ESCUELA POLITECNICA NACIONAL

FACULTAD DE INGENIERIA ELECTRICA

"ANALISIS SIMULACION Y DISEÑO DE SISTEMAS DE CONTROL ASISTIDO POR COMPUTADOR"

ROBERTO GUERRA ROBAYO

TESIS PREVIA A LA OBTENCION DEL TITULO DE INGENIERO EN ELECTRONICA Y CONTROL

MAYO DE 1994

QUITO

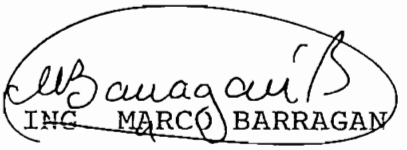
AGRADECIMIENTO

Mis más sinceros agradecimientos al Ing. Marco Barragán por sus acertados consejos y por todo el tiempo dedicado a la dirección de esta tesis.

A todas las personas que contribuyeron para que culmine un ciclo en mi vida, especialmente a mis padres.

CERTIFICACION

CERTIFICO QUE LA PRESENTE TESIS HA SIDO DESARROLLADA POR EL SR. ROBERTO GUERRA ROBAYO BAJO MI DIRECCION.



ING. MARCO BARRAGAN

DIRECTOR DE TESIS

INDICE

CAPITULO I
INTRODUCCION

1.1 INTRODUCCION

El objetivo de este trabajo de tesis es el de desarrollar un programa que ayude al análisis simulación y diseño de sistemas de control lineal invariante en el tiempo, tanto por métodos clásicos como haciendo uso de los métodos modernos.

Se ha venido usando el paquete CC para este propósito, pero este programa presenta ciertas limitaciones ergonómicas entre otras, que serán superadas al desarrollar un nuevo paquete.

Para superar estas limitaciones se desarrolla este programa en C++, con lo que se consigue mayor velocidad y flexibilidad en la ejecución del programa, además el compilador de C usado (Borland C++ 3.0) permite producir aplicaciones de windows, lo que produce ventajas adicionales como son: un sistema multitarea, pantallas redimensionables, ambiente gráfico, uso de comandos comunes a todos los programas de windows (ej. Alt+F4 para cerrar aplicación), barras de menús desplegables, ayudas tipo índice, comunicación mediante cajas de dialogo; el ingreso de matrices se lo hace usando un editor de texto lo que permite un fácil ingreso y edición.

Las opciones del programa se pueden dividir en cinco partes principales que son:

1) Ingreso de funciones de transferencia, y matrices.
2) Presentar la función de transferencia de diferentes maneras como son: en forma de polos y ceros, desarrollada, expandida en fracciones parciales, y obtener la trasformada inversa de Laplace.
3) Permite obtener la respuesta en el tiempo, el lugar geométrico de las raíces, los diagramas de Bode, Nichols, y Nyquist.

4) Para el análisis usando variables de estado, se puede obtener la matriz e^{At}, los valores propios, los vectores propios, la ecuación característica, determinar las matrices de controlabilidad y observabilidad, determinar si el sistema es controlable y observable, y la solución de ecuaciones de estado no homogéneas, invariantes en el tiempo.
5) Ingreso de compensadores y análisis de los efectos que produce en la planta.

El programa esta diseñado para funcionar en computadoras compatibles con IBM, requiere Windows 3.1, 1 Mb de espacio disponible en el disco, aprovecha el uso de un coprocesador matemático, se recomienda usar un computador 386, pero el programa puede funcionar en un computador 286.

En lo referente al contenido de esta tesis, los dos primeros capítulos dan una noción general sobre la parte teórica necesaria para entender, y desarrollar este tema de tesis, no se profundiza en los temas tratados ni se da una justificación matemática de los temas tratados, ya que este no es el objetivo de esta tesis, y sobre ellos existe abundante y clara explicación, en varios libros (ver bibliografía).

El capitulo I trata esencialmente de los sistemas de control en el dominio de la frecuencia, se dan las reglas generales para construir el lugar geométrico de las raíces, así como el criterio de estabilidad de Routh a pesar de no estar desarrollado en el programa, pero es necesario para trazar el lugar geométrico en forma manual; se aborda también los diagramas de Bode, Nyquist, Nichols, así como criterios de estabilidad.

En segundo lugar, se presenta las técnicas básicas de

compensación, en frecuencia, y usando el lugar geométrico de las raíces.

Por último se presenta un breve estudio sobre la respuesta en el tiempo, usando entradas impulso, escalón o rampa.

El capitulo II trata sobre los métodos modernos, se trata las variables de estado, valores y vectores propios, controlabilidad y observabilidad, matriz exponencial y solución de ecuaciones de estado no homogéneas invariantes en el tiempo.

El capitulo III presenta el listado del programa, y la explicación de los algoritmos usados.

En el capitulo IV se dan conclusiones sobre el trabajo desarrollado y se da recomendaciones para complementarlo.

Como anexo para que se pueda manejar el programa con una mayor facilidad se proporciona un manual del usuario, en el cual a manera de tutorial se incluyen algunos ejemplos.

1.2 METODOS DE CONTROL

La teoría de control clásica aborda esencialmente el problema de la estabilidad, empleando los métodos de respuesta en frecuencia o lugar geométrico de las raíces, los que además son utilizados para compensar sistemas lineales de control de lazo cerrado, con una entrada y una salida. Los sistemas compensados satisfacen ciertos requerimientos, estos sistemas producidos son aceptables pero no óptimos.

Las plantas reales usualmente tienen muchas entradas y

muchas salidas; para describir un sistema de este tipo se requiere una gran cantidad de ecuaciones; los métodos clásicos no son aplicables con estos sistemas. Desde los años 60 gracias a la disponibilidad de computadores digitales se pudo analizar sistemas complejos usando variables de estado.

Los desarrollos más recientes de la teoría de control moderna están en el campo del control óptimo de sistemas, tanto determinísticos como estocásticos, así como en sistemas de control complejos con adaptación y aprendizaje. Ahora que las computadoras se han abaratado y reducido en tamaño, estas pueden utilizarse de manera efectiva como parte integral de estos sistemas de control. Las aplicaciones recientes de la teoría de control moderna incluyen sistemas no ingenieriles, como los de biología, biomedicina, economía, y socioecnomía.

Esta tesis se limita a tratar los casos de control clásico y moderno (capítulo II, y III respectivamente) por razones de extensión y complejidad, pero se sugiere el uso de las subrutinas desarrolladas para ampliar el programa, ya que este de ningún modo se puede considerar completo, o que satisfaga todos los requisitos que el usuario pueda tener.

CAPITULO II
CONTROL CLASICO

2. METODOS CLASICOS.

Para el análisis de las funciones de transferencia usando métodos clásicos el programa permite presentar la función de transferencia en forma de polos y ceros, en forma desarrollada, su expansión a fracciones parciales, y obtener la transformada inversa de Laplace de la función de transferencia; en lo que respecta a gráficas se puede obtener la respuesta en el tiempo, el lugar geométrico de las raíces, los diagramas de Bode, Nichols, Nyquist. Todo lo dicho anteriormente es válido tanto para lazo abierto como para lazo cerrado (considerando realimentación unitaria), y con o sin compensación.

Cuando se obtiene la respuesta en el tiempo, la entrada al sistema puede ser una rampa, paso o impulso.

En este capitulo se da una descripción muy breve, a manera de repaso, de las reglas para construir el lugar geométrico de las raíces, y de los procedimientos generales para trazar los diagramas de Bode, Nichols y Nyquist, se incluye el criterio de estabilidad de Routh (a pesar de que el programa no lo realiza) por ser necesario para trazar el lugar geométrico de las raíces de manera manual. El criterio de estabilidad de Nyquist requiere una variación de frecuencia de $-\infty$ a $+\infty$, el programa sólo grafica valores positivos de frecuencia; para graficar los puntos correspondientes a frecuencias negativas trace la imagen espejo respecto al eje real en el diagrama de Nyquist.

No se incluye otros temas de teoría, por cuanto como ya se mencionó anteriormente es una teoría muy conocida, y no es el objetivo de la tesis.

2.1 METODO DEL LUGAR GEOMETRICO DE LAS RAICES.

Podemos hacer un análisis de la calidad de los sistemas de lazo cerrado, basándonos en la ubicación de las raíces de la ecuación característica (1 + G(s)H(s) = 0). Hallar las raíces de la ecuación característica de grado superior es difícil, y requiere resolverse mediante una computadora. Al requerir las raíces de la ecuación característica para diferentes valores de la ganancia el problema se complica aún más ya que el proceso debe repetirse.

W. R. Evans desarrolló un método simple para hallar las raíces de la ecuación característica. Este método denominado método del lugar geométrico de las raíces, consiste en un procedimiento en que se trazan las raíces de la ecuación característica para todos los valores de un parámetro del sistema usualmente real. Se ubican en una gráfica las raíces correspondientes a un valor determinado de este parámetro. El parámetro que se usa en el programa es la ganancia, por ser el parámetro de interés en el análisis de sistemas de control, pero se puede usar cualquier otro variable de la función de transferencia de lazo abierto, encontrando una función de transferencia equivalente.

2.1.1 RESUMEN DE LOS PASOS PARA CONSTRUIR EL LUGAR GEOMETRICO DE LAS RAICES

1. Ubicar los polos y ceros de G(s)H(s) en el plano s. Las ramas del lugar de las raíces parten de los polos de lazo abierto y terminan en los ceros (ceros finitos o ceros en el infinito).
2. Determinar el lugar de las raíces sobre el eje real.
3. Determinar las asíntotas de las ramas del lugar de las raíces.

4. Hallar los puntos de ruptura de partida y de llegada.
5. Determinar el ángulo de partida (ángulo de llegada) del lugar de las raíces desde un polo complejo (hacia un cero complejo).
6. Hallar los puntos donde el lugar de las raíces cruza el eje imaginario, usando el criterio de estabilidad de Routh Hurwitz.
7. Tomando una serie de puntos de prueba en una amplia zona alrededor del origen de s, trazar el lugar de las raíces.
8. Ubicar una raíz de la ecuación característica en una de las ramas del lugar de las raíces y determinar el valor de la ganancia K correspondiente, utilizando la condición de magnitud. O bien, con la condición de magnitud, determinar la ubicación de las raíces de la ecuación característica para un valor determinado de la ganancia K.

Para obtener el lugar geométrico usando el programa, se usa la tecla aceleradora CTRL+G, se desplegará una pantalla con la información necesaria para que se dibuje el lugar geométrico de manera manual, esta información incluye mostrar la función de transferencia en forma de polos y ceros, ángulos de partida (llegada) de polos (ceros) imaginarios, suma de polos y ceros, punto de intersección de asíntotas, ángulos de partida o llegada de asíntotas, puntos de ruptura en el eje real, o en una de las ramas del lugar geométrico de las raíces, con sus respectivas ganancias; en la pantalla principal se gráfica el lugar geométrico.

Para obtener la ganancia en un punto, se ubica el mouse en el punto deseado, y se presiona el botón izquierdo del mouse, el punto junto con su ganancia se indicarán en la esquina superior izquierda de la pantalla.

Para determinar los puntos que corresponden al lugar geométrico de las raíces, se determinan los puntos de ruptura con el eje real (haciendo dK/ds =0), y a partir de este punto, o de los polos o ceros complejos, se hace una busqueda del punto que cumple la condición de ángulo = 180°. El lugar de las raíces en el eje real es trazado, uniendo polos y ceros consecutivos en el eje real, si el número de polos y ceros a la izquierda del punto es impar.

El resto de la información necesaria para trazar el lugar geométrico se la obtiene de manera muy simple y no se considera necesario describirla aquí.

2.2 CRITERIO DE ESTABILIDAD DE ROUTH

Un sistema de control es estable si y sólo si, todos los raíces de la ecuación característica están ubicadas en el semiplano izquierdo del plano s, excluyendo el eje imaginario.

Para funciones de transferencia de lazo cerrado de la forma

$$\frac{C(s)}{R(s)} = \frac{b_0 s^m + b_1 s^{m-1} + \ldots + b_{m-1} s + b_m}{a_0 s^n + a_1 s^{n-1} + \ldots + a_{n-1} s + a_n} = \frac{B(s)}{A(s)} \qquad (1.1)$$

donde a y b son constantes y $m \leq n$, primeramente se debe factorar el polinomio A(s) para hallar los polos de lazo cerrado. Para evitar hallar las raíces del polinomio A(s) existe un criterio sencillo, denominado criterio de estabilidad de Routh, que permite determinar la cantidad de polos de lazo cerrado que hay en el semiplano derecho del plano s.

El criterio de estabilidad de Routh dice si hay o no raíces con parte real positiva en una ecuación polinómica, sin

tener que resolverla. Cuando se aplica a un sistema de control se puede tener información sobre estabilidad absoluta directamente a partir de los coeficientes de la ecuación característica.

El procedimiento es como sigue:

1. El polinomio en s de grado n se escribe de la siguiente forma:

$$a_0 s^n + a_1 s^{n-1} + \ldots + a_{n-1} s + a_n = 0 \qquad (1.2)$$

donde los coeficientes son cantidades reales. Se supone que $a_n \neq 0$; es decir, cualquier raíz nula ha sido eliminada.

2. Si cualquiera de los coeficientes son nulos o negativos en presencia de un coeficiente positivo al menos, hay una raiz o raíces imaginarias, o que tienen partes reales positivas. Por lo tanto en tal caso el sistema no es estable. Si solamente interesa la estabilidad absoluta, no hay necesidad de seguir adelante con el procedimiento.

3. Si todos los coeficientes son positivos, se colocan en filas y columnas de acuerdo al siguiente esquema:

s^n	a_0	a_2	a_4	a_6 ...
s^{n-1}	a_1	a_3	a_5	a_7 ...
s^{n-2}	b_1	b_2	b_3	b_4 ...
s^{n-3}	c_1	c_2	c_3	c_4 ...
s^{n-4}	d_1	d_2	d_3	d_4 ...
		. . .		
s^2	e_1	e_2		
s^1	f_1			
s^0	g_1			

Los coeficientes b_1, b_2, b_3, etc se evalúan como sigue;

$$b_1 = (a_1 a_2 - a_0 a_3)/a_1$$
$$b_2 = (a_1 a_4 - a_0 a_5)/a_1$$
$$b_3 = (a_1 a_6 - a_0 a_7)/a_1$$

La evaluación de b_i continua hasta que todos los coeficientes b_i sean cero. En las filas siguientes se sigue el mismo procedimiento de multiplicación cruzada de los

coeficientes para la evaluación de las c, d, e, etc. Es decir,

$$c_1 = (b_1a_3 - a_1b_2)/b_1$$
$$c_2 = (b_1a_5 - a_1b_3)/b_1$$
$$c_3 = (b_1a_7 - a_1b_4)/b_1$$

Este proceso continua hasta el renglón n-ésimo. El conjunto completo de coeficientes es triangular. Nótese que al elaborar el conjunto, una fila completa se puede multiplicar o dividir por un número positivo para simplificar los cálculos subsiguientes sin alterar la conclusión de estabilidad.

El criterio de estabilidad de Routh establece que la cantidad de raíces de la ecuación (1-2) con partes reales positivas es igual a la cantidad de cambios de signo en los coeficientes de la primera columna del conjunto. Nótese que no se necesita conocer los valores exactos de los términos de la primera columna; en su lugar solo se requieren los signos. La condición necesaria y suficiente para que todas las raíces de la ecuación (1-2) queden en el semiplano izquierdo del plano s, es que todos los coeficientes de la ecuación (1-2) sean positivos y que todos los términos de la primera columna del conjunto tengan signos positivos.

Existen casos especiales que no serán tratados, para información sobre estos ver la referencia [1].

2.3 METODOS DE RESPUESTA EN FRECUENCIA

Por respuesta en frecuencia se entiende la respuesta en estado estacionario de un sistema, ante una entrada sinusoidal. En los métodos de respuesta de frecuencia, que disponen los ingenieros de control para el análisis y diseño de sistemas; se varia la frecuencia de la señal de entrada dentro de un rango de interés, manteniendo constante la magnitud de la señal de entrada, y se estudia la respuesta resultante.

La función de transferencia sinusoidal, una función compleja de la frecuencia w, queda caracterizada por su

magnitud y ángulo de fase, con la frecuencia como parámetro. Hay tres representaciones utilizadas comúnmente de las funciones de transferencia sinusoidal, que son:

1. Diagrama de Bode o diagrama logarítmico.
2. Diagrama polar o de Nyquist.
3. Diagrama del logaritmo de la magnitud en función de la fase o de Nichols.

2.3.1 DIAGRAMA DE BODE O DIAGRAMA LOGARÍTMICO

Una función de transferencia sinusoidal se puede representar con dos diagramas separados, uno de la magnitud en función de la frecuencia y el otro del ángulo de fase (en grados) en función de la frecuencia. El diagrama de Bode consiste en dos gráficas: una es la representación del logaritmo de la magnitud de la función de transferencia sinusoidal; la otra es un diagrama del ángulo de fase, ambos en función de la frecuencia en escala logarítmica.

La representación habitual o normalizada de la magnitud logarítmica de $G(jw)$ es $20 \log|G(jw)|$, con 10 como base de logaritmos. La unidad utilizada en esta representación es el decibelio, abreviado usualmente como db. En la representación logarítmica, se trazan las curvas, utilizando la escala logarítmica para la frecuencia y la escala lineal ya sea para la magnitud (pero en db) o para el ángulo de fase en grados.

La ventaja principal de realizar un diagrama logarítmico, es que la multiplicación de magnitudes se convierte en suma. Además, se dispone de un procedimiento sencillo para bosquejar una curva del logaritmo de la magnitud en forma aproximada. Esa aproximación se basa en rectas asintóticas y es suficiente si sólo se requiere información superficial sobre las características de la respuesta en frecuencia.

La representación logarítmica es útil debido a que presenta las características de alta y baja frecuencia de la función de transferencia en un solo diagrama. Una gran ventaja

es la posibilidad de extender el rango de bajas frecuencias al utilizar una escala logarítmica, pues en los sistemas prácticos las características de baja frecuencia son muy importantes. Aún cuando no es posible trazar curvas hasta frecuencia cero debido a la frecuencia logarítmica (log $0^+ \rightarrow -\infty$) esto no constituye un problema serio.

Si $|G(jw)|$ tiene un valor pico en alguna frecuencia, esta se denomina frecuencia de resonancia. La magnitud de $G(jw)$ en la frecuencia de resonancia se denomina máximo de resonancia.

Procedimiento general para trazar el diagrama de bode

Primero se reescribe la función de transferencia sinusoidal $G(jw)H(jw)$ como un producto de factores (de primer o segundo grado). Luego se identifican las frecuencias de cruce asociadas con esos factores básicos. Finalmente se trazan las curvas asintóticas del logaritmo de la magnitud con las pendientes adecuadas entre las frecuencias de cruce. La curva exacta, que es próxima a la curva asintótica, se puede obtener agregando las correcciones apropiadas.

La curva de ángulo de fase de $G(jw)$ se puede dibujar sumando las curvas de ángulo de fase de los factores individuales.

El uso de aproximaciones asintóticas exige menos tiempo que otros métodos que pueden usarse para calcular la respuesta en frecuencia de una función de transferencia. Las principales razones por las que en la práctica se usan extensamente los diagramas de Bode, son la facilidad de trazar las curvas de respuesta en frecuencia para una función de transferencia dada y la facilidad de modificación de la curva de respuesta de frecuencia, cuando se agregan compensadores.

Basándose en las frecuencias de corte de las curvas asintóticas se ha escogido la escala automática de frecuencias como, diez veces mas pequeña que el polo o cero más pequeño, la frecuencia mínima, y como diez veces más grande que el polo

o cero más grande, la frecuencia máxima.

Para trazar el diagrama de bode, se reemplaza en la función de transferencia s por jw, y se procede a evaluarla para distintos valores de frecuencia usando aritmetica compleja. Usando sencillas transformaciones obtenemos la parte real, imaginaria, modulo y ángulo de la función de transferencia, esta información es suficiente para obtener todos los diagramas en frecuencia.

El programa no presenta algunas opciones para obtener el diagrama de Bode, pues se puede escoger entre:

- Escala manual o automática.
- Lazo abierto o cerrado.
- Modulo, ángulo o los dos a la vez.
- Sistema compensado o no compensado.

Para acceder a estas opciones presione la tecla aceleradora CTRL+B, se muestra una caja de dialogo con las opciones indicadas anteriormente; esta caja de dialogo sólo se presenta en el diagrama de Bode, pero las opciones seleccionadas son guardadas para usarse en los diagramas de Nichols o Nyquist.

2.3.2 DIAGRAMA POLAR O DE NYQUIST

El diagrama polar de una función de transferencia sinusoidal G(jw) es un diagrama de la magnitud de G(jw) en función del ángulo de fase de G(jw) en coordenadas polares al variar el valor de w de 0° a +∞. Nótese que en los diagramas polares, el ángulo de fase se mide como positivo (o negativo) en sentido antihorario (horario) desde el eje real positivo. Cada punto en el diagrama polar de G(jw) representa el extremo terminal de un vector para un valor determinado de w. En el diagrama polar, es importante indicar que las proyecciones de G(jw) sobre los ejes real e imaginario, son sus componentes real e imaginario. Como es fácil construir el diagrama logarítmico, los datos para construir el diagrama polar se

pueden obtener directamente de él si los decibelios se convierten en relaciones normales de magnitud. Se puede utilizar una computadora para obtener |G(jw)| y /G(jw) con gran exactitud para diversos valores de w en el rango de interés.

Una ventaja de utilizar un diagrama polar es que presenta las características de respuesta en frecuencia de un sistema en todo el rango de frecuencias, en un solo diagrama, la desventaja es que el mismo no indica claramente las contribuciones de cada factor individual de la función de transferencia de lazo abierto.

Para trazar el diagrama de Nyquist con ayuda del programa presione la tecla aceleradora CTRL+Y, el diagrama se presentará en la pantalla principal, la escala es automática, a no ser que hayamos graficado un diagrama de Bode con escala manual previamente, en tal caso la escala es la ingresada manualmente.

2.3.3 DIAGRAMA DEL LOGARITMO DE LA MAGNITUD EN FUNCIÓN DE LA FASE O DIAGRAMA DE NICHOLS

Otro modo de presentar gráficamente las características de respuesta en frecuencia, es utilizar el diagrama del logaritmo de la magnitud en decibelios en función de la fase. Este diagrama se denomina de Nichols.

En el diagrama de Bode, las características de respuesta en frecuencia de G(jw) aparecen en papel semilogarítmico como dos curvas separadas: la del logaritmo de la magnitud y la del ángulo de fase, en tanto que en el diagrama de Nichols, las dos curvas del diagrama de Bode se combinan en una sola. El logaritmo de la magnitud en función de la fase se puede trazar fácilmente, leyendo los valores del logaritmo de la magnitud y del ángulo de fase del diagrama de Bode, teniendo como parámetro w.

Las ventajas de este diagrama son que se puede determinar

rápidamente la estabilidad relativa del sistema de lazo cerrado, y que la compensación se puede establecer fácilmente.

Si presiona la tecla aceleradora CTRL+N en el programa, se obtiene el diagrama de Nichols.

2.3.4 CRITERIO DE ESTABILIDAD DE NYQUIST

Este criterio relaciona la respuesta de frecuencia de lazo abierto G(jw)H(jw) a la cantidad de polos y ceros de 1 + G(jw)H(jw) que hay en el semiplano derecho s. El criterio de Nyquist es útil porque permite determinar gráficamente de las curvas de respuesta de lazo abierto la estabilidad absoluta del sistema de lazo cerrado, sin determinar los polos de lazo cerrado. Las curvas se pueden obtener analíticamente o experimentalmente; esto es muy útil cuando no conocemos la expresión matemática de un sistema.

Reglas para determinar la estabilidad según el criterio de estabilidad de Nyquist

1. Este criterio se puede expresar como

$$Z = N + P$$

donde Z = cantidad de ceros de 1 + G(s)H(s) en el semiplano derecho del plano s (encerrado por el contorno de Nyquist).
N = cantidad de rodeos alrededor del punto -1 + j0 en sentido horario.
P = cantidad de polos de G(s)H(s) en el semiplano derecho del plano s

Si P no es cero, para que un sistema de control sea estable, se debe tener Z = 0, o N = -P, lo que significa que hay que tener p rodeos antihorarios alrededor del punto -1 + j0.

Si G(s)H(s) no tiene polos ni ceros en el semiplano

derecho del plano s, entonces Z = N. Por lo tanto para que haya estabilidad, no debe haber rodeos alrededor del punto -1 + j0 por parte de la gráfica G(jw)H(jw). En este caso no es necesario considerar la gráfica para el eje jw completo, pues basta solamente con la porción de frecuencia positiva. La estabilidad del sistema se puede determinar viendo si el punto -1 + j0 queda rodeado por el diagrama de Nyquist. Para que haya estabilidad, el punto -1 + j0 no debe quedar rodeado.

2. Se debe tener mucho cuidado al verificar la estabilidad de sistemas con lazos múltiples, ya que pueden incluir polos en el semiplano derecho del plano s. Nótese que aunque un lazo interior sea inestable, se puede hacer que todo el sistema de lazo cerrado sea estable con un diseño adecuado. Para determinar la inestabilidad de sistemas con múltiples lazos múltiples no basta la simple inspección de los rodeos alrededor del punto -1 + j0 por la gráfica G(jw)H(jw). En estos casos sin embargo, se puede determinar fácilmente si hay o no algún polo de 1 + G(S)H(S) en el semiplano derecho del plano s, al aplicar el criterio de estabilidad de Routh al denominador de G(s)H(s).

3. Si el lugar de G(jw)H(jw) pasa por el punto -1 + j0, hay raíces de la ecuación característica, ubicados sobre el eje jw. Esto no es deseable para sistemas de control prácticos. En un sistema de control de lazo cerrado bien diseñado, ninguna de las raíces de la ecuación característica debe quedar sobre el eje jw.

2.4 COMPENSACIÓN

Por compensación se entiende la modificación de la dinámica del sistema para satisfacer unas especificaciones requeridas. Las especificaciones están relacionadas con la exactitud, estabilidad relativa y velocidad de respuesta.

Una variación en la ganancia puede ser un procedimiento útil para lograr un mejor funcionamiento; sin embargo en muchos casos esto no es suficiente, en este caso se requiere

aumentar un compensador diferente.

Los compensadores pueden tener básicamente 2 esquemas, la figura 1.2 muestra el llamado compensador en serie; la figura 1.1 muestra el compensador en la realimentación o paralelo.

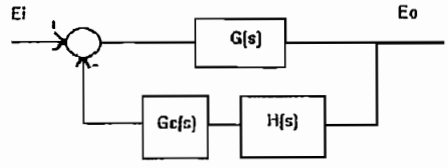


Figura 1.1

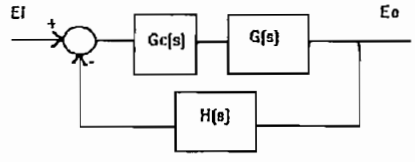


Figura 1.2

Si aplicamos una entrada sinusoidal e_i a la entrada de una red (ver figura 1.1), y la salida estacionaria e_o tiene un adelanto de fase, entonces, se denomina red de adelanto. Si la salida tiene un atraso de fase, entonces se llama red de atraso. En una red de atraso adelanto, se produce a la salida tanto atraso como adelanto de fase, pero en diferentes regiones de frecuencia; el atraso se da en bajas frecuencias y el adelanto en alta frecuencia. Si el compensador tiene características de red de adelanto, red de atraso, o red de atraso-adelanto, se denomina compensador de adelanto, compensador de atraso, o compensador de atraso-adelanto.

Esta sección se limita a tratar los compensadores en atraso, adelanto, y atraso-adelanto; estos en el esquema de compensación en serie.

2.4.1 COMPENSACION EN ADELANTO

Un compensador en adelanto tiene la siguiente función de transferencia

$$K_c\alpha \frac{Ts+1}{\alpha Ts+1} = K_c \frac{s+1/T}{s+1/\alpha T} \quad (0<\alpha<1) \qquad (1.3)$$

Al ser $\alpha < 1$, el cero siempre esta ubicado a la derecha del polo en el plano complejo. El valor mínimo de α está limitado por la construcción física de la red de adelanto; se toma usualmente α mínimo de 0.07. Cuando α es pequeño, es necesario un amplificador para compensar la atenuación de la red de adelanto.

Trazando el diagrama de bode de esta función de transferencia observamos que se comporta como un filtro pasa altos. Las frecuencias de cruce del compensador son en $w = 1/T$ y $w = 1/\alpha T$.

La compensación en adelanto básicamente aumenta el ancho de banda, acelera la respuesta y disminuye el sobreimpulso

máximo en la respuesta escalón.

Compensación en adelanto basadas en el método del lugar de las raíces.

Para el diseño del compensador de adelanto se sigue el siguiente procedimiento:

1. Con las especificaciones de funcionamiento, se determina la posición deseada de los polos dominantes de lazo cerrado.
2. Dibujando el lugar de las raíces se determina si un ajuste de ganancia, puede ser suficiente o no para lograr los polos deseados. Si no se logra se calcula el ángulo faltante, que debe ser provisto por la red de adelanto.
3. Si no se especifican coeficientes de error estático, se determina la ubicación del polo y cero de la red de adelanto, para que contribuya con el ángulo necesario. Si se especifica un coeficiente estático de error, es conveniente usar el método de respuesta de frecuencia.
4. Se determina la ganancia de lazo abierto del sistema compensado, con la condición de modulo.

Este procedimiento de compensación se usa cuando las especificaciones en el domino del tiempo, como máximo sobreimpulso, tiempo de establecimiento etc.

Compensación en adelanto basadas en el método de respuesta de frecuencia.

1. Determinar la ganancia de lazo abierto K para satisfacer los coeficientes de error.
2. Usando la ganancia K así determinada, calcular el margen de fase del sistema no compensado.
3. Determinar el ángulo de fase ϕ necesario que debe ser agregado al sistema.
4. Determinar el factor de atenuación α que cumpla la ecuación $\sin(\phi)=(1-\alpha)/(1+\alpha)$. Determinar la magnitud a la cual la magnitud del sistema no compensado es igual a -20

log($1/\sqrt{\alpha}$). Fijar esta frecuencia como la nueva frecuencia de transición de ganancia. Esta frecuencia corresponde a w_m y el de fase máximo ϕ_m se produce a esta frecuencia.

5. Determinara la frecuencia de transición de la red de adelanto de

$$w = 1/T, \qquad w = 1/\alpha T$$

6. Se inserta un amplificador con ganancia igual a $1/\alpha$.

2.4.2 COMPENSACION EN ATRASO

El compensador en atraso tiene la siguiente función de transferencia

$$K_c\beta \frac{Ts+1}{\beta Ts+1} = K_c \frac{s+1/T}{s+1/\beta T} \quad (\beta>1) \qquad (1.4)$$

Al ser $\beta > 1$ el polo está ubicado a la derecha del cero.

Trazando el diagrama de bode de esta función de transferencia observamos que se comporta como un filtro pasabajos. Las frecuencias de cruce del compensador en atraso están en $w = 1/T$ y $w = 1/(\beta T)$.

La compensación en atraso aumenta la ganancia en baja frecuencia y así mejora la exactitud en estado estacionario del sistema, pero reduce la velocidad de respuesta debido al reducido ancho de banda.

Compensación en atraso basadas en el método del lugar de las raíces.

Se usa este método cuando el sistema presenta características satisfactorias de respuesta transitoria, pero no en estado estacionario. El procedimiento de diseño en este caso es como sigue:

1. Trazar el diagrama del lugar de las raíces para el sistema sin compensación. Ubicar los polos de lazo cerrado deseados en el lugar de las raíces.
2. Determinar la ganancia de lazo abierto utilizando la condición de modulo.
3. Evaluar el coeficiente de error particular especificado en el problema.
4. Determinar el valor de incremento en el coeficiente de error necesario para satisfacer las especificaciones.
5. Determinar el polo y cero de la red de retardo que produce el incremento necesario en el coeficiente de error de que se trate, sin alterar apreciablemente los lugares de las raíces originales.
6. Trazar un nuevo lugar de las raíces para el sistema compensado. Ubicar los polos dominantes deseados de lazo cerrado en el lugar de las raíces. Si la contribución angular de la red de retardo es pequeña, los dos lugares de las raíces son casi idénticos, luego situar los polos dominantes de lazo cerrado deseados, basados en las especificaciones de respuesta transitoria.
7. Ajustar la ganancia del amplificador de la condición de módulo, de manera que los polos de lazo cerrado dominantes queden en las ubicaciones deseadas.

Compensación en atraso basada en el método de respuesta de frecuencia.

La función de una red de retardo es brindar atenuación en el rango de alta frecuencia para dar a un sistema suficiente margen de fase. El procedimiento para el diseño de un compensador en atraso es como sigue:

1. Determinar la ganancia de lazo abierto, que satisfaga el requisito de coeficiente de error determinado.
2. Con la ganancia así determinada trazar el diagrama de Bode del sistema no compensado y determinar los márgenes de fase y ganancia del sistema no compensado.
3. Si las especificaciones respecto a márgenes de fase y ganancia no son satisfactorios, hallar el punto de

frecuencia donde el ángulo de fase de la función de transferencia de lazo abierto, es igual a -180° más el margen de fase requerido. El margen de fase requerido es el margen de fase especificado más 5 a 12°. (Estos 5 a 12° compensan el retardo de fase de la red de retardo.) Se elige esta frecuencia como nueva frecuencia de transición de ganancia.

4. Se elige la frecuencia de corte w = 1/T, una década por debajo de la nueva frecuencia de transición de ganancia.
5. Determinar la atenuación necesaria para reducir la curva de amplitud a 0 db en la nueva frecuencia de transición de ganancia. Notando que esta atenuación es -20 log ß, determinar el valor de ß. Entonces la otra frecuencia de corte, es determinada de w = 1/ßT.

2.4.3 COMPENSACION EN ATRASO-ADELANTO

Cuando deseamos mejorar tanto la respuesta en estado estacionario como la transitoria, se debe usar un compensador en adelanto y uno en atraso a la vez. Es más económico (para el caso continuo) introducirlos como un solo elemento que hacerlo como elementos separados.

La compensación en atraso-adelanto combina las ventajas de los compensadores en atraso y adelanto. Incrementa el orden del sistema en dos a menos que haya una cancelación de polos y ceros.

El compensador en atraso adelanto esta dado por

$$K_c \left(\frac{s+1/T_1}{s+\gamma/T_1}\right) \left(\frac{s+1/T_2}{s+1/\beta T_2}\right) \qquad (1.5)$$

Donde $\gamma > 1$ y $\beta > 1$. Con frecuencia se elige $\gamma = \beta$; es este caso la frecuencia a la cual el ángulo de fase es cero está dado por

$$w_1 = 1/\sqrt{(T_1 T_2)} \qquad (1.6)$$

Compensación en atraso-adelanto basada en el método del lugar de las raíces.

El procedimiento para el diseño de un compensador en atraso adelanto es:

1. De las especificaciones dadas, determinar la ubicación deseada de los polos dominantes de lazo cerrado.
2. Para tener los polos dominantes de lazo cerrado en las ubicaciones deseadas, calcular las contribución angular ϕ necesaria de la porción de adelanto de fase de la red de atraso-adelanto.
3. Usando la ecuación 1.5, determinar la constante K_c del requisito de coeficiente de error determinado especificado en el problema de diseño.
4. Para el compensador en atraso-adelanto, elegir T_2 suficientemente grande como para que

$$(s_1 + 1/T_2)/(s_1 + 1/\beta T_2) \qquad (1.7)$$

sea aproximadamente unitaria, donde $s = s_1$ es uno de los polos dominantes de lazo cerrado.

Determinar los valores de T_1 y β de los requisitos de modulo y angulo.

5. Utilizando el valor de β determinado, elegir T_1 de manera que el modulo de la ecuación 1.7 sea 1 y el angulo este entre 0 y 3°. El valor de βT_2, la constante de tiempo más grande de la red de atraso-adelanto, no debe ser demasiado elevada, para que sea realizable físicamente.

Compensación en atraso basada en el método de respuesta de frecuencia.

El diseño de un compensador en atraso-adelanto está basado en una combinación de las técnicas de diseño tratadas en la compensación de adelanto y en la de atraso. El valor de a para la red de adelanto debe ser $1/\beta$, hecho así se pueden combinar los compensadores de adelanto y atraso diseñados

individualmente, para producir un único compensador en atraso-adelanto.

El programa sólo permite el ingreso de la función de transferencia del compensador (no compensa automáticamente) por lo que el diseño del compensador se lo deberá hacer de manera manual por el usuario.

Los comandos que requieren el uso del sistema compensado multiplican el compensador por la funcion de transferencia y usan esta función como la nueva funcion de transferencia.

2.5 RESPUESTA EN EL TIEMPO

La función de trasferencia G(s) para un sistema lineal invariante en el tiempo, es:

$$G(s) = Y(s) / X(s) \qquad (1-3)$$

siendo X(s) la transformada de Laplace de la entrada y Y(s) la transformada de Laplace de la salida. La salida se pude escribir como:

$$Y(s) = G(s)\, X(s) \qquad (1-4)$$

Para obtener la respuesta en el tiempo de un sistema con condiciones iniciales 0, tenemos que encontrar la transformada inversa de Laplace de la ecuación (1-4). Para obtener la respuesta al impulso del sistema en lazo abierto se obtiene la transformada inversa de Laplace de G(s), si la entrada es una función paso se obtendrá la transformada de Laplace de G(s)/s, y en el caso de una rampa de $G(s)/s^2$.

Cuando se analiza el comportamiento en lazo cerrado la función G(s) debe ser reemplazada por G(s)/[1 + G(s)H(s)] y proceder como se menciono en el párrafo anterior.

Para obtener la respuesta en el tiempo usando el programa se presiona la tecla aceleradora CTRL+T, se presenta una caja

de dialogo en la que puede indicar el tipo de entrada, escoger entre escala automática y manual, e indicar si se va a analizar en lazo abierto o cerrado.

En general la entrada a un sistema de control no puede conocerse previamente, ya que es de naturaleza aleatoria. Debido a esto se usan señales de prueba típicas ya que existe una relación entre las características de respuesta de un sistema a una señal de prueba típica, y la capacidad de un sistema para atender las señales de entrada reales que se le presentan.

La selección de la señal de entrada a utilizar para analizar las características de un sistema, depende de la forma de las señales de entrada más habituales a que el sistema estará sometido en condiciones normales de operación. Si las entradas a un sistema de control son funciones que cambian gradualmente en el tiempo, la señal adecuada para esta prueba debe ser la rampa. Si un sistema esta sometido a perturbaciones súbitas una buena señal de prueba puede ser el escalón, y si el sistema está sujeto a variaciones bruscas, la mejor señal es un impulso.

El programa analíticamente computa la transformada inversa de Laplace y numéricamente calcula la respuesta en el tiempo substituyendo los valores de tiempo a ser graficados en la formula de la transformada de Laplace, la opción escala automática escoge el tiempo máximo a graficar basándose en la dominancia de polos, el tiempo máximo es 5 veces la constante de tiempo más grande del sistema; para una explicación más detallada del algoritmo se encuentra en el listado del programa.

CAPITULO III
CONTROL MODERNO

3. METODOS MODERNOS

Un sistema real puede tener varias entradas y salidas relacionadas entre si, en forma muy complicada. Para analizar un sistema con estas características, se requiere reducir la complejidad de las expresiones matemáticas, así como recurrir a computadoras, para resolver los cálculos tediosos. Desde este punto de vista, el método más adecuado para el análisis de estos sistemas, es el método en el espacio de estado.

Mientras la teoría de control convencional se basa en la relación entre la entrada y la salida, o función de transferencia, la teoría de control moderna se aplica a sistemas de múltiples entradas y múltiples salidas, que pueden ser lineales o no lineales, variables o invariables en el tiempo.

En modelos lineales el uso de la notación matricial, simplifica mucho la representación de sistemas de ecuaciones. El aumento en la cantidad de variables de estado, de entradas o salidas, no incrementa la complejidad de ecuaciones. De hecho, es posible proseguir el análisis de sistemas complicados, con entradas y salida múltiples, con procedimientos ligeramente más complicados que los requeridos por el análisis de ecuaciones diferenciales escalares de primer orden.

El programa desarrollado, sólo permite el análisis de sistemas lineales e invariantes en el tiempo; dentro de estos sistemas se puede obtener los valores y vectores propios, el polinomio característico, la matriz exponencial, la solución de ecuaciones de estado homogéneas y no homogéneas, la matriz de controlabilidad, y la de observabilidad, así como la determinación de controlabilidad y observabilidad.

En este capítulo se revisa brevemente la representación de sistemas en el espacio de estado, valores y vectores propios, matriz exponencial, solución de ecuaciones de estado, controlabilidad y observabilidad. Además se indica como usar

las herramientas del programa para analizar sistemas descritos en variables de estado.

3.1 REPRESENTACIÓN DE SISTEMAS EN EL ESPACIO DE ESTADO.

Se puede describir un sistema dinámico, consistente en un número finito de elementos concentrados, por ecuaciones diferenciales ordinarias, en las que el tiempo es la variable independiente. Utilizando la notación vectorial matricial, se puede expresar una ecuación diferencial de enésimo orden por una ecuación diferencial vectorial-matricial de primer orden. En esta sección se presentan los métodos de obtención de representaciones en espacio de estado de sistemas continuos en el tiempo.

Un sistemas de ecuaciones diferenciales lineales de orden n en los que la función excitadora incluye términos derivativos se puede escribir de la forma

$$y^{(n)} + a_1 y^{(n-1)} + \ldots + a_{n-1} y^{(n)} + a_n y^{(n-1)} = b_0 u + b_1 u + \ldots + b_{n-1} u + b_n u$$

donde $y^{(n)}$ representa la derivada enésima de la salida del sistema

$b^{(n)}$ es la derivada enésima de la función excitadora.

La representación en el espacio de estado de la ecuación anterior es

$$\begin{array}{cccccccc}
x_1 & 0 & 1 & 0 & \ldots & 0 & x_1 & \beta_1 \\
x_2 & 0 & 0 & 1 & \ldots & 0 & x_2 & \beta_2 \\
\cdot & \cdot & & & & & \cdot & \cdot \\
\cdot & \cdot & & & & & \cdot & \cdot \\
\cdot & \cdot & & & & & \cdot & \cdot \\
x_{n-1} & 0 & 0 & 0 & \ldots & 1 & x_{n-1} & \beta_{n-1} \\
x_n & a_n & a_{n-1} & a_{n-2} & \ldots & a_1 & x_n & \beta_n
\end{array}$$

$$y = 1\ 0\ \ldots\ 0 \begin{matrix} x_1 \\ x_2 \\ \ldots \\ x_n \end{matrix} + \beta_0\, u$$

o en forma abreviada:

$$\dot{x} = Ax + Bu$$
$$y = Cx + Du$$

Para analizar sistemas descritos por variables de estado, usando el programa se debe ingresar las matrices A, B, C, D, el vector de entrada u y el vector de condiciones iniciales x0.

Para el ingreso de los valores se presiona la tecla aceleradora ALT+A y se escoge el comando matrices, con esto se abrirá un editor de texto, en el cual se ingresan los datos.

Los datos se los ingresa en la siguiente forma:

- Dimensiones de las matrices n, r, m. Siendo A de n x n, B de n x r, C de m x n, D de m x r.
- Coeficientes por filas de las matrices A, B, C, D.
- Vector de entradas u.
- Vector de valores iniciales x0.
- Tipos de entradas, donde 0 representa una entrada impulso, 1 una entrada paso, y 2 una entrada rampa.

3.2 VALORES y VECTORES PROPIOS.

VALORES PROPIOS

Los valores propios de una matriz A de n x n, son las raíces de la ecuación característica

$$|\lambda I - A| = 0$$

Si las ecuaciones de estado se representan en forma canónica asociada, los coeficientes de la ecuación

característica vienen dados sin dificultad por los elementos de la última fila de los elementos de la matriz A.

Otra propiedad importante de la ecuación característica y de los valores propios es que éstos son invariantes en una transformación no singular. En otras palabras, cuando la matriz A se transforma mediante una transformación no singular $x = Py$, de manera que

$$A' = P^{-1}AP$$

Entonces la ecuación característica y los valores propios de A' son idénticos a los de A.

VECTORES PROPIOS

El $n \times 1$ vector p_i que satisface la ecuación matricial

$$(\lambda_i I - A)p_i = 0$$

En donde λ_i es el i-ésimo valor propio de A, se denomina vector propio de A asociado con el valor propio i-ésimo.

El programa nos presenta la ecuación característica, en forma desarrollada, y en factores. Para acceder a esta opción presionamos la tecla aceleradora CTRL+V. De los factores de la ecuación característica, obtenemos los valores propios; los vectores propios no son obtenidos directamente del programa, pero teniendo los valores propios, existen procedmientos sencillos para obtener los vectores propios.

3.3 MATRIZ EXPONENCIAL

Se puede probar que la matriz exponencial de una matriz A de $n \times n$

$$e^{At} \quad \sum_{k=0}^{\infty} \frac{A^k t^k}{k!}$$

converge absolutamente para todo valor finito de t. (Por

tanto, se pueden realizar fácilmente en computadoras los cálculos para la evaluación de e^{At} utilizando el desarrollo de la serie).

Otro método para calcular e^{At}, usa la transformada de Laplace; e^{At} se pude escribir como

$$e^{At} = L^{-1}[(sI - A)^{-1}]$$

Por tanto, para obtener e^{At}, se debe invertir $(sI - A)$. Esto da como resultado una matriz cuyos elementos son funciones racionales de s. Luego se toma la transformada inversa de Laplace de cada elemento de la matriz.

El programa presenta la matriz e^{At} en forma literal (para calcular la se usa el segundo procedimiento). Para obtener esta matriz se presiona la tecla aceleradora CTRL+E.

El programa también nos presenta la matriz $(sI - A)^{-1}$ en forma literal. Para obtener esta matriz presionamos la tecla aceleradora CTRL+S.

3.4 SOLUCIÓN DE ECUACIÓN DE ESTADO INVARIANTE EN EL TIEMPO

Se considera la ecuación de estado no homogénea descrita por

$$\dot{x} = Ax + Bu \qquad (3-1)$$

donde

x : vector n-dimensional
u : vector r-dimensional
A : matriz constante n x n
B : matriz constante n x r

escribiendo la ecuación (3-1) como

$\dot{x}(t) - Ax(t) = Bu(t)$

y premultiplicando ambos lados de esta ecuación por e^{-At}, se obtiene

$$e^{-At}[\dot{x}(t) - Ax(t)] = d/dt[e^{-At}x(t)] = e^{-At}Bu(t)$$

Integrando la ecuación precedente entre 0 y t, se tiene

$$e^{-At}x(t) = x(0) + \int_0 e^{-At}Bu(\tau)d\tau$$

$$x(t) = e^{At}x(0) + \int_0 e^{A(t-\tau)}Bu(\tau)d\tau \qquad (3-2)$$

La ecuación (3-2) es la solución de (3-1)

El programa presenta $e^{At}x(0)$ (solución natural), y la $\int_0 e^{A(t-\tau)}Bu(\tau)d\tau$ (solución forzada) en forma literal; para ver estos vectores se presiona las teclas aceleradoras CTRL+N y CTRL+R respectivamente. Las entradas permitidas son la rampa, impulso, y paso. La solución total es la suma de los dos vectores anteriores. Para obtener la solución total se presiona la tecla aceleradora CTRL+X.

Para hallar (sI - A)^-1, utilzamos el algoritmo de Leverrier's que esta dado como sigue:

$$F_1 = I, \quad T_1 = -trAF_1/1$$
$$F_2 = AF_1 + T_1 I, \quad T_2 = -trAF_2/2$$
$$F_n = AF_{n-1} + T_{n-1}I, \quad T_n = -trAF_n/n$$
$$(sI - A)^{-1} = \frac{s^{n-1}F_1 + s^{n-2}F_2 + \ldots + sF_{n-1} + F_n}{s^n + T_1 s^{n-1} + \ldots + T_{n-1}s + T_n}$$

Para hallar e^{At} se calcula la transformada inversa de Laplace de cada uno de los términos de la matriz $(sI - A)^{-1}$.

La integral de 0 a t de $e^{(A(t-tao))}Bud(tao)$ la calculamos multiplicando $(sI - A)^{-1}$ por B y por u (en funcion de s), y calculando la transformada inversa de Laplace de cada uno de los elementos de la matriz.

3.5 CONTROLABILIDAD

Los conceptos de controlabilidad y observabilidad, fueron introducidos por Kalman. Estos juegan un papel importante en el diseño de sistemas de control en el espacio de estado. De hecho las condiciones de controlabilidad y observabilidad, pueden gobernar la existencia de una solución completa, en el problema de diseño de sistemas de control. La solución a este problema no puede existir si el sistema considerado no es controlable. Aunque la mayoría de los sistemas físicos son controlables y observables los modelos matemáticos correspondientes pueden no tener la propiedad de controlabilidad y observabilidad. Entonces se requiere conocer las condiciones bajo las cuales un sistema es controlable y observable.

Considere el sistema en tiempo continuo

$$\dot{x} = Ax + Bu \qquad (3\text{-}3)$$

donde x = vector de estado (vector n-dimensional)
u = señal de control (escalar)
A = matriz de n x n
B = matriz de n x 1

El sistema descrito por (3-3) se dice que es de estado completamente controlable en $t = t_0$ si es posible construir una señal de control no restringida, que puede transferir un estado inicial en cualquier estado final en un intervalo de tiempo finito $t_0 \leq t \leq t_1$. Si todo estado es controlable, entonces se dice que el sistema es completamente controlable.

Para la demostración ver referencia [1].

La condición para la controlabilidad de estado completo es que la siguiente matriz de n x n

$$[B \mid AB \mid \ldots \mid A^{n-1}B]$$

tenga determinante $\neq 0$.

El resultado anterior, se puede extender al caso en que el vector de control u, es de dimensión r y la matriz B es de dimensión n x r. Se puede probar que la condición para la controlabilidad del estado completo, es que la siguiente matriz de n x nr

$$[B \mid AB \mid \ldots \mid A^{n-1}B]$$

tenga n vectores columna, linealmente independientes. Esta matriz se denomina comúnmente matriz de controlabilidad.

El programa muestra la matriz de controlabilidad, y determina si el sistema es o no controlable. Para observar esta matriz presione la tecla aceleradora CTRL+C.

Para determinar si el sistema es controlable o no, se halla el determinante de la matriz de controlabilidad, si el la matriz de controlabilidad es cuadrada, utilizando la eliminacion gausiana; si la matriz de controlabilidad no es cuadrada, multiplicamos dicha matriz por la transpuesta obteniendose una matriz cuadrada y procediendose como en el caso anterior.

3.6 OBSERVABILIDAD

Se dice que un sistema es observable en el tiempo t_0 si, con el sistema en el estado $x(t_0)$, es posible determinar este estado a través de la observación de la salida durante un intervalo finito de tiempo.

Sea el sistema descrito por las ecuaciones siguientes:

$$\dot{x} = Ax \qquad (3-4)$$

$$y = Cx \qquad (3-5)$$

donde x = vector de estado (vector n-dimensional)

y = vector de salida (vector m-dimensional)

A = matriz n x n

_____ ____ C = matriz m x n

Se dice que el sistema es completamente observable si se puede determinar todo estado inicial **x**(0) a partir de la observación de **y**(t) en un intervalo de tiempo finito. El sistema es, por tanto, completamente observable si toda transición del estado puede afectar a todo elemento del vector de salida. El concepto de observabilidad es útil para resolver el problema de reconstruir variables de estado no medibles a partir de las medibles en el espacio mínimo de tiempo posible.

Se puede establecer la condición de observabilidad completa como sigue: el sistema descrito por las ecuaciones (3-4) y (3-5) es completamente observable si, y solamente si, la matriz de n x nm

_____ ____ _____ $[C^* \mid A^*C^* \mid \ldots \mid (A^*)^{n-1}C^*]$

tiene n vectores, columna linealmente independientes. Esta matriz se denomina matriz de observabilidad.

El programa nos puede presentar la matriz de observabilidad y determinar si el sistema es observable o no. Para obtener estos resultados presionamos la tecla aceleradora CTRL+B.

El programa determina si el sistema es obsevable con un procedimiento similar al detallado en controlabilidad.

3.7 Grammiano

El criterio de controlabilidad y observabilidad es bastante directo, exepto en el caso de entradas múltiples. Incluso con r = 2, se forma una matriz de controlabilidad u observabilidad con 2n columnas. Sin embargo, el hecho de no poder determinar el determinante de la matriz formada no significa que el sistema no sea controlable u observable. Una manera fácil de determinar si el sistema es controlable u observable, es hallar el deteminante de la matriz multiplicada

por su transpuesta (matriz cuadrada), si el determinante es cero, la matriz original es singular.

Este criterio es usado en el programa, en el caso de múltiples entradas, para determinar la controlabilidad y observabilidad de sistemas.

CAPITULO IV
EJEMPLOS DESARROLLADOS

EJEMPLOS DESARROLLADOS

En este capítulo se presentan diferentes ejemplos que buscan:

- Probar la validez del programa implementado.
- Explorar las diferentes opciones del programa.
- Indicar las limitaciones del programa.

Para cada opción del mismo se ha considerado necesario presentar desde los ejemplos más representativos y simples, hasta los más complejos.

Los ejemplos tratados a continuación son desarrollados en el libro INGENIERIA DE CONTROL MODERNA cuyo autor es Katsuhiko Ogata.

a) **Transformada inversa de Laplace y expansión en fracciones parciales:**

EJEMPLO 1.

Raíces reales y distintas.

G(s)=(s + 3)/(s + 1)(s + 2)

Para el ingreso de la función de transferencia, abrimos el menú archivo y usamos el comando nuevo; el ingreso de la función lo hacemos de manera literal escribiendo

(s+3)/(s+1)(s+2) o también (s+3)/(s+1)/(s+2)

El programa muestra la función desarrollada, es decir la multiplicación de todos sus factores el resultado es:

s + 3
s^2 + 3 s + 2

La expansión en fracciones parciales la obtenemos

presionando CTRL+F, y el resultado es:

2/(s + 1) - 1/(s + 2)

La transformada inversa de Laplace la obtenemos presionando CTRL+L, y el resultado es:

2e^-t - e^-2t

EJEMPLO 2

Raíces imaginarias distintas.

G(s) = (2s + 12)/(s^2 + 2s + 5)

Expansión a fracciones parciales.

(2s + 12)/((s + 1)^2 + 2^2)

Transformada inversa de Laplace.

5.38516 e^-t cos(2t - 68.1986)

EJEMPLO 3

Raíces reales repetidas.

G(s) = (s^2 + 2s + 3)/(s + 1)^3

Expansión en fracciones parciales.

2/(s + 0.99988t)^3 + 1/(s+1.00005)

Transformada inversa de Laplace obtenida:

t^2 e^-.99988t + 1e^-1.00005t
(extactamente el resultado es t^2 e^-t + e^-t)

Cuando las raíces son repetidas, el programa no las halla

exactamente, pero la presición obtenida es aceptable.

EJEMPLO 4

Grado del numerador mayor que el grado del denominador.

G(s) = (s^4+2s^3+3s^2+4s+5)/s(s+1)

Expansión a fracciones parciales.

s^2 + s + 2 + 5/s - 3/(s+1)

Transformada inversa de Laplace.

d^2/dt[δ(t)] + d/dt[δ(t)] + 2 δ(t) + 5 - 3 e^-t

El programa no halla la transformada de Laplace cuando las raíces son imaginarias repetidas, pero esta limitación no es muy importante ya que plantas con esta característica no son frecuentes.

b) Respuesta en el tiempo

EJEMPLO 5

Respuesta de sistemas de primer orden a entradas impulso, paso, o rampa.

G(s) = 1/s

En lazo cerrado la relación entrada salida esta dada por

1
s+1

La transformada inversa de Laplace es

1 e ^-t

Si la entrada es un escalón unitario entonces R(s) es 1/s por lo que C(s) es

1/s(s+1)

La transformada inversa de Laplace es

1 - e^-t

Para obtener la respuesta a la rampa tenemos que multiplicar 2 veces la función por una entrada escalón. En este caso G(s) en forma de pölos y ceros es

1
s^2(s+1)

La transformada inversa de Laplace es

t - 1 + e^-t

La repuesta en el tiempo la obtenemos presionando CTRL+T, y escogiendo de las opciones adecuadas de la caja de dialogo. (Ver figura 4.1)

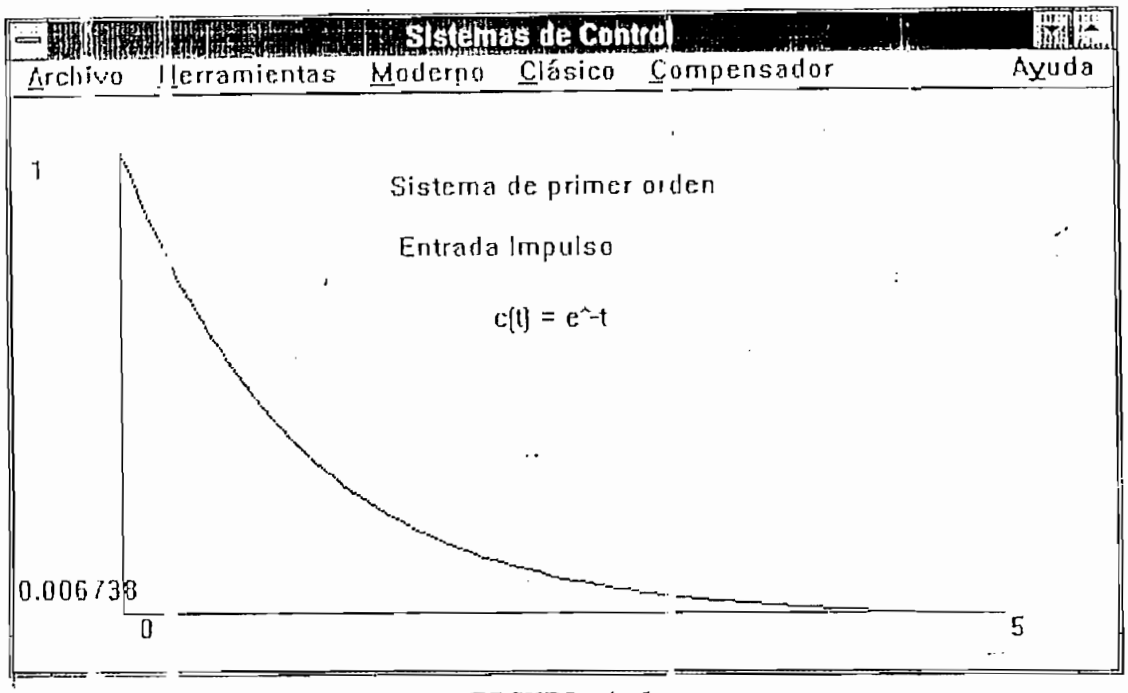


FIGURA 4.1

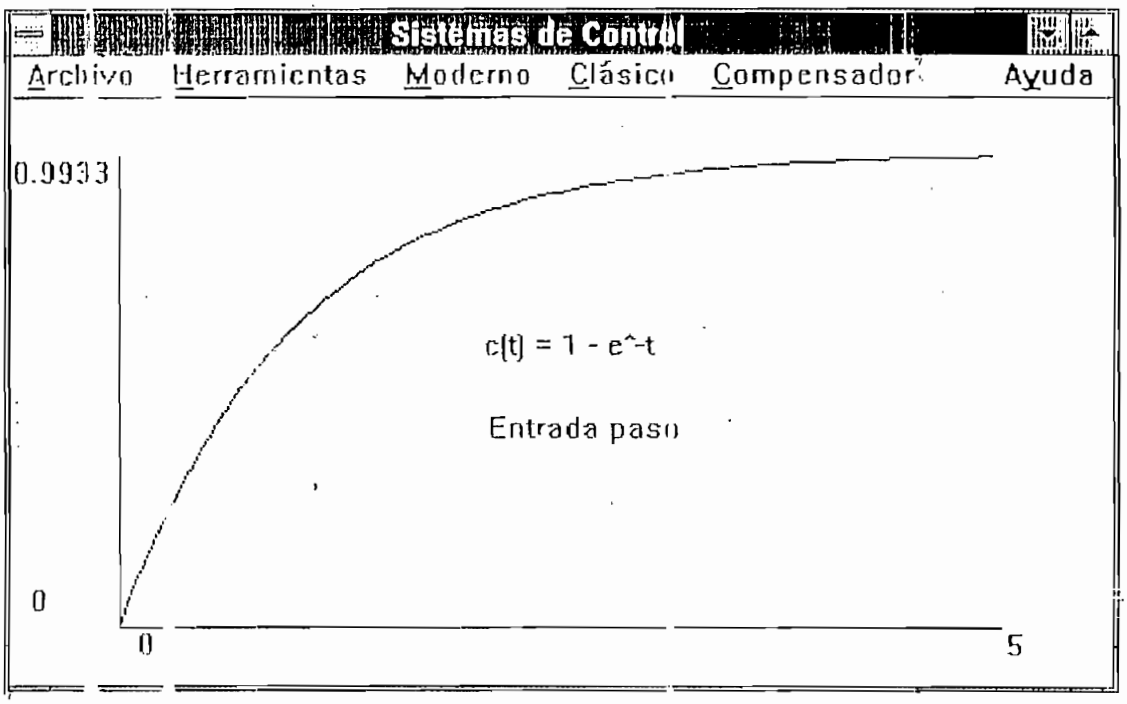


FIGURA 4.2

Al ser un sistema inestable se toma el tiempo máximo es 1.

La Figura 4.3 muestra la respuesta del sistema a una entrada rampa.

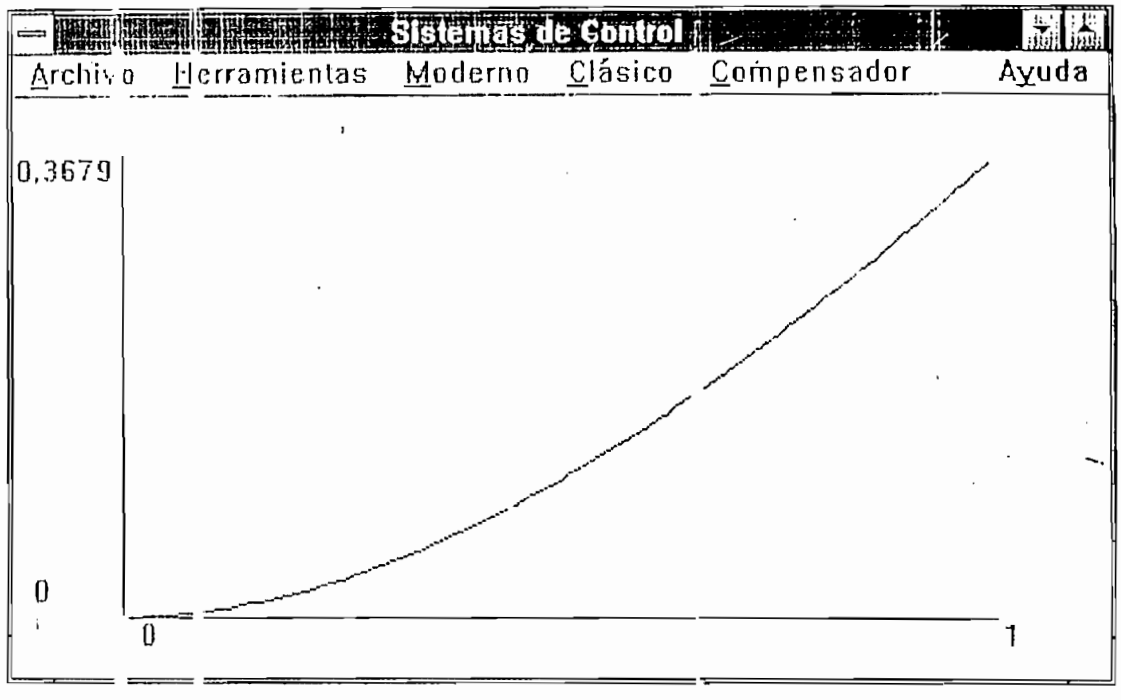


FIGURA 4.3

EJEMPLO 6

Obtenga la respuesta al escalón unitario, en un sistema de realimentación unitaria cuya función de transferencia de lazo abierto es

$$G(s) = \frac{25.04(s+0.2)}{s(s+5.02)(s+.01247)}$$

La curva proporcionada por el programa es

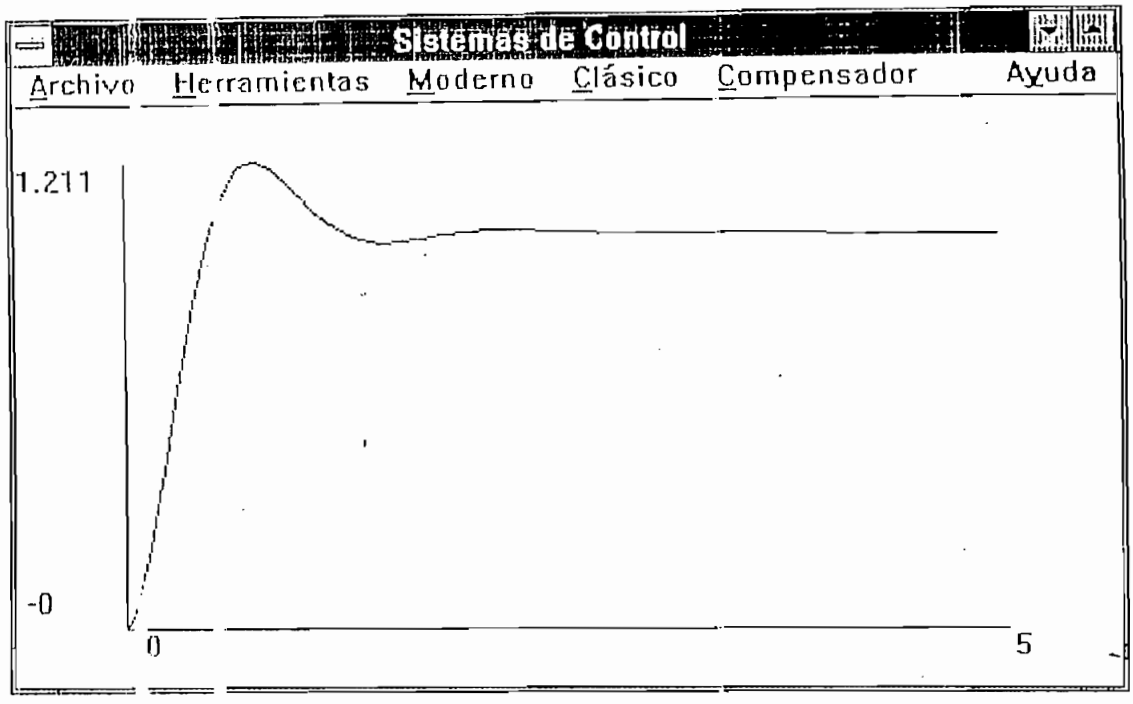


FIGURA 4.4

EJEMPLO 7

Obtenga la respuesta al escalón unitario, en un sistema de realimentación unitaria cuya función de transferencia de lazo abierto es

$$G(s) = \frac{5(s+20)}{s(s+4.59)(s^2+3.41s+16.35)}$$

La curva es

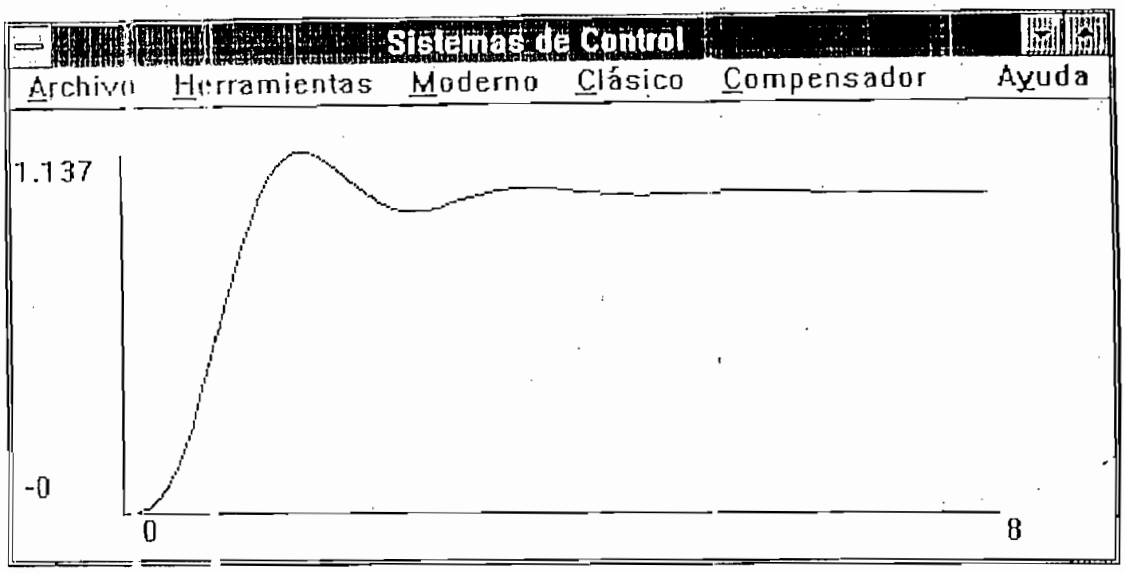


FIGURA 4.5

La función en lazo cerrado en forma de polos y ceros es

$$\frac{5(s+20)}{(s^2+2s+10)(s^2+6s+10)}$$

La transformada de Laplace es

$$1 + 2.13886\, e^{-2.99976t} \cos(0.99989t - 130.013) + 0.799788\, e^{-1.000241} \cos(3.0018t - 62.0217)$$

El programa sólo considera realimentación unitaria, ya que un sistema con realimentación diferente de uno se puede reducir a uno con realimentación unitaria de acuerdo al siguiente diagrama de bloques.

c) Lugar geométrico de las raíces

Trazar el diagrama del lugar geométrico de las raíces para los sistemas descritos por las siguientes funciones de trasferencia de lazo abierto:

8. $G(s) = \dfrac{1}{s(s+1)}$

9. $G(s) = \dfrac{1}{s(s+1)(s+2)}$

10. $G(s) = \dfrac{(s+1)}{s(s-1)(s^2+4s+16)}$

EJEMPLO 8

El programa nos da la información necesaria para trazar el lugar geométrico de manera manual (presionando CTRL+G) con la siguiente caja de dialogo

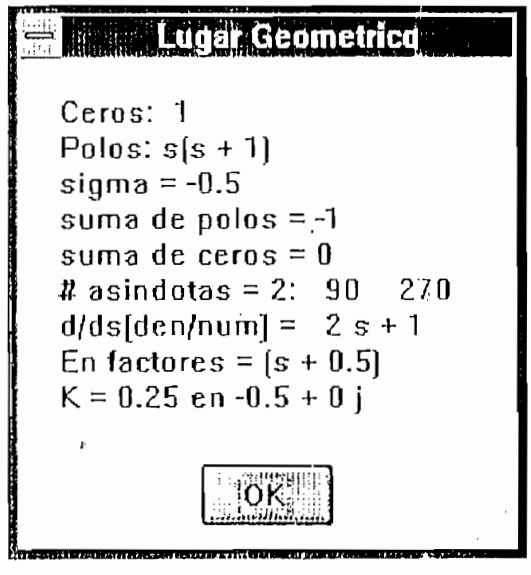


FIGURA.4.6

El diagrama que obtenemos es

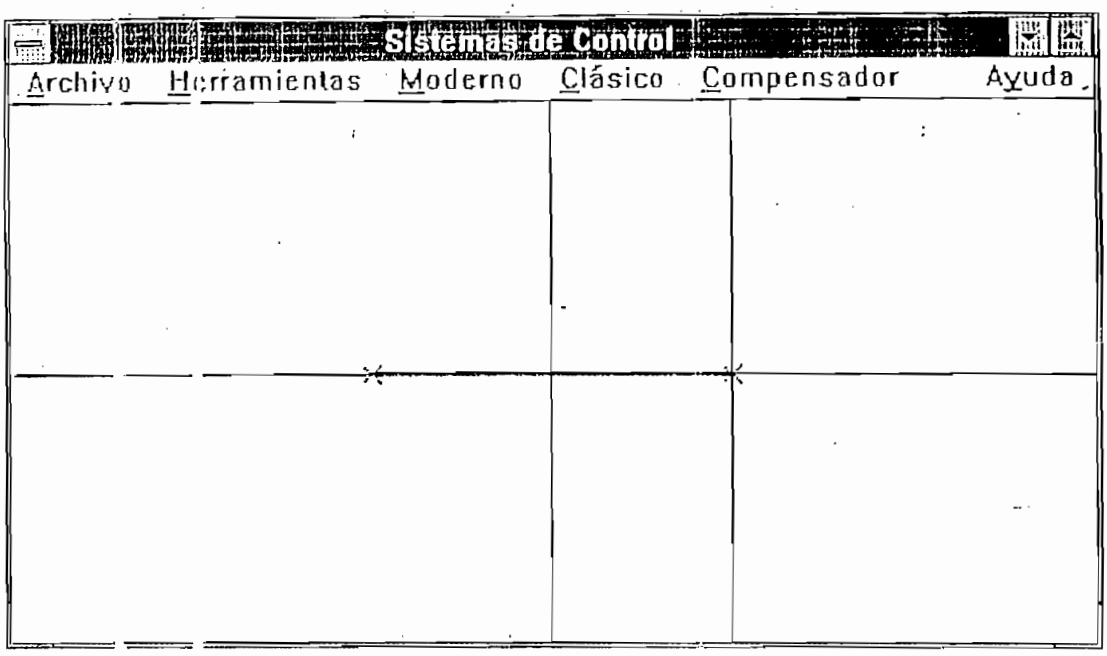


FIGURA 4.7

EJEMPLO 9

Los resultados que nos da el programa son

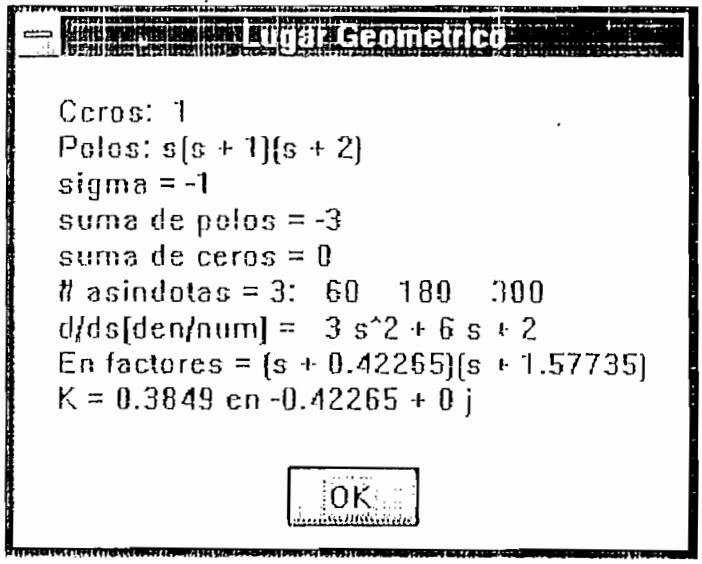


FIGURA 4.8

El diagrama es

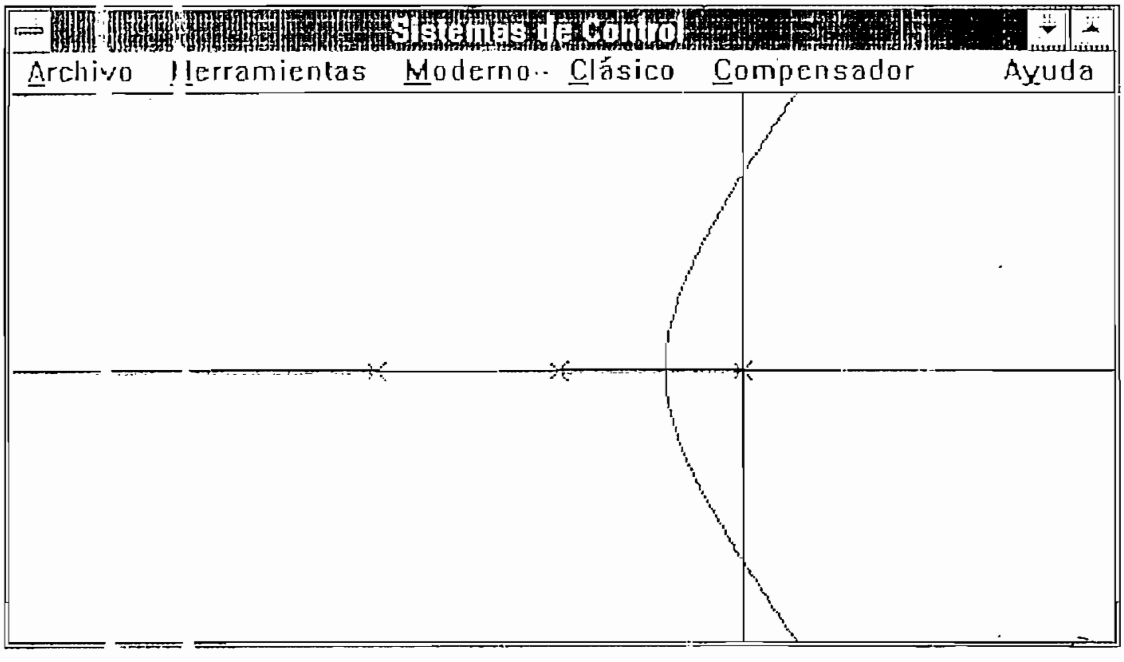


FIGURA 4.9

EJEMPLO 10

El programa nos da la siguiente información

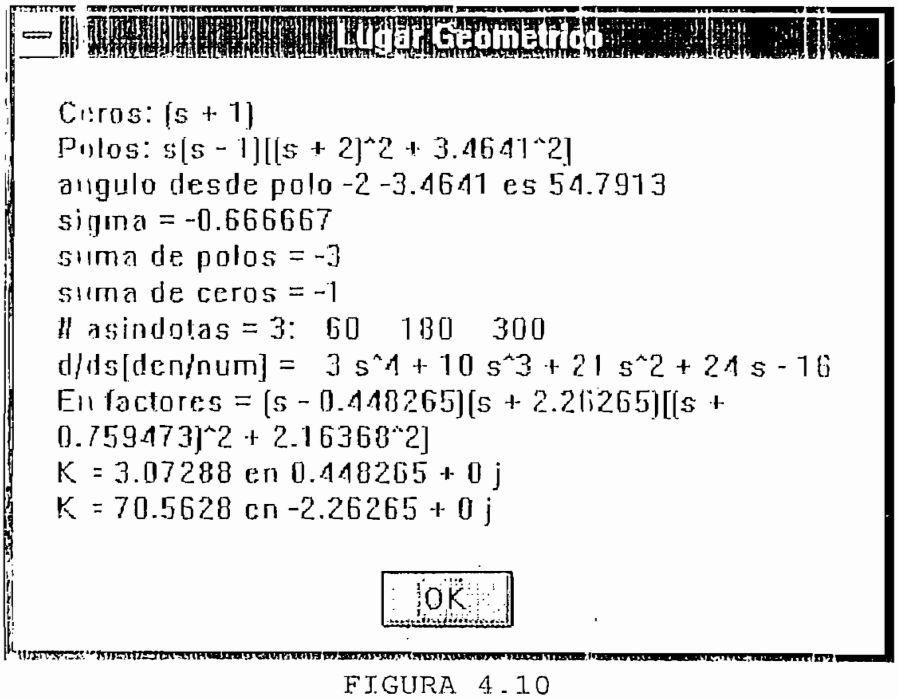


FIGURA 4.10

El diagrama que nos proporciona el programa es

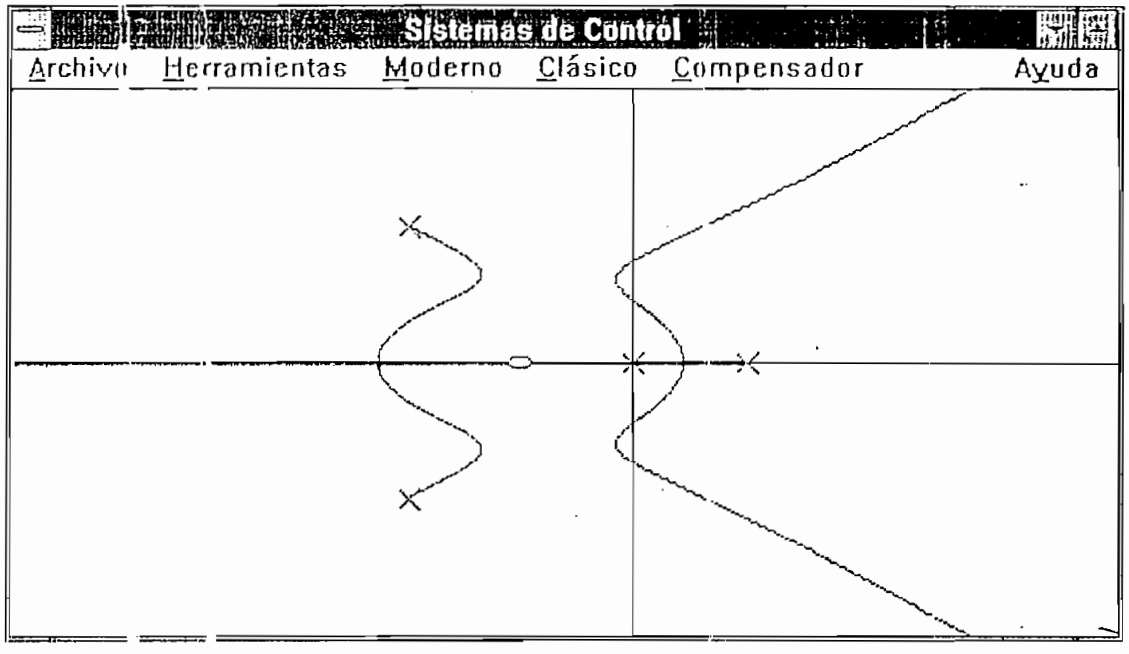


FIGURA 4.11

d) Diagrama de Bode

Trazar el diagrama de Bode de

$$G(jw) = \frac{1}{1 + jwT}$$

El programa solo permite el ingreso de las funciones de transferencia en terminos de s, por lo que substituyendo s por jw para ingresar la función de transferencia al programa tenemos el siguiente resultado:

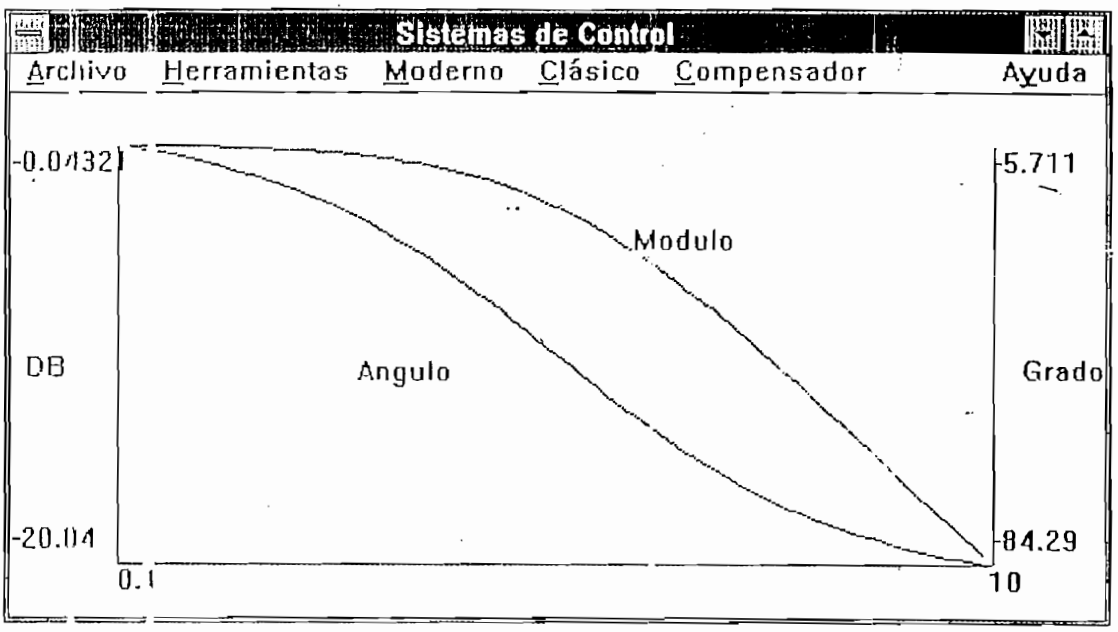


FIGURA 4.12

EJEMPLO 12

Trazar el diagrama de bode para una función con factores cuadráticos

$$G(s) = \frac{1}{1 + .4\,s + s^2}$$

El programa nos da la siguiente curva

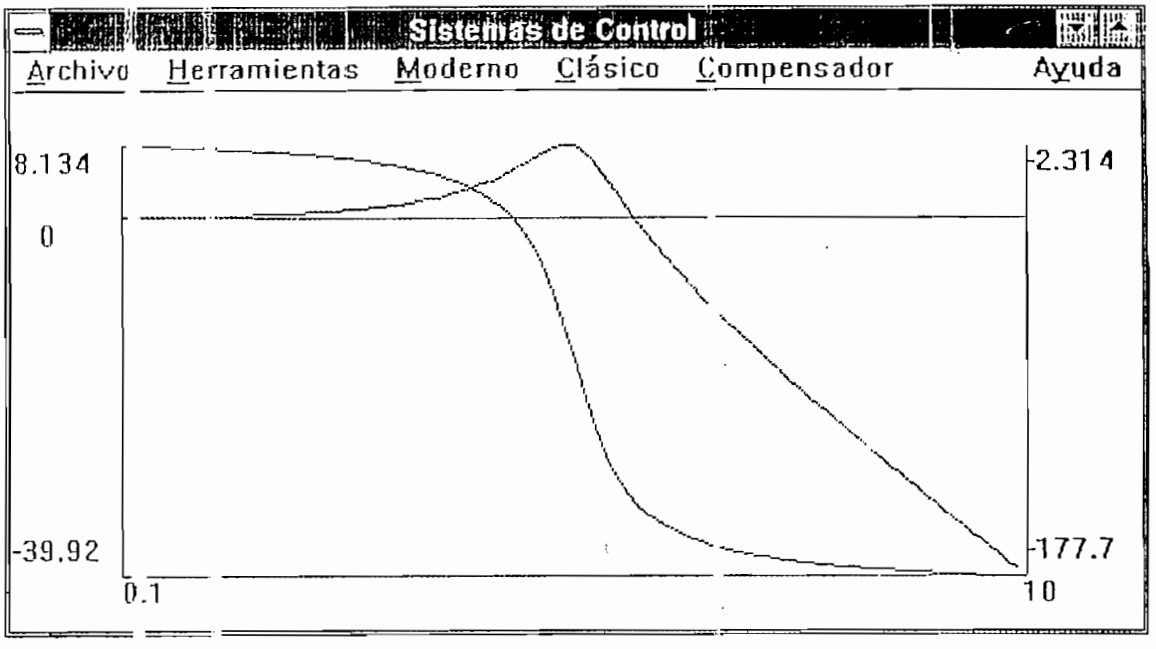


FIGURA 4.13

EJEMPLO 13

Trace el diagrama de Bode de

$$G(s) = \frac{20(s + 1)}{s(s + 5)(s^2 + 2s + 10)}$$

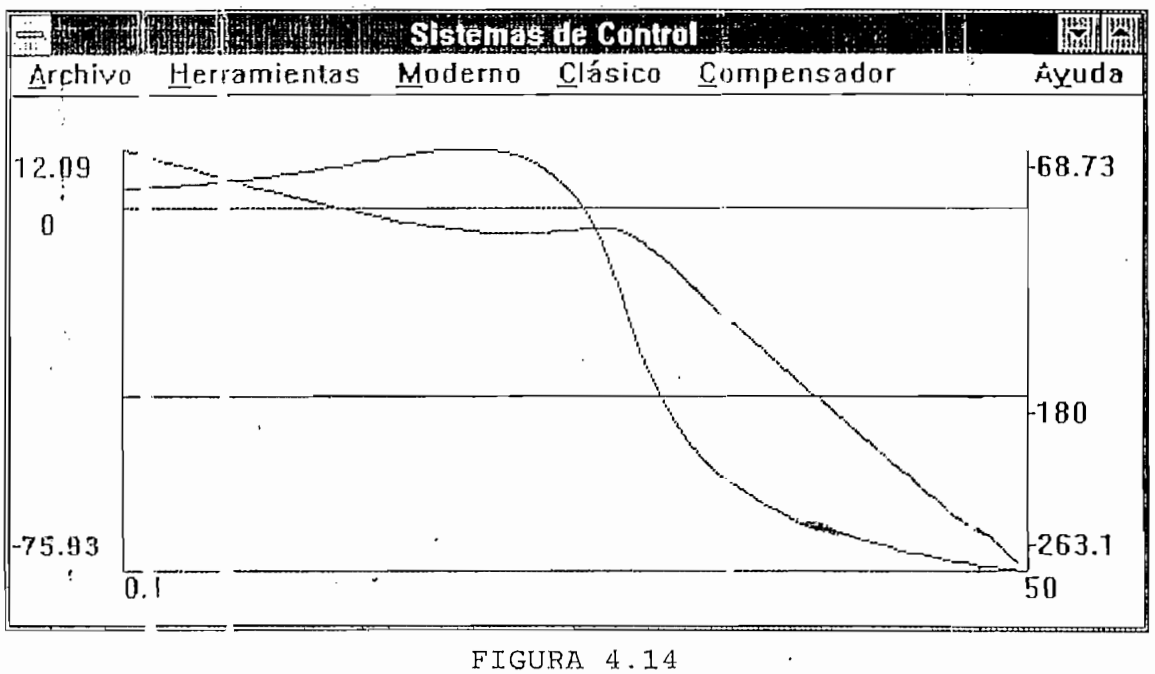


FIGURA 4.14

e) Diagrama de Nyquist

EJEMPLO 14

Trazar el diagrama de Nyquist de

$$G(s) = \frac{1}{1 + s}$$

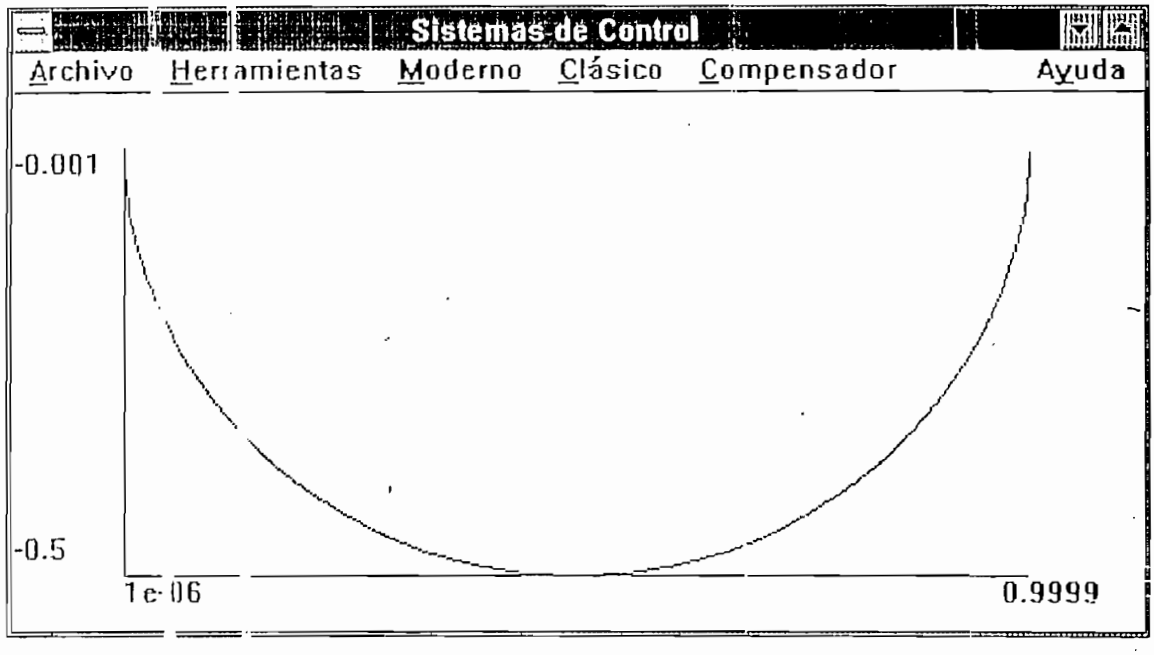


FIGURA 4.15

El programa marca el punto -1+0j, esto se muestra en el siguiente ejemplo.

EJEMPLO 15

Trace el diagrama de Nyquist para

$$G(s) = \frac{40}{(1+s)(1+2s)(1+3s)}$$

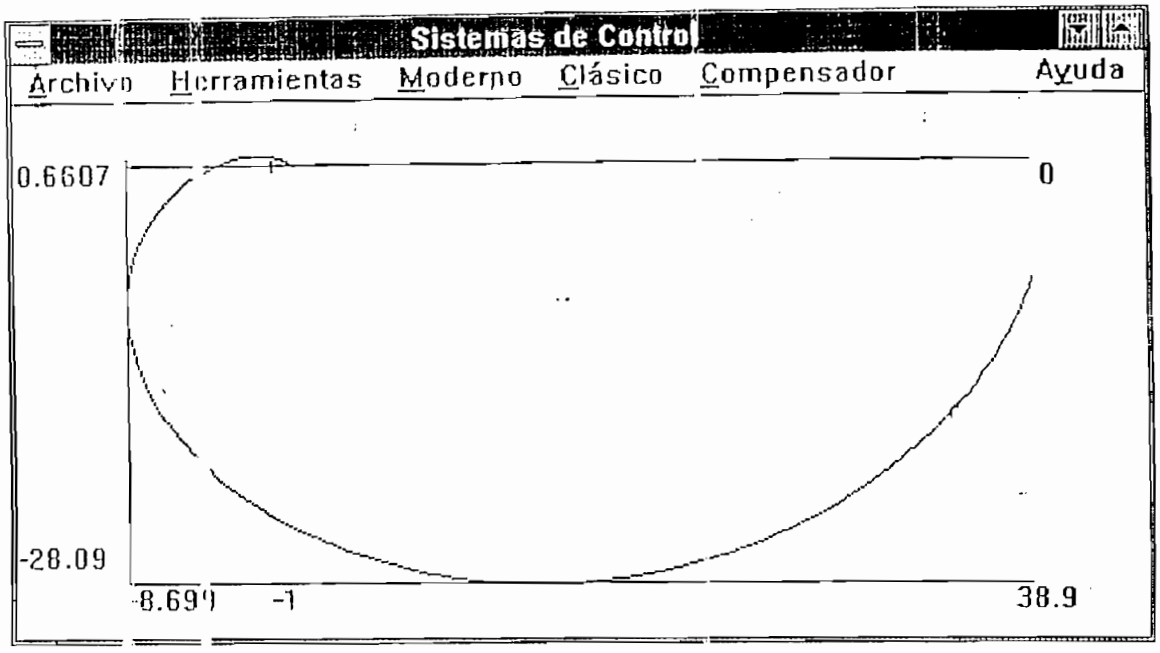


FIGURA 4.16

e) Diagrama de Nichols

EJEMPLO 16

Trace el diagrama de Nichols para el sistema dada en el ejemplo 14

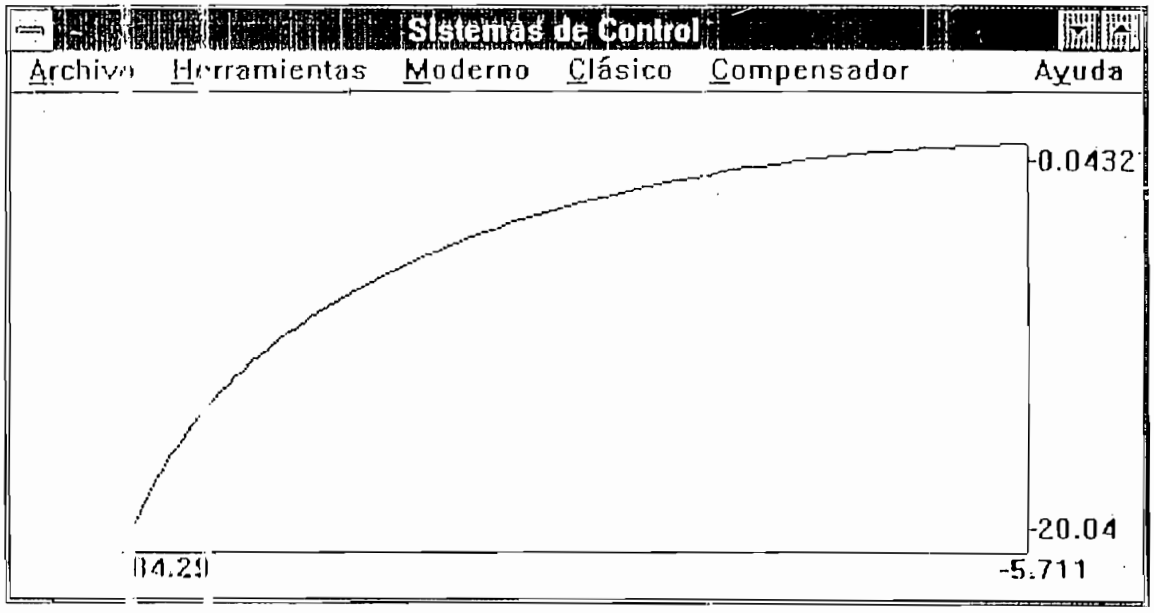


FIGURA 4.17

EJEMPLO 17

El diagrama de Nichols para el sistema descrito en el ejemplo 15 es

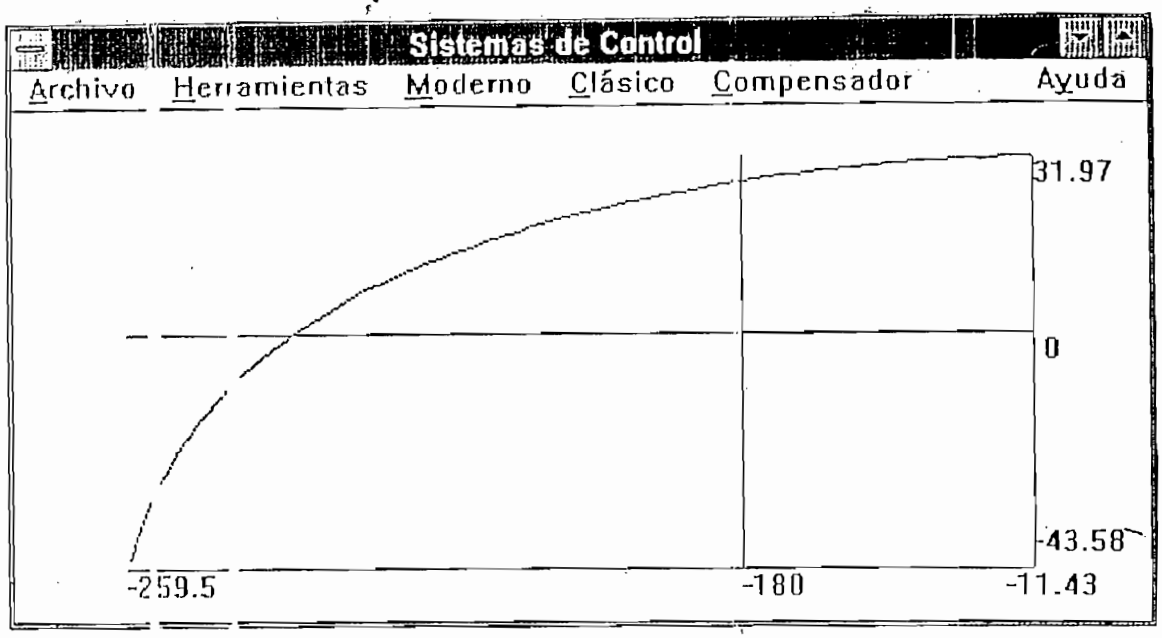


FIGURA 4.18

f) Compensación

EJEMPLO 18

La función de transferencia directa del sistema es

$$G(s) = \frac{4}{s(s+2)}$$

La relación de amortiguamiento de los polos de lazo cerrado es 0.5. La frecuencia natural no amortiguada es 2 rad/s. La constante de error estático de velocidad es $2s^{-1}$.

Se desea modificar los polos de lazo cerrado de modo que se obtenga la frecuencia natural no amortiguada $w_n = 4$ rad/s, sin cambiar el valor de la relación de amortiguamiento.

El compensador necesario para cumplir

estasespecificaciones es

$$G_c(s) = 4.68 \frac{(s+2.9)}{(s+5.4)}$$

Los diagramas del lugar de las raíces son:

Sistema sin compensar

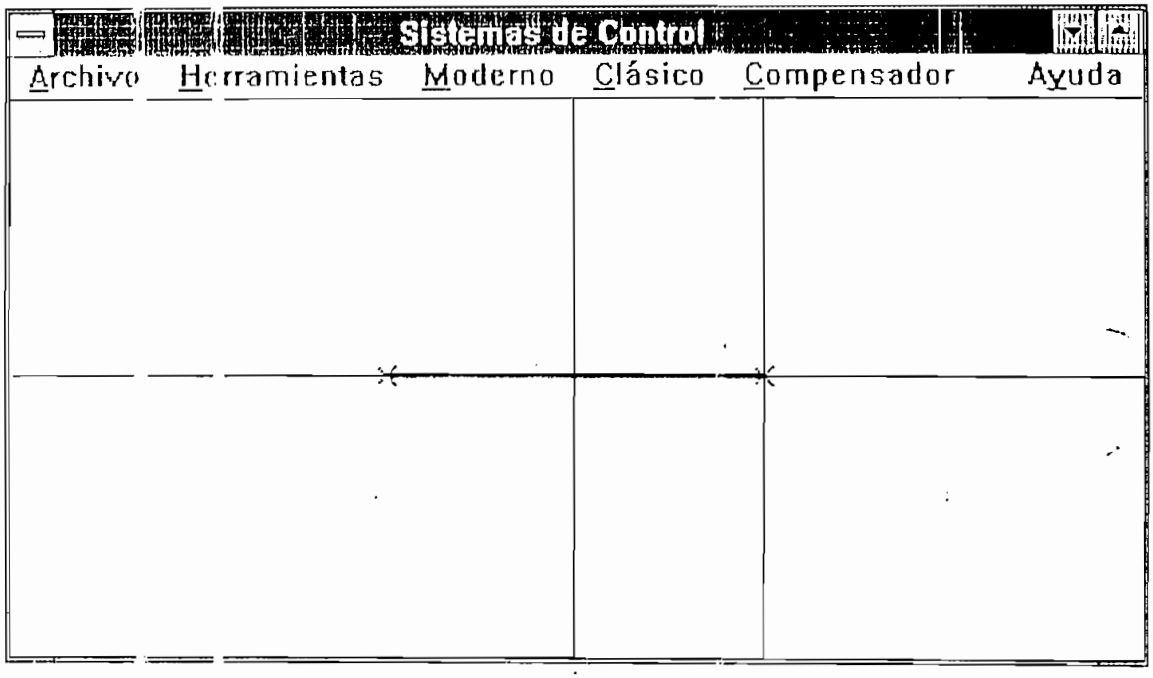
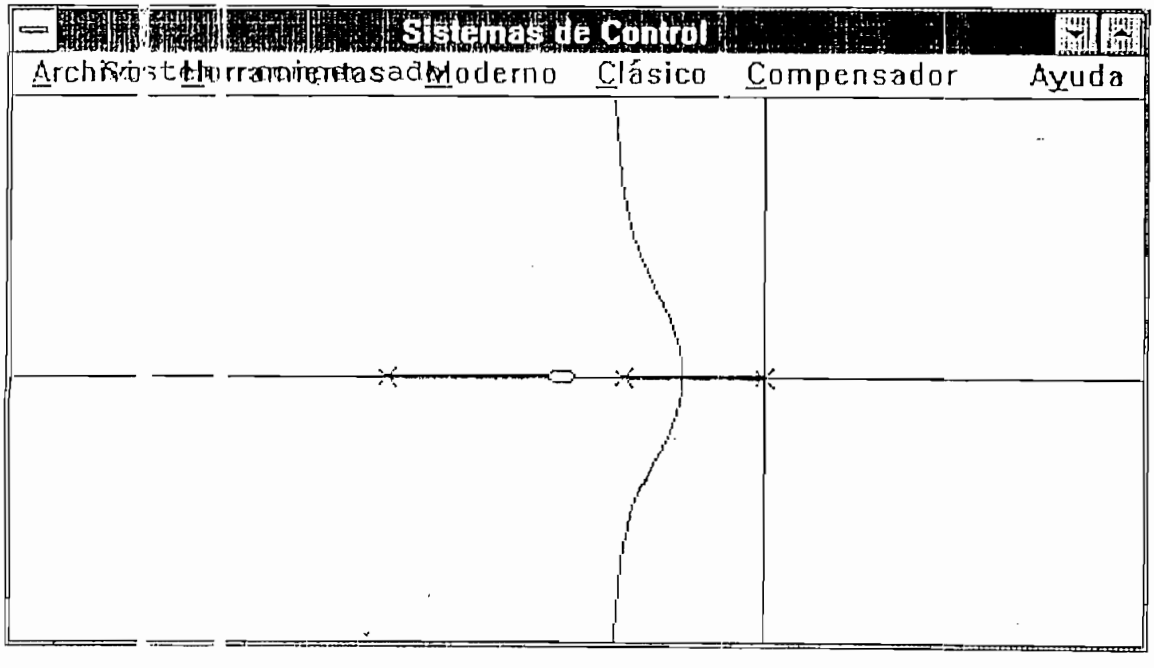


FIGURA 4.19



EJEMPLO 1)

La función de transferencia de lazo abierto es

$$G(s) = \frac{4}{s(s+2)}$$

Diseñar un compensador para el sistema tal que el coeficiente de error estático de velocidad K_v sea 20 s^{-1}, el margen de fase no sea menor a 50°, y el margen de ganancia sea por lo menos 10 db.

El compensador necesario es

$$G_c(s) = 41.7\ \frac{(s+4.41)}{(s+18.4)}$$

El diagrama de bode del sistema compensado es

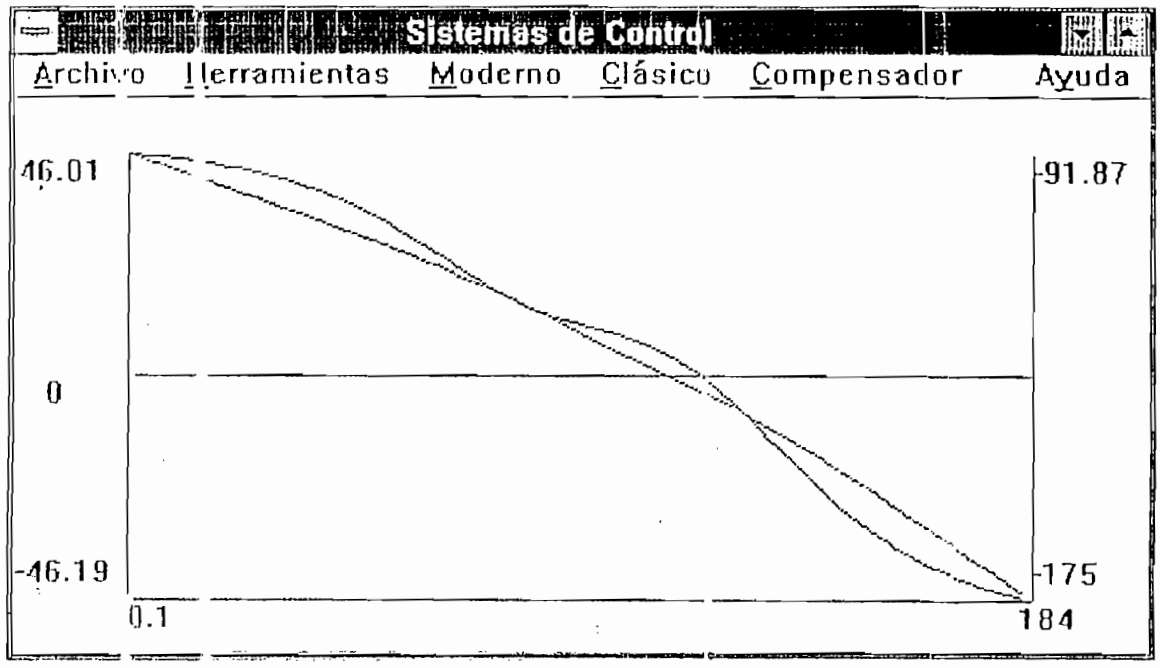


FIGURA 4.20

Que cumple las características especificadas

EJEMPLO 20

Compensar el siguiente sistema usando un controlador PID.

$$G(s) = \frac{1}{s(s+1)(s+5)}$$

La ganancia crítica es 30 y el período de oscilación es 2,7 (valores obtenidos del lugar de las raíces)

Esto se verifica con la respuesta en el tiempo del sistema compensado usando un compensador de ganancia 30.

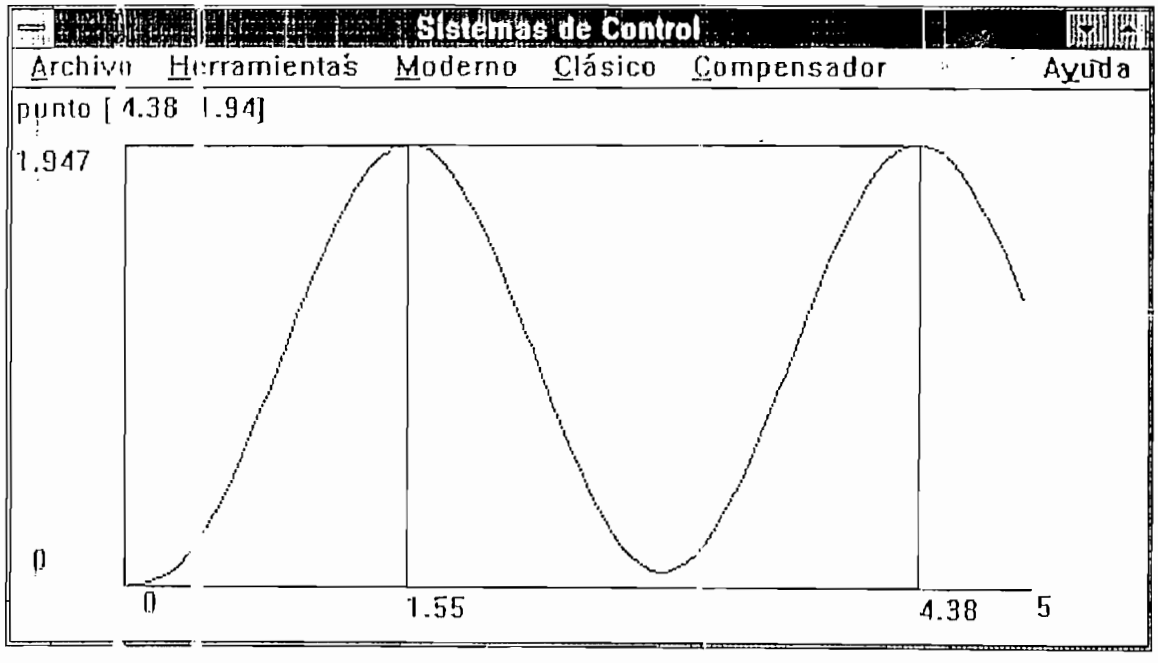


FIGURA 4.21

De el gráfico hallamos que el período critico es 2.8.

Usando el método de Zielger Nichols tenemos

Kp = 0.6 Kcr = 18
Ti = .5 Pcr = 1.4
Td = .125 Pcr = 0.351

Entonces el compensador es

$$Gc(s) = \frac{6.8223(s+1.4235)^2}{s}$$

La respuesta en el tiempo del sistema compensado es

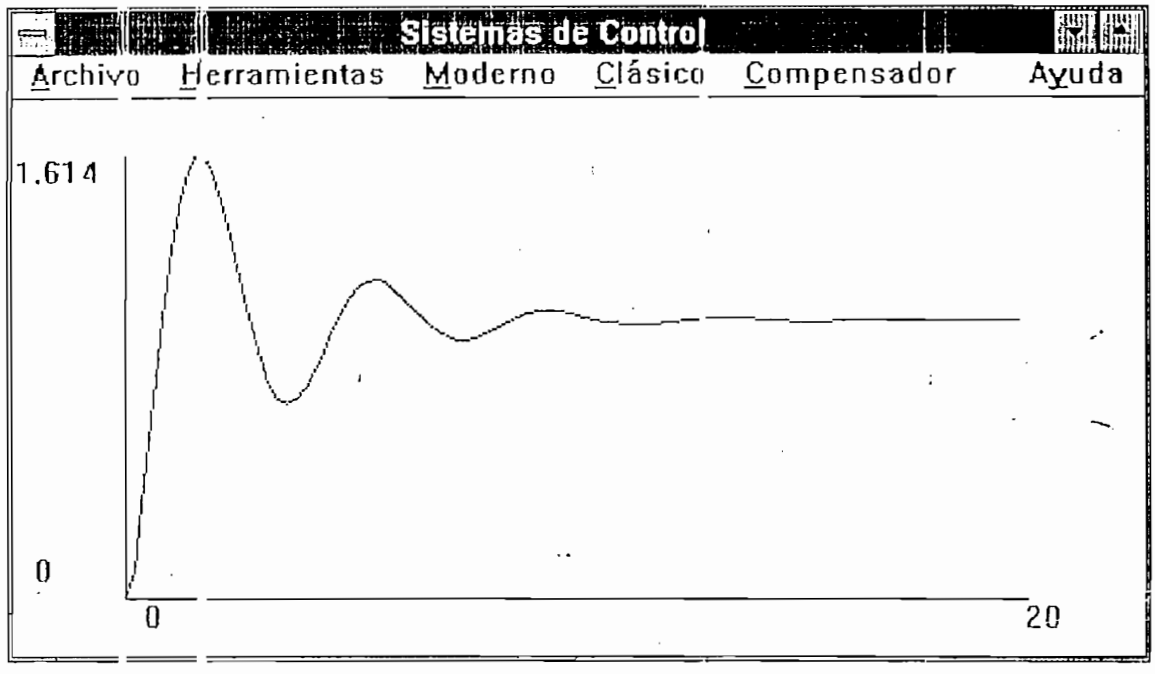


FIGURA 4.22

que tiene un sobreimpulso de 61.4 %

Para reducir el sobreimpulso a 18% se usa el compensador

Gc(s) = 13.846(s+0.65)^2/s

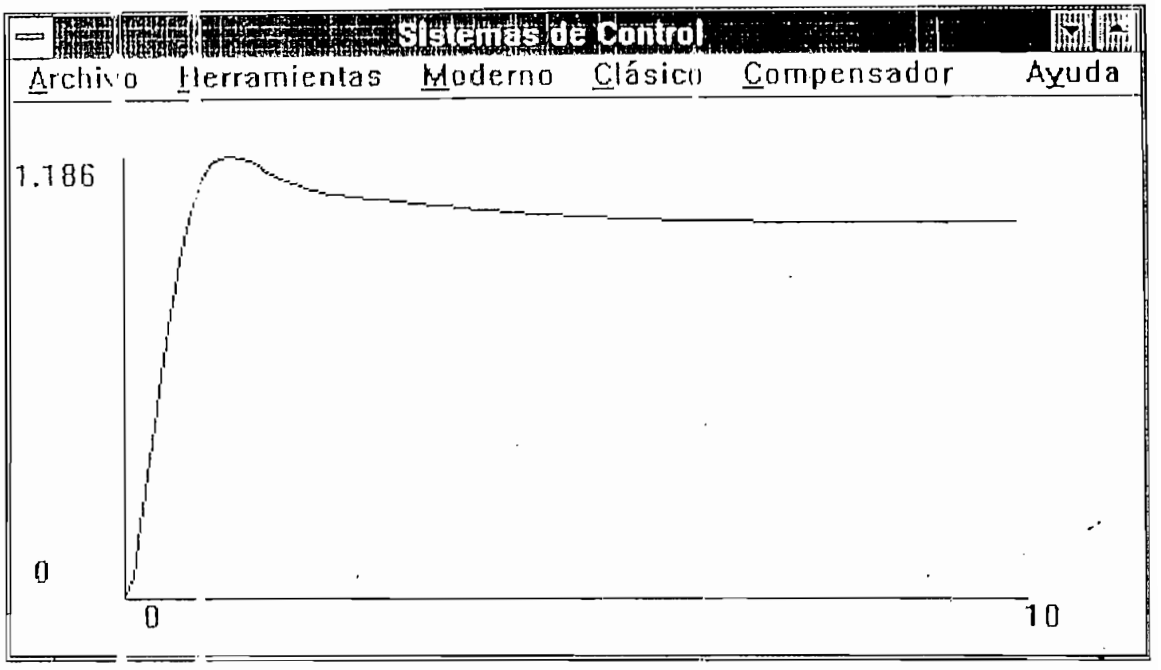


FIGURA 4.23

Si se incrementa la ganancia proporcional a 39.4 la velocidad de respuesta aumenta, pero se incrementa el sobreimpulso máximo hasta un 28 % como lo indica la figura

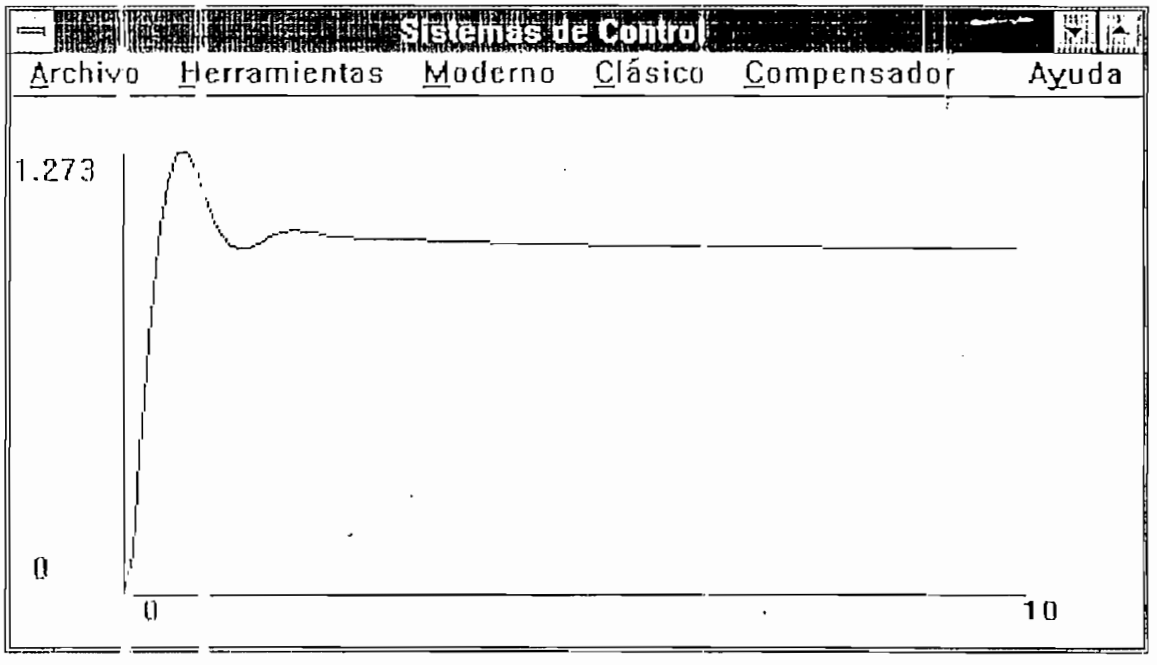


FIGURA 4.24

EJEMPLO 22

Considere el sistema con realimentación unitaria, cuya función de transferencia de lazo abierto es

$$G(s) = \frac{9}{s(s+0.5)(s^2+0.6s+10)}$$

Trace el diagrama de Bode para esta función de transferencia de lazo abierto, trace también el diagrama de Nichols. Finalmente, represente el diagrama de Bode para la función de transferencia de lazo cerrado.

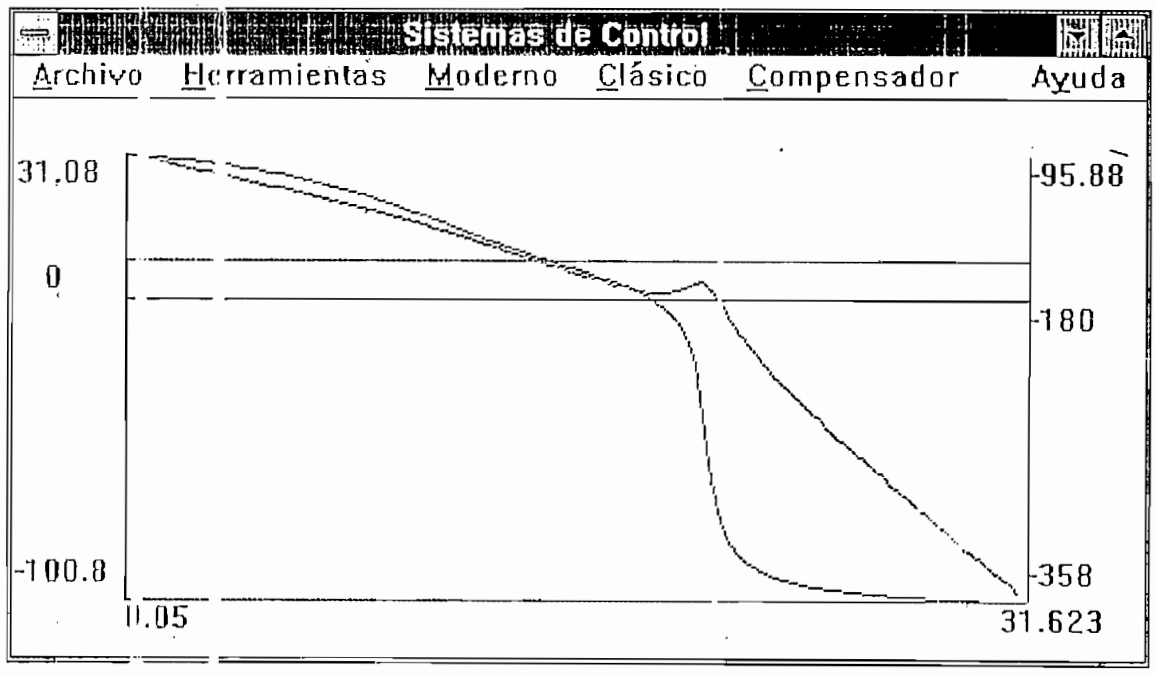


FIGURA 4.25

Diagrama de Nichols

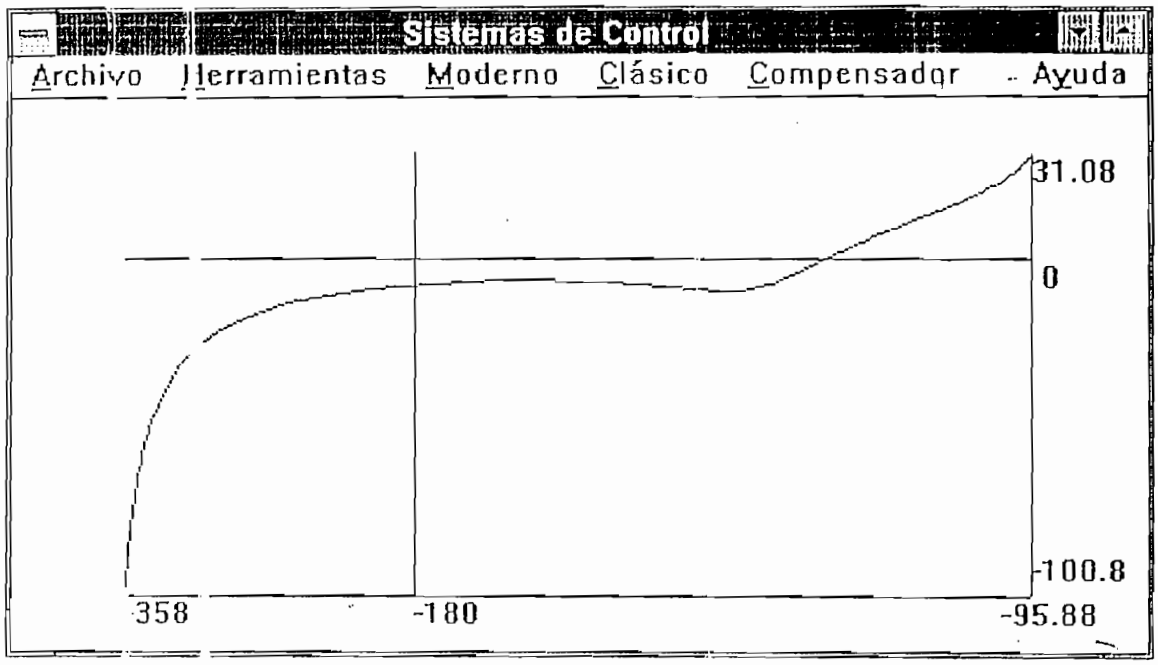


FIGURA 4.26

Diagrama de Bode de G(jw)/[1+G(jw)]

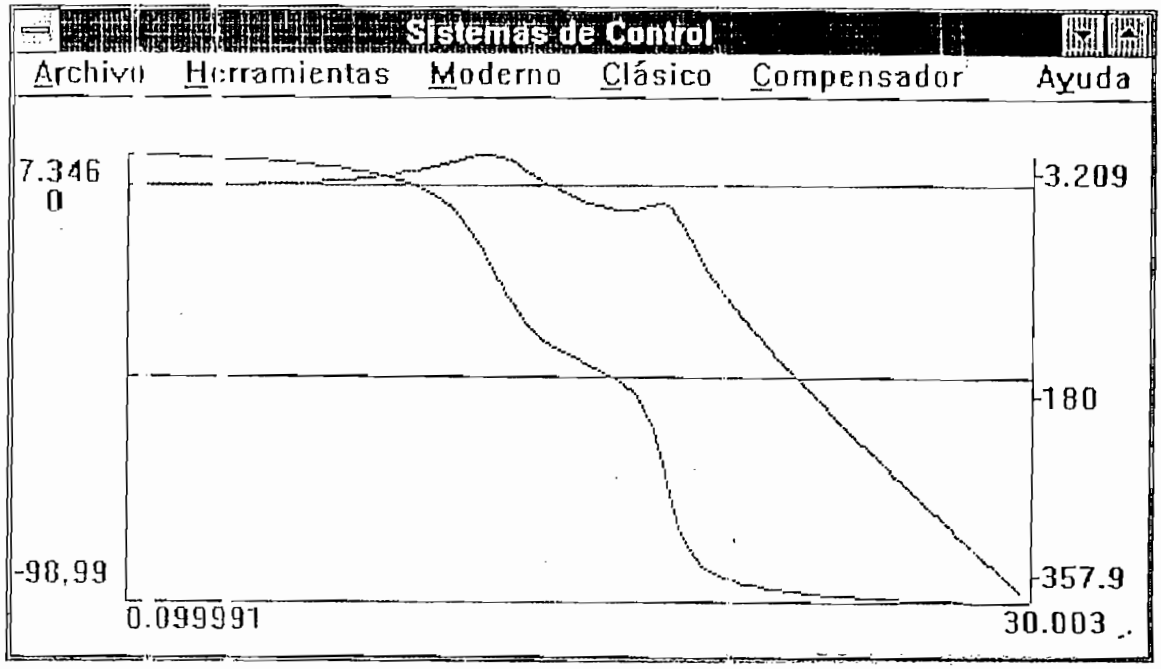


FIGURA 4.27

La respuesta en el tiempo en lazo abierto para una entrada impulso es:

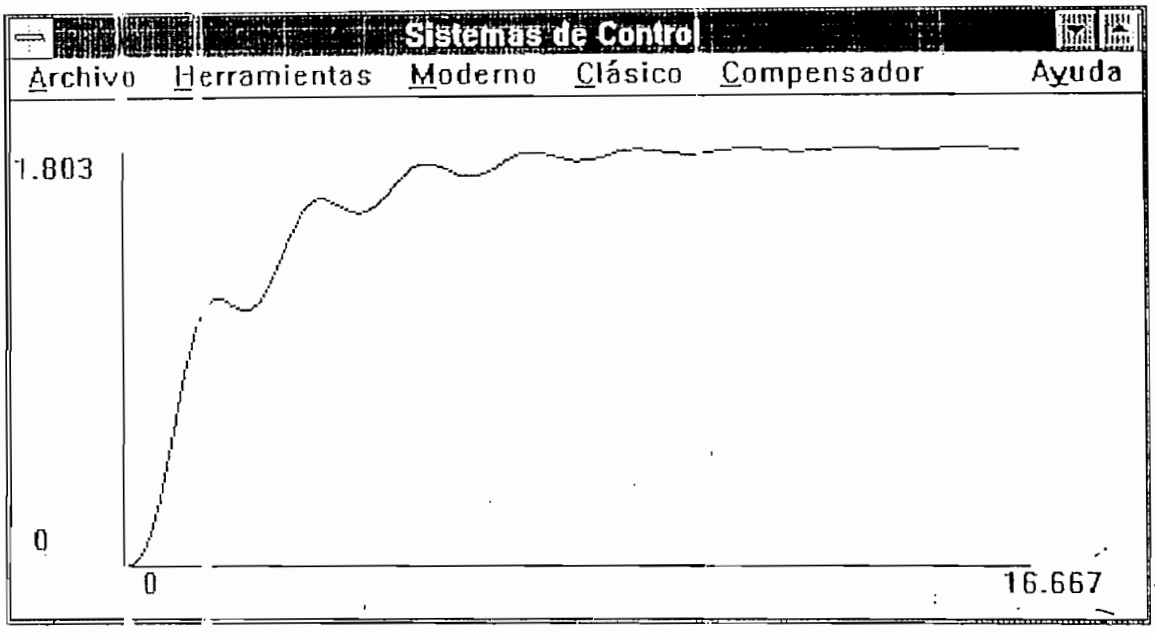


FIGURA 4.28

En lazo cerrado para una entrada impulso tenemos

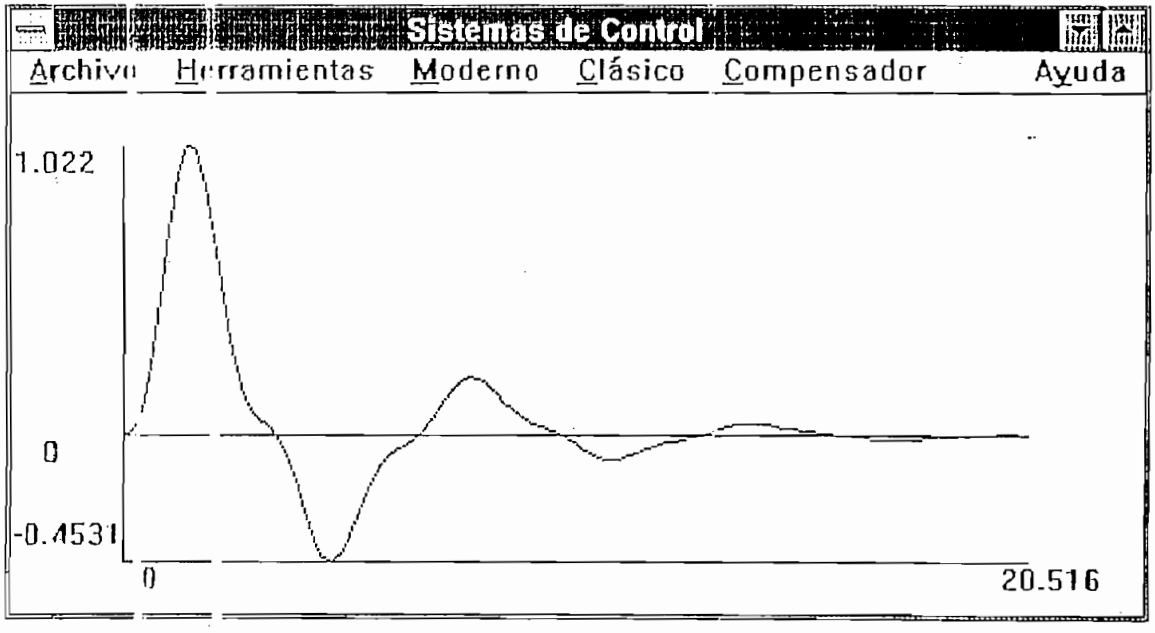


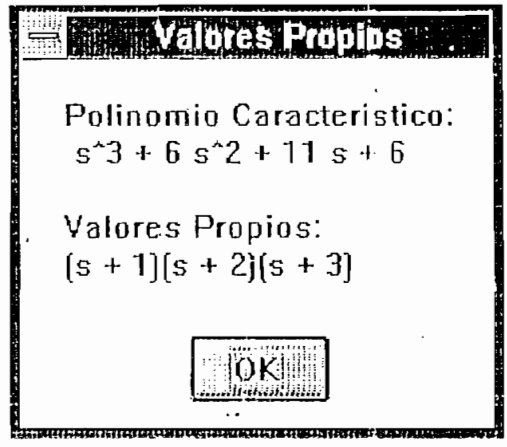
FIGURA 4.29

Análisis en variables de estado

Considere la siguiente matriz A:

$$A = \begin{matrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 6 & -11 & -6 \end{matrix}$$

Determine la ecuación característica y los valores propios. Los Resultados son:



Figur B.30

Dada la matriz A

$$A = \begin{matrix} 1 & 0 \\ 0 & 2 \end{matrix}$$

Determine el polinomio característico, los valores propios, la matriz $(sI - A)^{-1}$, y la matriz e^{At}

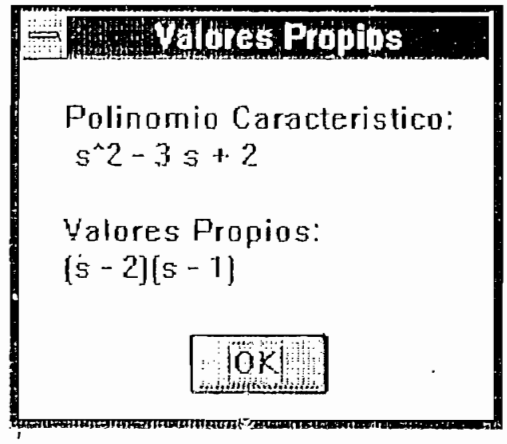


FIGURA 4.30

$(sI - A)^{-1}$ nos da

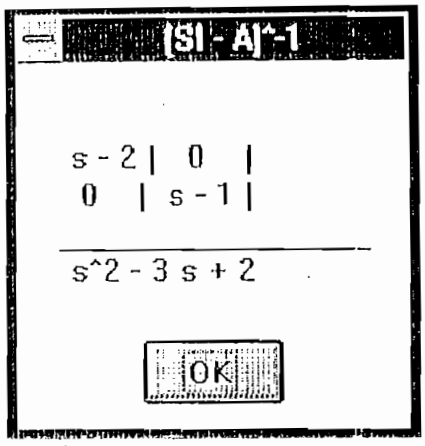


FIGURA 4.31

y la matriz e^{At}

FIGURA 4.32

Con valores múltiples tenemos

$$A = \begin{matrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & -3 & 3 \end{matrix}$$

El polinomio característico y los valores propios son:

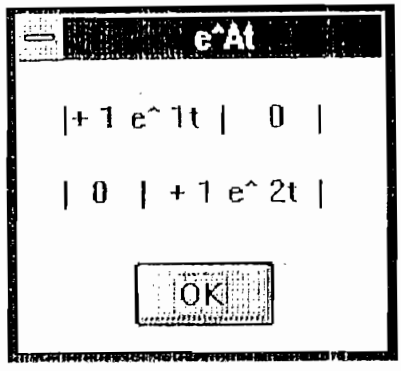


FIGURA 4.33

El calculo de e^{At} esta grabado en el archivo eat.mtz cuyo contenido es

|+ 0.5 t^2 e^0.99t + -1 t^1 e^1t + 1 e^ 1t | + -1 t^2 e^0.99t + 1 t^1 e^1t | + 0.5 t^2 e^0.99t |

|+ 0.5 t^2 e^0.99t | + -0.99 t^2 e^0.99t + -1 t^1 e^1t + 1 e^ 1t | + 0.49 t^2 e^0.99t + 1 t^1 e^1t |

|+ 0.49 t^2 e^0.99t + 1 t^1 e^1t | + -0.99 t^2 e^0.99t + -3 t^1 e^1t | + 0.49 t^2 e^0.99t + 1.99t^1 e^1t + 1 e^ 1t |

Obtenga la respuesta temporal del siguiente sistema

$$\begin{matrix} x1 \\ x2 \end{matrix} = \begin{matrix} 0 & 1 \\ -2 & -3 \end{matrix} \begin{matrix} X1 \\ X2 \end{matrix} \begin{matrix} 0 \\ 1 \end{matrix} u$$

$u(t) = 1(t)$

Primero obtenemos la matriz de transición de estado que esta dada por $L^{-1}[(Si - A)^{-1}]$

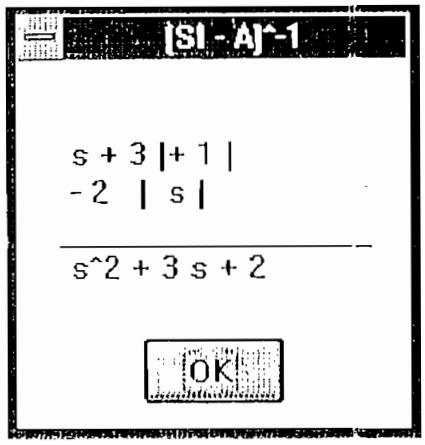


FIGURA 4.34

Si el estado inicial es cero entonces

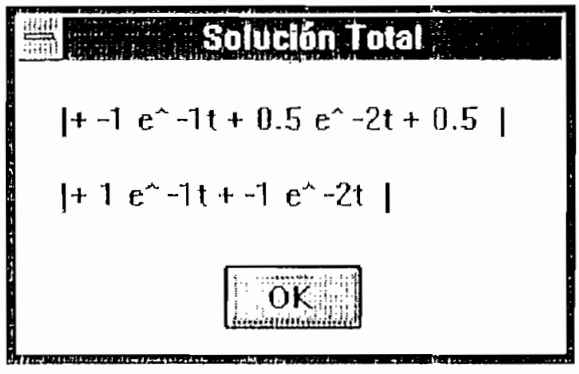


FIGURA 4.35

Determinar si el sistema es controlable

$$A = \begin{matrix} 1 & 1 \\ 0 & -1 \end{matrix} \qquad B = \begin{matrix} 1 \\ 0 \end{matrix}$$

Los resultados del programa son

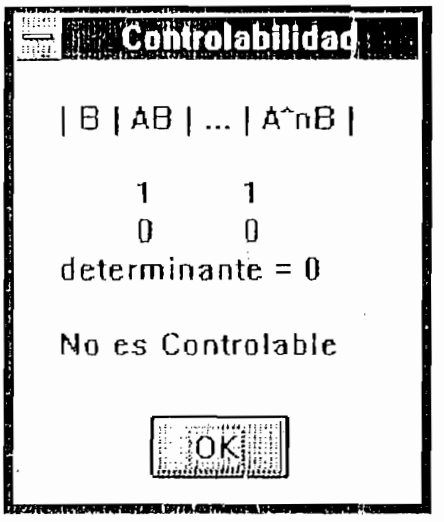


FIGURA 4.36

Para el sistema

$$A = \begin{matrix} 1 & 1 \\ 2 & -1 \end{matrix} \qquad B = \begin{matrix} 0 \\ 1 \end{matrix}$$

Los resultados son

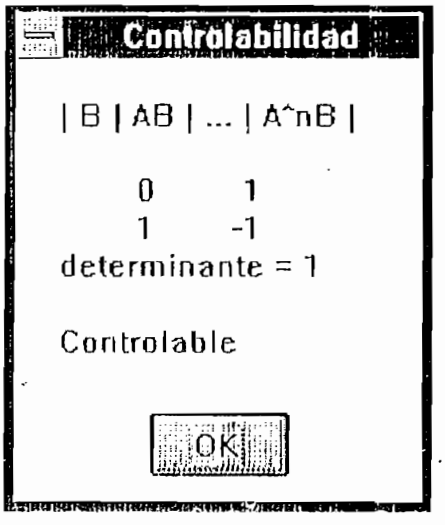


FIGURA 4.37

Es el sistema controlable

$$A = \begin{matrix} -1 & 1 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & -2 \end{matrix} \qquad b = \begin{matrix} 4 & 2 \\ 0 & 0 \\ 3 & 0 \end{matrix}$$

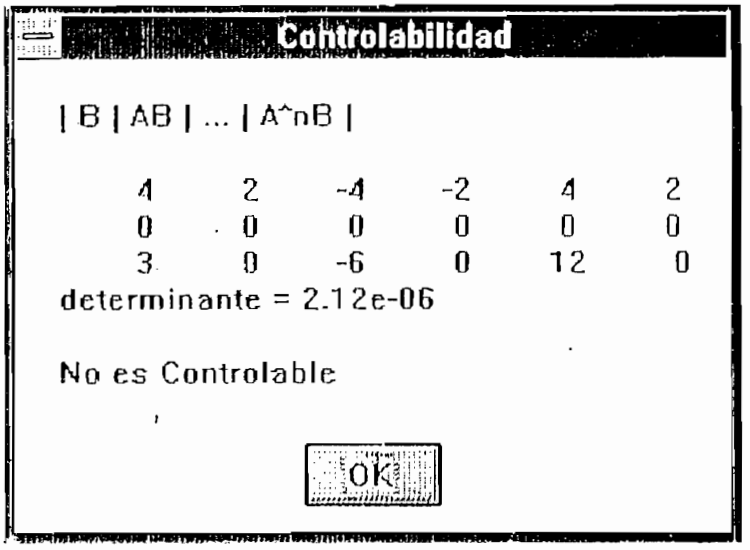


FIGURA 4.38

El determinante corresponde al del gramiano

Es el siguiente sistema observable

$$A = \begin{matrix} 1 & 1 \\ -2 & -1 \end{matrix} \qquad C = \begin{matrix} 1 & 0 \end{matrix}$$

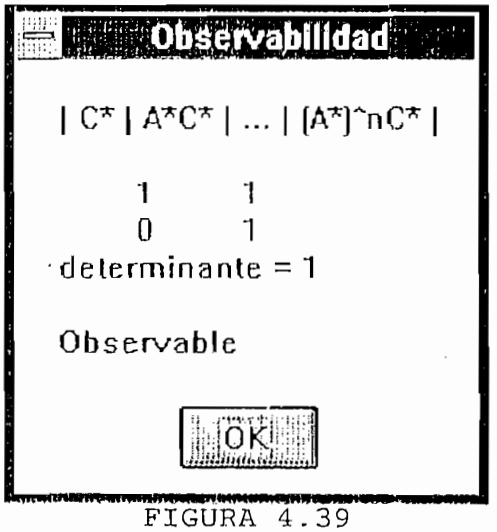


FIGURA 4.39

Muestre que el siguiente sistema no es observable

$$A = \begin{matrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ -6 & 11 & -6 \end{matrix} \qquad C = \begin{matrix} 4 & 5 & 1 \end{matrix}$$

el programa nos da

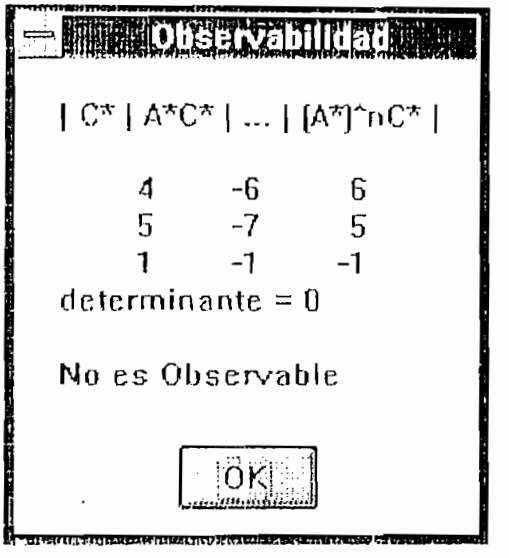


FIGURA 4.40

EJEMPLO 25

Considere el ejemplo del péndulo invertido, montado en un carro con propulsión a motor. Se desea mantener vertical al péndulo invertido en presencia de perturbaciones.

Diseñe un sistema de control de modo que dada cualquier condición inicial, el péndulo sea llevado de retorno a la posición vertical al igual que el vehículo

$$A = \begin{matrix} 0 & 1 & 0 & 0 \\ 20.601 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ -0.4905 & 0 & 0 & 0 \end{matrix}$$

$$B = \begin{matrix} 0 \\ \\ \\ \end{matrix} \qquad C = \begin{matrix} 1 & 0 & 0 & 0 \end{matrix}$$

-1 0 0 1 0

0

0.5

Se utiliza el control de realimentación de estado

Se verifica que el sistema es de estado completo controlable

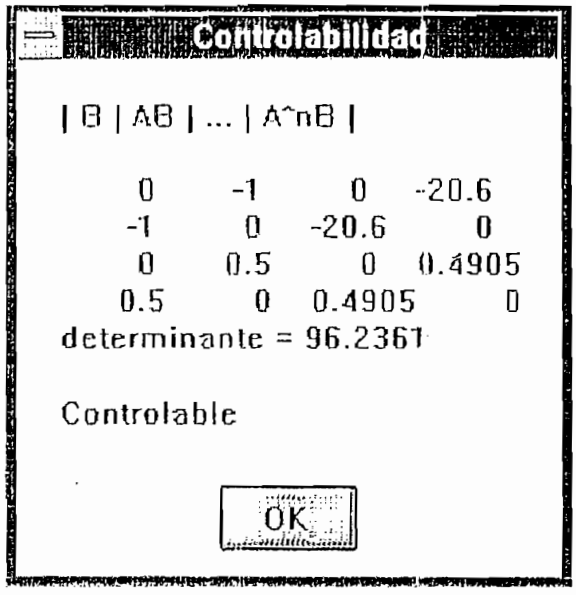


FIGURA 4.41

La ecuación característica esta dada por

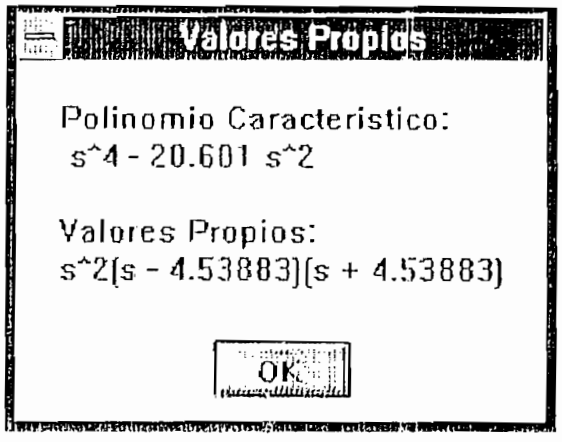


FIGURA 4.42

Con esto podemos obtener la matriz de ganancia de realimentación K = [-298.15 -60.697 -163.099 -73.394], con condiciones iniciales x1 = 0.1, x2 = 0, x3 = 0, x4 =0, La matriz A del sistema compensado es

$$A = \begin{matrix} 0 & 1 & 0 & 0 \\ -277.549 & -60.697 & -163.099 & -73.394 \\ 0 & 0 & 0 & 1 \\ 148.5845 & 30.3485 & 81.5495 & 36.697 \end{matrix}$$

La respuesta en el tiempo del sistema es

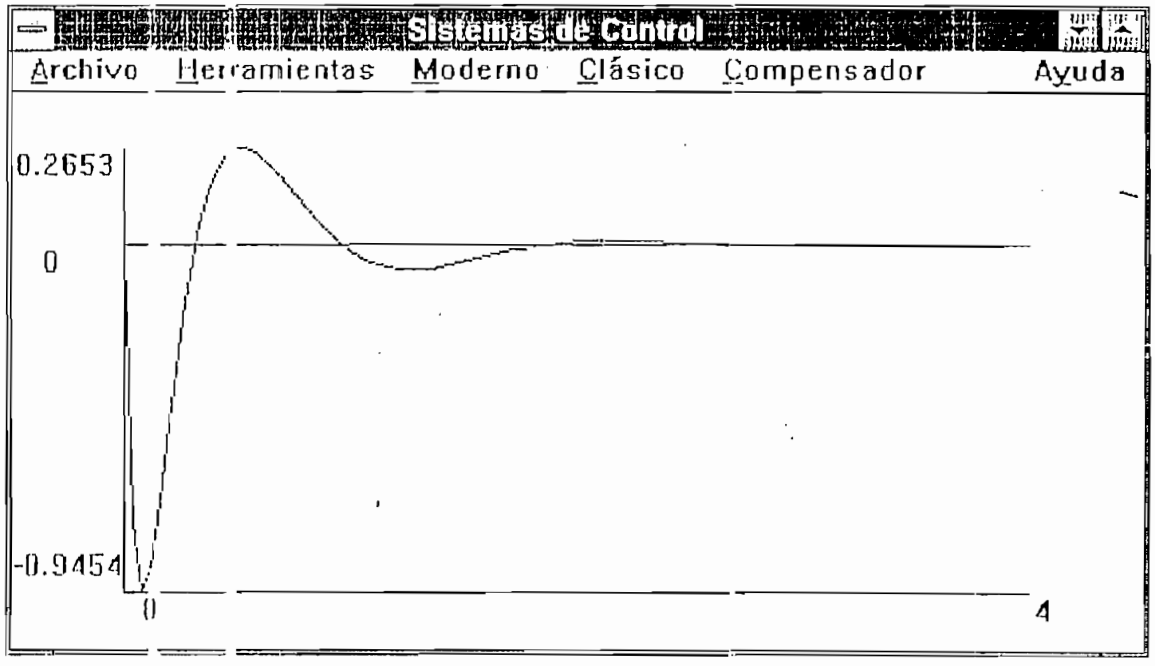


FIGURA 4.43

x(t) es

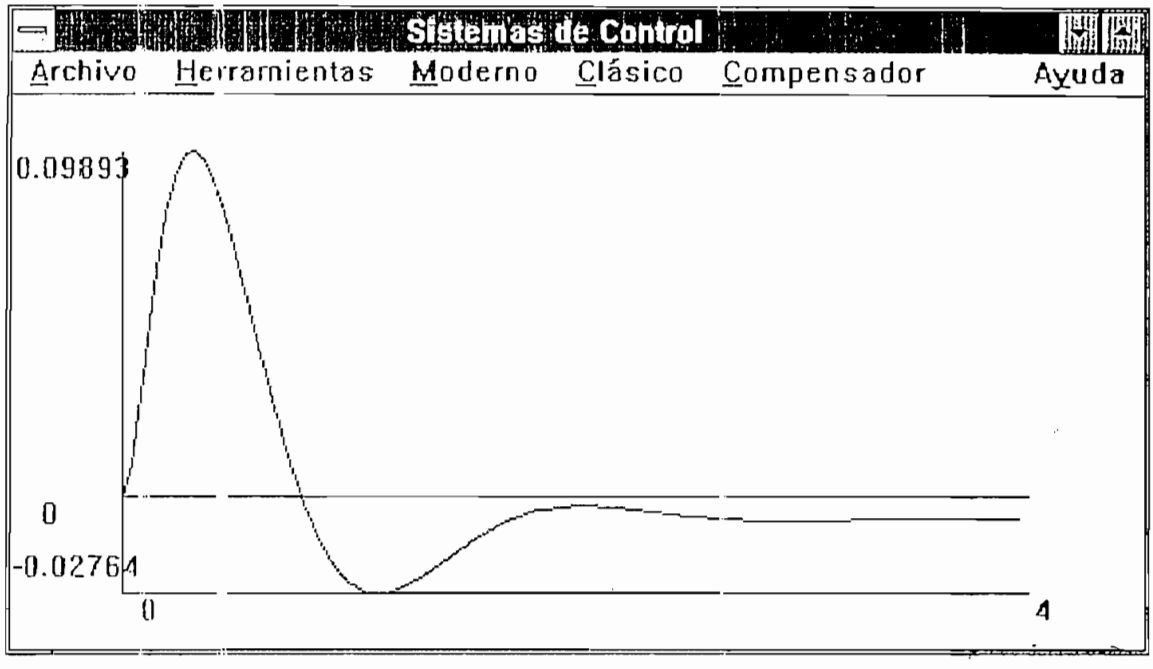


FIGURA 4.44

x4(t) es

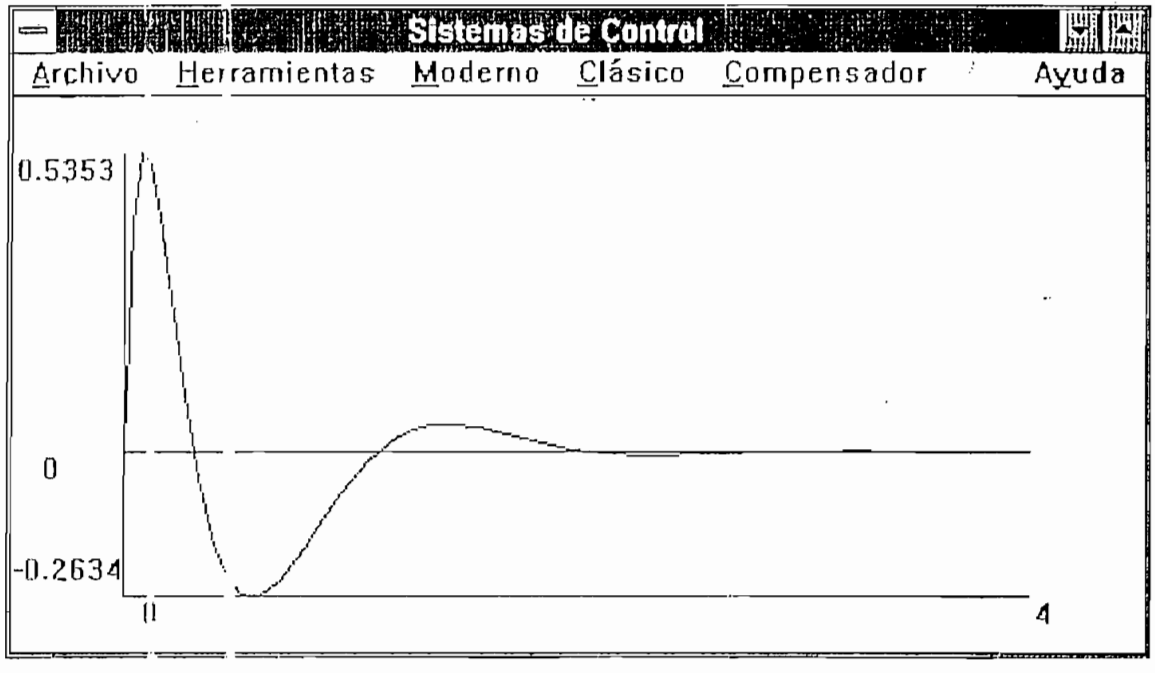


FIGURA 4.45

x1(t) es

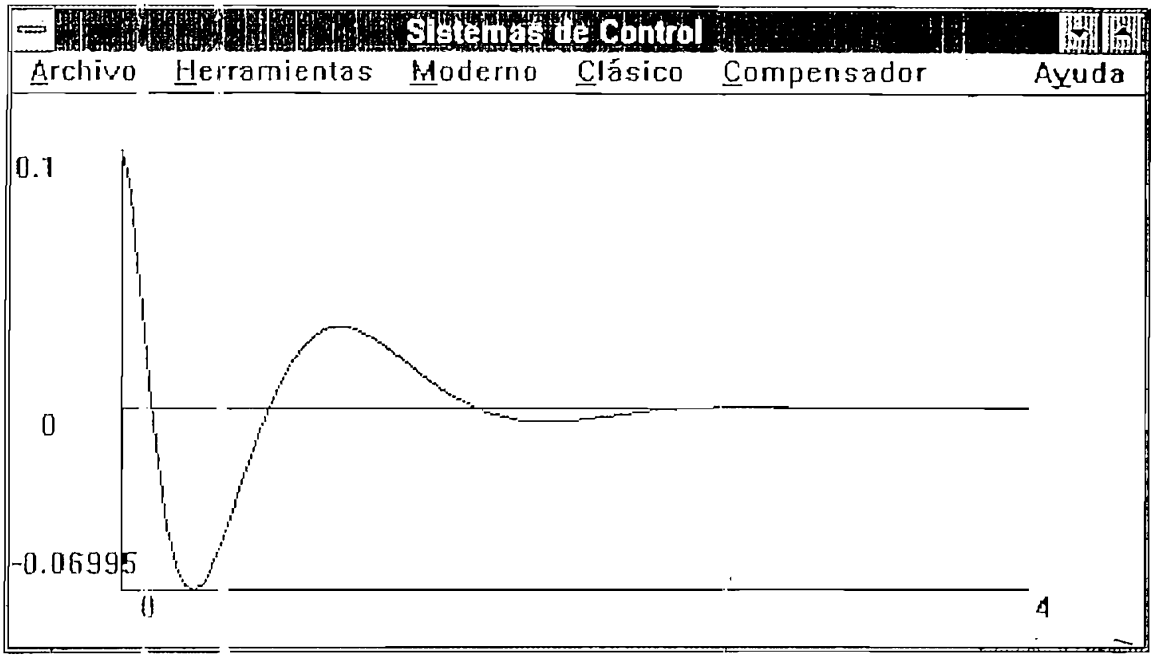


FIGURA 4.46

CAPITULO V
CONCLUSIONES Y RECOMENDACIONES

4.1 CONCLUSIONES

- La gran difusión de computadores personales, permite utilizar las computadoras personales para analizar sistemas de control con mucha facilidad y exactitud.

- El compilador usado (Borland C ++ 3.0) permite producir aplicaciones de Windows, lo que nos produce las siguietes ventajas con respecto a DOS, como son principalmente:

    - Sistema Multitarea.
    - Pantallas redimensionables.
    - No necesita capturadores gráficos.
    - Uso de comandos comunes a todas las aplicaciones de Windows.

- La principal desventaja, de trabajar bajo Windows es su velocidad, sin embargo este programa es más rápido que el programa CC que corre bajo DOS.

Respecto al programa usado como base para desarrollar esta tesis tenemos:

- El programa CC es más completo, que el desarrollado en esta tesis, pero debe recordarse que fue desarrollado por cientos de personas.

- El obtener la respuesta de sistemas de manera literal, permite dar facilidades al usuario para analizar sistemas, pero complica el programa. El análisis usando variables de estado no lo realiza de manera literal, el programa CC.

Como limitaciones del programa podemos mencionar lo siguiente:

- El grado máximo del numerador o denominador de la función de transferencia usada es 10, esto es suficiente ya que el modelo matemático, de un sistema físico generalmente se reduce a una función de transferencia de grado 2 o 3.

- El programa dará resultados erroneos cuando la función de transferencia tenga polos o ceros que contengan raices imaginarias repetidas.

- Si en el ingreso de la función de transferencia, existe un error no identificado, el archivo que contiene la información de la función de transferencia será destruido; al iniciar el programa nuevamente, el programa no se cargará, para solucionar el problema cree un archivo (en formato ASCII) llamado funcion.ft con la información 0 1 0 1.

Los ejemplos presentados en esta tesis y otros que se han desarrollado nos indican lo siguiente:

- Los ejemplos presentan resultados que son muy satisfactorios, lo que da validación al programa.

- El desarrollo de los ejemplos resultó muy sencillo gracias a las facilidades de uso y flexibilidad que presenta el programa.

Con respecto a las subrutinas desarrolladas tenemos:

- El tener el listado de un programa, permite usar las subrutinas desarrolladas en nuevas aplicaciones, e incluso modificar las mismas para cumplir los requerimientos deseados en una nueva aplicación.

- El nivel al que está desarrollado este programa (programación orientada a objetos), es similar al nivel en el que se realizan los programas en la actualidad.

- El programa presentado en esta tesis es una herramienta muy útil para el análisis y diseño de sistemas de control, tanto en lo que respecta a sistemas de una entrada una salida (usando métodos clásicos) o de múltiples entradas y múltiples salidas (teoría de control moderna). Las facilidades de uso, así como la velocidad

de respuesta son superiores a las del programa que se ha tomado como base el programa CC.

- Un programa complejo y extenso como el realizado en esta tesis no puede estar libre de errores en su primera versión, y siempre se pueden realizar mejoras.

4.2 RECOMENDACIONES

En base a la experiencia adquirida durante el desarrollo de esta tesis, se dá las siguientes recomendaciones en cuanto a la implementación de uno o varios programas que complementen al programa desarrollado. El programa podría cumplir las siguientes características:

- Determinar automáticamente el compensador necesario para cumplir ciertos requerimientos en tiempo y frecuencia.
- Compensación en la realimentación.
- Posibilidad de usar un compensador ON-OFF.
- Compensar un sistema descrito en variables de estado, usando realimentación de estado y/o control óptima.
- Encontrar la función de transferencia equivalente de un sistema descrito en variables de estado.
- Desarrollar un programa similar usando control discreto.
- Identificación e ingreso de la función de transferencia de una planta real, usando una tarjeta de adquisición de datos.

Este es un campo tan amplio que seguramente quedan muchas cosas más por hacer, por lo que no es necesario hacer más recomendaciones.

BIBLIOGRAFIA

1.- OGATA KATSUHIKO, "Ingeniería de Control Moderna", Pretince-Hall Internacional, 1973.

2.- KUO BENJAMIN, "Sistemas Automáticos de Control", Prentice-Hall Internacional, 1970.

3.- THOMPSON PETER, "Computer Aided Control Systems", System Technology, 1985.

4.- TED FAISON, "Borland C++ 3.1 Object-Oriented Programming", Prentice-Hall Computer Publishing, 1992.

5.- BEN EZZELL, "Turbo C++ Programming", Addison-Wesley, 1990.

ANEXO A
LISTADO DEL PROGRAMA

```
/*
** sima.cpp .- Subrutinas necesarias para el calculo de
**             Matrices en el espacio de estado
** Proyecto:   CONTROL MODERNO
** Fecha:      10 de Enero 1994
** Revision:   20 de Enero 1994
** Comenarios: Este archivo contiene 6 funciones necesarias
**           para obtener (SI-A)^-1, e^At, integral de e^At
** Funciones  bairstow, muestra de polinomios, transformada
** llamadas:   inversa de Laplace, expancion en fraciones
**             parciales
** Variables:  Matriz A, B, C, D, Vectores X0, u
*/

#include <iostream.h>
#include <string.h>
#include <stdio.h>
#include <stdlib.h>
#include <conio.h>
#include <complex.h>
#include "moderno.h"

void far escribe(char *arch[],float x_leer[], int n);
void far mostrar_matriz(int n, float matriz[][max_m], int
tipo_entrada, float pc[], char s[]);
void far mostrar_matriz(int n, int m, float matriz[][max_m],
char s[]);
void far mostrar_matriz(int n, float matriz[][max_m][max_m],
int tipo_entrada, float pc[], int m, char s[]);
void far mult_anxb(int n, int m, float a[][max_m], float
b[][max_m], float c[][max_m], int opcion, char s[]);
void far mult_bxu(int n, int m, float b[][max_m], float u[],
float bxu[]);
void far mult_exb(int n,   float matriz[][max_m][max_m],
float b[], float matriz1[max_m][max_m]);
void far sima(float a[][max_m],float f[][max_m][max_m],float
t[], int n);
```

```
void far suma(int m, float a[], int n, float b[], int &max,
float c[]);
void far resp_total(int n, float a[][max_m], float
b[][max_m], int tipo_entrada, float pc[], char s[]);

/*
** funcion traza retorna la suma de los elementros de la
** diagonal de la matriz tridimencional A
*/
float far traza(int n, float a[][max_m][max_m], int k)
{
     int i;
     float suma=0;

     for(i=0;i<n;i++) suma=a[i][i][k]+suma;
     for(i=0;i<n;i++) a[i][i][k]=a[i][i][k]-suma/k;
     return -suma/k;
}

/*
** Multiplica a*b(k) y lo almacena en b(k+1)
*/
void far producto(int n, float a[][max_m], float
b[][max_m][max_m], int k)
{
     int i, j, l;
     float suma;

     for(i=0;i<n;i++)
     {
          for(j=0;j<n;j++)
          {
               suma=0;
               for(l=0;l<n;l++)
                    suma=a[i][l]*b[l][j][k]+suma;
               b[i][j][k+1]=suma;
```

```
            }
      }
}

/*
**---------- Muestra (sI-A)^-1----------------
*/
void far mostrar_matriz(int n, float matriz[][max_m][max_m],
char s1[])
{
      int i,j,k,m,maxim[max_m],temp;
      float pol[max_m];
      char s[200];

      strcpy(s1,"");
      for(j=0;j<n;j++)
      {
            maxim[j]=2;
            for(i=0;i<n;i++)
            {
                  for(k=0;k<n;k++)
pol[k]=matriz[i][j][k];
                  mostrarp(n-1,pol,s);
                  if (strlen(s)>maxim[j]) maxim[j]=strlen(s);
            }
      }
      for(i=0;i<n;i++)
      {
            strcat(s1,"\n");
            for(j=0;j<n;j++)
            {
                  for(k=0;k<n;k++) pol[k]=matriz[i][j][k];
                  mostrarp(n-1,pol,s);
                  if (strlen(s)<2) strcpy(s,"0");
                  temp=strlen(s);
                  for(m=0;m<(maxim[j]-temp)/2;m++)strcat(s1,"
```

```
");
                strcat(s1, s);
                for(m=0;m<(maxim[j]-temp+1)/2;m++)strcat(s1,
" ");
                strcat(s1, "|");
          }
     }
}

/*
** integral de e^at si opcion =1 fraciones parciales
*/
void far mostrar_matriz(int n, float matriz[][max_m][max_m],
int tipo_entrada, float pc[max_m], int m1, char s1[])
{
     int i,j,k,m,l;
     float pol[max_m];
     char s[20*max_m];
     int opcion=0;

     strcpy(s1,"");
     for(i=0;i<n;i++){
          if (i!=0) strcat(s1, "\n\n");
          strcat(s1,"|");
          for(j=0;j<m1;j++){
               for(k=0;k<n;k++)    pol[k]=matriz[i][j][k];
                    l=0;
                    while (pol[l]==0) l++;
                    for (m=0;m<n;m++)   pol[m]=pol[m+l];
                    if (l>=n) strcpy(s,"  0  ");
                    e          l          s          e
pasar(pol,pc,n-l-1,n+tipo_entrada,opcion,s,0,0);
                    strcat(s1,s);
                    strcat(s1, " |  ");
          }
     }
```

```
}


/*
**-------evaluacion de (sI-A)^-1 ---------
*/
void far sima(float a[][max_m],float f[][max_m][max_m],float
t[], int n)
{
     int i,j;
//Matriz f0 identidad
     for(i=0;i<n;i++)
          for(j=0;j<n;j++)
               if (i==j) f[i][j][0]=1;
               else f[i][j][0]=0;

// F(n) = A*F(n-1) + T(n-1)*I ; en t[i]=traza
     for(i=1;i<=n;i++){
          producto(n,a,f,i-1);
          t[i]=traza(n,f,i);
     }
     t[0]=1;
}

/*
**------multiplica e^at*b---
*/
void far mult_exb(int n, float matriz[][max_m][max_m],
float b[], float matriz1[max_m][max_m])
{
     int i, k, l;
     float suma;

     for(k=0;k<n;k++)
          for(i=0;i<n;i++)
          {
```

```
                suma=0;
                for ( l = 0 ; l < n ; l + + )
suma=matriz[i][l][k]*b[l]+suma;
                matriz1[i][k]=suma;
          }
}

/*
**-----multiplica b*u ---
*/
void far mult_bxu(int n, int m, float b[][max_m], float u[],
float bxu[])
{
     int i,j;
     float suma;

     for(i=0;i<n;i++)
     {
          suma=0;
          for(j=0;j<m;j++)
          {
               suma=b[i][j]*u[j]+suma;
          }
          bxu[i]=suma;
     }
}

void far mult_anxb(int n, int m, float a[][max_m], float
b[][max_m], float c1[][max_m], int opcion, char s[])
{
     int i,j,l,k;
     float c[max_m][max_m];//posiblemente requiere aumentar
tamanio
     float    suma,temp[max_m][max_m],    an[max_m][max_m],
at[max_m][max_m];
```

```
if (opcion==1)
{
      for (i=0;i<m;i++)
            for(j=0;j<n;j++)
                  c[j][i]=b[i][j];
      for (i=0;i<n;i++)
            for(j=0;j<n;j++)
                  at[j][i]=a[i][j];
}
else
{
      for(i=0;i<n;i++)
            for(j=0;j<m;j++)
                  c[i][j]=b[i][j];
      for (i=0;i<n;i++)
            for(j=0;j<n;j++)
                  at[i][j]=a[i][j];
}

for(i=0;i<n;i++)
      for(j=0;j<n;j++)
      {
            if (i==j) temp[i][j]=1;
            else temp[i][j]=0;
      }

for(k=1;k<n;k++)
{
      for(i=0;i<n;i++)
            for(j=0;j<n;j++)
            {
                  suma=0;
                  for(l=0;l<n;l++)
                        suma=temp[i][l]*at[l][j]+suma;
                  an[i][j]=suma;
            }
```

```
            for(i=0;i<n;i++)
                  for(j=0;j<n;j++)
                        temp[i][j]=an[i][j];
//          mostrar_matriz(n,n,an);

            for(i=0;i<n;i++)
                  for(j=0;j<m;j++)
                  {
                        suma=0;
                        for(l=0;l<n;l++)
                              suma=an[i][l]*c[l][j]+suma;
                        c[i][j+k*m]=suma;
                  }
      }
//    strcpy(s,"");
      char s1[100];
      for (i=0;i<n;i++)
      {
            for(j=0;j<n*m;j++)
            {
                  sprintf(s1,"%10.4g", c[i][j]);
                  strcat(s,s1);
            }
            if (j!=n-1) strcat(s,"\n");
      }

// Multiplica b, ab .. anb por transpuesta

      if (m>1)
      {
            for(i=0;i<n;i++)
                  for(j=0;j<n;j++)
                  {
                        suma=0;
                        for(l=0;l<n*m;l++)
                              suma=c[i][l]*c[j][l]+suma;
```

```
                    temp[i][j]=suma;
               }
          for(i=0;i<n;i++)
               for(j=0;j<n;j++)
                    c1[i][j]=temp[i][j];
     }
     else
     {
          for(i=0;i<n;i++)
               for(j=0;j<n;j++)
                    c1[i][j]=c[i][j];
     }
}
void far multi_exb(int n, float matriz[][max_m][max_m], float
b[])
{
     int i, j, k, l;
     float suma;
     float matriz1[max_m][max_m][max_m];

     for(k=0;k<n;k++)
          for(i=0;i<n;i++)
          {
               suma=0;
               for ( l = 0 ; l < n ; l + + )
suma=matriz[i][l][k]*b[l]+suma;
               matriz1[i][0][k]=suma;
          }

     for (i=0; i<n; i++)
          for (j=0; j<n; j++)
               for (k=0;k<n; k++)
                    matriz[i][j][k]=matriz1[i][j][k];
}
void far mostrar_matriz(int n, int m, float matriz[][max_m],
char s[])
```

```
{
     int i,j;
     char s1[200];

     strcpy(s,"");
     for(i=0;i<n;i++)
     {
          for(j=0;j<m;j++)
          {
               sprintf(s1,"%g        ",matriz[i][j]);
               strcat(s,s1);
          }
          if (i!=(n-1)) strcat(s,"\n");
     }
}

void far resp_total(int n, float a[][max_m], float
b[][max_m], int tipo_entrada, float pc[max_m], char s1[])
{
     int i,k,m,l1,l2,max;
     float pol1[max_m], pol2[3*max_m], pol3[max_m];
     char s[20*max_m];
     extern char *pasa[1];
     int opcion=0;

     strcpy(s1,"");
     for(i=0;i<n;i++){
          if (i!=0) strcat(s1, "\n\n");
          strcat (s1,"|");
          for(k=0;k<n;k++)    pol1[k]=a[i][k];
          for(k=0;k<n;k++)    pol2[k]=b[i][k];
          l1=0;
          while (pol1[l1]==0) l1++;
          for (m=0;m<n;m++)   pol1[m]=pol1[m+l1];
          l2=0;
          while (pol2[l2]==0) l2++;
```

```
            for (m=0;m<n;m++)    pol2[m]=pol2[m+12];

suma(n-11-1+tipo_entrada,pol1,n-12-1,pol2,max,pol3);

            //------genera archivo de salida --------
            sprintf(s,"g_%d.ft",i+1);
            *pasa=s;
            pol2[0]=n+tipo_entrada;
            for (m=0;m<=n+tipo_entrada;m++) pol2[m+1]=pc[m];
            pol2[n+tipo_entrada+2]=max;
            f o r      ( m = 0 ; m < = m a x ; m + + )
pol2[m+n+tipo_entrada+3]=pol3[m];
            escribe(pasa,pol2,max+n+tipo_entrada+4);
            //------fin de generacion---------

            if (pol3[0]==0) strcpy(s,"  0  ");
            e                l                s                e
pasar(pol3,pc,max,n+tipo_entrada,opcion,s,0,0);
            strcat(s1,s);
            strcat(s1, " |  ");
      }
}

//--respuesta natural y forzada---
void far mostrar_matriz(int n, float matriz[][max_m], int
tipo_entrada, float pc[max_m], char s1[])
{
      int i,k,m,l;
      float pol[max_m];
      char s[20*max_m];
      int opcion=0;

      strcpy(s1,"");
      for(i=0;i<n;i++){
            if (i!=0) strcat(s1, "\n\n");
            strcat(s1,"|");
```

```
            for(k=0;k<n;k++)     pol[k]=matriz[i][k];
            l=0;
            while (pol[l]==0) l++;
            for (m=0;m<n;m++)    pol[m]=pol[m+l];
            if (l>=n) strcpy(s,"  0  ");
            e                l                s                e
pasar(pol,pc,n-l-1,n+tipo_entrada,opcion,s,0,0);
            strcat(s1,s);
            strcat(s1, " |  ");
      }
}
/*
**********************************************************
** Proyecto:    CONTROL CLASICO Y MODERNO
** Fecha:       3 de Diciembre 1993
** Revision:    22 de Febrero 1994
** Comenarios: Este archivo reune todos las subrutinas
**              necesarias para hacerlas funcionar en un
solo
**              programa integrado
**
** Funciones   bairstow, muestra de polinomios, transformada
** llamadas:    inversa de Laplace, expancion en fraciones
**              parciales, sima, lgr, etc.
**    Usa como variables gloabales las siguientes para
**    describir las funciones de transferencia
**    a ... vector en el que se almacena los coeficientes de
**    numerador
**    b ... vector en el que se almacena los coeficientes del
**    denominador
**    m ... grado del numerador
**    n ... grado del denominador
**    Las matrices que describen las variables de estado son:
**    a1 ... Representa la matriz A
**    b1 ... Representa la matriz B
**    c1 ... Representa la matriz C
```

```
**    d1 ... Representa la matriz D
*******************************************************
*/
#include <edit.h>
#include <chkbox.h>
#include <radio.h>
#include <owl.h>
#include <filewnd.h>
#include <mdi.h>
#include <string.h>
#include <io.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <owl.h>
#include <complex.h>
#include <inputdia.h>
#include "helpwind.h"
#include "midialog.h"
#include "dialogo.h"
#include "clasico.h"
#include "moderno.h"

float angulos(int m, float x1, float y1, float x[],  float
y[]);
float maximovalor (float a[],int m,float b[], int n, float
c[], int l)
float minimovalor (float a[],int m,float b[], int n, float
c[], int l)
void limites(float valor[], int n, float &minimo);
void far f_cadena(char s[], char *arch);
void far s_cadena(char s[], char *arch);
void far mostrar_matriz(int n, float matriz[][max_m], int
tipo_entrada, float pc[], char s[]);
void far mostrar_matriz(int n, int m, float matriz[][max_m],
char s[]);
```

```
void far mostrar_matriz(int n, float matriz[][max_m][max_m],
int tipo_entrada, float pc[], int m, char s[]);
void far mult_anxb(int n, int m, float a[][max_m], float
b[][max_m], float c[][max_m], int opcion, char s[]);
void far mult_bxu(int n, int m, float b[][max_m], float u[],
float bxu[]);
void far mult_exb(int n, float matriz[][max_m][max_m], float
b[], float matriz1[max_m][max_m]);
void far suma(float a[][max_m],float f[][max_m][max_m],float
t[], int n);
void far resp_total(int n, float a[][max_m], float
b[][max_m], int tipo_entrada, float pc[], char s[]);

void far leer_funcion();
void far suma(int m, float a[], int n, float b[], int &max,
float c[]);
void far leer_matriz();
void far lode(float a[], float b[], int m, int n, float
t_min, float t_max);
void far derivada_f(int m, float a[], int n, float b[],
float c[], float d[], int &max);

float far a1[max_m][max_m], b1[max_m][max_m];
float far c1[max_m][max_m], d1[max_m][max_m];
float far forzada[max_m][max_m], natural[max_m][max_m];
float far axb[max_m][max_m];
float far c0[max_m],u[max_m],bxu[max_m];
float far t[max_m], f[max_m][max_m][max_m];
float far x_leer[120], a[maximo], b[maximo], r[maximo],
im[maximo] modulo;
float far ac[maximo], bc[maximo];

int far n1 m1, r1, tipo_ent;
int far fa la_ent=0, existe_funcion=0;
int i, n, m, nc, mc, hecho=0, existe_compensador=0;
int hacer_graf=0;
```

```
char far s[14000],s2[200];
char *pas[1], *funcion[1], *compen[1];

class TAboutDialog: public TDialog {
public:
            TAboutDialog(PTWindowsObject AParent)
                                    :   TDialog(AParent,
"DIALOG_ABOUT") {}
};

const int WIDTHSIZE  = 6;
const int HEIGHTSIZE = 6;

struct {
      char x_maximo [WIDTHSIZE];
      char y_minimo [HEIGHTSIZE];
      WORD automatico;
      WORD impulso, paso, rampa;
      WORD abierto, cerrado;
      WORD limpiar, compensado;
} Data /* Estructura  en la cual se ingresan las opciones
para graficar en el tiempo */

struct {
      char  x_maxima [WIDTHSIZE];
      char  y_minima [HEIGHTSIZE];
      WORD modeauto;
      WORD modulo, angulo, ambos;
      WORD x_abierto, y cerrado;
      WORD  impiar, compensado;
} Dato /* Estructura  en la cual se ingresan las opciones
para graficar en frecuencia */

class TBitmapAttributesDialog: public TDialog {

            TEdit* widthEdit;
```

```
            TEdit* heightEdit;
            TBCheckBox* resizeCheckBox;
            TBRadioButton* colors2RadioButton;
            TBRadioButton* colors16RadioButton;
            TBRadioButton* colors256RadioButton;
            TBRadioButton* windowsRadioButton;
            TBRadioButton* os2RadioButton;
            TBRadioButton* noneRadioButton;
            TBRadioButton* rle4RadioButton;
public:
            TBitmapAttributesDialog(PTWindowsObject);
};
// Ingreso de valores a estructura
TBitmapAttributesDialog::TBitmapAttributesDialog(
                                        PTWindowsObject
AParent)
            : TDialog(AParent, "MIDIALOGO")
{
             widthEdit = new TEdit(this, ID_WIDTH, WIDTHSIZE);
             heightEdit   =   new   TEdit(this,   ID_HEIGHT,
HEIGHTSIZE);
            resizeCheckBox =
                 new TBCheckBox(this, ID_RESIZE, 0);
            colors2RadioButton =
                 new TBRadioButton(this, ID_2COLORS, 0);
            colors16RadioButton =
                 new TBRadioButton(this, ID_16COLORS, 0);
            colors256RadioButton =
                 new TBRadioButton(this, ID_256COLORS, 0);
            windowsRadioButton  =
                 new TBRadioButton(this, ID_WINDOWS, 0);
            os2RadioButton  =
                 new TBRadioButton(this, ID_OS2, 0);
            noneRadioButton  =
                 new TBRadioButton(this, ID_NONE, 0);
            rle4RadioButton  =
```

```
                new TBRadioButton(this, ID_RLE4, 0);
//          rle8RadioButton =                          new
TBRadioButton(this, ID_RLE8, 0);
            TransferBuffer = (LPSTR) &Data;
}
class BodeDialog: public TDialog {

            TEdit* widthEdit;
            TEdit* heightEdit;
            TBCheckBox* resizeCheckBox;
            TBRadioButton* colors2RadioButton;
            TBRadioButton* colors16RadioButton;
            TBRadioButton* colors256RadioButton;
            TBRadioButton* windowsRadioButton;
            TBRadioButton* os2RadioButton;
            TBRadioButton* noneRadioButton;
            TBRadioButton* rle4RadioButton;
public:
            BodeDialog(PTWindowsObject);
};

BodeDialog::BodeDialog(
                                        PTWindowsObject
AParent)
            : TDialog(AParent, "BODEDIALOGO")
{
             widthEdit = new TEdit(this, ID_WIDTH, WIDTHSIZE);
             heightEdit   =   new   TEdit(this,   ID_HEIGHT,
HEIGHTSIZE);
            resizeCheckBox =
                 new TBCheckBox(this, ID_RESIZE, 0);
            colors2RadioButton =
                 new TBRadioButton(this, ID_2COLORS, 0);
            colors16RadioButton =
                 new TBRadioButton(this, ID_16COLORS, 0);
            colors256RadioButton =
```

```
                new TBRadioButton(this, ID_256COLORS, 0);
            windowsRadioButton  =
                new TBRadioButton(this, ID_WINDOWS, 0);
            os2RadioButton  =
                new TBRadioButton(this, ID_OS2, 0);
            noneRadioButton  =
                new TBRadioButton(this, ID_NONE, 0);
            rle4RadioButton  =
                new TBRadioButton(this, ID_RLE4, 0);
//          rle8RadioButton  =     new  TBRadioButton(this,
ID_RLE8, 0);
            TransferBuffer = (LPSTR) &Dato;
}

_CLASSDEF(TMDIFileApp)
_CLASSDEF(TMDIFileWindow)

_CLASSDEF(TMyWindow)
class TMyWindow : public TWindow
{
public:
     char ingreso[100];
     char ingreso_com[100];
     WORD ChildNum;
     HDC DragDC;
     BOOL ButtonDown;
     HPEN ThePen;
     int Xmax, Ymax;
     int PenSize;

          TMyWindow(PTWindowsObject AParent,
                                        LPSTR Title)
                                : TWindow(AParent, Title) {
     ChildNum = 1;
     strcpy(ingreso,"");
     strcpy(ingreso_com,"");
```

```
funcion="funcion.fi";
leer_matriz();
leer_funcion();
ButtonDown = FALSE;
PenSize = 1;
ThePen = CreatePen(PS_SOLID, PenSize, 0);
}
        LPSTR GetClassName() {return "DialogTransfer";}

void GetWindowClass(WNDCLASS&);
void pantalla(int borrar);
void lgr();
void pantalla_x(int borrar);
virtual void TMyWindow::WMSize( TMessage& Message );
virtual BOOL CanClose();
void SetPenSize(int NewSize);
virtual void WMLButtonDown(RTMessage Msg)=[WM_FIRST +
WM_LBUTTONDOWN];
virtual void WMRButtonDown(RTMessage Msg)=[WM_FIRST +
WM_RBUTTONDOWN];
virtual void IDMRELOJ(RTMessage Msg) = [CM_FIRST +
IDM_RELOJ];
virtual void IDMCALCULAR(RTMessage Msg) = [CM_FIRST +
IDM_CALCULAR];
virtual void IDMINFORMAC(RTMessage Msg) = [CM_FIRST +
IDM_INFORMAC];
virtual void IDMFRACCION_P(RTMessage Msg) = [CM_FIRST
+ IDM_FRACCION_P];
virtual void IDMTRAN_INV_L(RTMessage Msg) = [CM_FIRST
+ IDM_TRAN_INV_L];
virtual void IDMPOLO_CERO(RTMessage Msg) = [CM_FIRST +
IDM_POLO_CERO];
virtual void IDMEXPANSION(RTMessage Msg) = [CM_FIRST +
IDM_EXPANSION];
virtual void IDMBODE(RTMessage Msg) = [CM_FIRST + 260];
virtual void IDMFuncion_matriz(RTMessage) = [CM_FIRST
```

```
+ 326];
     virtual void IDMTIEMPO(RTMessage Msg) = [CM_FIRST +
IDM_TIEMPO];
     virtual void IDMLUGAR_GEO(RTMessage Msg) = [CM_FIRST +
IDM_LUGAR_GEO];
     virtual void IDMNYQUIST(RTMessage Msg) = [CM_FIRST +
IDM_NYQUIST];
     virtual void IDMNICHOLS(RTMessage Msg) = [CM_FIRST +
IDM_NICHOLS];
     virtual void IDMCOMP_TIEM(RTMessage Msg) = [CM_FIRST +
IDM_COMP_TIEM];
     virtual void IDMABIERTO(RTMessage Msg) = [CM_FIRST +
222];
     virtual void IDMCERRADO(RTMessage Msg) = [CM_FIRST +
221];
     virtual void IDMCOMP_FREC(RTMessage Msg) = [CM_FIRST +
IDM_COMP_FREC];
     virtual void IDMENT_PASO(RTMessage Msg) = [CM_FIRST +
225];

     virtual void far IDMMATRIZ_A(RTMessage Msg) = [CM_FIRST
+ 301];
     virtual void far IDMMATRIZ_B(RTMessage Msg) = [CM_FIRST
+ 302];
     virtual void far IDMMATRIZ_C(RTMessage Msg) = [CM_FIRST
+ 303];
     virtual void far IDMMATRIZ_D(RTMessage Msg) = [CM_FIRST
+ 304];
     virtual void far IDMMATRIZ_U(RTMessage Msg) = [CM_FIRST
+ 305];
     virtual void far IDMMATRIZ_X0(RTMessage Msg) =
[CM_FIRST + 306];
     virtual void far IDMMATRIZ_TIPO(RTMessage Msg) =
[CM_FIRST + 307];
     virtual void far IDMVALOR_PROPIO(RTMessage Msg) =
[CM_FIRST + IDM_VALOR_PR];
```

```
     virtual void far IDME_AT(RTMessage Msg) = [CM_FIRST +
IDM_CAL_E_AT];
     virtual void far IDME_AT_X0(RTMessage Msg) = [CM_FIRST
+ 224];
     virtual void far IDMSIMA(RTMessage Msg) = [CM_FIRST +
223];
     virtual void far IDMCONTROLAB(RTMessage Msg) =
[CM_FIRST + IDM_CONTROLAB];
     virtual void far IDMOBSERVAB(RTMessage Msg) = [CM_FIRST
+ IDM_OBSERVABI];
     virtual void far IDMX_AX_B(RTMessage Msg) = [CM_FIRST
+ IDM_X1_AX_B];
     virtual void far IDMResp_total(RTMessage) = [CM_FIRST
+ 308];

     virtual void far CMHelp(RTMessage Msg)          =
[CM_FIRST + IDM_AYUDA];
     virtual void NewFile(RTMessage Msg) =[CM_FIRST +
CM_MDIFILENEW];
     virtual void OpenFile(RTMessage Msg) =[CM_FIRST +
CM_MDIFILEOPEN];
     virtual void Guardar(RTMessage Msg) =[CM_FIRST +
CM_FILESAVE];
     virtual void Matrices(RTMessage) =[CM_FIRST + 310];
};

// Verificación para terminar el programa
BOOL TMyWindow::CanClose()
{
     return MessageBox(HWindow, "Desea Salir?",
          "Terminar programa", MB_YESNO | MB_ICONQUESTION)
== IDYES;
}

/*   Carga del menu e icono */
void TMyWindow::GetWindowClass(WNDCLASS& wndClass)
```

```
{
        TWindow::GetWindowClass(wndClass);
        wndClass.lpszMenuName ="MIMENU";
        wndClass.hIcon =
LoadIcon(GetApplication()->hInstance, "calc");
}

/*  Redimencionamiento de ventana */
void TMyWindow::WMSize( TMessage& Message)
{
     DragDC = GetDC(HWindow);
     Xmax = Message.LP.Lo;
     Ymax = Message.LP.Hi;
     ReleaseDC(HWindow,DragDC);
}

/* Atiende al boton izquierdo del mouse */
void TMyWindow::WMLButtonDown(RTMessage Msg)
{
     POINT punto;
     HWND hwndnuevo;
     float eje_y, eje_x;
     char s[200];

if (hacer_graf!=0)
{
     DragDC = GetDC(HWindow);
     GetCursorPos(&punto);
     if ( ( hwndnuevo = WindowFromPoint( punto ) ).!= 0 )
          ScreenToClient(hwndnuevo,&punto);
     ReleaseDC(HWindow,DragDC);
     if (x_leer[4]==3)

eje_y=pow(10,log10(x_leer[3])+(float)(.9*Ymax-punto.y)*
          log10(x_leer[1]/x_leer[3])/Ymax/.8);
     else if(x_leer[4]==5)
```

```
            eje_y=x_leer[3]+(float)((Ymax-punto.y)*
                  (x_leer[1]-x_leer[3]))/Ymax;
      else
            eje_y=x_leer[3]+(float)((Ymax*.9-punto.y)*
                  (x_leer[1]-x_leer[3]))/Ymax/.8;

//------caso de bode en db con x4 = 0 mostrar puntos ------
      if (x_leer[4]<=0)

eje_x=pow(10,log10(x_leer[2])+(float)(-.1*Xmax+punto.x)*
                  log10(x_leer[0]/x_leer[2])/Xmax/.8);
      else if (x_leer[4]==5)
            eje_x=x_leer[2]+(float)((punto.x)*
                  (x_leer[0]-x_leer[2]))/Xmax;
      else
            eje_x=x_leer[2]+(float)((-.1*Xmax+punto.x)*
                  (x_leer[0]-x_leer[2]))/Xmax/.8;

      sprintf(s,"punto [%5.3g, %5.3g]                ",eje_x,
eje_y);
      TextOut(DragDC,1,1,s,strlen(s));
      if (x_leer[4]==5)
      {
            complex c[maximo], d[maximo], num, den;
            if (m!=0) {
                  bairstow(m,a,0,0);
                  for (i=0;i<m;i++)
                        c[i]=complex(r[i],im[i]);
            }
            if (n!=0) {
                  bairstow(n,b,0,0);
                  for (i=0;i<n;i++)
                        d[i]=complex(r[i],im[i]);
            }
            if (m>0)  num=ev_num(eje_x,eje_y,c,m);
            else num=complex(1,0);
```

```
        if (n>0)  den=ev_num(eje_x,eje_y,d,n);
        else den=complex(1,0);
        if (sqrt(norm(num))<.0001) strcpy(s,"K = INF ");
        else
        {
             den=-den/num*modulo;
             sprintf(s,"K = [%5.3g + %5.3g j]
",real(den),imag(den));
        }
        TextOut(DragDC,1,20,s,strlen(s));
    }

    if (Msg.WParam & MK_CONTROL)
    {
        sprintf(s,"%4.3g", eje_x);
        TextOut(DragDC,punto.x, .91*Ymax,s,strlen(s));
        MoveTo(DragDC, punto.x, punto.y);
        LineTo(DragDC, punto.x, Ymax*.9);
        sprintf(s,"%4.3g", eje_y);
        if ((x_leer[4]==2) || (x_leer[4]==-1))
        {
             TextOut(DragDC,.9*Xmax,punto.y,s,strlen(s));
             MoveTo(DragDC, Xmax*.91, punto.y);
             LineTo(DragDC, punto.x, punto.y);
        }
        else if (x_leer[4]==-6)
        {
             float temp;
             TextOut(DragDC,1,punto.y,s,strlen(s));
             temp=x_leer[8]+(float)((Ymax*.9-punto.y)*
                  (x_leer[6]-x_leer[8]))/Ymax/.8;
             sprintf(s,"%4.3g", temp);
             TextOut(DragDC,
.91*Xmax,punto.y,s,strlen(s));
             MoveTo(DragDC, Xmax*.1, punto.y);
             LineTo(DragDC, Xmax*.9, punto.y);
```

```
            }
            else
            {
                  TextOut(DragDC, 1,punto.y,s,strlen(s));
                  MoveTo(DragDC, .1*Xmax, punto.y);
                  LineTo(DragDC, punto.x, punto.y);
            }
      }
      ReleaseCapture();
      ReleaseDC(HWindow, DragDC);
}
}
/* Atiende a  la presion del boton derecho del mouse */
void TMyWindow::WMRButtonDown(RTMessage Msg)
{
      char InputText[40];

      sprintf(InputText, "Diagrama");
      if    (    GetApplication()->ExecDialog(new
TInputDialog(this, "Texto a Pantalla",
            "Ingrese Texto a mostrar:", InputText, sizeof
InputText)) == IDOK )
      {
                  DragDC = GetDC(HWindow);

TextOut(DragDC,Msg.LP.Lo,Msg.LP.Hi,InputText,strlen(InputT
ext));
                  ReleaseCapture();
                  ReleaseDC(HWindow,DragDC);
      }
}

/*    Cambia el color de las lineas */
void TMyWindow::SetPenSize(int NewSize)
{
      DeleteObject(ThePen);
```

```
      ThePen = CreatePen(PS_SOLID, 1, NewSize);
      PenSize = NewSize;
}

/*    Crea ventana de ayuda */
void TMyWindow::CMHelp(RTMessage)
{
      PTWindow HelpWindow;

      HelpWindow = new THelpWindow(this);
      HelpWindow->Attr.Style |= WS_POPUPWINDOW | WS_CAPTION;
      HelpWindow->Attr.X = 100;
      HelpWindow->Attr.Y = 100;
      HelpWindow->Attr.W = 300;
      HelpWindow->Attr.H = 300;
      GetApplication()->MakeWindow(HelpWindow);
}

/* Llama al reloj de windows */
void TMyWindow::IDMRELOJ(RTMessage){
      WinExec("clock.exe",SW_SHOWNORMAL);}

/* Llama al editor de texto */
void TMyWindow::Matrices(RTMessage){
      WinExec("mfileapp.exe",SW_SHOWNORMAL);}

/*    Llama a la calculadora */
void TMyWindow::IDMCALCULAR(RTMessage){
      WinExec("calc.exe",SW_SHOWNORMAL);}
//    Llama al mapa de bits
void TMyWindow::IDMINFORMAC(RTMessage){
      GetApplication()->ExecDialog(newTAboutDialog(this));}

/*    Muestra la funcion de trasferencia en fracciones
      parciales */
void TMyWindow::IDMFRACCION_P(RTMessage){
```

```
     if (n>0) pasar(a,b,m,n,1,s1,0,0);
     else mostrarp(m,a,s1);
     MessageBox(HWindow,   s1,"POLINOMIO   EN   FRACCIONES
PARCIALES",MB_OK);}

/*   Muestra la transformada inversa de Laplace de la funcin
     de transferencia ingresada */
void TMyWindow::IDMTRAN_INV_L(RTMessage){
     if (n>0) pasar(a,b,m,n,0,s1,0,0);
     else mostrarpl(m,a,s1);
     MessageBox(HWindow,   s1,"Transformada   Inversa   de
Laplace",MB_OK);}

/*   Muestra la funcion de trasferencia en forma de polos y
     ceros */
void TMyWindow::IDMPOLO_CERO(RTMessage) {
     if (modulo !=1) sprintf(s1,"(%g)",modulo);
     else strcpy(s1,"");
     if(m==0)
     {
          if (modulo==1) sprintf(s2,"%g",a[0]);
          else strcpy(s2,"");
     }
     else
     {
          bairstow(m,a,0,0);
          mostrarf(m,r,im,s2);
     }
     strcat(s1,s2);
     if (n!=0) {
          bairstow(n,b,0,0);
          mostrarf(n,r,im,s2);
          strcat(s1,"\n");
          strcat(s1,s2);
     }
     MessageBox(HWindow, s1, "Polos y Ceros",MB_OK);}
```

```
/*    Muestra la funcion de trasferencia multiplicando todos
      los factores tanto del numerador como del denominador
      */
void TMyWindow::IDMEXPANSION(RTMessage) {
      mostrarp(m,a,s1);
      mostrarp(n,b,s2);
      strcat(s1,"\n");
      strcat(s1,s2);
      MessageBox(HWindow,s1,"Polinomiodesarollado",MB_OK);}

/*    Procedimiento para obtener la respuesta forzada */
void TMyWindow::IDMX_AX_B(RTMessage){
      leer_matriz();
      sima(a1,f,t,n1);
      t[n1+1]=0;t[n1+2]=0;
      mult_bxu(n1,m1,b1,u,bxu);
      mult_exb(n1,f,bxu, forzada);
      mostrar_matriz(n1,forzada,tipo_ent,t,s1);
      s_cadena(s1, "forzada.mtz");
      if (strlen(s1)>400)
            strcpy(s1,"Respueta Forzada grabada en Archivo
forzada.mtz");
      MessageBox(HWindow, s1,"Resp. Forzada",MB_OK);}

/*    Procedimiento para obtener la respuesta total */
void TMyWindow::IDMResp_total(RTMessage){
      leer_matriz();
      sima(a1,f,t,n1);
      t[n1+1]=0;
      t[n1+2]=0;
      mult_exb(n1,f,x0, natural);
      mult_bxu(n1,m1,b1,u,bxu);
      mult_exb(n1,f,bxu, forzada);
      resp_total(n1,natural,forzada,tipo_ent,t,s1);
      s_cadena(s1, "total.mtz");
      if (strlen(s1)>400)
```

```
          strcpy(s1,"Respuesta total grabada en Archivo
total.mtz");
     MessageBox(HWindow, s1,"Solución Total",MB_OK);}
/*   Procedimiento para obtener la matriz e^At en forma
     literal */
void TMyWindow::IDME_AT(RTMessage){
     leer_matriz();
     sima(a1,f,t,n1);
     mostrar_matriz(n1, f, 0, t, n1, s1);
     s_cadena(s1, "eat.mtz");
     if (strlen(s1)>400)
          strcpy(s1,"Función  e^At  grabada  en  Archivo
eAt.mtz");
     MessageBox(HWindow, s1,"e^At",MB_OK);}

/*   Procedimiento para obtener la respuesta natural */
void TMyWindow::IDME_AT_X0(RTMessage){
     leer_matriz();
     sima(a1,f,t,n1);
     mult_exb(n1, f, x0, natural);
     mostrar_matriz(n1, natural, 0, t, s1);
     s_cadena(s1, "natural.mtz");
     if (strlen(s1)>400)
          strcpy(s1,"Respuesta natural grabada en Archivo
natural.mtz");
     MessageBox(HWindow, s1,"e^At * X0",MB_OK);}

/* Procedimientos para mostrar matrices y vectores */
void TMyWindow::IDMMATRIZ_A(RTMessage){
     leer_matriz();
     if (n1>0) mostrar_matriz(n1,n1,a1,s1);
     else strcpy(s1,"Matriz A no definida");
     MessageBox(HWindow, a1,"Matriz A",MB_OK);}
void TMyWindow::IDMMATRIZ_B(RTMessage){
     leer_matriz();
     if (m1>0) mostrar_matriz(n1,m1,b1,s1);
```

```
     else strcpy(s1,"Matriz B no definida");
     MessageBox(HWindow, s1,"Matriz B",MB_OK);}
void TMyWindow::IDMMATRIZ_C(RTMessage){
     leer_matriz();
     if (r1>0) mostrar_matriz(r1,n1,c1,s1);
     else strcpy(s1,"Matriz C no definida");
     MessageBox(HWindow, s1,"Matriz C",MB_OK);}
void TMyWindow::IDMMATRIZ_D(RTMessage){
     leer_matriz();
     if (r1>0) mostrar_matriz(m1,r1,d1,s1);
     else strcpy(s1,"Matriz D no definida");
     MessageBox(HWindow, s1,"Matriz D",MB_OK);}
void TMyWindow::IDMMATRIZ_U(RTMessage){
     leer_matriz();
     strcpy(s1,"Entrada Transpuesta:\n");
     for (i=0;i<m1;i++){
          sprintf(s2,"%g   ", u[i]);
          strcat(s1,s2);
     }
     MessageBox(HWindow, s1,"Entrada u",MB_OK);}
void TMyWindow::IDMMATRIZ_X0(RTMessage){
     leer_matriz();
     strcpy(s1,"Valores Iniciales Transpuestos:\n");
     for (i=0;i<n1;i++){
          sprintf(s2,"%g   ", x0[i]);
          strcat(s1,s2);
     }
     MessageBox(HWindow, s1,"Valores iniciales",MB_OK);}

void TMyWindow::IDMMATRIZ_TIPO(RTMessage){
     leer_matriz();
     if (falla_ent==1)
          strcpy(s1,"Entrada desconocida\nSe usarß entrada
Paso");
     else if (tipo_ent==0) strcpy(s1,"Entrada Impulso");
     else if (tipo_ent==1) strcpy(s1,"Entrada Paso");
```

```
    else if (tipo_ent==2) strcpy(s1,"Entrada Rampa");
    MessageBox(HWindow, s1,"Tipo de entrada",MB_OK);}
void TMyWindow::IDMENT_PASO(RTMessage){
    b[n+1]=0;
    n=n+1;
    mostrarp(m,a,s1);
    strcat(s1,"\n");
    mostrarp(n,b,s2);
    strcat(s1,s2);
    MessageBox(HWindow, s1,"G = G * L /s",MB_OK);}

/*  Procedimiento para convertir una función de
    transferencia en su equivalente en variables de estado
    */
void TMyWindow::IDMFuncion_matriz(RTMessage){
    int j;

    sprintf(s1,"%d 1  1\n\n",n);
    //---matriz a
    for (i=0;i<n-1;i++)
    {
        for (j=0;j<n;j++)
        {
            if (j==i+1) strcpy(s2,"1 ");
            else strcpy(s2,"0 ");
            strcat(s1,s2);
        }
        strcat(s1,"\n");
    }
    for (j=0;j<n;j++)
    {
        sprintf(s2,"%g ", -b[n-i]/b[0]);
        strcat(s1,s2);
    }
    strcat(s1,"\n\n");
```

```
      //---matriz b
      for (j=0; j<n-1; j++)
      {
            strcpy(s2,"0\n");
            strcat(s1,s2);
      }
      strcat(s1,"1\n\n");
      //---matriz c
      i=m;
      for (j=0;j<n;j++){
            if (i>=0) sprintf(s2,"%g ",a[i--]/b[0]);
            else sprintf(s2,"0 ");
            strcat(s1,s2);
      }
      //matriz d
      strcat(s1,"\n\n0\n\n1\n\n");
      for (j=0;j<n;j++) strcat(s1,"0 ");
      strcat(s1,"\n\n1");
      s_cadena(s1,"matriz");
}

/*    Relee la matriz original */
void TMyWindow::IDMABIERTO(RTMessage){
      *funcion="funcion.ft";
      leer_funcion();
      mostrarp(m,a,s1);
      strcat(s1,"\n");
      mostrarp(n,b,s2);
      strcat(s1,s2);
      hecho=0;
      MessageBox(HWindow, s1,"Funcion Original",MB_OK);}

/*    Convierte la funcion g en g/(1+gh) */
void TMyWindow::IDMCERRADO(RTMessage){
      float c[10];
      int l;
```

```
      suma(m,a,n,b,l,c);
      mostrarp(m,a,s1);
      strcat(s1,"\n");
      mostrarp(l,c,s2);
      strcat(s1,s2);
      for (i=0;i<=l;i++) b[i]=c[i];
      n=l;
      modulo=a[0]/b[0];
      hecho=0;
      MessageBox(HWindow, s1,"G(s)=1/[1+G(s)H(s)]",MB_OK);}

/*    Halla la matriz (sI - A)^-1 */
void TMyWindow::IDMSIMA(RTMessage){
      leer_matriz();
      sima(a1,f,t,n1);
      mostrar_matriz(n1, f, s1);
      mostrarp(n1,t,s2);
      strcat(s1,"\n");
      for (i=0;i<strlen(s2);i++) strcat(s1,"_");
      strcat(s1,"\n");
      strcat(s1,s2);
      s_cadena(s1, "sima.mtz");
      if (strlen(s1)>400)
           strcpy(s1,"(sI       A)^1   grabado   en   Archivo
sima.mtz");
      MessageBox(HWindow, s1,"(sI - A)^-1",MB_OK);}

/* Halla los valores propios */
void TMyWindow::IDMVALOR_PROPIO(RTMessage){
      leer_matriz();
      sima(a1,f,t,n1);
      strcpy(s1,"Polinomio Caracteristico:\n");
      mostrarp(n1,t,s2);
      strcat(s1,s2);
      bairstow(n1,t,0,0);
      mostrarf(n1,r,im,s2);
```

```
        strcat(s1, "\n\nValores Propios:\n");
        strcat(s1,s2);
        MessageBox(HWindow, s1,"Valores Propios",MB_OK);}

/*    Funcion para determinar la controlabilidad */
void TMyWindow::IDMCONTROLAR(RTMessage){
        float determinante;

        leer_matriz();
        if (m1>0)
        {
                strcpy(s1, "| B | AB | ... | A^nB |\n\n");
                mult_anxb(n1,m1,n1,b1,axb,0,s1);
                determinante=sist_ecuacion(n1, axb);
                sprintf(s2,"\ndeterminante = %g", determinante);
                strcat(s1,s2);
                if (fabs(determinante)<.001) strcat(s1,"\n\nNo es
Controlable");
                else strcat(s1, "\n\nControlable");
        }
        else strcpy(s1,"No existe Matriz B");
        MessageBox(HWindow, s1,"\Controlabilidad",MB_OK);}

/*    Funcion para determinar la controlabilidad */
void TMyWindow::IDMOBSERVAR(RTMessage){
        float determinante;

        leer_matriz();
        if (r1==0) strcpy(s1,"No existe Matriz C");
        else
        {
                strcpy(s1, "| C* | A*C* | ... | (A*)^nC* |\n\n");
                mult_anxb(n1,m1,n1,c1,axb,1,s1);
                determinante=sist_ecuacion(n1, axb);
                sprintf(s2,"\ndeterminante = %g", determinante);
                strcat(s1,s2);
```

```
            if (fabs(determinante)<.001) strcat(s1,"\n\nNo es
Observable");
            else strcat(s1, "\n\nObservable");
      }
      MessageBox(HWindow, s1,"Observabilidad",MB_OK);}

/*    Procedimiento para hallar el lugar geometrico de las
      raices */
void TMyWindow::IDMLUGAR_GEO(RTMessage) {
      root(a,b,m,n,s1);
      MessageBox(HWindow, s1,"Lugar Geometrico",MB_OK);
      hacer_graf=1;
      lgr();}

/* Obtiene el diagrama de nyquist */
void TMyWindow::IDMNYQUIST(RTMessage)
{
      if (hecho==0) bode (a,b,m,n,-1,-1);
      hecho=1;

      *pasa="sal_imag";
      pantalla(1);
      MessageBox(HWindow, "Real Vs Imaginaria","Diagrama de
Nyquist",MB_OK);
}

/* Obtiene el diagrama de nichols */
void TMyWindow::IDMNICHOLS(RTMessage)
{
      if (hecho==0) bode (a,b,m,n,-1,-1);
      hecho=1;

      *pasa="sal_mod";
      pantalla(1);
      MessageBox(HWindow, "Modulo en funcion de angulo",
"Nichols", MB_OK);
```

```
}

/*    Ingreso del compensador */
void TMyWindow::IDMCOMP_TIEM(RTMessage) {
     if  (  GetApplication()->ExecDialog(new  TInputDialog
(this , "Ingreso de compensador",
          "Ingrese Compensador en s:", ingreso_com, sizeof
ingreso_com)) == IDOK )
     {
          strcpy(s2,ingreso);
          strcat(s2,"\n");
          s_cadena(s2, "funcion.ini");
          WinExec("polinom2.exe  funcion.ini  funcion.cmp",
SW_SHOWNORMAL);
          *compen="funcion.cmp";
          leer_compesador();
          mostrarp(mc,ac,s1);
          mostrarp(nc,bc,s2);
          strcat(s1,"\n");
          strcat(s1,s2);
          existe_compensador=1;
          MessageBox(HWindow,   s1,"Compensador:",MB_OK   |
MB_ICONEXCLAMATION);
     }
     else
          MessageBox(HWindow,   "No   ha   ingresado   el
compensador","Operación    Cancelada",MB_OK    |
MB_ICONINFORMATION);

     ReleaseCapture();
     ReleaseDC(HWindow,DragDC);
}

/*    Muestra el compensador */
void TMyWindow::IDMCOMP_FREC(RTMessage) {
     if (existe_compensador==1){
```

```
            mostrarp(mc,ac,s1);
            mostrarp(nc,bc,s2);
            strcat(s1,"\n");
            strcat(s1,s2);}
      else strcpy(s1," 1 ");
      MessageBox(HWindow, s1,"Compensador es:",MB_OK);}

/*    Procedimiento para graficar */
void TMyWindow::pantalla_x(int borrar)
{
      float X1, X2, Y1, Y2;
      float maxim, linea_cero;
      char s[200];
      int alto=20;

      DragDC = GetDC(HWindow);
      SelectObject(DragDC, ThePen);
      if (borrar==1)
            InvalidateRect(HWindow, NULL, TRUE);

      UpdateWindow(HWindow);

      leer(pasa, x_leer, i);
//    DragDC = GetDC(HWindow);
      maxim=125;
      for(i=5;i<104;i++)
      {
            X1=(float)Xmax*(i-5)/maxim+.1*Xmax;
            X2=(float)Xmax*(i-4)/maxim+.1*Xmax;
            Y1=(float)x_leer[i]*Ymax/maxim+.1*Ymax;
            Y2=(float)x_leer[i+1]*Ymax/maxim+.1*Ymax;
            MoveTo(DragDC, X1, Ymax-Y1);
            LineTo(DragDC, X2, Ymax-Y2);
      }
//-------mostrar ejes--------------
```

```
     //--eje x-----
     sprintf(s,"%3.5g", x_leer[0]);
     TextOut(DragDC,   Xmax*.9-2*strlen(s),   .9*Ymax,   s,
strlen(s));
     sprintf(s,"%3.5g", x_leer[2]);
     TextOut(DragDC, .1*Xmax, .9*Ymax, s, strlen(s));
     MoveTo(DragDC, .1*Xmax, .9*Ymax);
     LineTo(DragDC, .9*Xmax, .9*Ymax);

     //--------eje y---------
     if ((x_leer[4]==0)||(x_leer[4]==4))//---caso de modulo
     {
          //-----linea para Mf-----
          if ((x_leer[1]*x_leer[3])<0)
          {

linea_cero=.9*Ymax+x_leer[3]*Ymax*.8/(x_leer[1]-x_leer[3]);
               MoveTo(DragDC, Xmax*.1, linea_cero);
               LineTo(DragDC, Xmax*.9, linea_cero);
               strcpy(s,"   0");
               TextOut(DragDC, 1, linea_cero, s, strlen(s));
          }
          sprintf(s,"%3.4g", x_leer[3]);
          TextOut(DragDC, 1, .9*Ymax-alto, s, strlen(s));
          sprintf(s,"%3.4g", x_leer[1]);
          TextOut(DragDC, 1, .1*Ymax, s, strlen(s));
          MoveTo(DragDC, .1*Xmax, .9*Ymax);
          LineTo(DragDC, .1*Xmax, .1*Ymax);
     }
     if (x_leer[4]==-1)//---caso de angulo
     {
          //-----linea para MG-----
          if (((x_leer[1]+180.)*(180.+x_leer[3]))<0)
          {
               linea_cero=.9*Ymax+(180.+x_leer[3])*Ymax*.8/
                    (x_leer[1]-x_leer[3]);
```

```
                MoveTo(DragDC, Xmax*.1, linea_cero);
                LineTo(DragDC, Xmax*.9, linea_cero);
                strcpy(s,"-180");
                TextOut(DragDC, Xmax*.9, linea_cero, s,
strlen(s));
          }
          sprintf(s,"%3.4g", x_leer[3]);
          TextOut(DragDC, Xmax*.9, .9*Ymax-alto, s,
strlen(s));
          sprintf(s,"%3.4g", x_leer[1]);
          TextOut(DragDC, Xmax*.9, .1*Ymax, s, strlen(s));
          MoveTo(DragDC, .9*Xmax, .9*Ymax);
          LineTo(DragDC, .9*Xmax, .1*Ymax);
     }

     ReleaseCapture();
     ReleaseDC(HWindow,DragDC);
     hacer_graf=1;
}
/*   Crea una nueva funcion de transferencia */
void TMyWindow::NewFile(RTMessage)
{
     if   (   GetApplication()->ExecDialog(new
TInputDialog(this, "Ingreso de función",
          "Función en s:", ingreso, sizeof ingreso)) == IDOK
)
     {
          strcpy(s2,ingreso);
          strcat(s2,"\n");
          s_cadena(s2, "funcion.ini");
          WinExec("polinom2.exe   funcion.ini
funcion.tt",SW_SHOWNORMAL);
          leer_funcion();
          mostrarp(m,a,s1);
          mostrarp(n,b,s2);
          strcat(s1,"\n");
```

```
            strcat(s1,s2);
            hecho=0;
            existe_funcion=1;
            MessageBox(HWindow, s1,"Funci≤n Ingresada",MB_OK
| MB_ICONEXCLAMATION);
      }
      else
            MessageBox(HWindow,    "No    ha    ingresado    la
funci≤n","Operación Cancelada",MB_OK | MB_ICONINFORMATION);

      ReleaseCapture();
      ReleaseDC(HWindow,DragDC);
}

/*    Abre un archivo ya existente */
void TMyWindow::OpenFile(RTMessage)
{
      char FileName[MAXPATH];
      char ChildName[14];

      if ( GetApplication()->ExecDialog(new TFileDialog(this,
SD_FILEOPEN,

_fstrcpy(FileName, "*.ft"))) == IDOK )
      {
            GetApplication()->MakeWindow(new TFileWindow(this,
ChildName, FileName));
            *funcion=FileName;
            f_cadena(s2, *funcion);
            s_cadena(s2, "funcion.ft");
            leer_funcion();
            mostrarp(m,a,s1);
            mostrarp(n,b,s2);
            strcat(s1,"\n");
            strcat(s1,s2);
            hecho=0;
```

```
          MessageBox(HWindow, s1,"Funcisn Ingresada",MB_OK
| MB_ICONQUESTION);
          *funcion="funcion.ft";
     }
     else
          MessageBox(HWindow,    "No    hay    cambio    en
funcisn","Accisn Cancelada",MB_OK | MB_ICONINFORMATION);
     ReleaseCapture();
     ReleaseDC(HWindow,DragDC);
}
void TMyWindow::Guardar(RTMessage)
{
     char FileName[MAXPATH];
     int extencion=0;

     if (existe_funcion)
     {
          if    (    GetApplication()->ExecDialog(new
TFileDialog(this, SD_FILESAVE,

_fstrcpy(FileName, "*.ft"))) == IDOK )
          {
               for (i=0;i< strlen (FileName);i++)
                    if (FileName[i]=='.') extencion=1;
               if (extencion==0) strcat(FileName,".ft");

               *funcion=FileName;
               f_cadena(s1, "funcion.ft");
               s_cadena(s1, *funcion);
               strcpy(s1,"Nombre de Archivo:\n");
               strcat(s1,FileName);
               MessageBox(HWindow,    s1,"Funcisn
Grabada",MB_OK | MB_ICONEXCLAMATION);
               *funcion="funcion.ft";
          }
          else
```

```
            {
                strcpy(s1,"No ha grabado la funci≤n:\n");
                MessageBox(HWindow,  s1,"Acci≤n
Cancelada",MB_OK | MB_ICONINFORMATION);
            }
        }
        else
            MessageBox(HWindow,  "No  existe  funci≤n  para
grabar","Acci≤n Cancelada",MB_OK | MB_ICONEXCLAMATION);

        ReleaseCapture();
        ReleaseDC(HWindow,DragDC);
}

/*    Obtiene el lugar geometrico */
void TMyWindow::lgr()
{
        float max_x, max_y, min_x, min_y, h, a[100];
        int m, n, l, k, j,i,ruptura, minimo, j1, hecho, cuenta;
        void leer1(char *arch[],float x_leer[]);
        float h1, x, y, inicial, delta, angulopolo, angulocero,
angulo, x1, y1;
        float asinr[10], asini[10], partida[10], xc[10]={0};
        float xp[10], yc[10]={0}, yp[10];
        float totalx,totaly;
        char *arch[1];
        const float grado = .1;
        const int puntos=800;
        const int radio=2;

        DragDC = GetDC(HWindow);
        SelectObject(DragDC, ThePen);
        InvalidateRect(HWindow, NULL, TRUE);
        UpdateWindow(HWindow);

        *arch="datps.lgr";
```

```
      leer1(arch,a);
      i=0;
//------lectura de valores--------
      n=a[i++];m=a[i++];
      for(j=0;j<n;j++){
            xc[j]=a[i++];
            yc[j]=a[i++];
      }
      for(j=0;j<m;j++){
            xp[j]=a[i++];
            yp[j]=a[i++];
      }

      ruptura=a[i++];
      for(j=0; j<ruptura; j++){
            asinr[j]=a[i++];
            asini[j]=a[i++];
            partida[j]=a[i++];
      }
//-------- calculo de limites ----------
      max_x=maximovalor(xp, m, xc, n, asinr, ruptura);
      min_x=minimovalor(xp, m, xc, n, asinr, ruptura);
      max_y=maximovalor(yp, m, yc, n, asini, 0);
      totalx = max_x - min_x;
      if (fabs(totalx)<.001) totalx = 2;
      max_x = max_x + totalx;
      min_x = min_x - totalx;
      max_y = 2 * max_y;
      if (max_y == 0) max_y = totalx;
      min_y = -max_y;
      totalx=Xmax/(max_x-min_x);
      totaly=Ymax/(max_y-min_y);
      h=radio*(max_x-min_x+max_y)/puntos;
      x_leer[3]=-max_y;
      x_leer[1]=max_y;
      x_leer[2]=min_x;
```

```
      x_leer[0]=max_x;
      x_leer[4]=5;

//-----Lgr en eje real--------
      cuenta=0;
      for (i=0;i<m; i++)
            if (fabs(yp[i])<0.0001) a[cuenta++]=xp[i];
      for (i=0;i<n; i++)
            if (fabs(yc[i])<.00001) a[cuenta++]=xc[i];
      float temp,max;
      while (cuenta>0)
      {
            max=maximovalor(a,cuenta,xc,0, asinr, 0);
            if (cuenta==1)
            {
                  temp=min_x;
                  cuenta=0;
            }
            else
            {
                  for (i=0;i<cuenta;i++)
                        if (a[i]==max)
                        {
                              cuenta--;
                              a[i]=a[cuenta];
                              i=cuenta;
                        }
                  temp=max;
                  max=maximovalor(a,cuenta,xc,0, asinr,0);
                  for (i=0;i<cuenta;i++)
                        if (a[i]==max)
                        {
                              cuenta--;
                              a[i]=a[cuenta];
                              i=cuenta;
                        }
```

```
                }
            MoveTo(DragDC,(max-min_x)*totalx,Ymax/2+1);
            LineTo(DragDC,(temp-min_x)*totalx,Ymax/2+1);
            MoveTo(DragDC,(max-min_x)*totalx,Ymax/2-1);
            LineTo(DragDC,(temp-min_x)*totalx,Ymax/2-1);
      }
//--------trazado de ejes, polos, ceros --------
      MoveTo (DragDC,Xmax,Ymax/2);
      LineTo (DragDC,0,Ymax/2);
      if (min_x*max_x<0)
      {
            MoveTo (DragDC, -min_x*totalx,0);
            LineTo (DragDC, -min_x*totalx,Ymax);
      }
      max=Xmax/100;
      for(j=0;j<m;j++){
            MoveTo(DragDC,(xp[j] -min_x)*totalx+max, (yp[j]
-min_y)*totaly+max);
            LineTo(DragDC,(xp[j] -min_x)*totalx-max, (yp[j]
-min_y)*totaly-max);
            MoveTo(DragDC,(xp[j] -min_x)*totalx+max, (yp[j]
-min_y)*totaly-max);
            LineTo(DragDC,(xp[j] - min_x)*totalx-max, (yp[j]
-min_y)*totaly+max);
      }
      for(j=0;j<n;j++)
            Ellipse (DragDC, (xc[j] -min_x)*totalx+Xmax/80,
(yc[j] -min_y)*totaly-Ymax/80,
                  (xc[j]    -min_x)*totalx-Xmax/80,    (yc[j]
-min_y)*totaly+Ymax/80);

//----programa principal----
      for(l = 0;l< ruptura;l++)
      {
            x = asinr[l];
            y = asini[l];
```

```
            inicial = partida[1];
            k = 0;
            cuenta=0;
            while (k < puntos)
            {
                  delta = 1000;
                  cuenta++;
                  hecho=0;
                  for (j = 0;j <= 2;j++)
                  {
                        angulo = inicial - (1 - j);
                        y1 = y + h * sin(angulo * M_PI / 180.);
                        x1 = x + h * cos(angulo * M_PI / 180.);
//                      cout << "\ninicial = " << angulo << " x=
" << x1 << " y= " << y1 ;
                        x1=x1;
                        angulopolo=angulos(m, x1, y1, xp, yp);
                        angulocero=angulos(n, x1, y1, xc, yc);
                        angulopolo = angulopolo - angulocero;
                        if(modulo<0)angulopolo=angulopolo+180;
                        i                                    f
((angulopolo>10000)||(angulopolo<-10000))
                              angulopolo=0;
                        while (angulopolo > 360) angulopolo  =
angulopolo - 360;
                        while (angulopolo < 0) angulopolo =
angulopolo + 360;
//                      cout << " equivalente = " << angulopolo
;
                        if    ((angulopolo    <    (180    +
grado))&&(angulopolo > (180 - grado)))
                        {
                              hecho=1;
                              if (k > 10)
                              {
                                    h1 = h * 4;
```

```
                    if (fabs(y) < h1)
                    {
                        k = puntos;
//                      cout << "en eje real";
                        MoveTo(DragDC,0, 0);
                        LineTo(DragDC,x, y);
                        MoveTo(DragDC,0, 0);
                        LineTo(DragDC,x, -y);
                    }
                }
                else h1 = 0;
                for (j1 = 0;j<n ;j++)
                    if ((fabs(xc[j1]-x) < h1) &&
(fabs(yc[j1]-y) < h1))
                    {
                        k = puntos;
//                      cout << "en cero";
                        MoveTo (DragDC, x1, y1);
                        LineTo (DragDC, xc[j1],
yc[j1]);
                        MoveTo (DragDC, x1, -y1);
                        LineTo  (DragDC,xc[j1],
-yc[j1]);
                    }
                for (j1 = 0;j<m ; j++)
                    if  ((fabs(xp[j1]-x)  <  h1)
&&(fabs(yp[j1] - y) < h1))
                    {
//                      cout << "en polo";
                        k = puntos;
                        MoveTo (DragDC,x1, y1);
                        LineTo  (DragDC,xp[j1],
yp[j1]);
                        MoveTo (DragDC, x1, -y1);
                        LineTo (DragDC, xp[j1],
-yp[j1]);
```

```
                    }
                }

                if (fabs(180-angulopolo) < delta)
                {
                    minimo = j;
                    delta = fabs(180-angulopolo);
                }
            }
//hacer en caso de no ser angulo 180 grados
            if (cuenta > 30)
                k = puntos;

            if (hecho==0)
            {
                if (minimo == 0) inicial = inicial -
2.4;
                if (minimo == 2) inicial = inicial +
2.6;
                if (minimo == 1) inicial = inicial +.1;
            }
            else
            {
                if (minimo == 0) inicial = inicial - 1;
                if (minimo == 2)  inicial = inicial + 1;
                k++;
                cuenta = 0;
                angulo = inicial;
                y1 = y + h * sin(angulo * M_PI / 180.);
                x1 = x + h * cos(angulo * M_PI / 180.);
                MoveTo    (DragDC,
(x-min_x)*totalx,(y-min_y)*totaly);
                LineTo    (DragDC,
(x1-min_x)*totalx,(y1-min_y)*totaly);
                MoveTo    (DragDC,
(x-min_x)*totalx,(-y-min_y)*totaly);
```

```
                    L i n e T o     ( D r a g D C ,
(x1-min_x)*totalx,(-y1-min_y)*totaly);
                    x = x1; y = y1;
//                  'cout << x << "," << y << "\n";
                    if ((x >= max_x) || (x <= min_x)) k =
puntos;
                    if ((y >= max_y) || (y <= min_y)) k =
puntos;
               }
          }
     }
}

/*   Graficación de diagramas */
void TMyWindow::pantalla(int borrar)
{
     float valores[120], X1, X2, Y1, Y2;
     float maxim, linea_cero, linea_uno;
     char s[200];
     int alto=20;

     DragDC = GetDC(HWindow);
     SelectObject(DragDC, ThePen);
     if (borrar==1)
          InvalidateRect(HWindow, NULL, TRUE);
     UpdateWindow(HWindow);

     leer(pasa,valores, i);
     if (!strcmp(*pasa,"sal_imag"))     *pasa="sal_real";
     if (!strcmp(*pasa,"sal_mod")) *pasa="sal_ang";
     leer(pasa,x_leer, i);
     maxim=125;

     for(i=5;i<104;i++){
          X1=(float)x_leer[i]*Xmax/maxim+.1*Xmax;
          X2=(float)x_leer[i+1]*Xmax/maxim+.1*Xmax;
```

```
            //grafica en y de .1 a .9 de Ymax
            Y1=(float)valores[i]*Ymax/maxim+.1*Ymax;
            Y2=(float)valores[i+1]*Ymax/maxim+.1*Ymax;
            MoveTo(DragDC, X1, Ymax-Y1);
            LineTo(DragDC, X2, Ymax-Y2);
      }

//-------mostrar ejes--------------
      if (x_leer[4]==1)//---caso de Nyquist
      {
            //-----linea para estabilidad-----
            if ((x_leer[1]*x_leer[3])<0)//---- linea cero
            {

linea_cero=.9*Ymax+x_leer[3]*Ymax*.8/(x_leer[1]-x_leer[3]);
                  MoveTo(DragDC, Xmax*.1, linea_cero);
                  LineTo(DragDC, Xmax*.9, linea_cero);
                  strcpy(s," 0");
                  TextOut(DragDC,   .9*Xmax,   linea_cero,   s,
strlen(s));
                  if (x_leer[0]>-1)//---------- mostrar -1
                        if (x_leer[2]<-1)
                        {
                              linea_uno=Xmax*.1+

Xmax*(1+x_leer[2])/(x_leer[2]-x_leer[0])*.8;
                              MoveTo(DragDC,    linea_uno,
linea_cero+3);
                              LineTo(DragDC,    linea_uno,
linea_cero-3);
                              strcpy(s,"-1");
                              TextOut(DragDC,linea_uno,.9*Ymax,
s, strlen(s));
                        }
            }
            sprintf(s,"%3.4g", x_leer[3]);
```

```
            TextOut(DragDC, 1, .9*Ymax-alto, s, strlen(s));
            sprintf(s,"%3.4g", x_leer[1]);
            TextOut(DragDC, 1, .1*Ymax, s, strlen(s));
            MoveTo(DragDC, .1*Xmax, .9*Ymax);
            LineTo(DragDC, .1*Xmax, .1*Ymax);
      }

      if (x_leer[4]==-1)//---caso de Nichols
      {
            x_leer[0]=x_leer[1];
            x_leer[1]=valores[1];
            x_leer[2]=x_leer[3];
            x_leer[3]=valores[3];
            x_leer[4]=2;

            //-------linea para MF
            if ((x_leer[1]*x_leer[3])<0)//---- linea cero
            {

linea_cero=.9*Ymax+x_leer[3]*Ymax*.8/(x_leer[1]-x_leer[3]);
                  MoveTo(DragDC, Xmax*.1, linea_cero);
                  LineTo(DragDC, Xmax*.9, linea_cero);
                  strcpy(s," 0");
                  TextOut(DragDC,   .9*Xmax,   linea_cero,   s,
strlen(s));
                  if (x_leer[0]>-1)//---------- mostrar -1
                        if (x_leer[2]<-1)
                        {
                              linea_uno=Xmax*.1+

Xmax*(1+x_leer[2])/(x_leer[2]-x_leer[0])*.8;
                              MoveTo(DragDC,    linea_uno,
linea_cero+3);
                              LineTo(DragDC,    linea_uno,
linea_cero-3);
                              strcpy(s,"-1");
```

```
                        TextOut(DragDC,linea_uno,.9*Ymax,
s, strlen(s));
                    }
            }
            //-----linea para MG-----
            if (((x_leer[0]+180.)*(180.+x_leer[2]))<0)
            {
                linea_cero=.9*Xmax+(180.+x_leer[2])*Xmax*.8/
                    (x_leer[0]-x_leer[2]);
                MoveTo(DragDC, linea_cero, Ymax*.1);
                LineTo(DragDC, linea_cero, Ymax*.9);
                strcpy(s,"-180");
                TextOut(DragDC,  linea_cero,  Ymax*.9,  s,
strlen(s));
            }
            sprintf(s,"%3.4g", x_leer[3]);
            TextOut(DragDC,   Xmax*.9,   .9*Ymax-alto,   s,
strlen(s));
            sprintf(s,"%3.4g", x_leer[1]);
            TextOut(DragDC, Xmax*.9, .1*Ymax, s, strlen(s));
            MoveTo(DragDC, .9*Xmax, .9*Ymax);
            LineTo(DragDC, .9*Xmax, .1*Ymax);
        }

        //--eje x-----
        sprintf(s,"%3.4g", x_leer[0]);
        TextOut(DragDC,  Xmax*.9-2*strlen(s),  .9*Ymax,  s,
strlen(s));
        sprintf(s,"%3.4g", x_leer[2]);
        TextOut(DragDC, .1*Xmax, .9*Ymax, s, strlen(s));
        MoveTo(DragDC, .1*Xmax, .9*Ymax);
        LineTo(DragDC, .9*Xmax, .9*Ymax);

        ReleaseCapture();
        ReleaseDC(HWindow,DragDC);
        hacer_graf=1;
```

```
}

/*    realiza los calculos correspondientes de acuerdo a las
      opciones ingresadas */
void TMyWindow::IDMTIEMPO(TMessage&)
{
      int resultado, borrar;

      resultado=GetApplication()->ExecDialog(
            new TBitmapAttributesDialog(this) );
      if(resultado==IDOK)
      {
            float tiempo_maximo;
            float tiempo_minimo;

           borrar=1;
           if (existe_compensador==1)
                if (Data.limpiar==1){
                     float c[10];
                     multiplica(m,mc,a,ac,c);
                     m=m+mc;
                     for (i=0;i<=m;i++) a[i]=c[i];
                     multiplica(n,nc,b,bc,c);
                     n=n+nc;
                     for (i=0;i<=n;i++) b[i]=c[i];
                     mostrarp(m,a,s1);
                     mostrarp(n,b,s2);
                     strcat(s1,"\n");
                     strcat(s1,s2);
                     MessageBox(HWindow,    s1,"Sistema
Compensado:",MB_OK | MB_ICONEXCLAMATION);
                }

           if(Data.cerrado==1)
           {
```

```
        float c[10];
        int l;

        suma(m,a,n,b,l,c);
        for (i=0;i<=l;i++) b[i]=c[i];
        n=l;
        modulo=a[0]/b[0];
    }
    if(Data.paso==1)
    {
        b[n+1]=0;
        n=n+1;
    }
    if(Data.rampa==1)
    {
        b[n+1]=0;
        b[n+2]=0;
        n=n+2;
    }
    tiempo_maximo=atof(Data.t_maximo);
    tiempo_minimo=atof(Data.t_minimo);
    if (tiempo_maximo<=tiempo_minimo)
        Data.automatico=0;

    if(Data.automatico==0)
    {
        bairstow(n,b,0,0);
        limites(r,n,tiempo_maximo);
        tiempo_minimo=0;
    }
    pasar(a,b,m,n,2,s1,tiempo_minimo,tiempo_maximo);
    *pasa="tiempo";
    pantalla_x(borrar);
    *funcion="funcion.ft";
    leer_funcion();
}
```

```
}

/*   Opciones de respuesta en frecuencia */
void TMyWindow::IDMBODE(TMessage&)
{
     int resultado, borrar;
     float fre_max;
     float fre_min;

     resultado=GetApplication()->ExecDialog(
          new BodeDialog(this) );
     if(resultado==IDOK)
     {
          borrar=1;
          if (existe_compensador==1)
          {
               if (Dato.limpiar==1){
                    float c[10];

                    multiplica(m,mc,a,ac,c);
                    m=m+mc;
                    for (i=0;i<=m;i++) a[i]=c[i];
                    multiplica(n,nc,b,bc,c);
                    n=n+nc;
                    for (i=0;i<=n;i++) b[i]=c[i];
               }
          }
          if(Dato.b_cerrado==1)
          {
               float c[10];
               int l;

               suma(m,a,n,b,l,c);
               for (i=0;i<=l;i++) b[i]=c[i];
               n=l;
               modulo=a[0]/b[0];
```

```
}

fre_max=atof(Dato.f_maxima);
fre_min=atof(Dato.f_minima);
if ((fre_max<=fre_min)||(fre_min<=0))
     Dato.bodeauto=0;
if(Dato.bodeauto==0)
{
     fre_max=-1;
     fre_min=-1;
}

if(Dato.angulo==1)
{
     bode (a,b,m,n,fre_min,fre_max);
     hecho=1;

     *pasa="sal_ang";
     pantalla_x(borrar);
}
if (Dato.modulo==1)
{
     bode (a,b,m,n,fre_min, fre_max);
     hecho=1;

     *pasa="sal_mod";
     pantalla_x(borrar);
}

if (Dato.ambos==1)
{
     float temp1,temp2;
     bode (a,b,m,n,fre_min, fre_max);
     hecho=1;

     *pasa="sal_ang";
```

```
                pantalla_x(borrar);
                SetPenSize(122);
                temp1=x_leer[1];temp2=x_leer[3];
                *pasa="sal_mod";
                pantalla_x(0);
                x_leer[6]=temp1;x_leer[8]=temp2;x_leer[4]=-6;
                SetPenSize(0);
            }
            *funcion="funcion.fE";
            leer_funcion();
        }
}

class TUserApplication: public TApplication {

public:

            TUserApplication(LPSTR AName,
            HANDLE AnInstance,
            HANDLE APrevInstance,
                                        LPSTR ACmdLine,
                                        int ACmdShow)

                        : TApplication(AName, AnInstance,

APrevInstance,
                                                    ACmdLine,
ACmdShow) {}
            virtual void InitMainWindow();
};


LPSTR APPLICATION_NAME = "Sistemas de Control";

void TUserApplication::InitMainWindow()
{
```

```
      MainWindow = new TMyWindow(NULL, APPLICATION_NAME);
      HAccTable = LoadAccelerators(hInstance,"MIMENU");
}

int PASCAL WinMain(HANDLE AnInstance, HANDLE APrevInstance,
                   LPSTR ACmdLine, int ACmdShow)
{
            TUserApplication Application(APPLICATION_NAME,
                  AnInstance,
                  APrevInstance,
                  ACmdLine,
                  ACmdShow);
            Application.nCmdShow = SW_SHOWMAXIMIZED;
            Application.Run();
            return Application.Status;
}

// Programa de presentacion de ayudas tipo indice
#include <stdlib.h>
#include <stdio.h>
#include <dos.h>
#include <string.h>
#include <owl.h>
#include <static.h>
#include <edit.h>
#include <listbox.h>
#include <button.h>
#include "helpwind.h"

THelpWindow::THelpWindow(PTWindowsObject AParent) :
   TWindow(AParent, "Ayuda General")
{
  DisableAutoCreate();
  Attr.Style |= WS_POPUPWINDOW | WS_CAPTION;
  Attr.X = 100;
  Attr.Y = 100;
```

```
    Attr.W = 300;
  Attr.H = 300;
    ListBox = new TListBox(this, ID_LISTBOX, 20, 20, 180, 80);
     new TButton(this, ID_BUTTON1, "Ayuda", 220, 20, 60, 30,
TRUE);
     new TButton(this, ID_BUTTON2, "Cancele", 220, 70, 60,
30, FALSE);
     Edit = new TEdit(this, ID_EDIT, "", 20, 180, 260, 90,
40, TRUE);
     new TStatic(this, -1, "Información de ayuda:", 20, 160,
160, 20, 0);
}

void THelpWindow::SetupWindow()
{
     TWindow::SetupWindow();
     ListBox->AddString("Acerca de CCW");
     ListBox->AddString("Ingreso de Función");
     ListBox->AddString("Función");
     ListBox->AddString("Fracciones Parciales");
     ListBox->AddString("Transf. Inv. de Laplace");
     ListBox->AddString("Polos y Ceros");
     ListBox->AddString("Expandida");
     ListBox->AddString("Gráficos");
     ListBox->AddString("Diagrama de Bode");
     ListBox->AddString("Respuesta en el Tiempo");
     ListBox->AddString("Lugar Geométrico Raices");
     ListBox->AddString("Diagrama de Nichols");
     ListBox->AddString("Diagrama de Nyquist");
     ListBox->AddString("Valores Propios");
     ListBox->AddString("Calculo de e^At");
     ListBox->AddString("Controlabilidad");
     ListBox->AddString("Observabilidad");
     ListBox->AddString("Integral de e^At");
     ListBox->AddString("Compensación");
     ListBox->SetSelIndex(0);
```

```
};

void THelpWindow::HandleListBoxMsg(RTMessage Msg)
{
     char SelString[25];

     if ( Msg.LP.Hi == LBN_DBLCLK )
     {
     ListBox->GetSelString(SelString, sizeof(SelString));
     FillEdit(SelString);
     }
}

void THelpWindow::HandleButton1Msg(RTMessage)
{
     char SelString[25];

     ListBox->GetSelString(SelString, sizeof(SelString));
     FillEdit(SelString);
}

void THelpWindow::HandleButton2Msg(RTMessage)
{
     CloseWindow();
}

void THelpWindow::FillEdit(PChar SelString)
{
     PChar HelpStr;

     if ( strcmp(SelString, "Acerca de CCW") == 0 )
     {  HelpStr =
          "Este programa le permite analizar\r\n"
          "sistemas de una entrada una salida\r\n"
          "descritos por funciones de transfe-\r\n"
          "rencia.\r\n\r\n"
```

```
            "Tambien perimite analizar sistemas\r\n"
            "multivariables descritos por varia-\r\n"
            "bles de estado.\r\n\r\n"
            "  Escuela politécnica nacional\r\n"
            "Facultad de Ingeniería eléctrica\r\n"
            "                    1994";
      }
      if ( strcmp(SelString, "Respuesta en el Tiempo") == 0
)
      {  HelpStr =
            "Presenta la salida del sistema a\r\n"
            "entradas impulso, paso, rampa.\r\n\r\n"
            "La escala automatica escoge el tiem-\r\n"
            "po máximo como 5 veces la constante\r\n"
            "de tiempo mas grande del sistema.\r\n\r\n"
            "Puede graficar la respuesta en lazo\r\n"
            "abierto o cerrado, si desea obtener\r\n"
            "la función analítica que describe la\r\n"
            "curva, use la transformada inversa\r\n"
            "de Laplace, y luego use la opción\r\n"
            "entrada paso una vez si la entrada\r\n"
            "es paso, o dos veces si la entrada\r\n"
            "es rampa";
      };
      if ( strcmp(SelString, "Ingreso de Función") == 0 )
      {  HelpStr =
            "Para ingresar una nueva función pre-\r\n"
            "sione ALT+A y luego N, se presenta\r\n"
            "una caja de dialogo en la que puede\r\n"
            "ingresar la función de transferencia\r\n"
            "en forma literal por ejemplo.\r\n\r\n"
            "(s^2+5s+6)/(s^3+2s^2+5s+3)\r\n\r\n"
            "Con esta función se trabajará hasta\r\n"
            "que cree una nueva o abra una fun-\r\n"
            "ción previamente grabada. Si cierra\r\n"
            "el programa y luego lo usa nueva-\r\n"
```

```
            "mente la función permanecera activa.";
    }

    if ( strcmp(SelString, "Fracciones Parciales") == 0 )
    {  HelpStr =
            "Muestra la función ingresada en\r\n"
            "Fracciones Parciales por ejemplo\r\n"
            "si se ha ingresado \r\n\r\n"
            "s^2 + 5 s + 6\r\n"
            "s^3 + 2 s^2 + 5 s + 3\r\n\r\n"
            "Se obtiene:\r\n\r\n"
            "    0.22 s + 3.8            +    0.77\r\n"
            "[(s + .63)^2 + 1.91^2]      (s + .73)";
    };
    if ( strcmp(SelString, "Transf. Inv. de Laplace") == 0
)
    {  HelpStr =
            "Obtiene la Transformada Inversa de\r\n"
            "Laplace. Si la función es\r\n\r\n"
            "s^2 + 5 s + 6\r\n"
            "s^3 + 2 s^2 + 5 s + 3\r\n\r\n"
            "Se obtiene:\r\n\r\n"
             "1.95 e ^-.063 t cos(1.91 t - 83)+0.77 e^-0.73
t";
    };
    if ( strcmp(SelString, "Polos y Ceros") == 0 )
    {  HelpStr =
            "Obtiene la función ingresada en\r\n"
            "forma de polos y ceros. Si la\r\n"
            "funcisn es:\r\n\r\n"
            "s^2 + 5 s + 6\r\n"
            "s^3 + 2 s^2 + 5 s + 3\r\n\r\n"
            "Se obtiene:\r\n\r\n"
            "(s + 2)(s + 3)\r\n"
            "[(s+0.63)^2 + 1.91](s + .737)";
    }
```

```
if ( strcmp(SelString, "Expandida") == 0 )
{  HelpStr =
      "Obtiene la función ingresada como\r\n"
      "un solo polinomio. Si la funci≤n\r\n"
      "ingresada es:\r\n\r\n"
      "(s+2)(s+3)\r\n"
      "Se obtiene:\r\n\r\n"
      "s^2 + 5 s + 6";
};
if (strcmp(SelString, "Diagrama de Bode") == 0 )
{  HelpStr =
      "Se puede obtener un diagrama tanto\r\n"
      "de la magnitud en [db], vs w como\r\n"
      "de el ángulo en grados vs w, tanto\r\n"
      "para la funci≤n en lazo abierto o\r\n"
      "lazo cerrado.\r\n\r\n"
      "La escala automática se basa en la\r\n"
      "frecuencia de corte de las aproxi-\r\n"
      "maciones asindóticas y es 10 veces\r\n"
      "más grande que el polo o cero más\r\n"
      "grande y 10 veces más pequeño que\r\n"
      "el polo o cero más pequeño.\r\n\r\n"
      "Para facilitar trazar los margenes\r\n"
      "de fase fase y ganancia se traza\r\n"
      "una linea en 0 db y 180 grados";
};
if ( strcmp(SelString, "Diagrama de Nyquist") == 0 )
{  HelpStr =
      "El diagrama de Nyquist presenta\r\n"
      "la parte imaginaria de G(jw) vs\r\n"
      "la parte real\r\n\r\n"
      "Si la curva rodea al punto -1, j0\r\n"
      "la función es inestable en caso\r\n"
      "contrario es estable.";
};
if ( strcmp(SelString, "Diagrama de Nichols") == 0 )
```

```
{  HelpStr =
       'Grafica el logaritmo del modulo vs\r\n"
       "la fase\r\n\r\n"
       "Con este diagrama tenemos informa-\r\n"
       "ción del Margen de fase, y margen\r\n"
       "de ganancia.";
};
if ( strcmp(SelString, "Compensación") == 0 )
{  HelpStr =
       "Permite ingresar los coeficientes de\r\n"
       "compensadores en adelanto, atraso, y\r\n"
       "atraso adelanto.";
};
if ( strcmp(SelString, "Valores Propios") == 0 )
{  HelpStr =
       "Se obtiene la ecuación caracteris-\r\n"
       'tica y la solución de esta que co-\r\n"
       "rresponde a los valores propios\r\n\r\n"
       'El polinomio caracteristico se lo\r\n"
       'obtiene por hallar el determinante\r\n"
       'de A - Landa I\r\n"
       'Siendo I la matriz identidad\r\n\r\n"
       'Al igualar el polinomio Caracte-\r\n"
       'ristico a cero se obtiene la ecua-\r\n"
       'ción caracteristica los valores de\r\n"
       'landas que satisfacen la ecuación\r\n"
       'caracteristica corresponden a los\r\n"
       'valores propios.";
};
if ( strcmp(SelString, "Calculo de e^At") == 0 )
{  HelpStr =
       'Obtiene la matriz e^At en forma li-\r\n"
       'teral esta matriz esta dada como la\r\n"
       'transformada inversa de Laplace de\r\n"
       "(sI - A)^-1\r\n"
       "siendo I la matriz identidad\r\n\r\n"
```

```
        "Ingresada la matriz A como\r\n\r\n"
        " | 0   1 |\r\n"
        " |-2  -3 |\r\n\r\n"
        "Se obtine\r\n"
        "|  2e^t - e^2t  | e^t - e^2t  |\r\n"
        "|-2e^t + 2e^2t |-e^t - 2e^2t |";
};
if ( strcmp(SelString, "Controlabilidad") == 0 )
{  HelpStr =
        "Presenta la matriz de controlabi-\r\n"
        "lidad y determina si el sistema es\r\n"
        "controlable o no.\r\n\r\n"
        "La matriz de controlabilidad está\r\n"
        "dada por:\r\n\r\n"
        "[ B | AB | ... | A^n-1B]\r\n\r\n"
        "Si las matrices A y B son\r\n"
        "  A = | 1  1 |     B = | 1 |\r\n"
        "      | 0 -1 |         | 0 |\r\n\r\n"
        "el resultado será\r\n"
        "  | 0   1 |\r\n"
        "  | 1  -1 |\r\n"
        "  Controlable";
}
if ( strcmp(SelString, "Observabilidad") == 0 )
{  HelpStr =
        "Presenta la matriz de observabili-\r\n"
        "dad y determina si el sistema es\r\n"
        "observable o no.\r\n\r\n"
        "La matriz de observabildad esta\r\n"
        "dada por:\r\n\r\n"
        "[ C* | A*C* | ... | (A*)^n-1C*]\r\n\r\n"
        "Si las matrices A y C son\r\n"
        "  A = | 1  1 |     C* | 1 |\r\n"
        "      |-2 -1 |        | 0 |\r\n\r\n"
        "el resultado será\r\n"
        "  | 1   1 |\r\n"
```

```
            "   | 0    1 |\r\n"
            "   Observable";
        }
        if ( strcmp(SelString, "Lugar Geométrico Raices") == 0
)
        {   HelpStr =
            "Determina si la matriz es obserbable";
        },
        if ( strcmp(SelString, "Función") == 0 )
        {   HelpStr =
            "Presenta la función de diversas\r\n"
            "formas";
        };
        if ( strcmp(SelString, "Gráficos") == 0 )
        {   HelpStr =
            "Presenta diferentes gráficos de la función";
        };
        if ( strcmp(SelString, "Integral de e^At") == 0 )
        {   HelpStr =
            "Presenta la integral de\r\n"
            "[e^A(t-tao)*b*u d(tao)\r\n"
            "entre o y t\r\n\r\n"
            "Dadas las matrices\r\n\r\n"
            "A = | 0 1|   B = |0|\r\n"
            "       |-2 -3|       |1|\r\n"
            "u = 1(t)\r\n\r\n"
            "Se obtiene\r\n"
            "|.5 - e^-t + .5 e^-2t|\r\n"
            "|    e^-t - e^-2t      |";
        };
        Edit->SetText(HelpStr);
}

/*   Subrutina para obtener raices reales o complejas de un
     polinomio de grado n /*
/*   q1 ...Vector de N+1 coeficientes en orden ascendente
```

```
      polinom ... vector que contiene coeficientes del
      polinomio.
      N ... Grado del polinpmio
      pl... Raices (parte real)
      ql... Raices (parte imaginaria)
      E ... Exactitud deseada*/

#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <iostream.h>
#include "datos.h"
const double E = .0001;
extern float far r[maximo],im[maximo];

void segundo(float u0,float v0,float nr);

void far bairstow(int n, float polinomio[], float u0, float
v0)
{
    int i,j,np;
    double      q1[maximo+1],   p1[maximo],   p2[maximo],
Q2[maximo], p[maximo+4];
    double  p0, P7, q7, p8, p9, p3, p4,p6, q3, q4, p5, q0,
Q8, q5, q6;

// Elimina raices = 0
     while(polinomio[n]==0)
     {
          n--;
          r[n]=0;
          im[n]=0;
     }
     if (n==0) return;
     for(i = 1; i <= n + 1; i++)
          q1[n+2-i]=polinomio[i-1];
```

```
switch (n)
{
case 1: // polinomio de grado 1
     r[0] = -polinomio[1] / polinomio[0];
     im[0] = 0;
     break;

case 2: // polinomio de grado 2
     u0 = -polinomio[1] / polinomio[0];
     v0 = -polinomio[2] / polinomio[0];
     segundo(u0, v0, 1);
     break;

default:
     p[9] = 0;
     P7 = n;
     p[7] = P7;
     p[8] = P7 + 1;
     p[6] = 1;
     q7 = P7 + 2;
     for(i = 1;i<= P7 + 1;i++)
          Q2[q7 - i] = q1[i];

NN:
     p[11] = .00500101;
     p[12] = .010000101;
     p[4] = 0;
GG:
     p8 = p[11];
     p[11] = -10 * p[12];
     p[12] = -10 * p8;
     p8 = p[11];
     p9 = p[12];
     p[4] = p[4] + 1;
     goto BB;
HH:
```

```
      p[9] = 1;
      p[13] = p8;
      p[14] = p9;
BB:
      p[5] = 0;
FF:
      p3 = 0;
      p4 = 0;
      p6 = 0;
      q3 = 1;
      q4 = 0;
      p5 = Q2[P7 + 1];
      if (p5==0) goto CC;
      for (i = 1;i <= P7;i++)
      {
            q0 = P7 + 1 - i;
            Q8 = Q2[q0];
            q5 = p8 * q3 - p9 * q4;
            q6 = p8 * q4 + p9 * q3;
            p5 = p5 + Q8 * q5;
            p6 = p6 + Q8 * q6;
            p3 = p3 + i * q3 * Q8;
            p4 = p4 - i * q4 * Q8;
            q3 = q5;
            q4 = q6;
      }
      p[10] = p3 * p3 + p4 * p4;
      if (p[10] == 0) goto EE;
      p[2] = (p6 * p4 - p5 * p3) / p[10];
      p8 = p8 + p[2];
      p[3] = -(p5 * p4 + p6 * p3) / p[10];
      p9 = p9 + p[3];
      if ((fabs(p[2]) + fabs(p[3])) < E) goto DD;
      p[5] = p[5] + 1;
      if (p[5] < 500) goto FF;
      if (p[9] != 0) goto DD;
```

```
            if (p[4] < 5) goto GG;
//          cout << "JG ERROR6 NO HAY RAICES PARA 100
INTERACCIONES Y 5 VALORES";
            exit(1);

      DD:
            p0 = n + 2;
            for (i = 1;i <= p[8]; i++)
            {
                  q7 = p0 - i;
                  Q8 = q1[q7];
                  q1[q7] = Q2[i];
                  Q2[i] = Q8;
            }
            q7 = P7;
            P7 = p[7];
            p[7] = q7;
            if (p[9] == 0) goto HH;
            goto II;
      EE:
            if (p[9] == 0) goto GG;
            p8 = p[13];
            p9 = p[14];
      II:
            p[9] = 0;
            if (fabs(p9) < (10*E*fabs(p8))) goto JJ;
            p[1] = p8 + p8;
            p[10] = p8 * p8 + p9 * p9;
            P7 = P7 - 2;
            goto KK;
      CC:
            p8 = 0;
            p[7] = p[7] - 1;
            p[8] = p[8] - 1;
      JJ:
            p9 = 0;
```

```
            p[10] = 0;
            p[1] = p8;
            P7  = P7 - 1;
      KK:
            Q2[2] = Q2[2] + p[1] * Q2[1];
            q7 = p[1];
            Q8 = p[10];
            for (i = 2; i<= P7;,i++)
                  Q2[i + 1] = Q2[i + 1] + q7 * Q2[i] - Q8 *
Q2[i - 1];
      MM:
            p1[p[6]] = p8;
            p2[p[6]] = p9;
            p[6] = p[6] + 1;
            if (p[10] == 0) goto LL;
            p9 = -p9;
            p[10] = 0;
            goto MM;
      LL:
            if (P7 > 0) goto NN;
/*          cout << "las raices son\n";
            cout << "cont       real       imag\n";*/

            for (i = 1; i<=n;i++)
            {
//              cout << i << "     " << p1[i] << "     " <<
p2[i] << '\n";
                  if (fabs(p1[i])>.000001) r[i-1]=p1[i];
                  else r[i-1]=0;
                  if (fabs(p2[i])>.000001) im[i-1]=p2[i];
                  else im[i-1]=0;
            }
      }
}

// resuelve ecuacion de segundo grado
```

```
void segundo(float u0,float v0,float nr)
{
     float det;

     det = u0 * u0 + 4 * v0;
     if(det >= 0){
          r[nr-1] = (u0 + pow(det, .5)) / 2;
          im[nr-1] = 0;
          r[nr] = (u0 - pow(det, .5)) / 2;
          im[nr] = 0;
     }
     else
     {
          det = -det;
          r[nr-1] = u0 / 2;
          im[nr-1] = pow(det, .5) / 2;
          r[nr] = r[nr-1];
          im[nr] = -im[nr-1];
     }
}

//   Programas que manejan archivos
#include <malloc.h>
#include <string.h>
#include <stdio.h>
#include <math.h>
#include <stdlib.h>
#include <conio.h>
#include <iostream.h>
#include <complex.h>
#include "clasico.h"
#include "datos.h"
#include "moderno.h"

void far leer(char *arch[],float x_leer[], int &cuenta);
void far g_tiempo(int m, float r[], float im[], complex
```

```
coef[], float t_min, float t_max);
void far escribe(char *arch[],float x_leer[], int n);
void far pasar(float a[], float b[], int m, int n, int caso,
char s1[], float t_min, float t_max);
void far en_fraccion(float a[],float b[],int m, int n, int
caso, char s1[], float t_min, float t_max);
void far _complejo(int m, complex cc[], float a[]);
void far root(float a[], float b[], int m, int n, char s[]);
void far divide(int n, int m, float a[],float b[], float
c[], int *m1);

extern float far r[maximo],im[maximo];

/*   Funcion escribe permite abrir un archivo donde se
     almacena un vector de numeros
     arch ... nombre de archivo a crear
     x_leer ... vector de n elementos */
void far escribe(char *arch[],float x_leer[],int n)
{
     FILE *fp;
//   char car;
     int i;

     if((fp=fopen(arch[0],"w"))==NULL){
//        cout <<"\n No se puede abrir el archivo";
          exit(1);
     }

//   cout << "\n Imprimiendo en Archivo " << arch[0] <<"\n";
     if (n<100) for(i=0;i<n;i++)
fprintf(fp,"%f\n",x_leer[i]);
     else for(i=0;i<n;i++)     fprintf(fp,"%f ",x_leer[i]);
     fclose(fp);
}

/*   Funcion leer permite leer un archivo con elementos
```

```
      numericos en un vector

      arch ... nombre del archivo a leer
      x_leer ... vector a almacenar los elementos
      cuenta ... retorna un entero que indica numero de
      elementos leidos del archvo */
void far leer(char *arch[],float x_leer[], int &cuenta)
{
      FILE *fp;
      int  ;//,numero_total;
      float xi;

      if((fp=fopen(arch[0],"r"))==NULL){
            cout << "\n No ha ingresado la funcion q no la ha
grabado";
            exit(1);
      }
//    cout << "\n Leyendo Archivo " << arch[0];

      i=0;
      while(fscanf(fp,"%f",&xi)!=EOF) {
            x_leer[i]=xi;i++;}
      cuenta=i;
      fclose(fp);
}

void far pasar(float a[], float b[], int m, int n, int caso,
char s1[], float t_min, float t_max)
{
      float c[maximo],d[maximo];
      int m1,i;
      m1=m;

      strcpy(s1,"");
      for(i=0;i<=m;i++) d[i]=a[i];
      if(m1>n){
```

```
            divide(n,m1,d,b,c,m1);
            if (caso==0) mostrarp1(m-n,c,s1);//laplace
            else mostrarp(m-n,c,s1);
      }
      en_fraccion(d, b, m1, n, caso, s1, t_min, t_max);
}

/*    Procedimiento general que permite crear la transformada
      inversa de Laplace, la respuesta en el tiempo, y las
      fracciones parciales
      a . . vector de coeficientes del numerador
      b . . vector de coeficientes del denominador
      m . . numero de elementos del numerador
      n . . numero de elementos del denominador
      caso ... entero usado de bandera para identificar el
      caso a realizar
      t_max, t_min ... numeros que identifican el tiempo
      maximo y minimo a graficar la respuesta en el tiempo */
void far en_fraccion(float a[],float b[],int m, int n, int
caso  char s1[], float t_min, float t_max)
{
      complex *coef, *cc;
      char s2[200];
      float modulo;

      coef = new complex[maximo];
      cc   new complex [maximo];
      if (!cc) {
            cout << "Fallo de asignacion de memoria";
            exit(1);
      }
      else {
            modulo=a[0]/b[0];
            a_complejo(m,cc,a);
            evalua(a,m,b,n,coef);
            if (n>0) bairstow(n,b,0,0);
```

```
            if (caso==1) mostrarfp(n,r,im,coef,s2);
            if (caso==0) mostrarilt(n,r,im,coef,s2);
            if (caso==2) g_tiempo(n,r,im,coef,t_min, t_max);
            strcat(s1,s2);
      }
}
/*    Crea un vector complejo compuesto por las partes reales
      e imaginarias de las raices de un polinomio */
void far a_complejo(int m, complex cc[], float a[])
{
      int i;

      barstow(m,a,0,0);
      for (i=0;i<m;i++) cc[i]=complex(r[i],im[i]);
}

/*    Procedimiento para trazar el Lugar geometrico de las
      raices
      a  .. vector de coeficientes del numerador
      b  .. vector de coeficientes del denominador
      m  .. numero de elementos del numerador
      n  .. numero de elementos del denominador
      s  .. vector que devuelve mensajes */
void far root(float a[], float b[], int m, int n, char s[])
{
      float c[maximo], f[maximo], e[maximo];
      float suma_polos=0, suma_ceros=0, asindota, sigma,
angulo;
      int i, max, j;
      complex cc[maximo], d[maximo], num, den;
      char s1[200];
      FILE *fp;

      if((fp=fopen("datos.lgr","w"))==NULL){
            cout <<"\n No se puede abrir el archivo";
            exit(1);
```

```
    }

    fprintf(fp,"%d %d ",m,n);
//----------Polos y ceros------------------------------
    strcpy(s,"");
    if (m!=0) {
        a_complejo(m,cc,a);
        mostrarf(m,r,im,s1);
    }
    else
        sprintf(s1," %g",a[0]);
    if (m>0)
        for(i=0;i<m;i++) suma_ceros=suma_ceros+r[i];
    else
        suma_ceros=0;
    for (i=0;i<m;i++)
        fprintf(fp," %g %g ", r[i], im[i]);
    strcpy(s,"Ceros: ");
    strcat(s,s1);
    a_complejo(n,d,b);
    if (n>0)
        for(i=0;i<n;i++) suma_polos=suma_polos+r[i];
    else
        suma_polos=0;
    mostrarf(n,r,im,s1);
    for (i=0;i<n;i++)
        fprintf(fp," %g %g ", r[i], im[i]);
    strcat(s,"\nPolos: ");
    strcat(s,s1);

//---------- angulos de partida y llegada --------------
    //caso de polos imaginarios
    char s4[100];
    strcpy(s4,"");
    int cuenta=0;
```

```
    for (i=0;i<n;i++){
        if (im[i]==0) continue;
        angulo=0;
        for (j=0;j<n;j++)
            angulo = angulo + arg(d[i]-d[j]);
            angulo = M_PI - angulo;
        for (j=0;j<m;j++)
            angulo = angulo + arg(d[i]-cc[j]);
        while (angulo>2*M_PI) angulo=angulo-2*M_PI;
        while (angulo<0) angulo=angulo + 2*M_PI;
        sprintf(s1,"\nangulo desde polo %g %g es %g",
            real(d[i]), imag(d[i]), angulo*180/M_PI);
        strcat(s,s1);
        sprintf(s1," %g %g %g ", real(d[i]), imag(d[i]),
angulo*180/M_PI);
        strcat(s4,s1);
        cuenta++;
        i++;
    }
    //caso de ceros imaginarios
    if (m!=0){
        bairstow(m,a,0,0);
        for (i=0;i<m;i++){
            if (im[i]==0) continue;
            angulo=0;
            for (j=0;j<n;j++)
                angulo = angulo + arg(cc[i]-d[j]);
            angulo = M_PI - angulo;
            for (j=0;j<m;j++)
                angulo = angulo + arg(cc[i]-cc[j]);
            sprintf(s1,"\nangulo desde cero %g + %gi es
%g",
                real(cc[i]), imag(cc[i]),
angulo*180/M_PI);
            strcat(s,s1);
        }
```

```
        }

        if (n!=m)
        {
        //presentacion de sigma
            sigma = (suma_polos - suma_ceros) / (n - m);
            sprintf(s1, "\n sigma = %g",sigma);
            strcat(s,s1);
            sprintf(s1,"\n suma de polos = %g \n suma de ceros
= %g", suma_polos, suma_ceros);
            strcat(s,s1);

        //--------Determinacion de asindotas------------
            sprintf(s1, "\n # asindotas = %d:  ", n-m);
            strcat(s,s1);
            for (i=0;i<n-m;i++){
                asindota=180.*(2*i+1)/(n-m);
                sprintf(s1," %g   ", asindota);
                strcat(s,s1);
            }
        }
        else strcpy(s1,"");

//-------Determinacion de y dK/ds y K de ruptura---------
        derivada(m,a,c);
        multiplica(m-1,n,c,b,e);
        derivada(n,b,c);
        multiplica(m,n-1,a,c,f);

        max=n+m-1;//grado de derivada de g

        for(i=0;i<=max;i++) c[i]=f[i]-e[i];

        while (c[0]==0){ //verificar cuando grado maximo se
anula
            max--;
```

```
            for (i=0;i<=max;i++)
                  c[i]=c[i+1];
      }

      strcpy(s1,"\n d/ds[den/num] = ");
      strcat(s,s1);
      mostrarp(max,c,s1);
      strcat(s,s1);

      bairstow(max,c,0,0);
      strcpy(s1, "\n En factores = ");
      strcat(s,s1);
      mostrarf(max,r,im,s1);
      strcat(s,s1);

//--------evaluacion en punto de ruptura------------
      char s2[100],s3[100];

      strcpy(s2,"");
      strcpy(s1,"");
      for(i=0;i<max;i++){
            if(im[i]>=0){
                  num=ev_num(r[i],im[i],cc,m);
                  den=ev_num(r[i],im[i],d,n);
                  den=-den/num*b[0]/a[0];
                  i                                          f
((real(den)>-.0001)&&(fabs(imag(den))<.00001))
                  {
                     '  if (fabs(im[i])<.0004)
                        {
                              angulo=90;
                              sprintf(s1,"\n K = %g en %g + %g
j", real(den), r[i],im[i]);
                              s p r i n t f ( s 3 ; "    % g    % g
%g",r[i],im[i],angulo);
                        }
```

```
                    else
                    {
                         angulo=180;
                         cuenta++;
                         sprintf(s3,"   %g   %g   0   %g   %g
180",r[i],im[i],r[i],im[i]);
                    }
                    cuenta++;
               }
               else
               {
                    strcpy(s1,"");
                    strcpy(s3,"");
               }
               strcat(s2,s3);
               strcat(s,s1);
          }
     }
     fprintf(fp," %d %s %s",cuenta, s4, s2);
     fclose(fp);
}

/*   Lee del archivo matriz los coeficientes que los
     convierte en las matrices A, B, C, D, que representan
     las variables de estado */
void leer_matriz()
{
     int i, j, inicial, cuenta;
     float datos[200];
     extern float far a1[max_m][max_m], b1[max_m][max_m];
     extern float far c1[max_m][max_m], d1[max_m][max_m];
     extern float far x0[max_m],u[max_m];
     extern int far n1, m1, r1, tipo_ent, falla_ent;
     char *lee_matriz[1];

     *lee_matriz="matriz";
```

```
     leer(lee_matriz,datos, cuenta);

     for (i=cuenta; i<200; i++) datos[i]=0;
//A de nxn B de nxm C=mxn D=mxr x=n u=m y=r
     n1=datos[0];
     m1=datos[1];
     r1=datos[2];
//--------------Elementos de matriz-------------
     inicial=3;
     for(i=0;i<n1;i++)
          for(j=0;j<n1;j++)
               a1[i][j]=datos[inicial+j+n1*i];

     inicial=inicial+n1*n1;
     for(i=0;i<n1;i++)
          for(j=0;j<m1;j++)
               b1[i][j]=datos[inicial+j+i*m1];

     inicial=inicial+m1*n1;
     for(i=0;i<r1;i++)
          for(j=0;j<n1;j++)
               c1[i][j]=datos[inicial+j+i*n1];

     inicial=inicial+n1*r1;
     for(i=0;i<r1;i++)
          for(j=0;j<m1;j++)
               d1[i][j]=datos[inicial+j+i*m1];

     inicial=inicial+m1*r1;
     for (i=0;i<m1;i++)
          u[i]=datos[inicial+i];

     inicial=inicial+m1;
     for (i=0;i<n1;i++)
          x0[i]=datos[inicial+i];
```

```
        inicial=inicial+n1;
        tipo_ent=datos[inicial];
        if ((tipo_ent<0)||(tipo_ent>2))
        {
              falla_ent=1;
              tipo_ent=1;
        }
}

/*    Convierte un archivo en los vectores A, B, m, n que
      representan el numerador y denominador de la función de
      transferencia */
void far leer_funcion()
{
      extern int n, m;
      int i;
      extern float far a[maximo], b[maximo], r[maximo],
im[maximo], modulo;
      extern char *funcion[1];

      float x_leer[100];
      leer(funcion,x_leer,n);
      n=x_leer[0];
      m=x_leer[n+2];
      for(i=0;i<=m;i++) a[i]=x_leer[i+3+n];
      for(i=0;i<=n;i++) b[i]=x_leer[i+1];
      modulo = a[0]/b[0];
}

/*    Funcion para ingresar los coeficientes de la funcion de
      transferencia del compensador */
void far leer_compesador()
{
      extern int nc, mc;
      extern float far ac[maximo], bc[maximo], r[maximo],
im[maximo];
```

```
     int i;
     float x_leer[100];
     extern char *compen[1];

     leer(compen,x_leer,nc);
     nc=x_leer[0];
     mc=x_leer[nc+2];
     for(i=0;i<=mc;i++) ac[i]=x_leer[i+3+nc];
     for(i=0;i<=nc;i++) bc[i]=x_leer[i+1];
}
/*   Escribe en un archivo cuyo nombre esta dada por arch la
     cadena esecificada por s */
void far s_cadena(char s[], char *arch)
{
     FILE *fp;

     if((fp = fopen(arch,"w"))==NULL) {
          cout << "No se puede abrir archivo\n";
          exit (1);
     }

     fputs(s, fp);
     fclose(fp);
}

/*   Copia en una cadena s el contenido del archivo indicado
     en arch */
void far f_cadena(char s[], char *arch)
{
     FILE *fp;
     char cad[128];

     if((fp = fopen(arch,"r"))==NULL) {
          cout << "No se puede abrir archivo\n";
          exit (1);
     }
```

```
        while (!feof(fp)){
              if (fgets(cad, 126, fp)) strcpy(s,cad);
        }
        fclose(fp);
  }

  /*    Calcula el los valores maximos y minimos del vector
        valor */
  void extremo(float valor[])
  {
        float minimo, max_y, delta;
        int i;

        minimo=valor[5];
        max_y=valor[5];
        for (i=6;i<105;i++){
              if (valor[i] > max_y) max_y = valor[i];
              if (valor[i] < minimo) minimo = valor[i];}
        delta=max_y-minimo;

        if (delta==0)
        {
              delta=2;
              max_y=max_y+1;
              minimo=minimo-1;
        }
        for (i=5;i<105;i++)
              valor[i]=100.*(valor[i]-minimo)/delta;

        valor[1]=max_y;
        valor[3]=minimo;
  }

  /*    Procedimientos para calcular los coeficientes de las
        fracciones parciales */
  #include <complex.h>
```

```
#include <stdio.h>
#include "datos.h"
#include "clasico.h"

int fact(int n);

void far divide(int n, int m, float d[],float b[], float
c[], int &m1);
void far derivada_f(int m, float a[], int n, float b[],
float c[], float d[], int &max);
complex far ev_num(float real, float imag, complex c[], int
m);
complex far ev_den(float real, float imag, int &cuenta, int
m);

void far evalua(float a[], int m, float b11[], int n,
complex coef[])
{
     int i,j,k,cuenta,guarda,nuevo_m, nuevo_n,cuenta1,m1;
     extern float far r[maximo], im[maximo];
     float modulo, modulo_rep, e[maximo];
     float d[2*maximo], f[2*maximo], g[2*maximo], b[maximo];
     complex den, num, c[maximo],c1[maximo];

     for (i=0;i<=n;i++) b[i]=b11[i];
     bairstow(m,a,0,0);
     for (i=0;i<m;i++) c[i]=complex(r[i],im[i]);

     for(i = 0;i<n;i++){
          bairstow(n,b,0,0);
          modulo=a[0]/b[0];
          //evaluacion de numerador
          num=ev_num(r[i],im[i],c,m);
          //evaluacion de denominador
          cuenta1=0;
          den=ev_den(r[i],im[i],cuenta1,n);
```

```
            cuenta=cuenta1;

            coef[i] = num / den * modulo;
            if    (fabs(imag(coef[i]))<por_mil/10)
coef[i]=complex(real(coef[i]),0);
            if    (fabs(real(coef[i]))<por_mil/10)
coef[i]=complex(0,imag(coef[i]));//cambiar en caso de error

            if (cuenta>1){
                 //en d el polinomio repetido y f el polinomio
^ cuenta
                 e[0]=d[0]=1;
                 e[1]=d[1]=-r[i];
                 for(j=0;j<cuenta-1;j++)
                 {
                      multiplica(1,1+j,d,e,f);
                      for (k=0;k<=j+2;k++) e[k]=f[k];
                 }
                 divide(cuenta,n,b,f,d,m1);
                 //se anula termino de b arreglar esto
                 for (j=0;j<=n;j++) b[j]=b11[j];
                 nuevo_m=m;
                 nuevo_n=n-cuenta;
                 for(j=0;j<=m;j++) g[j]=a[j];
                 for (j=1;j<cuenta;j++){

derivada_f(nuevo_m,g,nuevo_n,d,e,f,guarda);
                      nuevo_m=guarda;
                      nuevo_n=nuevo_n*2;
                      if(guarda>-1){
                           for(k=0;k<=nuevo_m;k++)g[k]=e[k];
                           for(k=0;k<=nuevo_n;k++)d[k]=f[k];
                           bairstow(nuevo_m,g,0,0);
                           for    (k=0;k<nuevo_m;k++)
c1[k]=complex(r[k],im[k]);
                           bairstow(n,b,0,0);
```

```
                    num=ev_num(r[i],im[i],c1,nuevo_m);
                    bairstow(nuevo_n,d,0,0);
                    for     (k=0;k<nuevo_n;k++)
c1[k]=complex(r[k],im[k]);
                    bairstow(n,b,0,0);
                    den=ev_num(r[i],im[i],c1,nuevo_n);
                    i++;
                    modulo_rep=g[0]/d[0];
                    coef[i] = num / den * modulo_rep/
fact(j);
               }
               else //caso 1/(s+1)^3
               {
                    i++;
                    coef[i] = complex(0,0);
               }
               if     (fabs(imag(coef[i]))<por_mil/10)
coef[i]=complex(real(coef[i]),0);
               if     (fabs(real(coef[i]))<por_mil/10)
coef[i]=complex(0,imag(coef[i]));//cambiar en caso de error
          }
     }
     }
     if   (norm(coef[i])<.0000001)
coef[i]=complex(0,0);//cambir en cas de error
}

/*   Evaluación del numerador */
complex far ev_num(float real,float imag,complex c[],int m)
{
     int j;
     complex num;

     num=complex(1,0);
     for(j=0;j<m;j++){
          num = (complex(real,imag) -  c[j]) * num;
```

```
        }
        return num;
}

/*    Evaluación del denominador */
complex far ev_den(float real,float imag,int &cuenta,int m)
{
        int j;
        extern float r[maximo],im[maximo];
        complex den;

        den = complex(1,0);
        cuenta=0;
        for(j = 0;j<m;j++){
             if    ((fabs(real-r[j]))>por_mil*10    ||
fabs((imag!=im[j]))>por_mil*10)
                  den =complex(real-r[j],imag-im[j])*den;
             else cuenta++;
        }
        return den;
}

/*    evaluacion de la derivada de g(s)*/
void far derivada_f(int m, float a[], int n, float b[],
float c[], float d[], int &max)
{
     float a1[30],b1[30],e[30],f[30];
     int i;

     derivada(m,a,a1);
     multiplica(m-1,n,a1,b,e);
     derivada(n,b,b1);
     multiplica(m,n-1,a,b1,f);
     max=n+m-1;
     for(i=0;i<=max;i++) c[i]=e[i]-f[i];
     while (c[0]==0)//caso de eliminar primer termino
```

```
        {
                for(i=1;i<=max;i++) c[i-1]=c[i];
                max--;
        }
        multiplica(n,n,b,b,d);
}

/*      Procedimientos para evaluar el angulo de la función de
        transferencia en un punto */
#include <math.h>
#include <stdio.h>
#include <stdlib.h>
#include <conio.h>
#include <iostream.h>

float maximovalor (float a[],int m,float b[], int n, float
c[], int l);
float minimovalor (float a[],int m,float b[], int n, float
c[], int l);
float angulos (int m, float x1, float y1, float x[], float
y[])
{
        float angulo=0, cada_angulo;
        int i;

        for (i = 0; i<m;i++)
        {
                if ((x1 - x[i]) == 0)
                {
                        if ((y1 - y[i]) > 0) angulo = angulo + 90.;
                        if ((y1 - y[i]) < 0) angulo = angulo + 270.;
                }
                else
                {
                        cada_angulo = atan2((y1 - y[i]) , (x1 -
x[i])) * 180. / M_PI;
```

```
                if    (cada_angulo < 0)
cada_angulo=cada_angulo+360;
                  angulo= angulo + cada_angulo;
            }
//          cout << "x = " << x[i] << " y = " << y[i] <<
"angulo = " << angulo;
      }
      return (angulo);
}

void leer1(char *arch[],float x_leer[])
{
      FILE *fp;
      int i;
      float xi;

      if((fp=fopen(arch[0],"r"))==NULL){
            cout << "\n No se puede abrir el archivo "
<<arch[0];
            exit(1);
      }
//    cout <<"\n Leyendo Archivo " << arch[0];

      i=0;
      while(fscanf(fp,"%f",&xi)!=EOF) {
            x_leer[i]=xi;
            i++;
            if (i>1800) break;
            }
 /*   numero_total=i-1;
      for(i=0;i<=numero_total;i++){
            cout << " " << x_leer[i] << "\n";
            x_leer[i]=x_leer[i]+1;
      }*/
      fclose(fp);
}
```

```
float maximovalor (float a[],int m,float b[], int n, float
c[], int l)
{
      float max;
      int i;

      max = a[0];
      for (i = 1; i<m; i++)
                  if (max < a[i]) max = a[i];
      for (i = 0; i< n;i++)
                  if (max < b[i]) max = b[i];
      for (i = 0; i< l;i++)
                  if (max < c[i]) max = c[i];
      return (max);
      }

float minimovalor (float a[],int m,float b[], int n, float
c[], int l)
{
      float min;
      int i;

      min = a[0];
      for (i = 1; i<m; i++)
                  if (min > a[i]) min = a[i];
      for (i = 0; i< n;i++)
                  if (min > b[i]) min = b[i];
      for (i = 0; i< l;i++)
                  if (min > c[i]) min = c[i];
      return (min);
}

/*
**  eliminacion gausiana
**  Resolucion de sistema de ecuaciones n x n
*/
```

```
#include <iostream.h>
#include <stdlib.h>
#include <conio.h>
#include <math.h>
const int max_m = 10;

float far sist_ecuacion(int n, float a[][max_m])
{
     int i,j,k;
     float m[max_m][max_m];
     float p,cambio, determinante;

     for(i=0;i<n-1;i++){
          //p menor entero con i<=p<=n y a[p,i]!=0
          p = i;
          while (a[p][i]==0){
               p++;
               if (p>=n)
                    return 0.00000212; //cambiar por cero
          }

          //cambio de filas
          if (p!=i)
          {
               for(j=0;j<n;j++)
               {
                    cambio = a[i][j];
                    a[i][j] = a[p][j];
                    a[p][j] = cambio;
               }
//             return 1.2;//a[0][0];
          }

          for(j=i+1;j<n;j++)
          {
               m[j][i] = a[j][i]/a[i][i];
```

```
                for(k=0;k<n;k++)
 a[j][k]=a[j][k]-m[j][i]*a[i][k];
            }
       }

       if (a[n-1][n-1]==0)
                 return 0.0;
       determinante=1.0;
       for (i=0;i<n;i++)
            determinante=determinante*a[i][i];
       return determinante;
}

/*   Procedimiento para crear un archivo con los datos
     correspondientes a la respuesta en frecuencia */
#include <complex.h>
#include "datos.h"
#include "clasico.h"

extern float r[maximo], im[maximo];
void bode(float a[], float b[], int m, int n, float f_min,
float f_max)
{
     complex c[maximo],d[maximo], den, num;
     float  db,  valor_mod[120],  valor_ang[120],  max_y,
minimo, x,y;
     float valor_real[120], valor_imag[120], modulo, temp;
     int i;
     char *pasa[1];

     modulo=a[0]/b[0];
     if (m!=0) {
          bairstow(m,a,0,0);
          for (i=0;i<m;i++)
               c[i]=complex(r[i],im[i]);
     }
```

```
        if (n!=0) {
              bairstow(n,b,0,0);
              for (i=0;i<n;i++)
                    d[i]=complex(r[i],im[i]);
        }

  if (f_min<=0)
  {
        minimo=10000;
        max_y=0;
        if (m!=0) {
              bairstow(m,a,0,0);
              for (i=0;i<m;i++)
              {
                    temp=fabs(r[i]);
                    //caso de j(w/wn)2
                    if (fabs(im[i])>.0001)
                          temp=sqrt(im[i]*im[i]+temp*temp);
                    // .001 para considerar cero en este caso
  asindota cruza en 1
                    if (temp<.001) temp=1;
                    if (temp > max_y) max_y = temp;
                    if (temp < minimo) minimo = temp;
              }
        }
        if (n!=0) {
              bairstow(n,b,0,0);
              for (i=0;i<n;i++)
              {
                    temp=fabs(r[i]);
                    //caso de j(w/wn)2
                    if (fabs(im[i])>.0001)
                          temp=sqrt(im[i]*im[i]+temp*temp);
                    // .001 para considerar cero en este caso
  asindota cruza en 1
                    if (temp<.001) temp=1;
```

```
                if (temp > max_y) max_y = temp;
                if (temp < minimo) minimo = temp;
            }
        }

        //caso de g= 1  w = 0.1 ,  10
        if (minimo==10000) minimo=1;
        if (max_y==0) max_y=1;

        minimo=.1*fabs(minimo);
        max_y=10*fabs(max_y);
}
else
{
        minimo=f_min;
        max_y=f_max;
}
        valor_ang[0]=valor_mod[0]=max_y;
        valor_ang[2]=valor_mod[2]=minimo;
        valor_mod[4]=0;//por ser diagrama de bode
        valor_ang[4]=-1;
        x=-100.*log10(minimo);
        y=log10(max_y)+x/100.;
        for(i=5;i<106;i++){
            db=pow(10,(y*(i-5.)-x)/100.);
            if (m>0)  num=ev_num(0,db,c,m);
            else num=complex(1.,0);
            num=modulo*num;
            if (n>0)  den=ev_num(0,db,d,n);
            else den=complex(b[0],0);
            //caso de 1/(s2+1) se trunca funcion
            if (norm(den)==0) den=.0001;
            den=num/den;
            valor_mod[i]=20*log10(sqrt(norm(den)));
            valor_ang[i]=atan2(imag(den), real(den));
            if (valor_ang[i] > 0)
```

```
  valor_ang[i]=valor_ang[i]-M_PI*2;
            valor_imag[i]=imag(den);
            valor_real[i]=real(den);
       }

       extremo(valor_mod);
       *pasa="sal_mod";
       escribe(pasa,valor_mod,106);//escribe  en  sal_mod  el
 modulo de g()

       extremo(valor_ang);
       valor_ang[1]=valor_ang[1]*180*M_1_PI;
       valor_ang[3]=valor_ang[3]*180*M_1_PI;
       *pasa="sal_ang";
       escribe(pasa,valor_ang,106);//escribe  en  sal_ang  el
 angulo de g()

       extremo(valor_real);
       extremo(valor_imag);
       valor_real[0]=valor_real[1];
       valor_real[2]=valor_real[3];
       valor_real[1]=valor_imag[1];
       valor_real[3]=valor_imag[3];
       valor_real[4]=1; //escala no logaritmica
       *pasa="sal_real";
       escribe(pasa,valor_real,106);//escribe   en   sal_real
 parte imaginaria de g()

      valor_imag[0]=valor_real[1];
      valor_imag[2]=valor_real[3];
      valor_imag[4]=1;
      *pasa="sal_imag";
      escribe(pasa,valor_imag,106);//escribe   en   sal_imag
 parte real de g[]
 }
```

```
void extremo(float valor[])
{
     float minimo, max_y, delta;
     int i;

     minimo=valor[5];
     max_y=valor[5];
     for (i=6;i<106;i++){
          if (valor[i] > max_y) max_y = valor[i];
          if (valor[i] < minimo) minimo = valor[i];}
     delta=max_y-minimo;

     if (delta==0)
     {
          minimo=minimo-1;
          max_y=max_y+1;
          delta=2.;
     }
     for (i=5;i<=106;i++)
          valor[i]=100.*(valor[i]-minimo)/delta;

     valor[1]=max_y;
     valor[3]=minimo;
}

/*   Determina la constante de tiempo maxima */
void limites(float valor[], int n, float &minimo)
{
     int i,cuenta=0;

     minimo=10000.;

     if (n==0) minimo=1;
     for(i=0; i<n; i++)
     {
          if (valor[i]==0) cuenta++;
```

```
            else
                if (minimo>-valor[i]) minimo=-valor[i];
        }

        if ((minimo<=0)||(minimo==10000))
            minimo=1.;
        else
            minimo=5./minimo;
        if (cuenta>1)
            minimo=1.;
}

/*   Programa para generar un procesador de texto */
#include <owl.h>
#include <filewnd.h>
#include <mdi.h>
#include <string.h>
#include <io.h>
#include "mfileapp.h"

_CLASSDEF(TMDIFileApp)
_CLASSDEF(TMDIFileWindow)

class _CLASSTYPE TMDIFileApp : public TApplication {
public:
        TMDIFileApp(LPSTR name, HANDLE hInstance,
            HANDLE hPrevInstance, LPSTR lpCmd,
            int nCmdShow)
                : TApplication(name, hInstance,
                    hPrevInstance,    lpCmd,
nCmdShow) {};
        virtual void InitMainWindow();
        virtual void InitInstance();
};

class _CLASSTYPE TMDIFileWindow : public TMDIFrame {
```

```
public:
          WORD ChildNum;

          TMDIFileWindow(LPSTR ATitle, LPSTR MenuName);
          virtual void NewFile(RTMessage Msg) =[CM_FIRST +
CM_MDIFILENEW];
          virtual void OpenFile(RTMessage Msg) =[CM_FIRST +
CM_MDIFILEOPEN];
};

TMDIFileWindow::TMDIFileWindow(LPSTR ATitle, LPSTR MenuName)
                                                          :
TMDIFrame(ATitle, MenuName)
{
   ChildNum = 1;
}

void TMDIFileWindow::NewFile(RTMessage)
{
     char ChildName[14];

     wsprintf(ChildName,"Funcion %d", ChildNum++);
     GetApplication()->MakeWindow(new     TFileWindow(this,
ChildName, ""));
}

void TMDIFileWindow::OpenFile(RTMessage)
{
     char FileName[MAXPATH];
     char ChildName[14];

     wsprintf(ChildName,"Funcion %d", ChildNum++);
     if ( GetApplication()->ExecDialog(new TFileDialog(this,
SD_FILEOPEN,

_fstrcpy(FileName, "*.ccp"))) == IDOK )
```

```
      GetApplication()->MakeWindow(new    TFileWindow(this,
 ChildName, FileName));
 }

 void TMDIFileApp::InitMainWindow()
 {
      MainWindow = new TMDIFileWindow("Ingreso de Funciones
 y Matrices", "Commands");
      ((PTMDIFileWindow)MainWindow)->ChildMenuPos = 3;
 }

 void TMDIFileApp::InitInstance()
 {
      TApplication::InitInstance();
      if ( Status == 0 )
      {
      HAccTable    =    LoadAccelerators(hInstance,
"FileCommands");
           if ( HAccTable == 0 )
                Status = EM_INVALIDWINDOW;
      }
}

int PASCAL WinMain(HANDLE hInstance, HANDLE hPrevInstance,
                LPSTR lpCmd, int nCmdShow)
{
          TMDIFileApp  MDIFileApp("MDIFileApp",  hInstance,
hPrevInstance,
          lpCmd, nCmdShow);
          MDIFileApp.Run();
          return MDIFileApp.Status;
}

/*   procedimiento para graficar en el tiempo  y mostrar las
     funciones en diferentes formas*/
#include <iostream.h>
```

```
#include <stdio.h>
#include <string.h>
#include <math.h>
#include <complex.h>
#include <conio.h>
#include "datos.h"

void far escribe(char *arch[],float x_leer[], int n);
void far extremo(float valor[]);
int fact(int i);

void far g_tiempo(int m, float r[], float im[], complex
coef[], float t_min, float t_max)
{
     int i,k,cuenta,j;
     float temp;
     float tiempo[120],x, var, pasos;

     for (i=0;i<120;i++) tiempo[i]=0;
tiempo[0]=t_max;
tiempo[2]=t_min;
pasos=(tiempo[0]-tiempo[2])/100.;
x=tiempo[2]-pasos;
for(j=5;j<=105;j++)
{
x=x+pasos;
     for(i=0;i<m;i++){
          cuenta=1;
          if    ((fabs(imag(coef[i])<por_mil)    &&
fabs(im[i])<por_mil)){
              for (k=i+1;k<m;k++)

if((fabs(r[i]-r[k])<por_mil)&&(fabs(im[i]-im[k])<por_mil))
cuenta++;
              while (cuenta>0)
              {
```

```
                    temp=real(coef[i])/fact(cuenta-1);
                    if (fabs(temp)>por_mil)
                         if ((x==0)&&(cuenta==1)) var=temp;
                         else

 var=temp*pow(x,cuenta-1)*exp(r[i]*x);
                    else
                         var=0;
                    tiempo[j]=tiempo[j]+var;
                    cuenta--;
                    i++;
               }
          }
          else
          {
               cuenta=1;
               while (cuenta>0){

 temp=2.*sqrt(norm(coef[i]))/fact(cuenta-1);
                    if (fabs(temp)>por_mil)
                         if    ((x==0)&&(cuenta==1))
 var=temp*cos(arg(coef[i]));
                         else

 var=temp*pow(x,cuenta-1)*exp(r[i]*x)*cos(im[i]*x+arg(coef[
 i]));
                    cuenta=cuenta-2;
                    i=i+2;
                    tiempo[j]=tiempo[j]+var;
               }
          }
          i--;
     }
}
/*
for (j=5;j<105;j++)
```

```
      tiempo[j]=inp(800);
 */
 extremo(tiempo);
 char *pasa[1];
 *pasa="tiempo";
 tiempo[4]=4;
 escribe(pasa,tiempo,105);//escribe en sal_mod el modulo de
 g()
 }

 /*   Procedimiento para mostrar un polinomio */
 void mostrarp(int m, float c[], char s[])
 {
      int i;
      char signo, s1[20];
      float temp;

      strcpy(s,"");
      for(i=0;i<=m;i++){
           temp=c[i];
           if (temp==0) continue;
           if (temp>0) signo='+';
           else {signo='-';temp=-temp;}
           if  ((i==0)  &&  (signo=='+'))  signo='  ';//grado
maximo en i=0
           if (temp!=1) {
                if ((m-i)>1)
                     sprintf(s1,"%c    %g    s^%d
",signo,temp,m-i);
                else
                {
                     if ((m-i)==1)
                          sprintf(s1,"%c %g s ",signo,temp);
                     else
                          sprintf(s1,"%c %g ",signo,temp);
                }
```

```
            }
            else
            {
                if((m-i)>1)
                    sprintf(s1,"%c s^%d ",signo ,m-i);
                else
                {
                    if((m-i)==1)
                        sprintf(s1,"%c s ",signo);
                    else
                        sprintf(s1,"%c 1 ",signo);
                }
            }
            strcat(s,s1);
        }
    }
    /*   Procedimiento para mostrar las fracciones parciales */
    void mostrarf(int m, float r[], float im[],char s[])
    {
        int i,j,cuenta;
        char signo, s1[200];
        float temp, r1[maximo],im1[maximo];

        strcpy(s,"");
        if (m==0) sprintf(s,"%g",r[0]);//grado cero solo real
        cuenta=0;
        //raices = 0, 0, copia de valores
        for(i=0;i<m;i++){
            r1[i]=r[i];
            im1[i]=im[i];
            if ((fabs(r1[i])<por_mil)    &&
    (fabs(im1[i])<por_mil))
            {
                r1[i]=0;
                im1[i]=0;
                cuenta++;
```

```
            }
      }
//     imprime s^k
      if(cuenta>=1)
      {
            if(cuenta==1) strcpy(s,"s");
            else sprintf(s,"s^%d",cuenta);
      }

      for(i=0;i<m;i++){
            if(fabs(im1[i])<por_mil) im1[i]=0;//aproxima
            if(fabs(r1[i])<por_mil) r1[i]=0;//aproxima
            temp=r[i];
            cuenta=0;
            if ((r1[i]==0) && (im1[i]==0))
                  continue;//raiz repetida
            else
            {
                  if (r1[i]!=0)
                        for(j=i+1;j<m;j++)
                              i                                    f
((fabs((temp-r1[j])/temp)<por_mil)&&(fabs(im1[i]-im1[j])<p
or_mil))
                              {
                                    cuenta++;
                                    r1[j]=0;
                              }
                  if (im1[i]!=0)
                        for(j=i+1;j<m;j++)
                              i                                    f
(fabs((im1[i]-im1[j])/im1[i])<por_mil)
                              {
                                    cuenta++;
                                    im1[j]=0;}
            }
```

```
            if (temp<0) signo='+';
            if (temp>0){signo='-';temp=-temp;}
            if ((temp==0) && (im1[i]==0)) {strcat(s1," s
");continue;}
            //impresion de imaginarios
            if (im[i]!=0){
                 if    (r1[i]==0.)    sprintf(s1,"[s^2    +
%g^2",fabs(im1[i]));
                 else    sprintf(s1,"[(s    %c    %g)^2    +
%g^2",signo,fabs(r1[i]),fabs(im1[i]));
                 strcat(s1,"]");
                 i=i+1+cuenta*2;
            }
            //impresion de reales
            else
            {
                 sprintf(s1,"(s %c %g)",signo,-temp);
            }
            strcat(s,s1);
            if(cuenta>0)
            {
                 sprintf(s1,"^%d",cuenta+1);
                 strcat(s,s1);
            }
      }
}

/*   Procedimiento para mostrar la transformada inversa de
     Laplace */
void far mostrarilt(int m, float r[], float im[], complex
coef[], char s[])
{
     int i,k,cuenta;
     char s1[60], s2[60];
     float temp;
     strcpy(s,"");
```

```
      for(i=0;i<m;i++){
           //imprime coeficientes
           cuenta=1;
           if    ((fabs(imag(coef[i])<por_mil)    &&
fabs(im[i])<por_mil))
           {
                for (k=i+1;k<m;k++){

if((fabs(r[i]-r[k])<por_mil)&&(fabs(im[i]-im[k])<por_mil))
                          cuenta++;
                }
                while (cuenta>0){
                     temp=real(coef[i])/fact(cuenta-1);
                     if (fabs(temp)>por_mil){
                          if (cuenta==1)
                          {
                               if     (fabs(r[i])<por_mil)
sprintf(s1,"+ %g ", temp);
                               else sprintf(s1,"+ %g e^ %gt
", temp, r[i]);
                          }
                          else
                          {
                               if     (fabs(r[i])<por_mil)
sprintf(s1,"+ %g t^%d ", temp, cuenta-1);
                               else  sprintf(s1,"+  %g  t^%d
e^%gt ",temp,cuenta-1,r[i]);
                          }
                     }
                     else strcpy(s1,"");
                     cuenta--;
                     i++;
                     strcat(s,s1);
                }
           }
           else
```

```
          {
          //averiguar como hacer cuando hay raices complejas
repetidas
               cuenta=1;
               while (cuenta>0){

temp=2.*sqrt(norm(coef[i]))/fact(cuenta-1);
                    if (temp!=0){
                         if (arg(coef[i])>0) strcpy(s2," +
");
                         if (arg(coef[i])<0) strcpy(s2," -
");

sprintf(s1,"%g",fabs(arg(coef[i])*180/M_PI));
                         strcat(s2,s1);
                         if  (fabs(arg(coef[i])*180)<por_mil)
strcpy(s2,"");

                        if (cuenta==1)
                        {
                             if (fabs(r[i])<por_mil)
                                  sprintf(s1,"+    %g
cos(%gt%s)",temp, fabs(im[i]),s2);
                             else
                                  sprintf(s1,"+  %g  e^%gt
cos(%gt%s)",temp,r[i],
                                       fabs(im[i]),s2);
                        }
                        else
                        {
                             if (fabs(r[i])<por_mil)
                                  sprintf(s1,"+  %g  t^%d
cos(%gt%s)",temp, cuenta-1,im[i],s2);
                             else
                                  sprintf(s1,"+  %g  t^%d
e^%gt cos(%gt%s)",temp, cuenta-1, r[i],
```

```
                                        im[i],s2);
                                }
                        }
                        else strcat (s1,"");
                        cuenta=cuenta-2;
                        i=i+2;
                        strcat(s,s1);
                }
        }
        i--;
    }
}

void far mostrarfp(int m, float r[], float im[], complex
coef[], char s[])
{
    int i,k,cuenta;
    char signo, s1[60],s2[60];
    float temp, num_s, num_t;

    strcpy(s,"");
    for(i=0;i<m;i++){
        //imprime coeficientes
        temp=-real(coef[i]);
        signo=' ';
        if (temp<0) signo='+';
        if (temp>0) signo=' ';
        if    ((fabs(temp)<por_mil)    &&
fabs(imag(coef[i]))<por_mil)  continue;

//      if(fabs(imag(coef[i]))<por_mil)
        if(fabs(im[i])<por_mil)
        {
            sprintf(s1,"%c %lg/",signo,-temp);
        }
        else
```

```
            {

 num_t=-2*(real(coef[i])*r[i]+imag(coef[i])*im[i]);
                num_s=2*real(coef[i]);
                if (num_s>0) sprintf(s2,"+ %g s ",num_s);
                if (num_s<0) sprintf(s2,"- %g s ",-num_s);
                if (fabs(num_s)<por_mil) strcpy(s2,"");

                if (num_t>0) sprintf(s1,"%c (%s+ %g)/",signo,
 s2, num_t);
                if (num_t<0) sprintf(s1,"%c (%s- %g)/",signo,
 s2, -num_t);
                if (fabs(num_t)<por_mil) sprintf(s1," %s/",
 s2);
                //real(coef[i])*2,
 -2*(real(coef[i])*r[i]+imag(coef[i])*im[i]));
           } |
           strcat(s,s1);

           temp=r[i];
           if (temp<0) signo='+';
           if (temp>0){signo='-';temp=-temp;}
           if ((fabs(temp)<por_mil)&&(fabs(im[i])<por_mil))
                strcat(s," s ");
           else
           {
                if (fabs(im[i])>por_mil)
                {
                     if (fabs(temp)<por_mil)
                          sprintf(s1,"(s^2    +    %g^2)",
fabs(im[i]));
                     else
                          sprintf(s1,"([s    %c    %g]^2    +
%g^2)",signo,-temp,fabs(im[i]));
                     i++;
                }
```

```
                else
                    sprintf(s1,"(s %c %g)",signo,-temp);
                strcat(s,s1);
            }
            //imprime el grado del factor
            cuenta=1;
            for (k=i+1;k<m;k++)
            {
                i                                                    f
((fabs(temp-r[k])<por_mil)&&(fabs(im[i]-im[k])<por_mil))
                    cuenta++;
            }

            if (cuenta>1){
                sprintf(s1,"^%d ",cuenta);
                strcat(s,s1);}
        }
}

/*.  Caso de impulsos */
void mostrarpl(int m, float c[], char s[])
{
    int i;
    char signo, s1[20];
    float temp;

    strcpy(s,"");
    for(i=0;i<=m;i++){
        temp=c[i];
        if (temp==0) continue;
        if (temp>0) signo='+';
        else {signo='-';temp=-temp;}
        if ((i==0) && (signo=='+')) signo=' ';//grado
maximo en i=0
        if (temp!=1) {
            if ((m-i)>1)
```

```
                    sprintf(s1,"%c    %g    d^%d/dt[ú(t)]
",signo,temp,m-i);
                else
                {
                    if ((m-i)==1)
                        sprintf(s1,"%c   %g   d/dt   [ú(t)]
",signo,temp);
                    else
                        sprintf(s1,"%c    %g    ú(t)
",signo,temp);
                }
            }
            else
            {
                if((m-i)>1)
                    sprintf(s1,"%c  d^%d/dt[ú(t)]  ",signo
,m-i);
                else
                {
                    if((m-i)==1)
                        sprintf(s1,"%cd/dt[ú(t)]",signo);
                    else
                        sprintf(s1,"%c ú(t) ",signo);
                }
            }
            strcat(s,s1);
        }
}

//   Factorial
int fact(int i)
{
    if(i==0) return 1;
        return (i*fact(i-1));
}
```

```
/* MULTIPLICACION DE DOS POLINOMIOS A,B Y DERIVADA DE
POLINOMIO */
#include <iostream.h>
#include <conio.h>
#include "datos.h"

void far derivada(int n,float a[],float d[])
{
     int i;
     for(i=0;i<n;i++){
          d[i]=(n-i)*a[i];
     }
}

void far multiplica(int n, int m, float a[], float b[],
float c[])
{
     int i,j;
     float sum;

     for(i=0;i<=n+m;i++){
          sum=0;
          c[i]=0;
          if (i>n) a[i]=0;
          if (i>m) b[i]=0;
          for(j=0;j<=i;j++){
               sum=a[j]*b[i-j];
               c[i]=c[i]+sum;
          }
     }
}

/*   Division de polinomios */
void far divide(int n, int m, float d[],float b[], float
c[], int &m1)
{
```

```
        int i,j;
        float a[maximo];

        for(i=0;i<=m;i++) a[i]=d[i];

        for(i = 1;i<=m;i++){
             c[i - 1] = a[i - 1] / b[0];
             for(j = 0;j<= i - 1;j++){
                  if (j>m-n) break;
                  if ((i-j)>n) b[i-j]=0;
                  a[i] = a[i] - c[j] * b[i - j];
             }
        }
        for(i = m - n + 1;i<=m;i++)
        {
             d[i-m+n-1]=a[i];
        }
        m1=n-1;
        while (d[0]==0)
        {
             if (m1<=0) break;
             m1--;
             for (j=0;j<=m1;j++) d[j]=d[j+1];
        }
}

/*   Suma de polinomios */
void far suma(int m, float a[], int n, float b[], int &max,
float c[])
{
     int minimo, i;

     if (m>n)
     {
          minimo=n;
          max=m;
```

```
        }
        else
        {
                minimo=m;
                max=n;
        }

        for(i=0;i<=max;i++)
        {
                if (i<=minimo)
                        c[max-i]=a[m-i]+b[n-i];
                if ((i>minimo)&(m>n))
                        c[max-i]=a[m-i];
                if ((i>minimo)&(n>=m))
                        c[max-i]=b[n-i];
        }
        while (c[0]==0)
        {
                if (max>0){
                        max--;
                        for (i=0;i<=max;i++) c[i]=c[i+1];
                }
                else break;
        }
}

//    eliminacion gausiana con pivoteo
//    Resolucion de sistema de ecuaciones complejo de n x n
#include <iostream.h>
#include <math.h>
#include <stdlib.h>
#include <conio.h>
#include <complex.h>

void ecuacion(complex a[][10], int n, complex b[])
{
```

```
int i,j,k,p;
complex x[10],m[10][10];
int nrow[10],ncopy;
complex suma;
float maximo;

cout << "Matriz original\n";
for(i=0;i<n;i++) a[i][n]=b[i];
for(i=0;i<n;i++){
     for(j=0;j<=n;j++)   cout << a[i][j] << "   ";
     cout << "\n";
}

for(i=0;i<n;i++)
               nrow[i] = i;

for(i=0;i<n-1;i++){
     //p menor entero con i<=p<=n y a[p,i]!=0
     p = i;
     maximo = norm(a[nrow[i]][i]) ;
     for(j=i+1;j<n;j++){
          if (maximo<norm(a[nrow[j]][i])){
                    maximo=norm(a[nrow[j]][i]);
                    p = j;
          }
     }

     if (norm(a[nrow[p]][i])==0){
          cout << "no existe solucion unica";
          exit (1);
     }

     if (nrow[i]!=nrow[p]){
          ncopy = nrow[i];
          nrow[i] = nrow[p];
          nrow[p] = ncopy;
```

```
            }

            for(j=i+1;j<n;j++){
                m[nrow[j]] [i]=a[nrow[j]][i]/a[nrow[i]] [i];
                for(k=0;k<=n;k++)

 a[nrow[j]][k]=a[nrow[j]][k]-m[nrow[j]][i]*a[nrow[i]][k];
            }
      }
      cout << "\nTriangular superior\n";
      for(i=0;i<n;i++){
                  for(j=0;j<=n;j++)
                                    cout << a[i][j] <<
"   ";
                  cout << "\n";
      }

      if (norm(a[nrow[n-1]][n-1])==0) {
                  cout << "No existe solucion unica";
                  exit (1);
      }

      x[n-1]=a[nrow[n-1]][n]/a[nrow[n-1]][n-1];
      for(i=n-2;i>=0;i--){
            suma = 0;
            for(j=i+1;j<n;j++)
                suma=suma+a[nrow[i]][j]*x[j];

            x[i]=(a[nrow[i]][n]-suma)/a[nrow[i]][i];
      }
      cout << "\nSolucion de sistema de ecuaciones\n";
      for(i=0;i<n;i++)
            cout << "x [" <<i <<"] = " << x[i] << "\n";
}

#include <iostream.h>
```

```
#include <string.h>
#include <conio.h>

void mostrarp(int m, float c[], char pasa_s[]);
void mostrarf(int m, float r[],float im[],char s[]);
void bairstow(int n, float a[], float u0, float v0);

void routh(float a[], float b[], int m, int n, char s[])
{
     float matriz[10][10][10], suma,resta, c[10];
     int m1,i,j,max,k,l;
     extern float r[10],im[10];
     char s1[20];

     strcpy(s,"");
     for(i=0;i<10;i++)
          for(j=0;j<10;j++)
               for (k=0;k<10;k++)
                    matriz[i][j][k]=0;

    m1=n/2;
    for(j=0;j<=m1;j++) matriz[0][j][0]=b[2*j];
    for(j=0;j<=(n-1)/2;j++) matriz[1][j][0]=b[2*j+1];

    for(i=0;i<=n;i++) if (i<=m) c[n-i] = a[m-i];
         else c[n-i]=0;

    for(j=0;j<=m1;j++) matriz[0][j][1]=c[2*j];
    for(j=0;j<=(n-1)/2;j++) matriz[1][j][1]=c[2*j+1];

   for (i=2;i<=n;i++)
        for (j=0;j<m1;j++)
             for (l=0;l<=8;l++){
                  suma = 0; resta = 0;
                  for (k=0;k<=l;k++){
                       s     u     m     a                =
```

```
 matriz[i-1][0][l-k]*matriz[i-2][j+1][k]+suma;
                              r    e    s    t    a              =
 matriz[i-2][0][l-k]*matriz[i-1][j+1][k]+resta;
                         }
                         matriz[i][j][l]=suma-resta;
                    }

      for (i=2;i<=n;i++){
           max=8;
           while (matriz[i][0][max]==0) max--;
           for (k=0;k<=max;k++)      c[max-k]=matriz[i][0][k];
           mostrarp(max,c,s1);
           strcat(s,s1);
           strcat(s," = ");
           bairstow(max,c,0,0);
           mostrarf(max,r,im,s1);
           strcat(s,s1);
           strcat(s,"\n");
      }
      for (i=strlen(s);i>=0;i--)    if (s[i]=='s') s[i]='K';
}

/*
      El siguiente programa es generado en LEX para utilizar
      un analizador sintáctico, y luego crear un codigo en C
*/

/*
 * Copyright 1988 Mortice Kern Systems Inc.
 * All rights reserved.
 */
#include <stdio.h>
/* include string.h to get definition of memmove()
  * If you do not have string.h or it does not contain
memmove,
 * you will have to declare memmove here
```

```
 */
#include <string.h>
/*
  * If you have a typedef for size_t in your source or
headers,
  * compile your scanner with _SIZE_T=1 or _SIZE_T_DEFINED=1,
 * to prevent multiple type definitions
  * Turbo C defines size_t in stdio.h, and in lots of other
places.
 * MSC does the same.
 */
#ifndef _SIZE_T_DEFINED        /* MSC defined */
#ifndef _SIZE_T                /* Turbo C defined */
typedef int    size_t;
#define _SIZE_T 1
#endif
#endif

#if __STDC__
#define   YY_ARGS(_args_)     _args_
#else
#define   YY_ARGS(_args_)     ()
#endif

#define   YY_FATAL(msg)  { fprintf(stderr, "yylex: %s\n",
msg); exit(1); }

#define   YY_INTERACTIVE 1    /* save micro-seconds if 0 */

extern FILE    *yyin;         /* scanner input file */
extern FILE    *yyout;        /* scanner output file */

#define   YYLMAX 100          /* token and pushback buffer
size */

#define   YY_RETURN 0
```

```
#define   YY_SKIP        1
#define   YY_REJECT 2
#define   YY_MORE        3

#define   yytext    yy_tbuf        /* token string */

#define   ECHO      fputs(yytext, yyout)
#define   BEGIN          yy_start =
#define   REJECT         (yy_actype = YY_REJECT)
#define   NLSTATE        (yy_lastc = YYNEWLINE)
#define   YY_INIT        (yy_start = 0, yyleng = yy_end = 0,
yy_lastc = YYNEWLINE)

/* user-callable macros */
#define   yymore()  (yy_actype = YY_MORE)
#define   yygetc()  getc(yyin)     /* yylex input source */
#define   output(c) putc((c), yyout) /* yylex sink for
unmatched chars */

/* functions defined in yylex.c */
void yyless    YY_ARGS((int n));
int  input     YY_ARGS((void));
int  unput     YY_ARGS((int c));

/* functions defined in libl.lib */
int  yywrap    YY_ARGS((void));
void yyerror   YY_ARGS((char *fmt, ...));
void yycomment YY_ARGS((char *term));
int  yymapch   YY_ARGS((int delim, int escape));

extern    int  yy_actype;
extern    int  yylineno, yyleng;
extern    int  yy_start, yy_lastc;
extern    unsigned char yy_tbuf [];

#define   INITIAL   0
```

```
#include <stdlib.h>
#include <math.h>
float coeficiente1[20]={0};
int exponente[20]={0};
int ncoefpol[20]={0};
int grado[20]={0};
int npnum,npol=0;
int potencia[20]={0};
char *basura;
int i=0;
int raya=0;
int mierror=0;
int noerror=0;
#define YYNEWLINE 012
static int yy_action(__na__) {
     yy_actype = YY_RETURN;
     switch (__na__) {
     case 0:
     {
               noerror+=yyleng;
               basura=yytext+1;
               exponente[i-1]=atoi(basura);
               if(    exponente[i-1]>grado[npol]    )
grado[npol]=exponente[i-1];
          }
break;
     case 1:
     {
               noerror+=yyleng;
               coeficiente1[i]=1.;
               exponente[i]=1;
               if(    exponente[i]>grado[npol]    )
grado[npol]=exponente[i];
               i++;
          }
break;
```

```
      case 2:
      {
                noerror+=yyleng;
                coeficiente1[i]=-1.;
                exponente[i]=1;
                if(    exponente[i]>grado[npol]    )
grado[npol]=exponente[i];
                i++;
           }
break;
      case 3:
      {
                noerror+=yyleng;
                coeficiente1[i]=atof(yytext);
                exponente[i]=1;
                if(    exponente[i]>grado[npol]    )
grado[npol]=exponente[i];
                i++;
           }
break;
     case 4:
     {
                noerror+=yyleng;
                coeficiente1[i]=atof(yytext);
                exponente[i]=0;
                i++;
           }
break;
     case 5:
     {
                noerror+=yyleng;
                basura=yytext+2;
                potencia[npol]=atoi(basura);
                ncoefpol[npol+1]=i;
                npol++;
                if( raya==0 ) npnum=npol;
```

```
            }
break;
      case 6:
      {
                  noerror+=yyleng;
                  basura=yytext+1;
                  potencia[npol]=atoi(basura);
                  ncoefpol[npol+1]=i;
                  npol++;
                  if( raya==0 ) npnum=npol;
            }
break;
      case 7:
      {
                  noerror+=yyleng;
                  potencia[npol]=1;
                  ncoefpol[npol+1]=i;
                  npol++;
                  if( raya==0 ) npnum=npol;
            }
break;
      case 8:
      {
                  noerror+=yyleng;
                  basura=yytext+2;
                  potencia[npol]=atoi(basura);
                  ncoefpol[npol+1]=i;
                  npol++;
                  raya++;
            }
break;
      case 9:
      {
                  noerror+=yyleng;
                  basura=yytext+1;
                  potencia[npol]=atoi(basura);
```

```
                ncoefpol[npol+1]=i;
                npol++;
                raya++;
            }
 break;
      case 10:
      {
                noerror+=yyleng;
                potencia[npol]=1;
                ncoefpol[npol+1]=i;
                npol++;
                raya++;
            }
 break;
      case 11:
      {
                noerror+=yyleng;
                potencia[npol]=1;
                ncoefpol[npol+1]=i;
                if( raya==0 ) npnum=npol;
                return 1;
            }
 break;
      case 12:
      {
                noerror+=yyleng;
                basura=yytext+1;
                potencia[npol]=atoi(basura);
                ncoefpol[npol+1]=i;
                if( raya==0 ) npnum=npol;
                return 1;
            }
 break;
     case 13:
     {
                noerror+=yyleng;
```

```
                basura=yytext+2;
                potencia[npol]=atoi(basura);
                ncoefpol[npol+1]=i;
                if( raya==0 ) npnum=npol;
                return 1;
            }
break;
    case 14:
    ;
break;
    case 15:
    case 16:
    case 17:
    case 18:
    case 19:
    case 20:
    case 21:
    {
                mierror=noerror+yyleng;
                return 1;
            }
break;
    }
    if (yy_actype == YY_RETURN) yy_actype = YY_SKIP;
    return 0;
}
#define YY_LA_SIZE 8

static unsigned short yy_la_act [] = {
  21|0<<9,  21|0<<9,  1|0<<9,  21|0<<9,  21|0<<9,  4|0<<9,
21|0<<9,  21|0<<9,  21|0<<9,  21|0<<9,  16|0<<9,  7|0<<9,
11|1<<9, 18|0<<9, 10|0<<9, 6|0<<9,
  12|2<<9, 9|0<<9, 5|0<<9, 13|3<<9, 8|0<<9, 4|0<<9, 4|0<<9,
3|0<<9, 20|0<<9, 20|0<<9, 20|0<<9, 20|0<<9, 20|0<<9, 4|0<<9,
4|0<<9, 2|0<<9,
  4|0<<9, 19|0<<9, 1|0<<9, 15|0<<9, 0|0<<9, 17|0<<9, 14|0<<9,
```

```
 21|0<<9,
 };

 static unsigned char yy_look [] = {
   0x40, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x08,
 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x80, 0x00, 0x00, 0x00, 0x00, 0x00, 0
 };

 static short yy_final [] = {
  0, 0, 1, 2, 4, 5, 7, 8, 9, 10, 11, 11, 11, 11, 12, 12,
   13, 14, 15, 15, 16, 16, 17, 18, 18, 18, 19, 19, 20, 21, 22,
23,
   23, 23, 24, 24, 24, 25, 26, 27, 27, 27, 28, 29, 29, 30, 31,
32,
  33, 33, 33, 33, 34, 35, 35, 36, 37, 38, 38, 40,
 };
 typedef unsigned char yy_state_t;
 #define   yy_endst 59
 #define   yy_nxtmax 404

 static yy_state_t yy_begin [] = {
  0, 57,
 };

 static yy_state_t yy_next [] = {
  8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 59, 8, 8, 8, 8, 8,
  8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8,
  8, 8, 8, 8, 8, 8, 8, 8, 7, 6, 8, 2, 8, 4, 8, 8,
  5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 8, 8, 8, 8, 8, 8,
  8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8,
  8, 8, 8, 3, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 1, 8,
  8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8,
  8, 8, 8, 3, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8,
   9, 15, 17, 13, 22, 19, 23, 23, 23, 23, 23, 23, 23, 23,
23,
```

```
  28, 25, 48, 21, 27, 59, 59, 59, 59, 59, 59, 59, 59, 50, 58,
13,
  59, 12, 16, 59, 59, 59, 14, 11, 11, 11, 11, 11, 11, 11, 11,
11,
  11, 19, 25, 18, 24, 59, 32, 33, 20, 26, 30, 59, 29, 29, 29,
29,
  29, 29, 29, 29, 29, 29, 59, 42, 42, 42, 42, 42, 42, 42, 42,
42,
  42, 31, 59, 59, 59, 10, 53, 33, 59, 59, 59, 35, 39, 35, 59,
33,
  36, 36, 36, 36, 36, 36, 36, 36, 36, 36, 34, 59, 59, 59, 59,
59,
  38, 31, 37, 37, 37, 37, 37, 37, 37, 37, 37, 37, 39, 32, 59,
33,
  59, 52, 59, 40, 59, 40, 59, 39, 41, 41, 41, 41, 41, 41, 41,
41,
  41, 41, 43, 59, 43, 59, 59, 44, 44, 44, 44, 44, 44, 44, 44,
44,
  44, 52, 59, 59, 59, 59, 33, 39, 59, 59, 59, 59, 59, 59, 59,
59,
  59, 34, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 59, 59, 59,
59,
  59, 59, 59, 59, 59, 59, 33, 45, 45, 45, 45, 45, 45, 45, 45,
45,
  45, 47, 47, 47, 47, 47, 47, 47, 47, 47, 47, 49, 49, 49, 49,
49,
  49, 49, 49, 49, 49, 51, 51, 51, 51, 51, 51, 51, 51, 51, 51,
59,
  59, 59, 59, 59, 46, 54, 54, 54, 54, 54, 54, 54, 54, 54, 54,
56,
  55, 55, 55, 55, 55, 55, 55, 55, 55, 55, 59, 59, 59, 59, 59,
59,
 59, 59, 59, 59, 46,
};

static yy_state_t yy_check [] = {
```

0, 0, 0, 0, 0, 0, 0, 0, 0, 0, -1, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0; 0, 0, 0, 0, 0, 0, 0; 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
7, 6, 14, 12, 20, 18, 10, 10, 10, 10, 10, 10, 10, 10, 10,
10,
26, 24, 46, 11, 23, 41, 40, 41, 40, 35, 43, 35, 43, 49, 57,
6,
-1, 6, 6, -1, -1, -1, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6,
6, 11, 23, 11, 23, -1, 5, 32, 11, 23, 5, -1, 5, 5, 5, 5,
5, 5, 5, 5, 5, 5, -1, 38, 38, 38, 38, 38, 38, 38, 38, 38,
38, 5, -1, -1, -1, 6, 2, 32, -1, -1, -1, 34, 38, 34, -1, 5,
34, 34, 34, 34, 34, 34, 34, 34, 34, 34, 5, -1, -1, -1, -1,
-1,
36, 5, 36, 36, 36, 36, 36, 36, 36, 36, 36, 36, 38, 44, 44,
5,.
44, 2, -1, 39, -1, 39, -1, 36, 39, 39, 39, 39, 39, 39, 39,
39,
39, 39, 31, -1, 31, -1, -1, 31, 31, 31, 31, 31, 31, 31, 31,
31,
31, 2, -1, -1, -1, -1, 44, 36, -1, -1, -1, -1, -1, -1, -1,
-1,
30, 44, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, -1, -1, -1,
-1,
-1, -1, -1, -1, -1, 45, 44, 45, 45, 45, 45, 45, 45, 45, 45,
45,
45, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 48, 48, 48, 48, 48,
48, 48, 48, 48, 48, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50,
-1,
-1, -1, -1, -1, 4, 53, 53, 53, 53, 53, 53, 53, 53, 53, 53,
1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, -1, -1, -1, -1, -1, -1,

```
  -1, -1, -1, -1, 4,
};

static yy_state_t yy_default [] = {
  59, 59, 4, 46, 59, 59, 59, 59, 59, 59, 59, 59, 59, 59, 59,
59,
  59, 59, 59, 59, 59, 59, 59, 59, 59, 59, 59, 59, 5, 5,
59,
  59, 59, 59, 34, 59, 36, 59, 59, 39, 39, 38, 31, 31, 5, 59,
5,
 59, 48, 59, 50, 46, 59, 59, 59, 59, 0, 7,
};

static short yy_base [] = {
  0, 336, 174, 405, 289, 140, 119, 87, 405, 405, 86, 137, 91,
405, 90, 405,
  405, 405, 93, 405, 92, 405, 405, 138, 105, 405, 104, 405,
405, 405, 258, 231,
  100, 405, 176, 110, 194, 405, 151, 216, 107, 106, 405, 111,
211, 279, 52, 405,
  299, 63, 309, 405, 405, 325, 405, 405, 405, 118, 405, 405,
};
/*
 * Copyright 1988 Mortice Kern Systems Inc.
 * All rights reserved.
 */
/*
 * table interpreter code
 */

#ifdef  YY_DEBUG
#undef  YY_DEBUG
#define YY_DEBUG(fmt, a1, a2)   fprintf(stderr, fmt, a1, a2)
#else
#define YY_DEBUG(fmt, a1, a2)
#endif
```

```
#ifndef YY_NOSCANNER

#ifndef YYLEX
#define YYLEX yylex                    /* lexical analyser name */
#endif

/* stdin and stdout may not be constants */
FILE    *yyin =  stdin;
FILE    *yyout = stdout;

/*
 * yytext is defined to be yy_tbuf.
  * yy_sbuf[0:yyleng-1] contains the states corresponding to
yy_tbuf.
 * yy_tbuf[0:yyleng-1] contains the current token.
  * yy_tbuf[yyleng:yy_end-1] contains pushed-back characters.
 * When the user action routine is active,
 * yy_save contains yy_tbuf[yyleng], which is set to '\0'.
 * Things are different when YY_PRESERVE is defined.
 */

static  yy_state_t yy_sbuf [YYLMAX+1];  /* state buffer */
unsigned char yy_tbuf [YYLMAX+1]; /* text buffer */

#ifndef YY_PRESERVE     /* the efficient default push-back
scheme */

static  unsigned char yy_save;  /* saved yytext[yyleng] */

#define YY_USER { /* set up yytext for user */ \
          yy_save = yytext[yyleng]; \
          yytext[yyleng] = 0; \
     }
#define YY_SCANNER { /* set up yytext for scanner */ \
          yytext[yyleng] = yy_save; \
     }
```

```
#else       /* not-so efficient push-back for yytext
mungers */

static  unsigned char yy_save [YYLMAX];
static  unsigned char *yy_push = yy_save+YYLMAX;

#define YY_USER { \
          size_t n = yy_end - yyleng; \
          yy_push = yy_save+YYLMAX - n; \
          if (n > 0) \
               memmove(yy_push, yytext+yyleng, n); \
          yytext[yyleng] = 0; \
     }
#define YY_SCANNER { \
          size_t n = yy_save+YYLMAX - yy_push; \
          if (n > 0) \
               memmove(yytext+yyleng, yy_push, n); \
          yy_end = yyleng + n; \
     }

#endif

int     yy_actype;              /* special action */
int     yylineno = 1;           /* line number */
int     yyleng = 0;             /* yytext token length */
int     yy_end = 0;             /* end of pushback */
int     yy_start = 0;           /* start state */
int     yy_lastc = YYNEWLINE;   /* previous char */

#endif

#if __STDC__
int YYLEX(void) {
#else
int YYLEX()
{
```

```
#endif
     register int c, i, st, base;
     int fmin, fmax;                    /* yy_la_act indices of
final states */
     int oldi, oleng;                        /* base i, yyleng
before look-ahead */

     i = yyleng;
     YY_SCANNER;

  start:
     yyleng = i;
     /* determine previous char. */
     if (i > 0)
          yy_lastc = yytext[i-1];
     /* scan previously accepted token adjusting yylineno */
     while (i > 0)
          if (yytext[--i] == YYNEWLINE)
               yylineno++;
     /* adjust pushback */
     yy_end -= yyleng;
     memmove(yytext, yytext+yyleng, (size_t) yy_end);
     i = 0;

  more:
     oldi = i;

     /* run the state machine until it jams */
     for (yy_sbuf[i] = st = yy_begin[yy_start + (yy_lastc ==
YYNEWLINE)];
               !(st == yy_endst || YY_INTERACTIVE && yy_base[st]
> yy_nxtmax && yy_default[st] == yy_endst);
          yy_sbuf[++i] = st) {
          YY_DEBUG("<state %d, i = %d>\n", st, i);
          if (i >= YYLMAX)
               YY_FATAL("Token buffer overflow");
```

```
            /* get input char */
            if (i < yy_end)
                 c = yy_tbuf[i];         /* get pushback char
*/
            else if ((c = yygetc()) != EOF) {
                 yy_end = i+1;
                 yy_tbuf[i] = c = (unsigned char) c;
            } else /* c == EOF */ {
                 if (i == oldi)  /* no token */
                      if (yywrap())
                           return 0;
                      else
                           goto start;
                 else
                      break;
            }
            YY_DEBUG("<input %d = 0x%02x>\n", c, c);

            /* look up next state */
            while ((base = yy_base[st]+c) > yy_nxtmax ||
yy_check[base] != st) {
                 if (st == yy_endst)
                      goto jammed;
                 st = yy_default[st];
            }
            st = yy_next[base];
       jammed: ;
     }
     YY_DEBUG("<stopped %d, i = %d>\n", st, i);
     if (st != yy_endst)
          ++i;

 reject:
     /* search backward for a final state */
     while (--i > oldi) {
          st = yy_sbuf[i];
```

```
            if    ((fmin    =    yy_final[st])    <    (fmax    =
 yy_final[st+1]))
                goto final;      /* found final state(s) */
        }
        /* no match, default action */
        i = oldi + 1;
        output(yy_tbuf[oldi]);
        goto start;

    final:
        YY_DEBUG("<final state %d, i = %d>\n", st, i);
        oleng = i;                /* save length for REJECT */

        /* pushback look-ahead RHS */
        if ((c = (int)(yy_la_act[fmin]>>9) - 1) >= 0) { /*
trailing context? */
            unsigned char *bv = yy_look + c*YY_LA_SIZE;
            static unsigned char bits [8] = {
                1<<0, 1<<1, 1<<2, 1<<3, 1<<4, 1<<5, 1<<6,
1<<7
            };
            while (i > oldi) {
                st = yy_sbuf[--i];
                if    (bv[(unsigned)st/8]    &
bits[(unsigned)st%8])
                    break;
            }
        }

        /* perform action */
        yyleng = i;
        YY_USER;
        c = yy_action(yy_la_act[fmin] & 0777);
        YY_SCANNER;
        i = yyleng;
```

```
        /* process special actions */
        YY_DEBUG("<actype %d, return %d>\n", yy_actype, c);
        switch (yy_actype) {
            case YY_RETURN:                 /* action did return */
             YY_USER;
             return c;

            case YY_SKIP:                   /* action fell through
 */
             goto start;

          case YY_REJECT:                   /* REJECT */
             i = oleng;                   /* restore original yytext
 */
             if (++fmin < fmax)
                  goto final;      /* another final state, same
 length */
             else
                  goto reject;     /* try shorter yytext */

          case YY_MORE:                     /* yymore() */
             if (i > 0)
                  yy_lastc = yytext[i-1];
             goto more;
        }
    }

#ifndef YY_NOSCANNER

/*
 * user callable routines
 */

/* get input char with pushback */
int input()
{
```

```
      int c;
#ifndef YY_PRESERVE
      if (yy_end > yyleng) {
            yy_end--;
            memmove(yytext+yyleng, yytext+yyleng+1,
                  (size_t) (yy_end-yyleng));
            c = yy_save;
            YY_USER;
#else
      if (yy_push < yy_save+YYLMAX) {
            c = *yy_push++;
#endif
      } else
            c = yygetc();
      yy_lastc = c;
      if (c == YYNEWLINE)
            yylineno++;
      return c;
}

/* pushback char */
int unput(c)
      int c;
{
#ifndef YY_PRESERVE
      if (yy_end >= YYLMAX)
            YY_FATAL("Push-back buffer overflow");
      if (yy_end > yyleng) {
            yytext[yyleng] = yy_save;
            memmove(yytext+yyleng+1, yytext+yyleng,
                  (size_t) (yy_end-yyleng));
            yytext[yyleng] = 0;
      }
      yy_end++;
      yy_save = c;
#else
```

```
     if (yy_push <= yy_save)
          YY_FATAL("Push-back buffer overflow");
     *--yy_push = c;
#endif
     if (c == YYNEWLINE)
          yylineno--;
     return c;
}

/* accept first `n' chars of token */
void yyless(int n)
{
     if (n < 0 || n > yy_end)
          return;
     YY_SCANNER;
     yyleng = n;                    /* adjust pushback */
     YY_USER;
}

#endif
#line 156 "polinom2.lex"

yywrap(){return 1;}

void multiplica2(int na, int nb, int *nc,float a[], float
b[], float c[]);
void main(int argc,char *argv[])
{
 /* (coeficiente*s^exponente...)^potencia */
 int k,p,q;
 int m;
 int g1,g2,g3,g4,gn,gd;
  float aux1[20],aux2[20],aux3[20],aux4[20],num[20],den[20];
 int npden;
 FILE *salida;
```

```
 if(argc!=3)
 {
      printf("error en numero de argumentos\npolinom2.exe
archin archout");
   exit(1);
 }

 if((yyin=fopen(argv[1],"r"))==NULL)
 {
      printf("error:no se puede abrir el archivo de entrada");
   exit(1);
 }
 yylex();
 fclose(yyin);

 if((salida=fopen(argv[2],"w"))==NULL)
 {
  printf("error:no se puede abrir el archivo de salida");
   exit(1);
 }

 if( mierror!=0 )
   fprintf(salida,"0 1 0 1");
 else
 {
   npden=npol-npnum-1;

   /* EXPANSION DEL NUMERADOR */
   g1=grado[0];
   for( p=0;p<=g1;p++) aux1[p]=0;
   for( p=ncoefpol[0];p<ncoefpol[1];p++)

aux1[g1-exponente[p]]=aux1[g1-exponente[p]]+coeficiente1[p];
   gn=g1;
   for( p=0;p<=gn;p++) num[p]=aux1[p];
```

```
    if( potencia[0]>=2 )
    {
      for( p=1;p<potencia[0];p++)
      {
        multiplica2(g1,gn,&g3,aux1,num,aux3);
        gn=g3;
        for( q=0;q<=gn;q++) num[q]=aux3[q];
      }
    }
    if( npnum>=1 )
    {
      for( k=1;k<=npnum;k++)
      {
        g1=grado[k];
        for( p=0;p<=g1;p++) aux1[p]=0;
        for( p=ncoefpol[k];p<ncoefpol[k+1];p++)

aux1[g1-exponente[p]]=aux1[g1-exponente[p]]+coeficiente1[p];
          g4=g1;
        for( p=0;p<=g4;p++) aux4[p]=aux1[p];
        if( potencia[k]>=2 )
        {
       for( p=1;p<potencia[k];p++)
       {
         multiplica2(g1,g4,&g3,aux1,aux4,aux3);
         g4=g3;
         for( q=0;q<=g4;q++) aux4[q]=aux3[q];
       }
        }
        multiplica2(gn,g4,&g2,num,aux4,aux2);
        gn=g2;
        for( p=0;p<=gn;p++) num[p]=aux2[p];
      }
    }

    /* EXPANSION DEL DENOMINADOR */
```

```
    if( npden>0 )
    {
      g1=grado[npnum+1];
      for( p=0;p<=g1;p++) aux1[p]=0;
      for( p=ncoefpol[npnum+1];p<ncoefpol[npnum+2];p++)

aux1[g1-exponente[p]]=aux1[g1-exponente[p]]+coeficiente1[p];
      gd=g1;
      for( p=0;p<=gd;p++) den[p]=aux1[p];
      if( potencia[npnum+1]>=2 )
      {
        for( p=1;p<potencia[npnum+1];p++)
        {
       multiplica2(g1,gd,&g3,aux1,den,aux3);
       gd=g3;
       for( q=0;q<=gd;q++) den[q]=aux3[q];
        }
      }

     if( npden>=1 )
     {
       for( k=npnum+2;k<=npol;k++)
       {
      g1=grado[k];
      for( p=0;p<=g1;p++) aux1[p]=0;
      for( p=ncoefpol[k];p<ncoefpol[k+1];p++)

aux1[g1-exponente[p]]=aux1[g1-exponente[p]]+coeficiente1[p];
      g4=g1;
      for( p=0;p<=g4;p++) aux4[p]=aux1[p];
      if( potencia[k]>=2 )
      {
        for( p=1;p<potencia[k];p++)
        {
          multiplica2(g1,g4,&g3,aux1,aux4,aux3);
          g4=g3;
```

```
            for( q=0;q<=g4;q++) aux4[q]=aux3[q];
          }
        }
        multiplica2(gd,g4,&g2,den,aux4,aux2);
        gd=g2;
        for( p=0;p<=gd;p++) den[p]=aux2[p];
         }
       }
    }
    else
    {
      gd=0;
      den[0]=1;
    }

    fprintf(salida,"%d",gd);
    for (p=0;p<=gd;p++)
      fprintf(salida," %f",den[p]);
    fprintf(salida," %d",gn);
    for (p=0;p<=gn;p++)
      fprintf(salida," %f",num[p]);
  }
 fclose(salida);
}


void multiplica2(int na, int nb, int *nc, float a[], float
b[], float c[])
{
  int i,j;

  *nc=na+nb;
  for(i=0;i<=na+nb;i++) c[i]=0;
  for(i=0;i<=na;i++)
  {
    for(j=0;j<=nb;j++)
```

```
        c[i+j]=c[i+j]+a[i]*b[j];
    }
 }

 /*    Archivo generado en C */

 /*
  * Copyright 1988 Mortice Kern Systems Inc.
  * All rights reserved.
  */
 #include <stdio.h>
 /* include string.h to get definition of memmove()
   * If you do not have string.h or it does not contain
 memmove,
  * you will have to declare memmove here
  */
 #include <string.h>
 /*
   * If you have a typedef for size_t in your source or
 headers,
   * compile your scanner with _SIZE_T=1 or _SIZE_T_DEFINED=1,
  * to prevent multiple type definitions
   * Turbo C defines size_t in stdio.h, and in lots of other
 places.
  * MSC does the same.
  */
 #ifndef _SIZE_T_DEFINED        /* MSC defined */
 #ifndef _SIZE_T                /* Turbo C defined */
 typedef int    size_t;
 #define _SIZE_T 1
 #endif
 #endif

 #if __STDC__
 #define   YY_ARGS(_args_)     _args_
 #else
```

```
#define   YY_ARGS(_args_)     ()
#endif

#define   YY_FATAL(msg)  { fprintf(stderr, "yylex: %s\n",
msg); exit(1); }

#define   YY_INTERACTIVE 1    /* save micro-seconds if 0 */

extern FILE   *yyin;          /* scanner input file */
extern FILE   *yyout;         /* scanner output file */

#define   YYLMAX 100          /* token and pushback buffer
size */

#define   YY_RETURN 0
#define   YY_SKIP       1
#define   YY_REJECT 2
#define   YY_MORE       3

#define   yytext    yy_tbuf         /* token string */

#define   ECHO      fputs(yytext, yyout)
#define   BEGIN           yy_start =
#define   REJECT          (yy_actype = YY_REJECT)
#define   NLSTATE         (yy_lastc = YYNEWLINE)
#define   YY_INIT         (yy_start = 0, yyleng = yy_end = 0,
yy_lastc = YYNEWLINE)

/* user-callable macros */
#define   yymore()  (yy_actype = YY_MORE)
#define   yygetc()  getc(yyin)      /* yylex input source */
#define   output(c) putc((c), yyout)  /*  yylex  sink  for
unmatched chars */

/* functions defined in yylex.c */
void yyless    YY_ARGS((int n));
```

```
 int  input    YY_ARGS((void));
 int  unput    YY_ARGS((int c));

 /* functions defined in libl.lib */
 int  yywrap   YY_ARGS((void));
 void yyerror  YY_ARGS((char *fmt, ...));
 void yycomment YY_ARGS((char *term));
 int  yymapch  YY_ARGS((int delim, int escape));

 extern    int  yy_actype;
 extern    int  yylineno, yyleng;
 extern    int  yy_start, yy_lastc;
 extern    unsigned char yy_tbuf [];

#define   INITIAL   0

  #include <stdlib.h>
  #include <math.h>
  float coeficiente1[20]={0};
  int exponente[20]={0};
  int ncoefpol[20]={0};
  int grado[20]={0};
  int npnum,npol=0;
  int potencia[20]={0};
  char *basura;
  int i=0;
  int raya=0;
  int mierror=0;
  int noerror=0;
#define YYNEWLINE 012
static int yy_action(__na__) {
     yy_actype = YY_RETURN;
     switch (__na__) {
     case 0:
     {
               noerror+=yyleng;
```

```
                basura=yytext+1;
                exponente[i-1]=atoi(basura);
                if(     exponente[i-1]>grado[npol]     )
 grado[npol]=exponente[i-1];
            }
 break;
      case 1:
      {
                noerror+=yyleng;
                coeficiente1[i]=1.;
                exponente[i]=1;
                if(     exponente[i]>grado[npol]     )
 grado[npol]=exponente[i];
                i++;
            }
 break;
      case 2:
      {
                noerror+=yyleng;
                coeficiente1[i]=-1.;
                exponente[i]=1;
                if(     exponente[i]>grado[npol]     )
grado[npol]=exponente[i];
                i++;
            }
break;
      case 3:
      {
                noerror+=yyleng;
                coeficiente1[i]=atof(yytext);
                exponente[i]=1;
                if(     exponente[i]>grado[npol]     )
grado[npol]=exponente[i];
                i++;
            }
break;
```

```
        case 4:
        {
                noerror+=yyleng;
                coeficiente1[i]=atof(yytext);
                exponente[i]=0;
                i++;
            }
 break;
        case 5:
        {
                noerror+=yyleng;
                basura=yytext+2;
                potencia[npol]=atoi(basura);
                ncoefpol[npol+1]=i;
                npol++;
                if( raya==0 ) npnum=npol;
            }
 break;
        case 6:
        {
                noerror+=yyleng;
                basura=yytext+1;
                potencia[npol]=atoi(basura);
                ncoefpol[npol+1]=i;
                npol++;
                if( raya==0 ) npnum=npol;
            }
 break;
        case 7:
        {
                noerror+=yyleng;
                potencia[npol]=1;
                ncoefpol[npol+1]=i;
                npol++;
                if( raya==0 ) npnum=npol;
            }
```

```
 break;
       case 8:
       {
              noerror+=yyleng;
              basura=yytext+2;
              potencia[npol]=atoi(basura);
              ncoefpol[npol+1]=i;
              npol++;
              raya++;
            }
 break;
       case 9:
       {
              noerror+=yyleng;
              basura=yytext+1;
              potencia[npol]=atoi(basura);
              ncoefpol[npol+1]=i;
              npol++;
              raya++;
            }
 break;
       case 10:
       {
              noerror+=yyleng;
              potencia[npol]=1;
              ncoefpol[npol+1]=i;
              npol++;
              raya++;
            }
 break;
       case 11:
       {
              noerror+=yyleng;
              potencia[npol]=1;
              ncoefpol[npol+1]=i;
              if( raya==0 ) npnum=npol;
```

```
                return 1;
            }
 break;
      case 12:
      {
                noerror+=yyleng;
                basura=yytext+1;
                potencia[npol]=atoi(basura);
                ncoefpol[npol+1]=i;
                if( raya==0 ) npnum=npol;
                return 1;
            }
 break;
      case 13:
      {
                noerror+=yyleng;
                basura=yytext+2;
                potencia[npol]=atoi(basura);
                ncoefpol[npol+1]=i;
                if( raya==0 ) npnum=npol;
                return 1;
            }
 break;
      case 14:
      ;
 break;
      case 15:
      case 16:
      case 17:
      case 18:
      case 19:
      case 20:
      case 21:
      {
                mierror=noerror+yyleng;
                return 1;
```

```
            }
 break;
      }
      if (yy_actype == YY_RETURN) yy_actype = YY_SKIP;
      return 0;
 }
 #define YY_LA_SIZE 8

 static unsigned short yy_la_act [] = {
   21|0<<9,  21|0<<9,  1|0<<9,  21|0<<9,  21|0<<9,  4|0<<9,
 21|0<<9,  21|0<<9,  21|0<<9,  21|0<<9,  16|0<<9,  7|0<<9,
 11|1<<9, 18|0<<9, 10|0<<9, 6|0<<9,
   12|2<<9, 9|0<<9, 5|0<<9, 13|3<<9, 8|0<<9, 4|0<<9, 4|0<<9,
 3|0<<9, 20|0<<9, 20|0<<9, 20|0<<9, 20|0<<9, 20|0<<9, 4|0<<9,
 4|0<<9, 2|0<<9,
   4|0<<9, 19|0<<9, 1|0<<9, 15|0<<9, 0|0<<9, 17|0<<9, 14|0<<9,
 21|0<<9,
 };

 static unsigned char yy_look [] = {
   0x40, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x08,
 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x80, 0x00, 0x00, 0x00, 0x00, 0x00, 0
 };

 static short yy_final [] = {
  0, 0, 1, 2, 4, 5, 7, 8, 9, 10, 11, 11, 11, 11, 12, 12,
   13, 14, 15, 15, 16, 16, 17, 18, 18, 18, 19, 19, 20, 21, 22,
 23,
  23, 23, 24, 24, 24, 25, 26, 27, 27, 27, 28, 29, 29, 30, 31,
 32,
  33, 33, 33, 33, 34, 35, 35, 36, 37, 38, 38, 40,
 };
 typedef unsigned char yy_state_t;
 #define   yy_endst 59
 #define   yy_nxtmax 404
```

```
59,
  59, 59, 59, 59, 59, 59, 33, 45, 45, 45, 45, 45, 45, 45, 45,
45,
  45, 47, 47, 47, 47, 47, 47, 47, 47, 47, 47, 49, 49, 49, 49,
49,
  49, 49, 49, 49, 49, 51, 51, 51, 51, 51, 51, 51, 51, 51, 51,
59,
  59, 59, 59, 59, 46, 54, 54, 54, 54, 54, 54, 54, 54, 54, 54,
56,
  55, 55, 55, 55, 55, 55, 55, 55, 55, 55, 59, 59, 59, 59, 59,
59,
 59, 59, 59, 59, 46,
};

static yy_state_t yy_check [] = {
 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, -1, 0, 0, 0, 0, 0,
 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
  7, 6, 14, 12, 20, 18, 10, 10, 10, 10, 10, 10, 10, 10, 10,
10,
  26, 24, 46, 11, 23, 41, 40, 41, 40, 35, 43, 35, 43, 49, 57,
6,
 -1, 6, 6, -1, -1, -1, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6,
 6, 11, 23, 11, 23, -1, 5, 32, 11, 23, 5, -1, 5, 5, 5, 5,
 5, 5, 5, 5, 5, 5, -1, 38, 38, 38, 38, 38, 38, 38, 38, 38,
  38, 5, -1, -1, -1, 6, 2, 32, -1, -1, -1, 34, 38, 34, -1, 5,
  34, 34, 34, 34, 34, 34, 34, 34, 34, 34, 5, -1, -1, -1, -1,
-1,
  36, 5, 36, 36, 36, 36, 36, 36, 36, 36, 36, 36, 38, 44, 44,
5,
  44, 2, -1, 39, -1, 39, -1, 36, 39, 39, 39, 39, 39, 39, 39,
```

```
 39,
   39, 39, 31, -1, 31, -1, -1, 31, 31, 31, 31, 31, 31, 31, 31,
 31,
   31, 2, -1, -1, -1, -1, 44, 36, -1, -1, -1, -1, -1, -1, -1,
 -1,
   30, 44, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, -1, -1, -1,
 -1,
   -1, -1, -1, -1, -1, 45, 44, 45, 45, 45, 45, 45, 45, 45, 45,
 45,
  45, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 48, 48, 48, 48, 48,
   48, 48, 48, 48, 48, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50,
 -1,
   -1, -1, -1, -1, 4, 53, 53, 53, 53, 53, 53, 53, 53, 53, 53,
 1,
  1, 1, 1, 1, 1, 1, 1, 1, 1, 1, -1, -1, -1, -1, -1, -1,
  -1, -1, -1, -1, 4,
};

static yy_state_t yy_default [] = {
   59, 59, 4, 46, 59, 59, 59, 59, 59, 59, 59, 59, 59, 59, 59,
59,
   59, 59, 59, 59, 59, 59, 59, 59, 59, 59, 59, 59, 5, 5,
59,
   59, 59, 59, 34, 59, 36, 59, 59, 39, 39, 38, 31, 31, 5, 59,
5,
  59, 48, 59, 50, 46, 59, 59, 59, 59, 0, 7,
};

static short yy_base [] = {
   0, 336, 174, 405, 289, 140, 119, 87, 405, 405, 86, 137, 91,
405, 90, 405,
   405, 405, 93, 405, 92, 405, 405, 138, 105, 405, 104, 405,
405, 405, 258, 231,
   100, 405, 176, 110, 194, 405, 151, 216, 107, 106, 405, 111,
211, 279, 52, 405,
   299, 63, 309, 405, 405, 325, 405, 405, 405, 118, 405, 405,
```

```
};
/*
 * Copyright 1988 Mortice Kern Systems Inc.
 * All rights reserved.
 */
/*
 * table interpreter code
 */

#ifdef  YY_DEBUG
#undef  YY_DEBUG
#define YY_DEBUG(fmt, a1, a2)   fprintf(stderr, fmt, a1, a2)
#else
#define YY_DEBUG(fmt, a1, a2)
#endif

#ifndef YY_NOSCANNER

#ifndef YYLEX
#define YYLEX yylex                 /* lexical analyser name */
#endif

/* stdin and stdout may not be constants */
FILE    *yyin =  stdin;
FILE    *yyout = stdout;

/*
 * yytext is defined to be yy_tbuf.
 * yy_sbuf[0:yyleng-1] contains the states corresponding to
yy_tbuf.
 * yy_tbuf[0:yyleng-1] contains the current token.
 * yy_tbuf[yyleng:yy_end-1] contains pushed-back characters.
 * When the user action routine is active,
 * yy_save contains yy_tbuf[yyleng], which is set to '\0'.
 * Things are different when YY_PRESERVE is defined.
 */
```

```
static  yy_state_t yy_sbuf [YYLMAX+1];  /* state buffer */
unsigned char yy_tbuf [YYLMAX+1]; /* text buffer */

#ifndef YY_PRESERVE     /* the efficient default push-back
scheme */

static  unsigned char yy_save;  /* saved yytext[yyleng] */

#define YY_USER { /* set up yytext for user */ \
          yy_save = yytext[yyleng]; \
          yytext[yyleng] = 0; \
     }
#define YY_SCANNER { /* set up yytext for scanner */ \
          yytext[yyleng] = yy_save; \
     }

#else           /* not-so efficient push-back for yytext
mungers */

static  unsigned char yy_save [YYLMAX];
static  unsigned char *yy_push = yy_save+YYLMAX;

#define YY_USER { \
          size_t n = yy_end - yyleng; \
          yy_push = yy_save+YYLMAX - n; \
          if (n > 0) \
               memmove(yy_push, yytext+yyleng, n); \
          yytext[yyleng] = 0; \
     }
#define YY_SCANNER { \
          size_t n = yy_save+YYLMAX - yy_push; \
          if (n > 0) \
               memmove(yytext+yyleng, yy_push, n); \
          yy_end = yyleng + n; \
     }
```

```
#endif

int     yy_actype;              /* special action */
int     yylineno = 1;           /* line number */
int     yyleng = 0;             /* yytext token length */
int     yy_end = 0;             /* end of pushback */
int     yy_start = 0;           /* start state */
int     yy_lastc = YYNEWLINE;   /* previous char */

#endif

#if __STDC__
int YYLEX(void) {
#else
int YYLEX()
{
#endif
    register int c, i, st, base;
    int fmin, fmax;                 /* yy_la_act indices of
final states */
    int oldi, oleng;                    /* base i, yyleng
before look-ahead */

    i = yyleng;
    YY_SCANNER;

 start:
    yyleng = i;
    /* determine previous char. */
    if (i > 0)
        yy_lastc = yytext[i-1];
    /* scan previously accepted token adjusting yylineno */
    while (i > 0)
        if (yytext[--i] == YYNEWLINE)
            yylineno++;
    /* adjust pushback */
```

```
    yy_end -= yyleng;
    memmove(yytext, yytext+yyleng, (size_t) yy_end);
    i = 0;

more:
    oldi = i;

    /* run the state machine until it jams */
    for (yy_sbuf[i] = st = yy_begin[yy_start + (yy_lastc ==
YYNEWLINE)];
            !(st == yy_endst || YY_INTERACTIVE && yy_base[st]
> yy_nxtmax && yy_default[st] == yy_endst);
        yy_sbuf[++i] = st) {
        YY_DEBUG("<state %d, i = %d>\n", st, i);
        if (i >= YYLMAX)
            YY_FATAL("Token buffer overflow");

        /* get input char */
        if (i < yy_end)
            c = yy_tbuf[i];         /* get pushback char
*/
        else if ((c = yygetc()) != EOF) {
            yy_end = i+1;
            yy_tbuf[i] = c = (unsigned char) c;
        } else /* c == EOF */ {
            if (i == oldi)  /* no token */
                if (yywrap())
                    return 0;
                else
                    goto start;
            else
                break;
        }
        YY_DEBUG("<input %d = 0x%02x>\n", c, c);

        /* look up next state */
```

```
          while ((base = yy_base[st]+c) > yy_nxtmax ||
yy_check[base] != st) {
               if (st == yy_endst)
                    goto jammed;
               st = yy_default[st];
          }
          st = yy_next[base];
        jammed: ;
     }
     YY_DEBUG("<stopped %d, i = %d>\n", st, i);
     if (st != yy_endst)
          ++i;

  reject:
     /* search backward for a final state */
     while (--i > oldi) {
          st = yy_sbuf[i];
          if ((fmin = yy_final[st]) < (fmax =
yy_final[st+1]))
               goto final;      /* found final state(s) */
     }
     /* no match, default action */
     i = oldi + 1;
     output(yy_tbuf[oldi]);
     goto start;

  final:
     YY_DEBUG("<final state %d, i = %d>\n", st, i);
     oleng = i;                 /* save length for REJECT */

     /* pushback look-ahead RHS */
     if ((c = (int)(yy_la_act[fmin]>>9) - 1) >= 0) { /*
trailing context? */
          unsigned char *bv = yy_look + c*YY_LA_SIZE;
          static unsigned char bits [8] = {
               1<<0, 1<<1, 1<<2, 1<<3, 1<<4, 1<<5, 1<<6,
```

```
1<<7
            };
            while (i > oldi) {
                 st = yy_sbuf[--i];
                 if    ( bv[ (unsigned) st / 8 ]    &
bits[(unsigned)st%8])
                      break;
            }
      }

      /* perform action */
      yyleng = i;
      YY_USER;
      c = yy_action(yy_la_act[fmin] & 0777);
      YY_SCANNER;
      i = yyleng;

      /* process special actions */
      YY_DEBUG("<actype %d, return %d>\n", yy_actype, c);
      switch (yy_actype) {
          case YY_RETURN:              /* action did return */
           YY_USER;
           return c;

          case YY_SKIP:                /* action fell through
*/
           goto start;

        case YY_REJECT:                /* REJECT */
           i = oleng;                /* restore original yytext
*/
           if (++fmin < fmax)
                goto final;     /* another final state, same
length */
           else
                goto reject;     /* try shorter yytext */
```

```
        case YY_MORE:                   /* yymore() */
            if (i > 0)
                yy_lastc = yytext[i-1];
            goto more;
        }
    }

    #ifndef YY_NOSCANNER

    /*
     * user callable routines
     */

    /* get input char with pushback */
    int input()
    {
        int c;
    #ifndef YY_PRESERVE
        if (yy_end > yyleng) {
            yy_end--;
            memmove(yytext+yyleng, yytext+yyleng+1,
                (size_t) (yy_end-yyleng));
            c = yy_save;
            YY_USER;
    #else
        if (yy_push < yy_save+YYLMAX) {
            c = *yy_push++;
    #endif
        } else
            c = yygetc();
        yy_lastc = c;
        if (c == YYNEWLINE)
            yylineno++;
        return c;
    }
```

```
/* pushback char */
int unput(c)
     int c;
{
#ifndef YY_PRESERVE
     if (yy_end >= YYLMAX)
          YY_FATAL("Push-back buffer overflow");
     if (yy_end > yyleng) {
          yytext[yyleng] = yy_save;
          memmove(yytext+yyleng+1, yytext+yyleng,
               (size_t) (yy_end-yyleng));
          yytext[yyleng] = 0;
     }
     yy_end++;
     yy_save = c;
#else
     if (yy_push <= yy_save)
          YY_FATAL("Push-back buffer overflow");
     *--yy_push = c;
#endif
     if (c == YYNEWLINE)
          yylineno--;
     return c;
}

/* accept first `n' chars of token */
void yyless(int n)
{
     if (n < 0 || n > yy_end)
          return;
     YY_SCANNER;
     yyleng = n;                         /* adjust pushback */
     YY_USER;
}

#endif
```

```
#line 156 "polinom2.lex"

yywrap(){return 1;}

void multiplica2(int na, int nb, int *nc,float a[], float
b[], float c[]);
void main(int argc,char *argv[]) :
{
 /* (coeficiente*s^exponente...)^potencia */
 int k,p,q;
 int m;
 int g1,g2,g3,g4,gn,gd;
  float aux1[20],aux2[20],aux3[20],aux4[20],num[20],den[20];
 int npden;
 FILE *salida;

 if(argc!=3)
 {
      printf("error en numero de argumentos\npolinom2.exe
archin archout");
   exit(1);
 }

 if((yyin=fopen(argv[1],"r"))==NULL)
 {
      printf("error:no se puede abrir el archivo de entrada");
   exit(1);
 }
yylex();
fclose(yyin);

if((salida=fopen(argv[2],"w"))==NULL)
{
  printf("error:no se puede abrir el archivo de salida");
  exit(1);
}
```

```
  if( mierror!=0 )
    fprintf(salida,"0 1 0 1");
  else
  {
    npden=npol-npnum-1;

    /* EXPANSION DEL NUMERADOR */
    g1=grado[0];
    for( p=0;p<=g1;p++) aux1[p]=0;
    for( p=ncoefpol[0];p<ncoefpol[1];p++)

aux1[g1-exponente[p]]=aux1[g1-exponente[p]]+coeficiente1[p];
    gn=g1;
    for( p=0;p<=gn;p++) num[p]=aux1[p];

    if( potencia[0]>=2 )
    {
      for( p=1;p<potencia[0];p++)
      {
        multiplica2(g1,gn,&g3,aux1,num,aux3);
        gn=g3;
        for( q=0;q<=gn;q++) num[q]=aux3[q];
      }
    }
    if( npnum>=1 )
    {
      for( k=1;k<=npnum;k++)
      {
        g1=grado[k];
        for( p=0;p<=g1;p++) aux1[p]=0;
        for( p=ncoefpol[k];p<ncoefpol[k+1];p++)

aux1[g1-exponente[p]]=aux1[g1-exponente[p]]+coeficiente1[p];
          g4=g1;
        for( p=0;p<=g4;p++) aux4[p]=aux1[p];
        if( potencia[k]>=2 )
```

```
        {
       for( p=1;p<potencia[k];p++)
       {
         multiplica2(g1,g4,&g3,aux1,aux4,aux3);
         g4=g3;
         for( q=0;q<=g4;q++) aux4[q]=aux3[q];
       }
        }
        multiplica2(gn,g4,&g2,num,aux4,aux2);
        gn=g2;
        for( p=0;p<=gn;p++) num[p]=aux2[p];
     }
   }

   /* EXPANSION DEL DENOMINADOR */
   if( npden>0 )
   {
     g1=grado[npnum+1];
     for( p=0;p<=g1;p++) aux1[p]=0;
     for( p=ncoefpol[npnum+1];p<ncoefpol[npnum+2];p++)

aux1[g1-exponente[p]]=aux1[g1-exponente[p]]+coeficiente1[p];
     gd=g1;
     for( p=0;p<=gd;p++) den[p]=aux1[p];
     if( potencia[npnum+1]>=2 )
     {
       for( p=1;p<potencia[npnum+1];p++)
       {
      multiplica2(g1,gd,&g3,aux1,den,aux3);
      gd=g3;
      for( q=0;q<=gd;q++) den[q]=aux3[q];
       }
     }

     if( npden>=1 )
     {
```

```
        for( k=npnum+2;k<=npol;k++)
        {
       g1=grado[k];
       for( p=0;p<=g1;p++) aux1[p]=0;
       for( p=ncoefpol[k];p<ncoefpol[k+1];p++)

aux1[g1-exponente[p]]=aux1[g1-exponente[p]]+coeficiente1[p];
       g4=g1;
       for( p=0;p<=g4;p++) aux4[p]=aux1[p];
       if( potencia[k]>=2 )
       {
         for( p=1;p<potencia[k];p++)
         {
           multiplica2(g1,g4,&g3,aux1,aux4,aux3);
           g4=g3;
           for( q=0;q<=g4;q++) aux4[q]=aux3[q];
         }
       }
       multiplica2(gd,g4,&g2,den,aux4,aux2);
       gd=g2;
       for( p=0;p<=gd;p++) den[p]=aux2[p];
        }
     }
   }
   else
   {
     gd=0;
     den[0]=1;
   }

   fprintf(salida,"%d",gd);
   for (p=0;p<=gd;p++)
     fprintf(salida," %f",den[p]);
   fprintf(salida," %d",gn);
   for (p=0;p<=gn;p++)
     fprintf(salida," %f",num[p]);
```

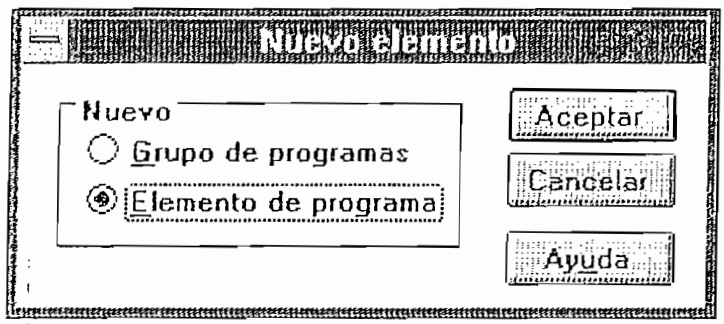
```
  }
  fclose(salida);
}

void multiplica2(int na, int nb, int *nc, float a[], float
b[], float c[])
{
  int i,j;

  *nc=na+nb;
  for(i=0;i<=na+nb;i++) c[i]=0;
  for(i=0;i<=na;i++)
  {
    for(j=0;j<=nb;j++)
      c[i+j]=c[i+j]+a[i]*b[j];
  }
}
```

ANEXO B
MANUAL DEL USUARIO

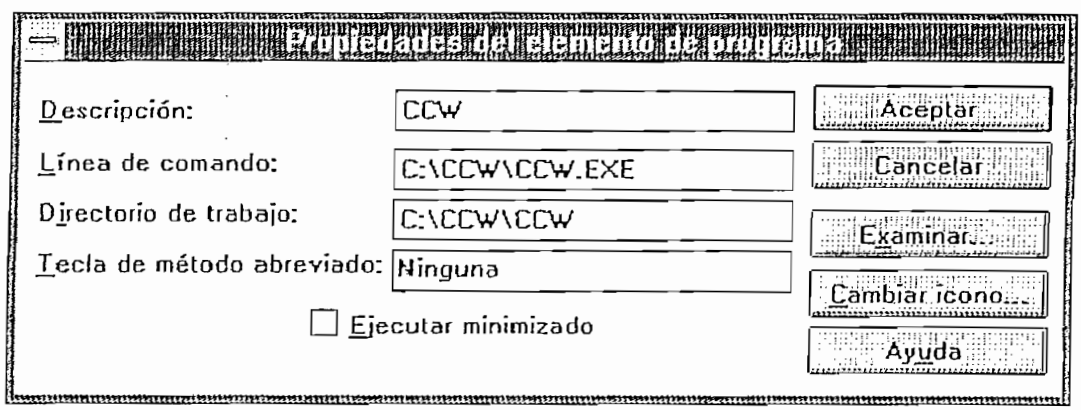
1. MANUAL DE USO DEL PROGRAMA

Para la instalación del programa se crea un directorio (usar el comando MD del DOS o el administrador de archivos de Windows), por ejemplo CCW, y se copia todos los archivos a este directorio, estos archivos vienen comprimidos en el diskette, para proceder a descomprimirlos y copiarlos al directorio creado, se tipea en el DOS a:instalar a: c:.

Se ejecuta Windows, por medio del comando "win"; en el administrador de programas, se escoge el comando "nuevo" del menú "Archivo", se presenta una caja de dialogo como se muestra en la siguiente figura.



Si desea aumentar el programa a un grupo ya existente, se escoge la opción "Elemento de Programa", y se presiona el botón "Aceptar"; A continuación se presenta la siguiente caja de dialogo, que puede ser llenada como se indica a continuación.



Si todo se ha realizado correctamente, se creará en el grupo de programas un ícono que representa al programa.

Para iniciar el programa hacemos un doble click sobre el ícono del programa. Se desplegará una pantalla como se muestra en la figura A.1.

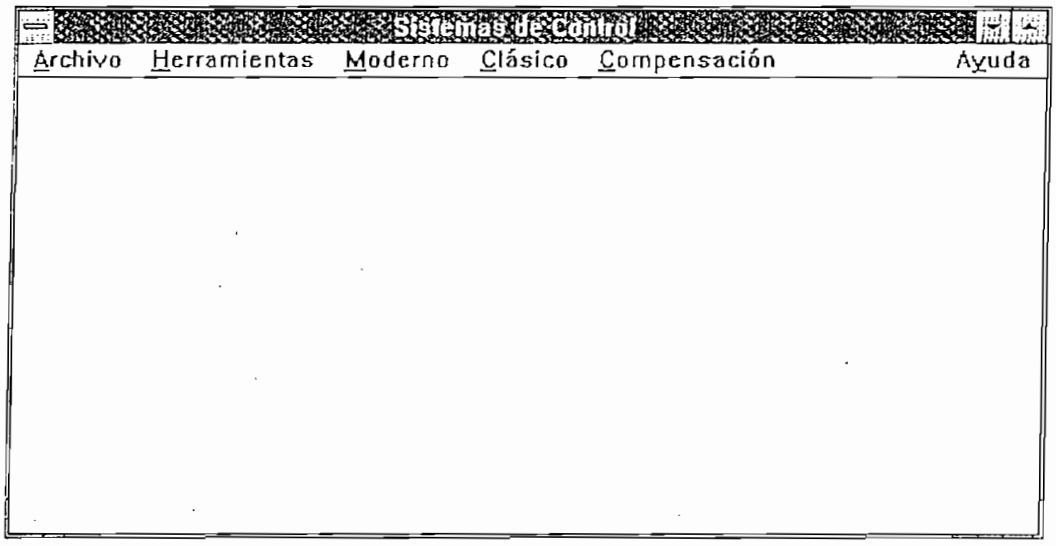


Figura A.1

Para que la aplicación ocupe toda la pantalla haga un click sobre el botón maximizar (cuyo símbolo es ▲, ubicado en la esquina superior derecha), hecho esto el botón maximizar se transformará en el botón restaurar (cuyo símbolo es ♦); toda la aplicación puede ser transformada a un ícono, para lograr esto use el botón minimizar (cuyo símbolo es ▼). Si la aplicación se encuentra minimizada puede restaurarla haciendo un doble click sobre el ícono.

La barra de título nos muestra el nombre "sistemas de control", esta a más de mostrar el título de la aplicación, permite mover la ventana (cuando no se encuentra maximizada); para lograr esto coloque el mouse sobre la

barra del título y mantenga presionado el botón izquierdo del mouse mientras lo mueve a una posición más adecuada.

Se puede fijar la altura y ancho de la ventana, para esto ubique el mouse en el limite de la ventana, y cuando el mouse cambie su forma y se forme una doble flecha, presione el botón izquierdo y muévalo con esto el tamaño de la ventana cambiará.

Este manual analiza las opciones que se encuentran en la barra de menús.

1) **Archivo**. Este menú contiene comandos para el ingreso, edición, grabación, y recuperación de funciones de transferencia o, de las matrices que describen las variables de estado. Al desplegar este menú la pantalla se muestra como en la figura A.2.

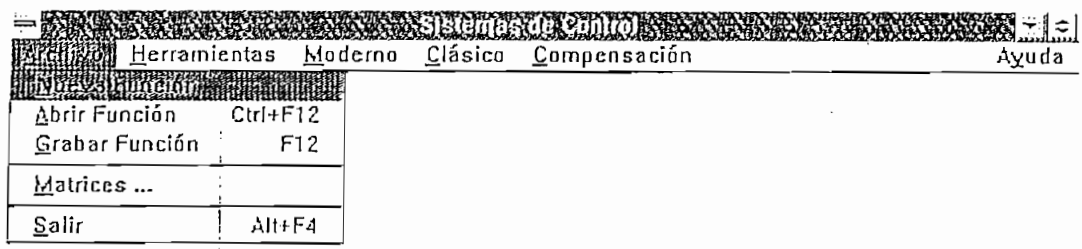


Figura A.2

El comando Nueva Función permite el ingreso de nuevas funciones de transferencia. Si se escoge este comando se presenta el siguiente cuadro de diálogo, en el cual debe ser ingresada la función de transferencia, un ejemplo es mostrado en la figura A.3.

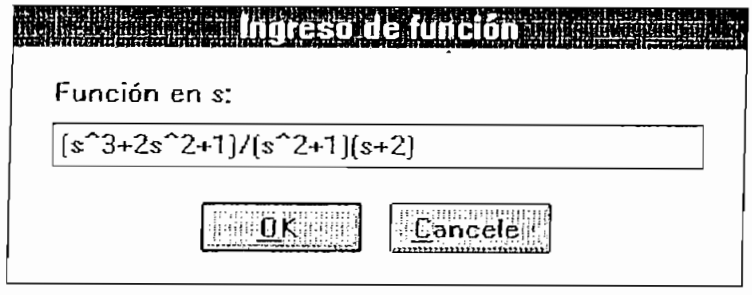


Figura A.3

Con esta función se ha desarrollado los ejemplos tratados en el menú clásico.

Si presiona el botón OK se presenta el cuadro de diálogo que muestra la figura A.4, en el que se indica la función de transferencia ingresada en forma desarrollada, si existe un error en el ingreso de la función de transferencia, la función ingresada será 1/1. Esta nueva función substituirá a la función que haya existido previamente.

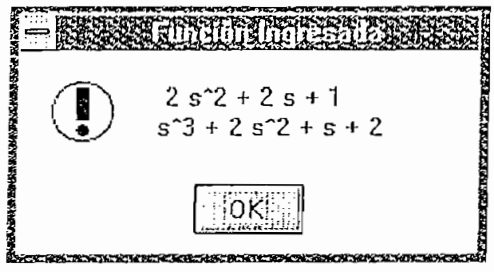


Figura A.4

y, si presiona Cancele se presenta el mensaje que indica la figura A.5.

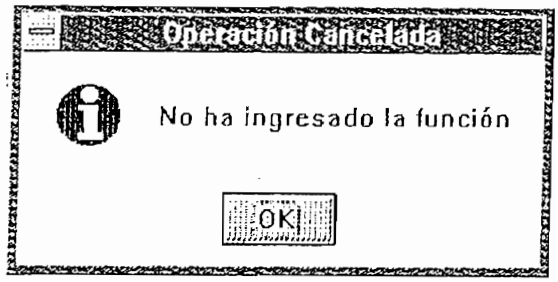


Figura A.5

Al presionar cancele, no existe cambio en la función de transferencia que estaba usando previamente, pero si usamos nuevamente el comando nuevo, la función permanecerá lista para que realice cualquier cambio.

Una vez que ha ingresado una nueva función la puede grabar para usarla en una próxima ocasión, el comando que permite hacer esto es el comando grabar (puede usar la tecla aceleradora F12); a continuación se le pedirá que ingrese el nombre del archivo en el que va a grabar la función, si no pone la extensión en el nombre de archivo el programa añadirá automáticamente la extensión "FT" al nombre del archivo. La caja de diálogo se muestra como en la figura A.6.

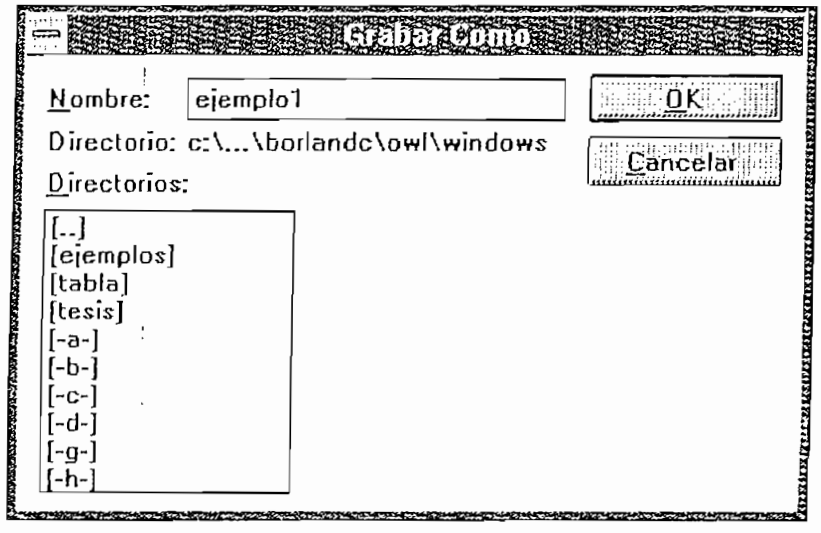


Figura A.6

Si desea que el programa sea grabado en una directorio distinto, al cual se encuentra el programa, haga un doble click sobre el directorio deseado, o escriba la vía de acceso al directorio deseado; este programa no permite la creación de directorios, para crear un nuevo directorio use otro programa, como el administrador de archivos de Windows, o el comando MD del DOS.

Inicialmente el botón OK no se encuentra iluminado; este se iluminará (indicando que puede usarlo), cuando ingrese un nombre para el archivo que contiene la función de transferencia.

Si presiona el botón OK, se procederá a grabar la función con el nombre del archivo que ha ingresado; se presenta un mensaje que le indica el nombre del archivo y el directorio en el cual la función ha sido grabada.

Figura A.7

En caso de no haber ingresado una función el programa muestra el siguiente mensaje

Figura A.8

En caso de presionar el botón cancelar se presenta el mensaje que indica la figura A.9, este mensaje es independiente de que exista o no una función de transferencia.

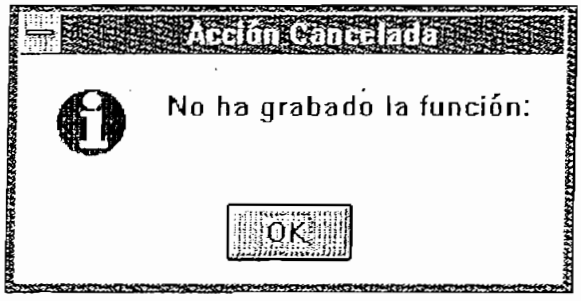


Figura A.9

El programa no verifica si existe el nombre de la función, por lo que si usa un nombre existente borrará la primera función.

Para recuperar funciones previamente grabadas, usamos el comando Abrir Función, entonces se presenta el cuadro de dialogo que muestra la figura A.10:

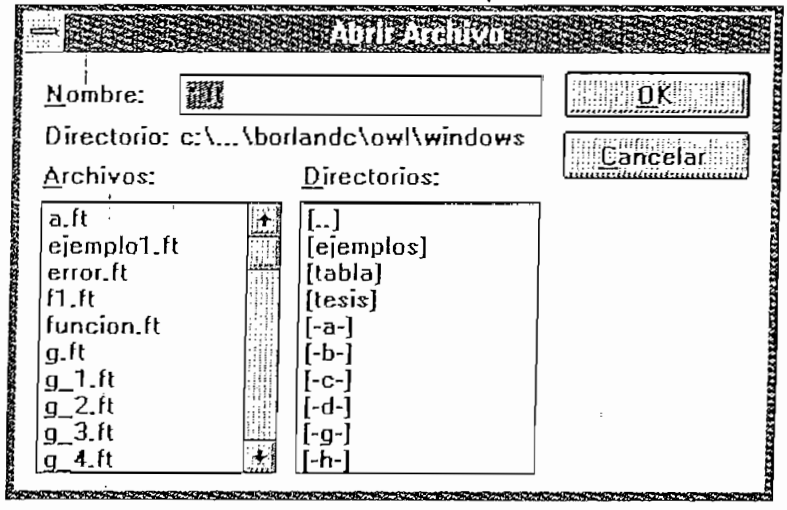


Figura A.10

En el cuadro correspondiente a archivos se presenta una lista de las funciones existentes en un directorio (puede usar el Scroll Bar para buscar una función) clasificadas en orden alfabético, si la función deseada se encuentra en un directorio diferente haga un doble click sobre el directorio deseado.

En la casilla nombre puede escribir el nombre de la función que desea recuperar, o iluminar el nombre de la función; para recuperar la función presione el botón OK a continuación se muestra una caja de dialogo que contiene la función contenida en el archivo, como se muestra en la figura A.11.

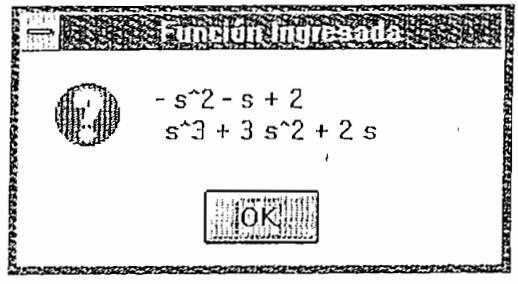


Figura A.11

Si escoge el botón cancelar el mensaje que se muestra es el siguiente:

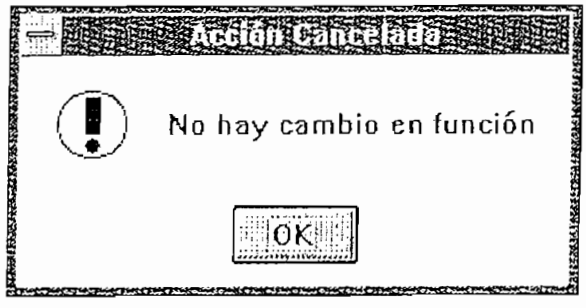


Figura A.12

Para salir del programa podemos presionar ALT+F4 o en el menú de archivos escoger el comando salir; en cualquier caso se le pedirá verificación con el cuadro mostrado en la figura A. 13.

Figura A.13

Si presionamos enter o el botón Yes, se terminará el programa; si presiona el botón No, retornará al programa.

La última función que hemos usado permanecerá como activa cuando iniciemos el programa nuevamente, aún en el caso que no la hayamos grabado.

2) **Herramientas.** Este menú contiene tres comandos que le ayudarán en su trabajo como son: una calculadora y un reloj; además presenta un cuadro de información general del programa.

El comando calculadora presenta la calculadora que muestra la figura A.14.

Esta es la calculadora que viene incluida en el programa Windows, el reloj también corresponde al reloj de Windows.

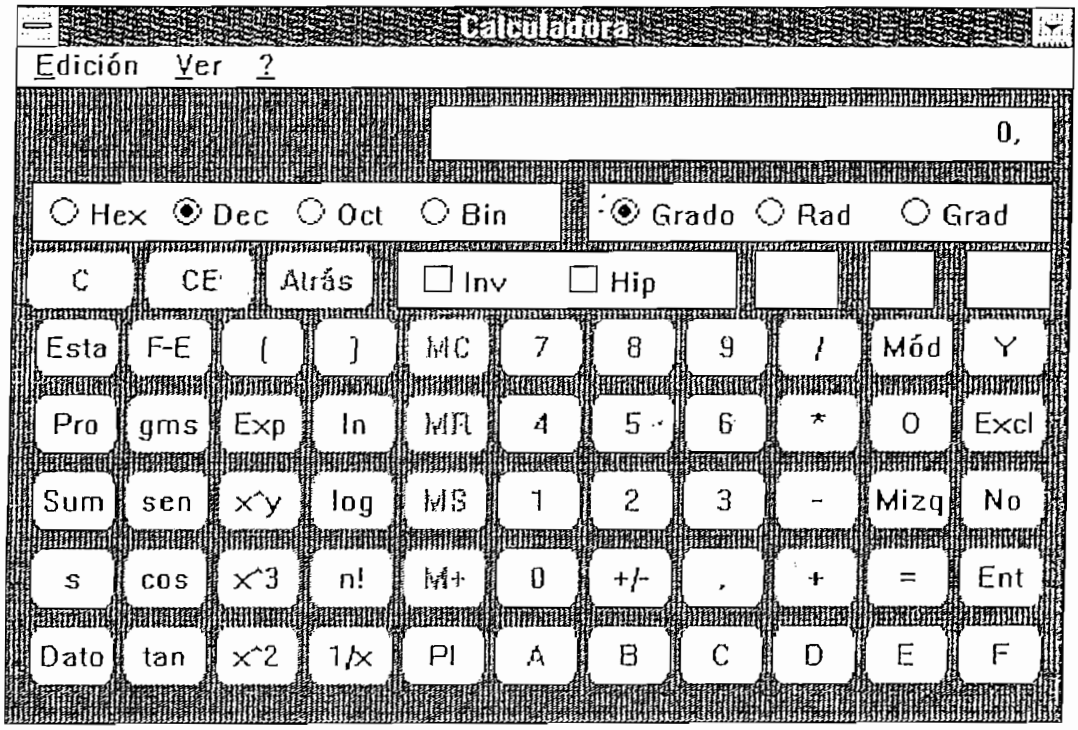


Figura A.14

El comando información nos muestra el siguiente mapa de bits:

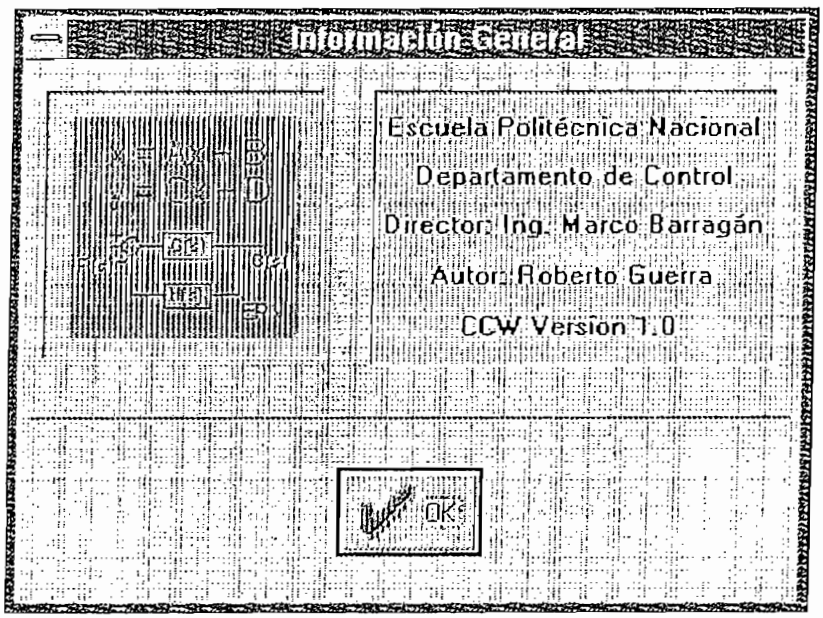


Figura A.15

3) Moderno. Contiene los comandos que permiten el análisis de sistemas descritos por variables de estado si despliega este menú observará la siguiente pantalla.

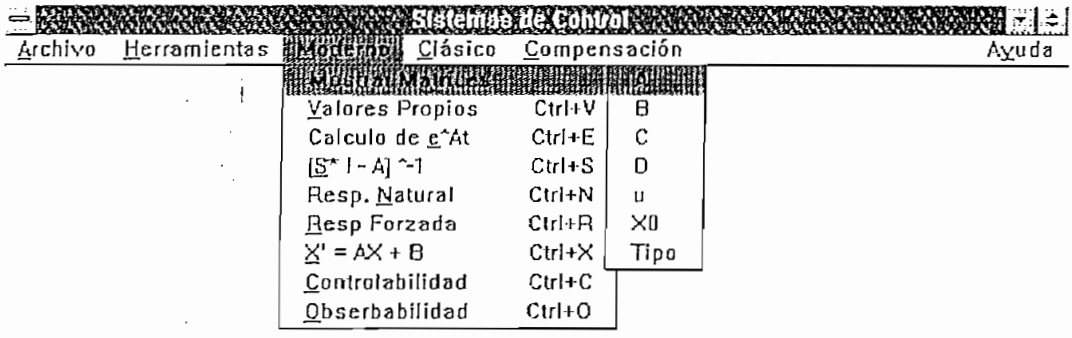


Figura A.16

Si un sistema esta descrito por variables de estado tiene la siguiente forma:

$$\dot{x} = A x + Bu$$
$$y = C x + Du$$

donde A = matriz n x n
B = matriz de n x r
C = matriz de m x n
D = matriz de m x r
x = vector de estado (vector de dimensión n)
u = vector de control (vector de dimensión m)
y = vector de salida (vector de dimensión m)

Si se va a ingresar un nuevo sistema descrito por variables de estado despliegue el menú de archivos y escoja la opción matrices, con esto se desplegará un editor de texto en el cual puede ingresar las matrices.

Para el ingreso de estas matrices se debe introducir primero los coeficientes n, r, m; luego los coeficientes de la matrices A, B, C, D por filas, a continuación se introduce las m entradas, los coeficientes correspondientes a las n condiciones iniciales del sistema, por último se ingresan los coeficientes correspondientes al tipo de entrada; donde 1 corresponde a una entrada paso, 2 para la rampa y 0 para la entrada impulso.

Para verificar si las entradas son las correctas despliegue los comandos contenidos en el menú moderno, dentro de este en el menú mostrar, aquí se permiten mostrar las matrices, los tipos de entradas, y las condiciones iniciales. Si existe un numero menor de entradas que las requeridas las entradas serán tomadas como cero.

3) Clásico. Este menú contiene comandos para analizar sistemas descritos por funciones de transferencia. Escogiendo este menú se desplegará la pantalla que se muestra en la figura A.17.

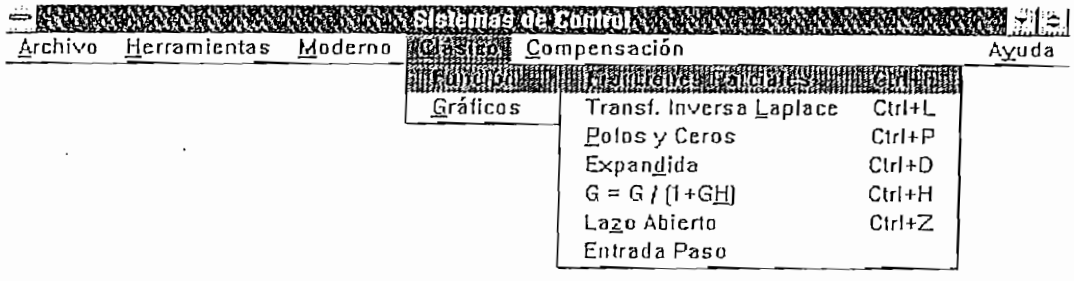


Figura A.17

La opción función permite presentar la función de transferencia en las formas más usuales usadas por los ingenieros en control, cada uno de estos comandos los tratamos a continuación.

El resultado del comando fracciones parciales se indica en la figura A.18, esta forma de presentación permite obtener fácilmente la transformada inversa de Laplace de la función de transferencia.

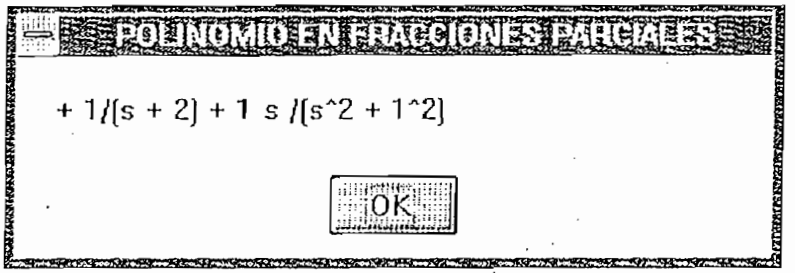


Figura A.18

La transformada inversa de laplace se indica en la figura A.19:

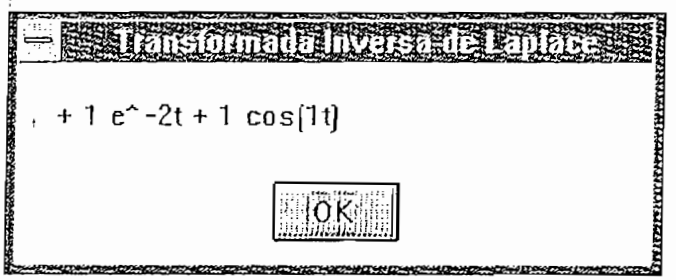


Figura A.19

El comando polos y ceros, presenta la función de transferencia como un producto de polinomios, de primer y segundo grado, los polinomios se indican en forma tal que sea sencillo obtener los polos y ceros de la función de transferencia. El resultado de este comando se muestra en la

figura A.20.

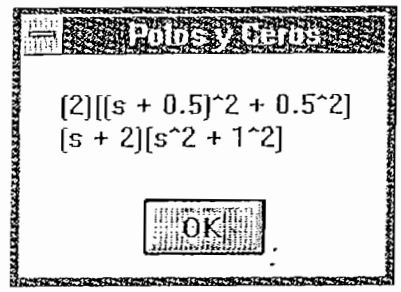


Figura A.20

El comando expandida multiplica todos los factores de la función de transferencia para producir un polinomio en el numerador y otro en el denominador, produce el resultado que se muestra en la figura A.21.

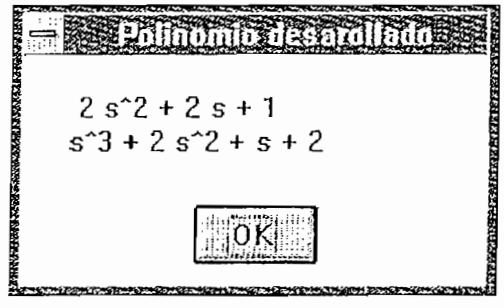


Figura A.21

El comando lazo cerrado transforma la función de transferencia G a G/[1+G], el resultado se indica en la figura A.22.

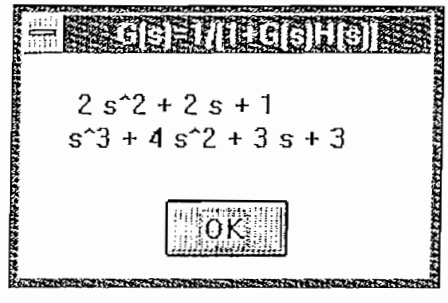


Figura A.22

Si ha usado el comando lazo cerrado o entrada paso y quiere reconstruir la función original use la opción lazo abierto el resultado es el siguiente:

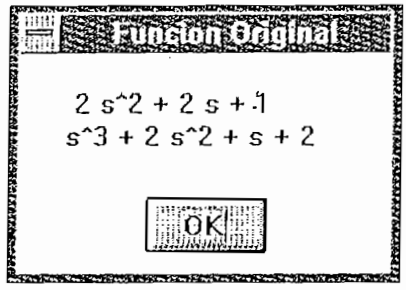


Figura A.23

Puede multiplicar la función por una entrada paso, el resultado se indica en la figura A.24.

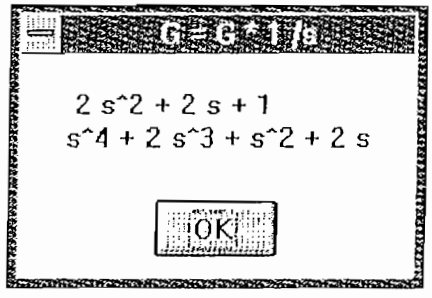


Figura A. 24

Para obtener los diferentes gráficos abra el menú de control clásico y use la opción gráficos la pantalla se

presenta como se muestra en la figura A.25.

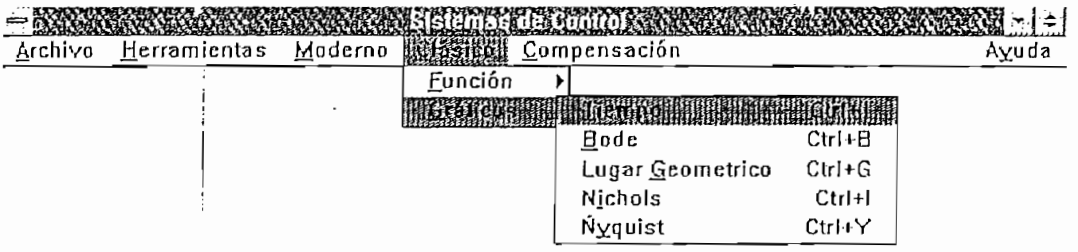


Figura A. 25

El comando tiempo se usa para obtener la respuesta enel

tiempo, presenta el siguiente cuadro de dialogo

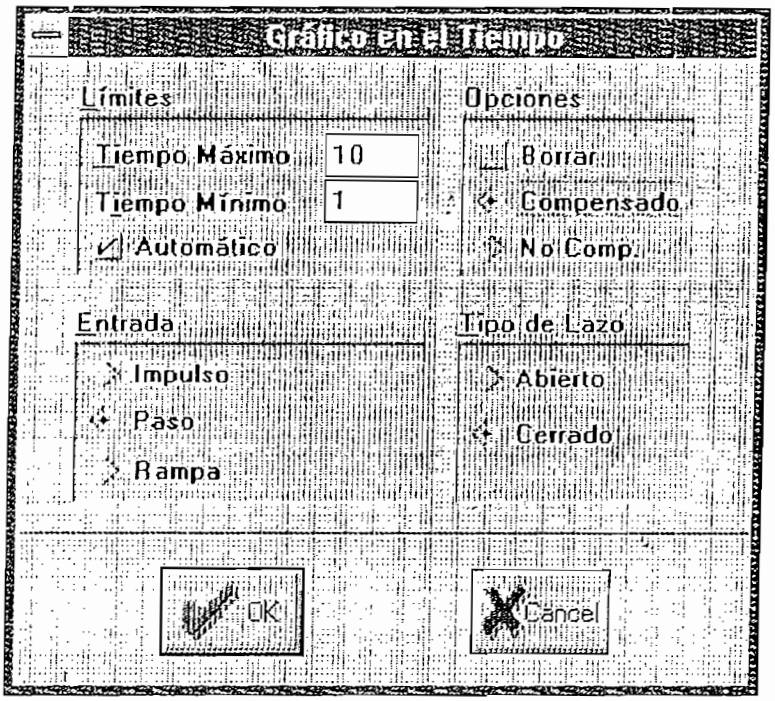


Figura A.26

Este cuadro permite elegir el tipo de entrada al sistema (paso, impulso, rampa), la respuesta en lazo abierto o cerrado, si el sistema es compensado o no, y la escala automática o manual, si eligió escala manual debe ingresar los limites, en caso de ingresar número erróneo para tiempo mínimo, el tiempo mínimo será 0, si el tiempo máximo es erróneo (menor que cero por ejemplo) se tomará el tiempo dado automáticamente.

Si elige el botón de OK se presentará en la pantalla principal el un gráfico como el de la figura A.27.

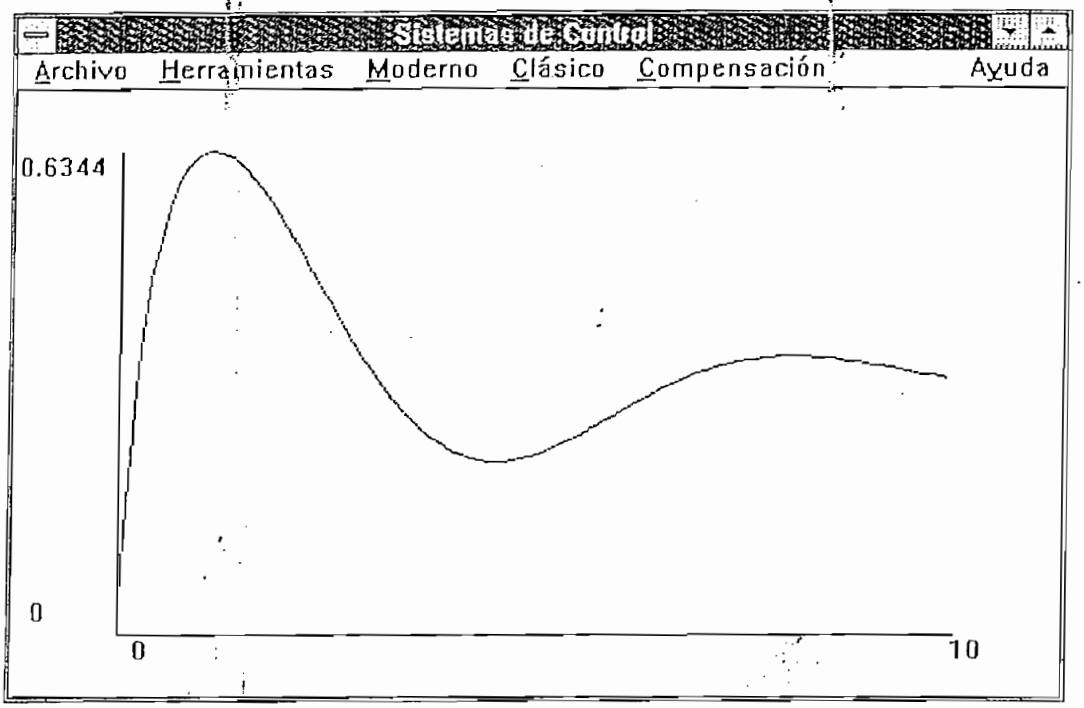


Figura A.27

Si presiona el botón cancelar no se hará ningún gráfico.

Las opciones indicadas en la cuadro de dialogo anterior se mantienen hasta que se termine el programa o ingrese otras.

Se puede señalar los puntos en la pantalla presionando el botón izquierdo del mouse; para trazar líneas hacia los ejes se presiona la tecla control y se presiona el botón izquierdo del mouse, esto es muy útil para trazar puntos de interés como máximo sobreimpulso, tiempo de establecimiento, etc.

Para introducir texto en la pantalla gráfica se presiona el botón derecho del mouse se presenta el cuadro de dialogo que se muestra en la figura A.28.

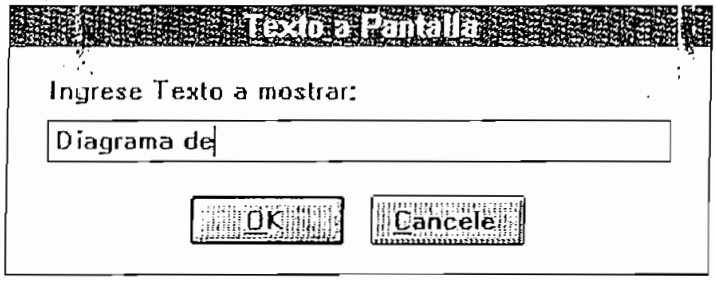


Figura A.28

El texto introducido será mostrado en la pantalla en la posición en que se ubico el mouse.

Como ejemplo de los 2 últimos procedimiento se presenta el siguiente diagrama:

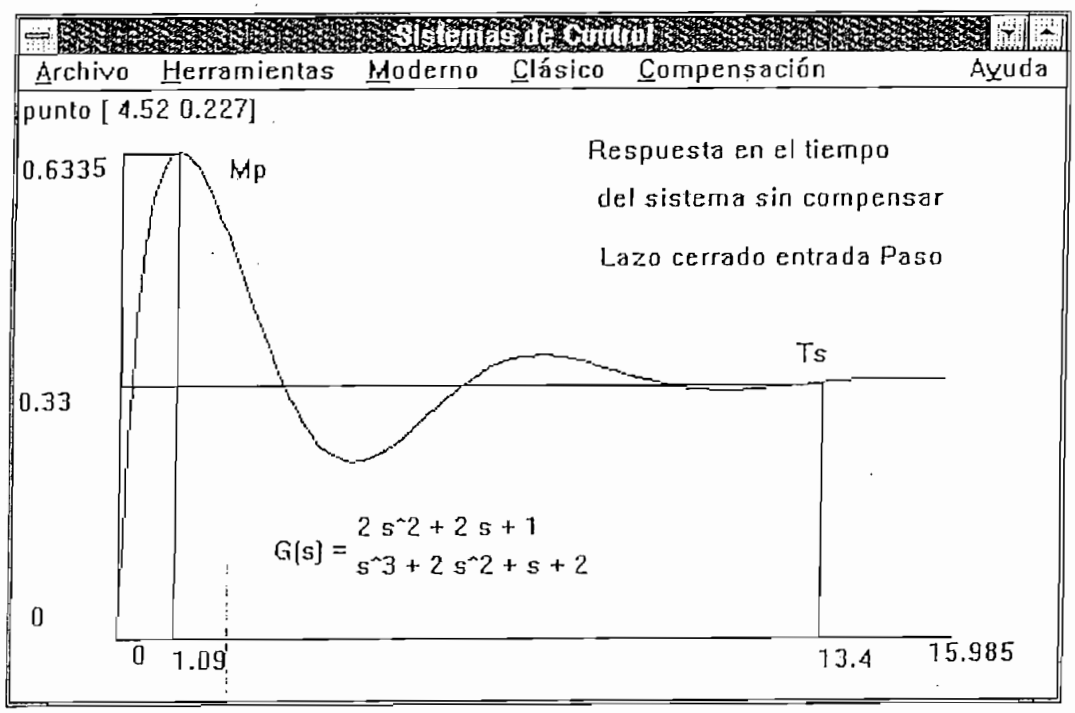


Figura A.29

Cada punto en el gráfico es mostrado en la esquina superior izquierda con sólo presionar el botón izquierdo del mouse en el punto deseado.

Para analizar la respuesta en frecuencia del sistema, se puede analizar los diagramas de Bode, Nichols, Nyquist.

Si desea trazar el diagrama de Bode presione CTRL+B, se presenta el siguiente cuadro de dialogo

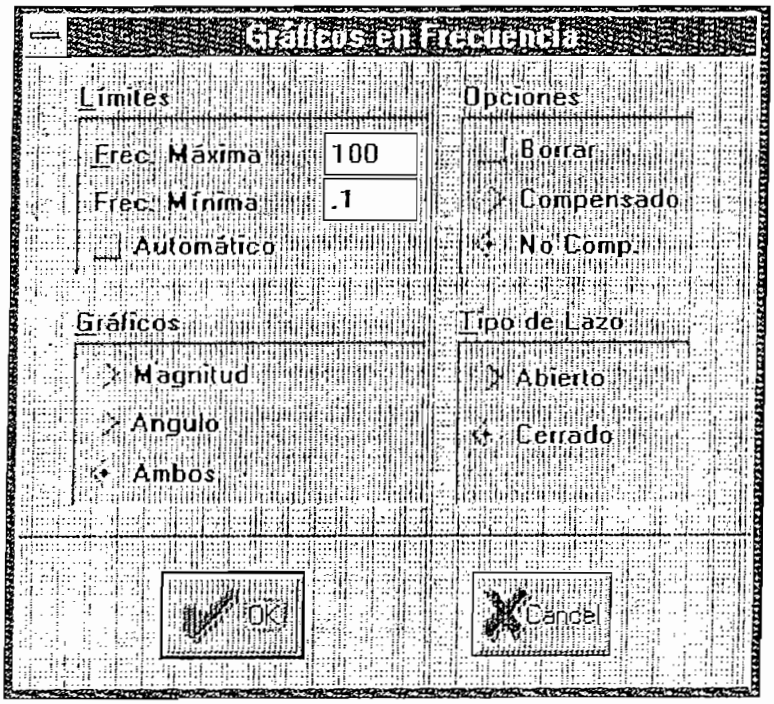


Figura A.30

Se obtiene una respuesta como la siguiente, el ángulo es dibujado con línea roja

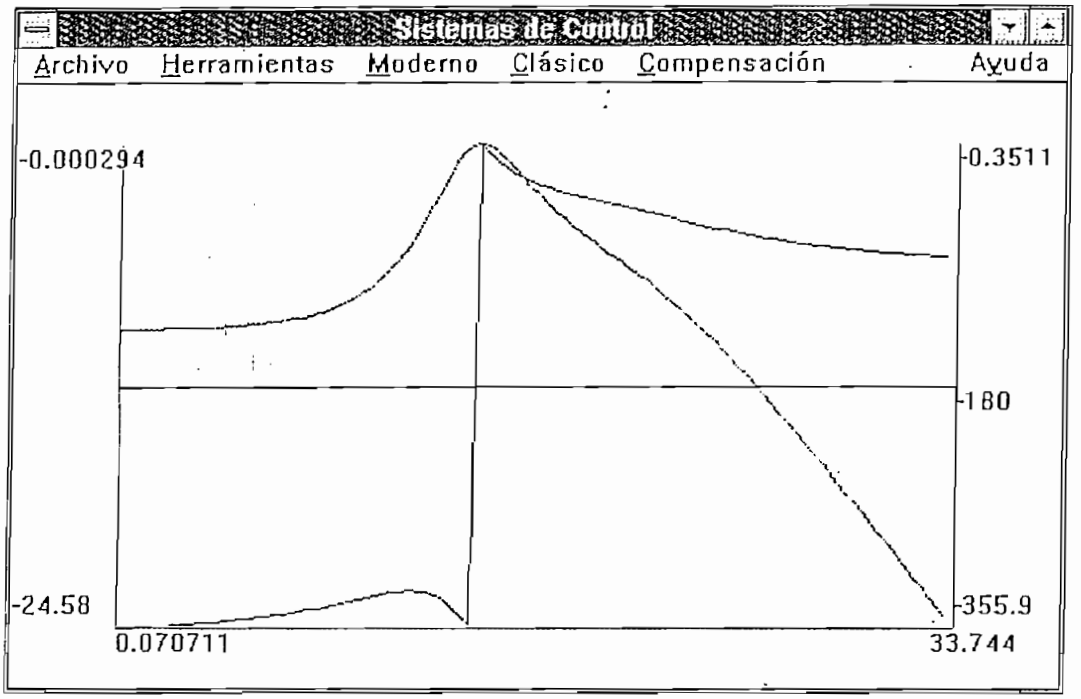


Figura A.31

El diagrama de bode representa el ángulo en color rojo y el modulo en negro.

Se traza una línea en 180 grados y en 0 db para facilitar el trazado del margen de fase y margen de ganancia.

El escala para el modulo se encuentra en la parte izquierda de la pantalla, y para el ángulo en la parte derecha.

El diagrama de Nichols se lo obtiene presionando CTRL+I del ejemplo se obtiene.

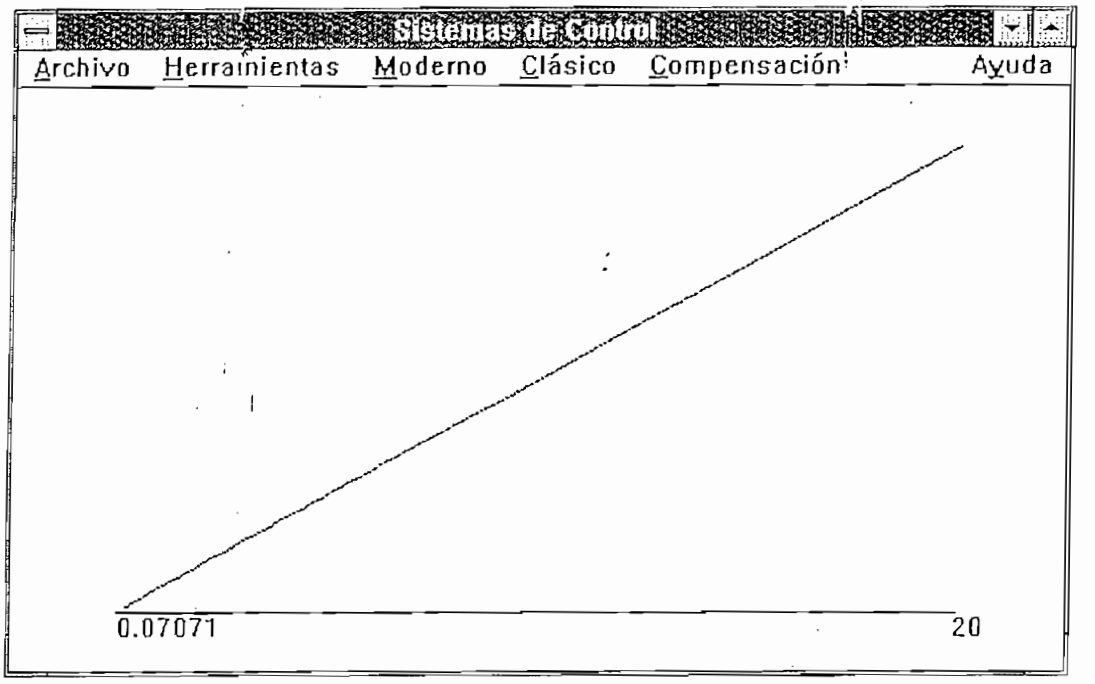


Figura A.32

El diagrama de Nyquist se lo obtiene presionando CTRL+Y.

Para el ingreso de las matrices que describen un sistema por variables de estado, use el comando matrices, que se encuentra en el menú archivo, este comando desplegará un editor de texto como se muestra como en la figura A.34.

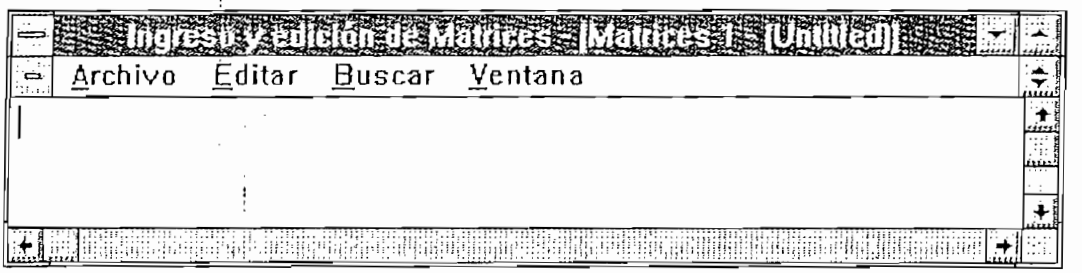


Figura A.34

Este editor de texto permite una fácil ingreso y edición de las matrices, puede marcar un segmento de texto, usando el mouse, y moverlo borrarlo o copiarlo, para copiar el texto marcado presione CTRL+INS, y SHIFT+INS en el punto en que desea copiar el texto; para borrar el texto marcado presione SHIFT+DEL; para restaurar el texto borrado presione SHIFT+INS.

Dentro del editor de texto el menú archivo presenta los comandos Nueva, Abrir, grabar, grabar como, y salir, el comando grabar permite grabar la matriz, y el comando grabar como le permite que Ud. entre el nuevo nombre del archivo a grabar. Para que un archivo sea el activo lo grabamos con el nombre "matriz", para que sea mas fácil su ubicación grabamos con cualquier nombre y la extensión "MTZ".

Usted puede cargar diversas matrices en el editor de texto y mostrarlas en dos formas en forma de mosaico (figura A.35) y en forma de cascada (figura A.36) para acceder a estas opciones despliegue el menú de ventanas, a más de lo dicho anteriormente se permite cerrar todas las ventanas abiertas. Estos comandos se hallan en el menú ventana.

En forma de mosaico.

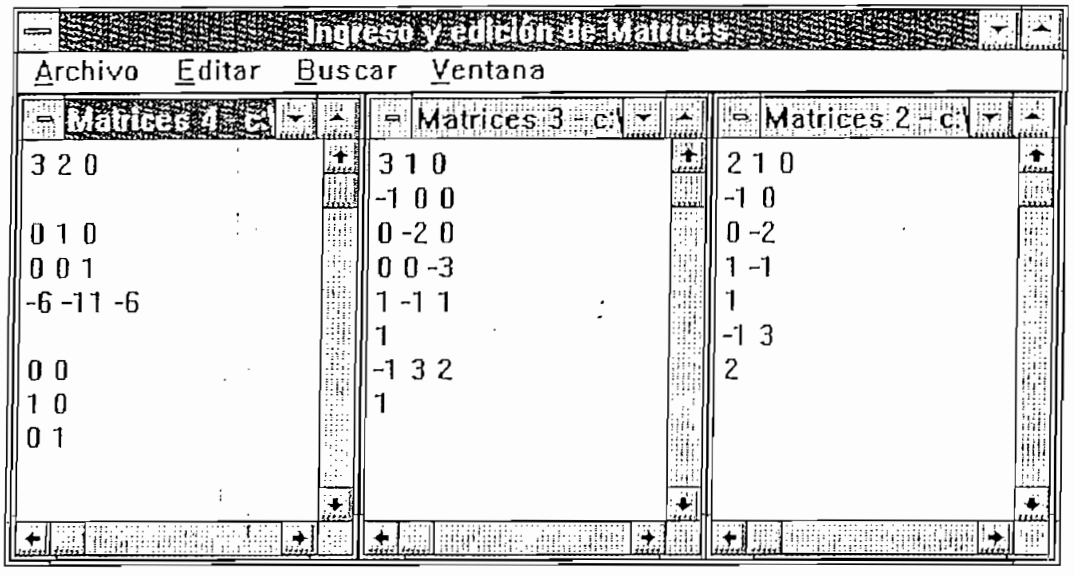


Figura A.35

En forma de cascada

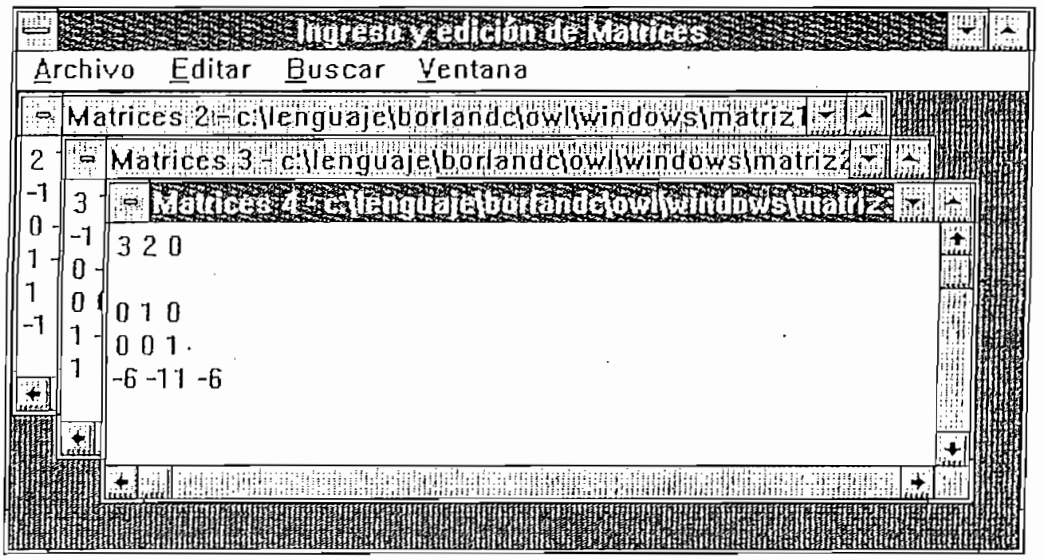


Figura A.36

Para moverse entre las distintas ventanas abiertas puede presionar CTRL+F6, para cerrar una ventana asegúrese que esta sea la activa, si no ha realizado cambios se cerrara la ventana, en caso de que exista un cambio se presentara el siguiente mensaje

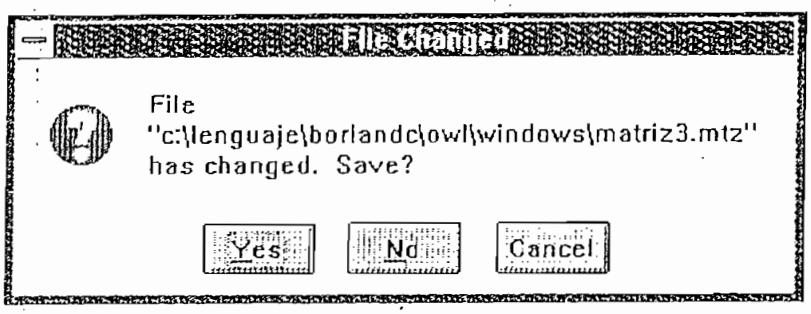


Figura A.37

Este editor de texto también sirve para ver los resultados que no alcanzan en la pantalla o editarlos.

Si el nombre que ingresa en el comando "grabar como", ya existe se presenta el mensaje

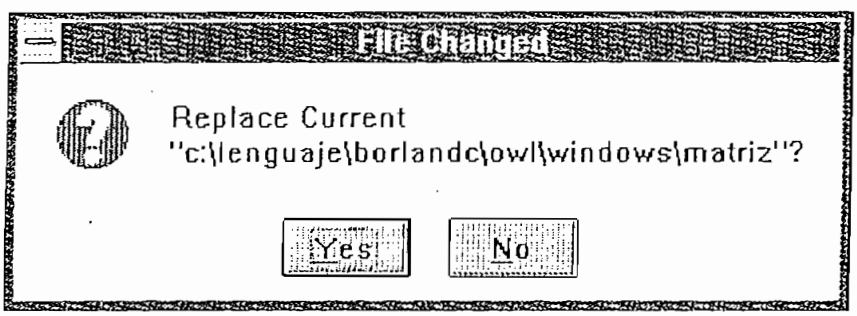


Figura A.38

Para el análisis de sistemas descritos por variables de estado, el menú moderno tiene las siguientes opciones:

- Mostrar matriz presenta las matrices A, B, C, D, las condiciones iniciales, las entradas, y los tipos de entradas.

Para el sistema siguiente:

$$\begin{vmatrix} x_1 \\ x_2 \end{vmatrix} = \begin{vmatrix} -1 & 0 \\ 0 & -2 \end{vmatrix} \begin{vmatrix} x_1 \\ x_2 \end{vmatrix} + \begin{vmatrix} -1 \\ 1 \end{vmatrix} [u]$$

donde u es una entrada paso, y las condiciones iniciales son -1, 3; para ingresar este sistema en el editor de texto debe ingresar.

```
2 1 0

-1 0
0 -2

1
-1

1

-1 3

1
```

En mostrar la matriz A se tiene lo siguiente

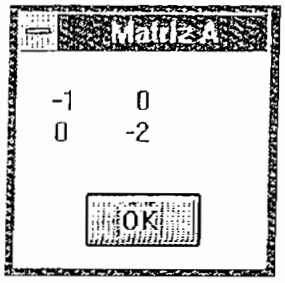


Figura A.39

La matriz B es

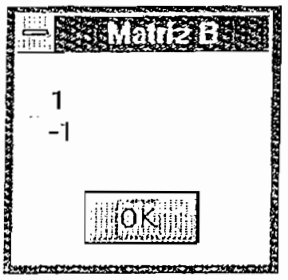


Figura A.40

La matriz C es

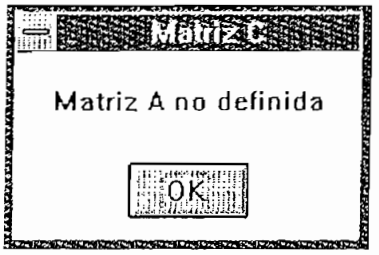


Figura A.41

Igual mensaje se tiene con la matriz D

La magnitud de la entrada está dada por

Figura A.42

Y el tipo de entrada se muestra como

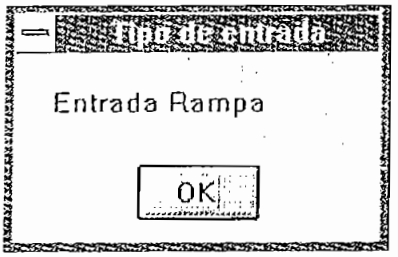


Figura A.43

Las condiciones iniciales se las obtiene con el comando XO dentro del menú mostrar matriz, el resultado se muestra así:

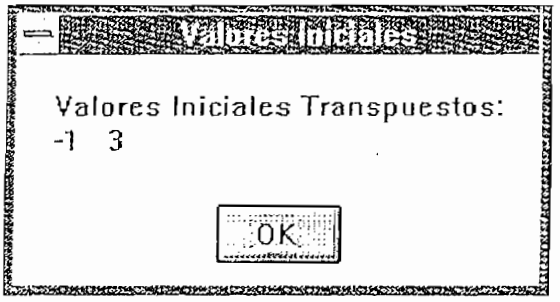


Figura A.44

- $(SI - A)^{-1}$ presenta la inversa de la matriz (sI - A) donde s es el operador de Laplace, I es la matriz identidad para el ejemplo ingresado tenemos.

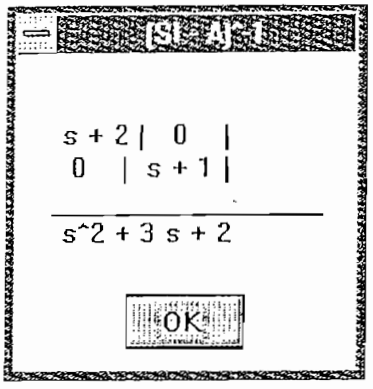


Figura A.45

- También presenta el polinomio característico, tanto expandido como en factores de los cuales se puede obtener los valores propios fácilmente.

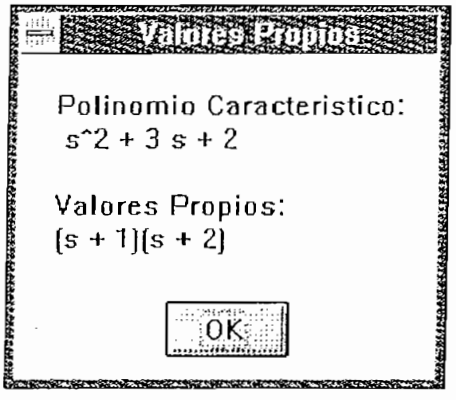


Figura A.46

- e^{At} es la transformada inversa de Laplace de $(sI- A)^{-1}$

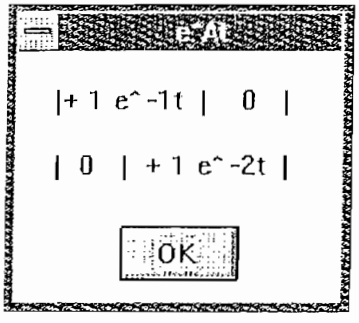


Figura A.47

- Respuesta natural. Es el resultado de $e^{At}X0$

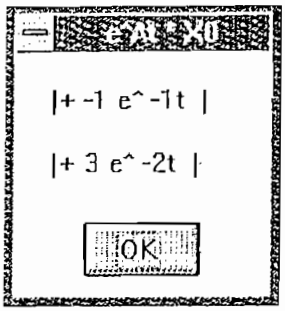


Figura A.48

- Respuesta forzada. corresponde a la integral de 0 a t de $e^{A(t-tao)}*B*u$

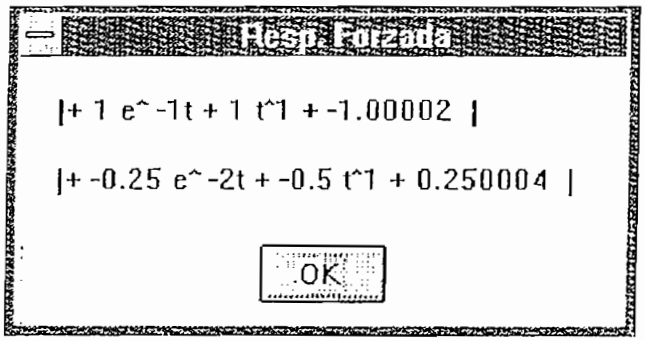


Figura A.49

- X = AX + BU. Es la suma de la respuesta natural + la forzada

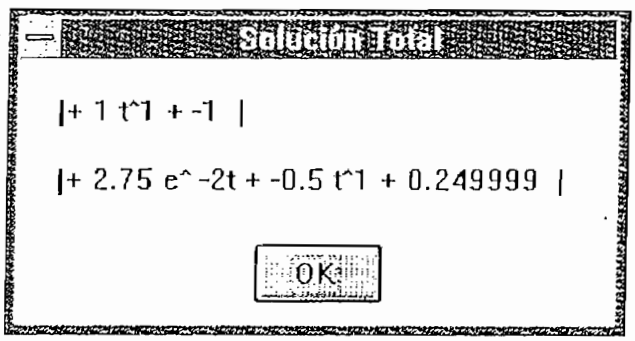


Figura A.50

Además se obtiene una función equivalente de cada fila (con el nombre g_0, g_1, etc) para analizarla usando los métodos de control clásico.

- Controlabilidad. Presenta la matriz de controlabilidad y determina si la matriz es controlable, además presenta el determinante de la matriz, (en caso de no ser una matriz cuadrada el determinante corresponde al del gramiano).

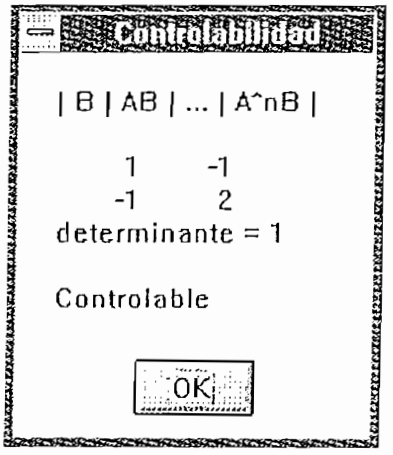


Figura A.51

- Observabilidad. Presenta la matriz de observabilidad y determina si la matriz es observable o no, además presenta el determinante de la matriz, (en caso de no ser una matriz cuadrada el determinante corresponde al del gramiano). En el ejemplo tratado el resultado es por no estar definida la matriz C

Figura A.52