que la elección del número de intervalos solo se produce en al integración de datos ya que en la integración de una función se realizan todos intervalos disponibles es decir hasta seis que corresponden a los diferentes algoritmos así cuando el número de intervalos es 1 la regla se llama Regla del trapecio, si es 2 Regla de 1/3 de Simpson, etc.

ECUACIONES DIFERENCIALES ORDINARIAS

La pantalla será:



En este se implementan los siguientes algoritmos:

- Euler
- Euler modificado
- Heun
- Runge-Kutta de cuarto orden
- Adams-Bashtorth-Moulton
- Milene-Simpson
- Runge-Kutta de cuarto orden (sistemas de ecuaciones diferenciales ordinarias)
- Disparo Lineal

Estos algoritmos permiten resolver ecuaciones diferenciales ordinarias de primer orden y sistemas de ecuaciones diferenciales ordinarias mediante el método de Runge-Kutta. El significado de los parámetros de entrada es:

Y' = función que representa la ecuación diferencial ordinaria de primer orden, Las funciones del sistema a resolver, debe tener la misma sintaxis que la del ingreso de funciones en el módulo de graficación pero puede incluir dos variables t y Y ya que

Y' = dy/dt.

a = Punto inicial del intervalo donde se obtendrá la solución

b = Punto final del intervalo donde se obtendrá la solución.

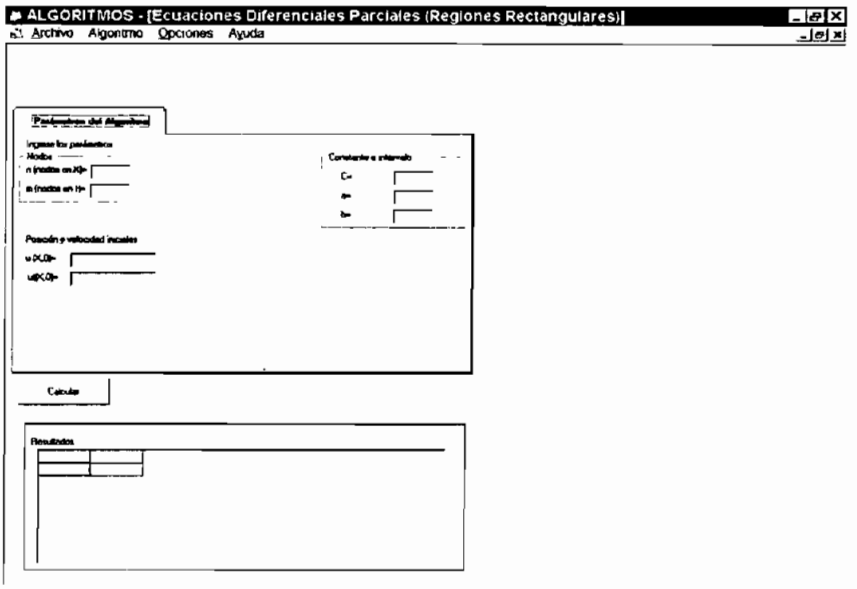
h = incremento con el cual sé hallara la solución a la ecuación diferencial.

Número de EDOs = número de ecuaciones diferenciales a resolver.

En los casilleros vacíos del método de disparo lineal se deben colocar las funciones p, q y r que cumplan con $x''(t)=p(t)X'(t)+q(t)X(t)+r(t)$, p, q y r deben solo ser funciones de t.

ECUACIONES DIFERENCIALES PARCIALES

La interfaz será:



En este se implementan los siguientes algoritmos:

- Ecuación de Laplace (Método de Sobrerelajación Sucesiva SOR)
- Ecuación de Ondas (Método de Diferencias finitas)
- Ecuación del Calor (Método Explicito, Implicito y Crank Nicholson)

Estos métodos permiten resolver ecuaciones diferenciales parciales ya seán estas elípticas, hiperbólicas o parabólicas. El significado de los parámetros de entrada es:

h= incremento con el que se hallará la solución

n= número de nodos en x

m= número de nodos en y

c= constante de la ecuación de ondas y del calor

a= Longitud del intervalo en X

b= Longitud del intervalo en Y

Iteraciones máximas = número de iteraciones máximas que se realizarán

Tolerancia = error que debe tener la respuesta, ingrese en el formato 0.001

u(x,a);u(X,0);u(0,y);(a,y)= condiciones de contorno

Condiciones de Neuman: permiten setear el lugar donde seda la condición de Neuman(arriba, abajo, izquierda, derecha).

u(X,0)= posición inicial

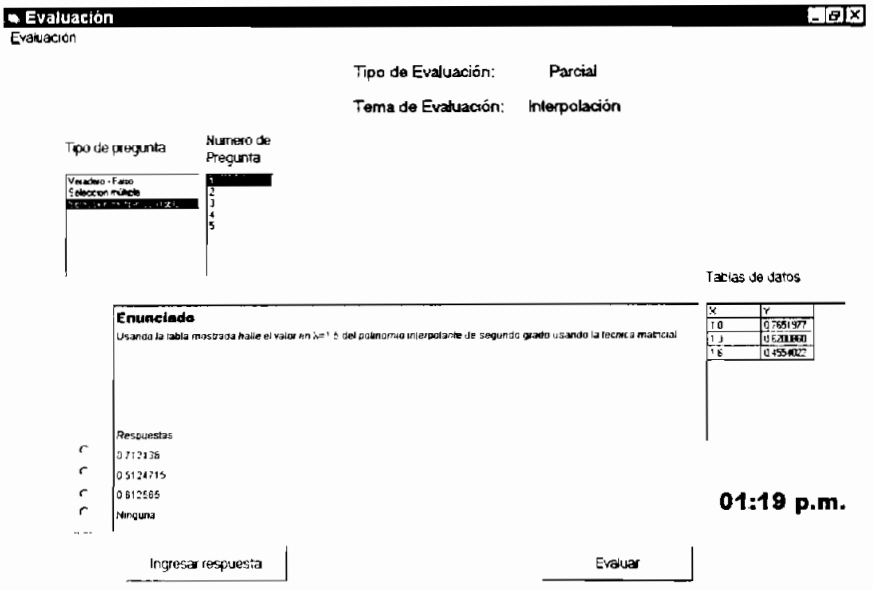
ut(X,0)= velocidad inicial

MÓDULO DE EVALUACIÓN

El módulo de evaluación le permite evaluar sus conocimientos mediante dos tipos de evaluaciones.

- Parcial
- Total

Para ambas se tiene la siguiente pantalla:



EVALUACIÓN PARCIAL

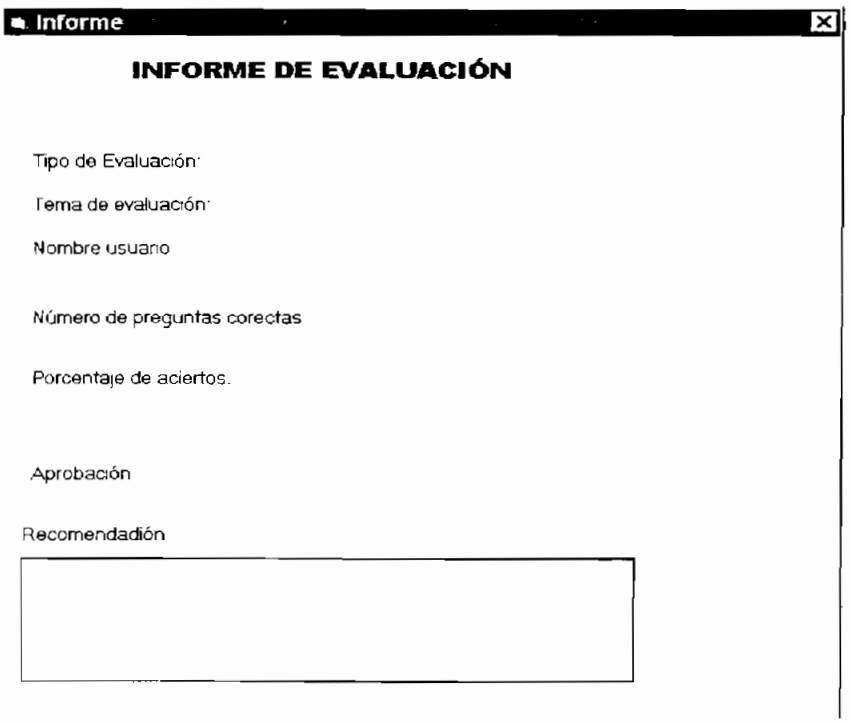
La evaluación parcial le permite evaluarse en cada uno de los temas que comprende el tutorial del Paquete Didáctico de Software para la Enseñaza-Aprendizaje de Métodos Numérico.

En la pantalla usted puede elegir el tema que desea para su evaluación, mediante un clik con el ratón sobre el, presentándose el tipo de pregunta que deberá contestar en las que se encuentran tres tipo:

- Verdadero y Falso
- Selección múltiple
- Selección múltiple con tabla

Para las preguntas de verdadero y falso se presentan dos alternativas una verdadera y otra falsa, en las de selección múltiple y selección múltiple con tabla se presentan cuatro posibilidades, la diferencia radica en la presentación de una tabla para la resolución del problema por parte de las preguntas de selección múltiple con Tabla.

Para este tipo de evaluación usted dispone de una hora para contestar las preguntas luego de lo cual se procede a evaluar sus respuestas. La aprobación de la misma se da si al menos ha contestado correctamente el 70% de las preguntas.
Para tal efecto se le mostrará un informe de evaluación indicándole su aprobación o no de la misma.



EVALUACIÓN TOTAL

La evaluación Total es una evaluación acumulativa de todas y cada uno de los temas que comprende el tutorial del Paquete Didáctico de Software para la Enseñaza-Aprendizaje de Métodos Numérico.

En la pantalla usted puede el tipo de pregunta que deberá contestar en las que se encuentran tres tipo:

- Verdadero y Falso
- Selección múltiple
- Selección múltiple con tabla

Para las preguntas de verdadero y falso se presentan dos alternativas una verdadera y otra falsa, en las de selección múltiple y selección múltiple con tabla se presentan cuatro posibilidades, la diferencia radica en la presentación de una tabla para la resolución del problema por parte de las preguntas de selección múltiple con Tabla.

Para este tipo de evaluación usted dispone de una hora y treinta minutos para contestar las preguntas luego de lo cual se procede a evaluar sus respuestas. La aprobación de la misma se da si al menos ha contestado correctamente el 70% de las preguntas.

Para tal efecto se le mostrará un informe de evaluación indicándole su aprobación o no de la misma que es el mismo del de evaluación parcial.

Encuesta para Adquisición de Requerimientos del Paquete Didáctico para la Enseñanza-Aprendizaje de Métodos Numéricos.

Esta encuesta se origina como parte del Proyecto de Titulación denominado Paquete Didáctico para la Enseñanza-Aprendizaje de Métodos Numéricos dicha encuesta tiene como fin recabar la información necesaria para obtener los requerimientos que deberá contemplar este paquete.
El Paquete Didáctico para la Enseñanza-Aprendizaje de Métodos Numéricos permitirá al usuario facilitar su aprendizaje para lo cual contara con interfaces amigables al usuario que contendrán la sustentación teórica de raíces de funciones no lineales, manipulación de polinomios, sistemas de ecuaciones lineales y no lineales, interpolación estadística y polinomial, resolución de ecuaciones diferenciales ordinarias y diferenciación e integración numérica, así también se implementarán algoritmos de uso frecuente de cada tema de la parte teórica como sigue:

Raíces de Funciones no lineales: Bisección, Falsa Posición, Primer Orden, Newton-Raphson, Secante.
Manipulación de Polinomios: Regla de Horner, Newton-Horner, Newton-Bairstow, Aplicación de Newton-Bairstow.
Sistemas de Ecuaciones Lineales y No Lineales: Eliminación Gaussiana, Gauss-Jordan, Factorización, Jacobi, Gauss-Seidel (para S.E.L), Jacobi, Gauss-Seidel, Newton (para S.E.N.L).
Regresión e Interpolación Polinomial: Técnica de Mínimos Cuadrados (para Regresión), Técnica Matricial de Vandermonde, Polinomio de Interpolación de Lagrange, Polinomio de Interpolación de Newton (Para Interpolación).
Resolución de Ecuaciones Diferenciales Ordinarias: Euler, Euler Modificado, Runge-Kutta (para E.D.O.), Euler, Runge-Kutta (para S.E.D.O.).
Integración Numérica: Método del Rectángulo, Método del Trapecio, Regla de Simpson.

Permitirá realizar evaluaciones de cada uno de los tópicos de la parte teórica con lo cual se podrán restringir el uso de los algoritmos implementados solamente a aquellos tópicos que se hayan aprobado.

También se podrá gráficar cualquier tipo de función o datos que de acuerdo al algoritmo se requiera, la misma estará disponible al inicio y como opción en la interfaz de los algoritmos.

El paquete permitirá opciones tales como guardar, imprimir, borrar ingresos mal realizados en pantalla, continuar o repetir el algoritmo, búsqueda de tópicos en la parte teórica.

Finalmente dispondrá de un acceso mediante palabra clave (password) para cada usuario con el objetivo de que el programa sepa que tópicos de la materia ha aprobado y que algoritmos estarán accesibles para este usuario, este password se pedirá al iniciar la sesión en caso de no disponer del mismo podrá suscribirse sin ningún problema en una opción que aparecerá en esta interfaz.

Nombre: .....

Seleccione la respuesta o complete de acuerdo a su criterio

Parte Teórica

1. Se deben incluir otros temas Si No
   Cuales: .....
   .....
2. El ingreso a los diferentes temas debe ser secuencial Si No
3. El contenido de los temas debe ser profundo Si No
4. Otras características especifíquelas

Parte de Algoritmos

1. Se deben implementar otros algoritmos Si No
   Cuales: .....
   .....
2. Debe restringirse el uso de algoritmos no aprobados Si No
3. Indique que características o rangos (de que tipo de función se puede extraer sus raíces, hasta que grado se puede evaluar el polinomio, hasta que grado debe ser el polinomio interpolante, cuantas ecuaciones simultaneas podrá resolver el algoritmo, el algoritmo de Runge-Kutta de que orden debe ser, etc.) debe cumplir el algoritmo de acuerdo a la función que este realiza.

Raíces de Funciones no lineales:
Bisección

Falsa Posición

Primer Orden

Newton-Raphson

Secante.

Manipulación de Polinomios:
Regla de Horner

Newton-Horner

Newton-Bairstow

Aplicación de Newton-Bairstow

Sistemas de Ecuaciones Lineales y No Lineales:

Para S.E.L

Eliminación Gaussiana

Gauss-Jordan

Factorización

Jacobi

Gauss-Seidel

Para S.E.N.L

Jacobi

Gauss-Seidel

Newton.

Regresión e Interpolación Polinomial:

Para Regresión

Técnica de Mínimos Cuadrados

Para Interpolación

Técnica Matricial de Vandermonde

Polinomio de Interpolación de Lagrange

Polinomio de Interpolación de Newton

Resolución de Ecuaciones Diferenciales Ordinarias:

Para E.D.O.

Euler

Euler Modificado

Runge-Kutta

Para S.E.D.O

Euler

Runge-Kutta

Integración Numérica:
Método del Rectángulo _____

_____

Método del Trapecio _____

_____

Regla de Simpson. _____

_____

4. Se debe mostrar los resultados de las iteraciones como:
   a) Opcional b) Obligatoriamente

5. Otras características especifíquelas

_____

_____

_____

_____

Parte de Evaluación

1. Se debe realizar la evaluación por temas Si No
2. Cuantas preguntas debe tener la evaluación: .....
3. Cuantas deben contestarse correctamente para su aprobación: .....
4. Debe limitarse el tiempo para la evaluación Si No
   Cuanto debe ser este tiempo: .....
5. Debe permitirse regresar a la parte teórica en el momento de la evaluación. Si No

6. La evaluación debe ser objetiva y usando:
   a) Verdadero Falso b) Selección múltiple c) ambas
7. Otras características especifíquelas

_____

_____

_____

_____

Parte de *Graficación*

1. Debe permitir gráficar varias funciones a la vez Si No
   Cuantas: .....
2. Se debe poder cambiar las escalas de graficación Si No
3. Cuantos pares de datos debe permitir graficar: .....
4. Otras características especifíquelas.

_____

_____

_____

_____