ESCUELA POLITECNICA NACIONAL

FACULTAD DE INGENIERIA ELECTRICA

SIMULACION DINAMICA DEL PROBLEMA DE LA BOLA SUSPENDIDA

MEDIANTE UN PROGRAMA PARA WINDOWS

Realizado por:

JAIME OSWALDO AZANZA GUALPA

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

QUITO, NOVIEMBRE DE 1994

CERTIFICO QUE EL PRESENTE TRABAJO HA SIDO REALIZADO EN SU TOTALIDAD POR EL SR. JAIME OSWALDO AZANZA GUALPA.

Ing. Mario Barragán B.

DIRECTOR DE TESIS

AGRADECIMIENTO

Un sincero agradecimiento al Ing. Marco Barragán Bedoya por su acertada dirección en esta tesis, ya que sin su apoyo e interés no hubiera sido posible este trabajo.

INDICE

CAPITULO I

INTRODUCCION

versatil, más aun al emplear la herramienta en el entorno Windows los cambios se hacen de manera más estructurada; así se puede preveer que pasará en una planta si se desarrollan diferentes cambios en la misma.

El programa presenta la simulación dinámica del Problema de la Bola Suspendida en el monitor del Computador, el programa desarrollado es versátil, de fácil comprensión y utilización para el usuario, posee menús desplegables para elegir las opciones, los cuales introducen al usuario en cuadros de diálogo para el ingreso de datos. Dispone además de opciones para presentar los resultados, gráficos, copiarlos al portapapeles, e impresión directa desde el menú de cualquier texto o gráfico de la ventana del programa sin tener que utilizar otra aplicación. Es de mensionar que se dispone de una ayuda práctica e ilustrativa que se selecciona directamente desde el menú del programa para su utilización, el programa tiene las características de enlace y encriptación. Para cumplir éste objetivo de ser ejecutable en el entorno Windows ver. 3.1, el programa es desarrollado en su totalidad en el Lenguaje C++, con la versión 3.1 del compilador Turbo C++ de la BORLAND, con la técnica de programación dirigida a objetos.

El programa fue implementado en un computador IBM 486 DX2 con targeta de gráficos V.G.A., configuración que se dispone en el Laboratorio de Control. Para utilizar el programa se debe tener instalado el entorno operativo Windows ver 3.1 en el

computador, el mismo que debe tener la suficiente memoria (RAM) para su ejecución (recomendado 4 Mbytes). Una de las ventajas que ofrece este entorno es la selección automática del controlador de la tarjeta de video, la ejecución de varios programas simultáneamente (multitarea), y la posibilidad de enlace e incrustación entre ellos.

La simulación dinámica se emplea como una herramienta en el aprendizaje de la Teoria de Control, cimientando de esta manera los conocimientos teóricos, ésto permite al estudiante diseñar las diferentes clases de controladores que existen, sin tomar encuenta las conexiones en la Planta y por tanto reforzando los criterios de las diferentes compensaciones, observando en la simulación el funcionamiento de la planta con el controlador diseñado y la respuesta en el tiempo. Su simulación es agradable en el uso, ádemas de amigable con el usuario. De esta manera la investigación se centra en el problema fundamental, con ello se dedica mayor tiempo a la busqueda de las alternativas de control, siendo por tanto el aprendizaje de la Ingeniería de Control más fácil.

Mediante el presente trabajo de tesis se pone a consideración del estudiante de Control un programa de computación que le servirá como una herramienta útil con la que pueda aplicar los conocimientos adquiridos de control tales como redes de adelanto de fase, acciones de control y control con realimentación de estados, a un problema clásico de control: una

esfera en suspención.

Debido a que existen una gran variedad de sistemas que pueden ser simulados, se ha puesto atención en uno de ellos, el cual nos permite cumplir con las características antes mencionadas.

1.2.- JUSTIFICACION SOBRE LA SELECCION DEL PROBLEMA.

Existen diferentes sistemas reales que poseen una inestabilidad inherente, los cuales se prestan para el estudio de los sistemas de control, ya que a esos sistemas los podemos estabilizar al aplicar compensaciones diferentes, dependiendo de las especificaciones que se necesite. El problema de la bola suspendida, esto es una esfera de material ferromagnético que flota en un espacio en el cual se encuentra un campo electromagnético generado por un electroimán, figura 2.1; este es un caso típico de inestabilidad, el cual es el tema de análisis en el presente trabajo. Esta planta se presta para la simulación gráfica en el computador, ádemas se dispone de la información detallada de los parámetros y constitución del sistema.

La solución de un problema de control no sólo debe quedarse en el diseño de determinado compensador, sino que el resultado obtenido será más comprensible al tener una idea de cómo

reaccionará la planta frente a éste, lo que se observará en la simulación dinámica. Los cálculos se facilitan al utilizar un programa que permita obtener los resultados teóricos esperados, como la respuesta en el tiempo, parámetros del controlador, y realizar un análisis de los estados del sistema.

1.3.- DEFINICIONES PRELIMINARES.

Definimos como modelo a una representación analítica de la dinámica de una planta, en la cual se trata de extraer las características más importantes del comportamiento dinámico de la misma, con las aproximaciones necesarias. Se entiende por sistema dinámico aquel capaz de almacenar energía en sus elementos y por tanto producir retardo en su respuesta.

Una clasificación de los modelos es la que se presenta a continuación:

Modelo Físico. Se basa en un prototipo usualmente costoso que mantiene la característica del sistema real con sus señales escaladas en magnitud y/o tiempo.

Modelo Analítico. Es un tipo de modelo que procura la abstracción analítica de las características a propiedades de la dinámica del sistema original. Se lo puede representar en el campo contínuo o en el campo discreto.

Modelo Numérico. Es la descripción de un componente, pieza o grupo de componentes mediante números que representan movimientos, es decir representan desplazamientos, velocidades y aceleraciones.

Uno de los métodos utilizados para la modelación de sistema es, el Método Directo, el cual consiste en plantear las ecuaciones diferenciales que describen la dinámica de la planta partiendo de las relaciones de interconexión de los elementos o componentes básicos de la misma.

Los modelos matemáticos de los sitemas físicos (mecánicos, eléctricos, etc) pueden derivarse de consideraciones de energía sin aplicarles las leyes de Newton o de Kirchhoff. Al derivar las ecuaciones de movimiento para un sistema mecánico complicado, conviene hacerlo aplicando dos métodos diferentes (uno basado en la segunda ley de Newton y el otro en consideraciones de energía) para asegurarse de que sean correctos. En este aspecto, el método de Lagrange es un recurso adecuado para derivar las ecuaciones del sistema. Para derivar las ecuaciones de movimiento de Lagrange, es necesario definir las coordenadas generalizadas y el Lagrangiano, para establecer el principio de Hamilton.

1.4.- CONTENIDO.

A continuación se indica el contenido del presente trabajo

de tesis, en el cual se desarrollan los capítulos.

En el capítulo II se realiza la demostración matemática que permite encontrar las ecuaciones diferenciales que definen al sistema (Problema de la Bola Suspendida), partiendo de las consideraciones de Lagrange, para aplicarlas a la esfera en suspención y al circuito que determina que corriente circule por el electroimán. Luego se realiza la linealización de las ecuaciones diferenciales encontradas, bajo ciertas consideraciones, y por último se realiza una descripción del sistema a variables de estado.

En el capítulo III se realiza la implementación de las alternativas de control propuestas, para que el sistema compensado responda a características deseadas.

En el capítulo IV se revisan los diversos algorítmos empleados en el programa, indicando para ello los diagranmas de flujo de las rutinas principales, además se revisa la animación del sistema, y la obtención, presentación de los resultados. Se describe de manera general las principales funciones utilizadas en el programa, y se revisa la elección del compilador empleado.

En el capítulo V se presenta los resultados obtenidos con las diferentes compensaciones, para luego presentar una serie de conclusiones obtenidas en el desarrollo del programa.

Como complemento a lo anterior se presentan cuatro anexos, el primero es un pequeño manual de usuario del programa, el mismo que tiene como objetivo indicar la mejor manera de utilizar el programa. En el segundo anexo se presenta un listado del codigo fuente del programa, desarrollado con el paquete de la BORLAND, el compilador Turbo C++ ver. 3.1. En el tercer anexo se presenta una lista completa de la nomenclatura utilizada para obtener una rápida referencia o conseguir información de algúna variable que se requiera. Por último en cuarto anexo se presenta una descripción en diagramas de bloques de consideraciones de condiciones iniciales para la obtención de las compensaciones con control clásico.

CAPITULO II

ANALISIS MATEMATICO

2.1.- DESCRIPCION FISICA DEL PROBLEMA.

El sistema se halla conformado por un electroimán, el cual ejerce una fuerza de atracción sobre una esfera metálica, la cual depende de la posición que tenga la masa. Se emplea una fuente luminosa y una fotocelda que nos permiten tener un voltaje cuya magnitud sea función directa de la posición que ha adoptado la esfera. Es decir es un sistema que puede suspender una masa de material magnético por medio de un electroimán, cuya corriente es controlada por la posición de la masa.

El servosistema puede inducir un estado de posición determinada en objetos metálicos pequeños; el sistema funciona en base a los principios electromagnético y fotoeléctrico

La figura muestra un esquema del servosistema fotoeléctrico y electromagnético.

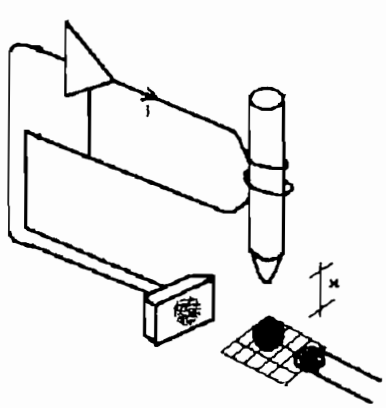


Figura 2.1.

En el circuito de la figura 2.1., la fotocelda "vigila" la esfera en el espacio, detectando su posición según el grado de

sombra que en esta se produzca. Si la esfera tiende a caer, el voltaje generado por el sensor (fotocelda) trasmite esta información al sistema de control, y este aumenta la corriente del electroimán para incrementar la fuerza de atracción, evitandose de esta manera la caida de la esfera. Si la esfera se aproxima demasiado al electroimán, la fotocelda capta la situación informando de esto al sistema de control, produciendose una disminución de la corriente de realimentación, y se reduce el tiro magnético, estableciendose así nuevamente el equilibrio.

2.2.- CRITERIOS PARA LA OBTENCION DE LAS ECUACIONES QUE DESCRIBEN EL SISTEMA.

2.2.1.- Observaciones Preliminares.

Para formular las leyes fundamentales de la dinámica de un sistema electromecánico, pueden tomarse como base las relaciones de Lagrange, estas relaciones se deducen de las leyes de movimiento de Newton y de la definición de la ecuación de d´Alembert.

Del Principio de d´Alembert se puede decir que es un ligero replanteo de la segunda ley de Newton, que algunas veces conduce a un camino más simple para obtener la ecuación de movimiento de un cuerpo. Así, cuando una fuerza actúa sobre una masa, la acelera; en lugar de pensar en la aceleración como resultado de la aplicación de una fuerza, se puede convertir ésta situación

dinámica en una situación de equilibrio en la cual la suma. de las fuerzas externas se iguala por una fuerza de inercia ficticia. La ecuación que resulta de la aplicación del principio de d´Alembert es aquella en la cual la suma de todas las fuerzas, incluyendo la fuerza de inercia ficticia, se hace igual a cero. El enfoque de d´Alembert se aplica tanto a sistemas traslacionales como rotacionales y proporciona una simplificación analítica importante en situaciones complicadas que involucran traslación y rotación combinadas. La principal ventaja del método de d´Alembert sobre la aplicación directa de la segunda ley de Newton es que no se necesita considerar la acción de fuerzas y pares respecto a un eje a través de su centro de gravedad. En lugar de esto se puede resumir tal acción con respecto al eje que consideremos conveniente.

Ahora para obtener las relaciones de Lagrange, es conveniente mensionar que la utilización de la ley de conservación de la energía para obtener ecuaciones de movimiento es fácil en sistemas simples. Lagrange desarrolló una forma más general de abordarlo basada en el principio de energía, que puede usarse para sistemas más complejos, en donde se derivan dichas ecuaciones de movimiento, del conocimiento de las energías potencial y cinética del sistema.

Para derivar las ecuaciones de movimiento de Lagrange, es necesario definir las coordenadas generalizadas y el Lagrangiano, para establecer el principio de Hamilton.

Coordenadas generalizadas. Las coordenadas generalizadas de un sistema son un conjunto de coordenadas independientes que se necesita para describir completamente el movimiento del sistema. El número de coordenadas generalizadas necesario para describir el movimiento del sistema es igual al número de grados de libertad, que se representará por n

Si las n coordenadas generalizadas son q_1, q_2,..., q_n, entonces, en cualquier instante el sistema se caracteriza mediante un punto en un espacio n-dimensional. Al transcurrir el tiempo, el punto se mueve y describe una curva en el espacio.

Lagrangiano. El Lagrangiano L de un sistema se define por:

$$L = T - U \qquad (2.1)$$

donde T es la energía cinética y U es la energía potencial del sistema. El Lagrangiano en forma general es una función de las coordenadas generalizadas q_i, velocidades $\dot{q}_i$, i = 1,2,..,n, y del tiempo t, o bien.

$$L = L(q_i, \dot{q}_i, t) \qquad (2.2)$$

Utilizando el Principio de Hamilton el cual establece que el movimiento del punto del sistema en el espacio n-dimensional de $t = t_1$ a $t = t_2$ es tal que la integral. Ref [1]

---

[1] Dinamica de Sistemas, OBATA, 596 - 606.

$$I = \int_{t_1}^{t_2} L(q_i,\ \dot{q}_i,\ t)\ dt \qquad (i=1,2,\ldots,n)$$

es un extremo (máximo o mínimo) de la trayectoria del movimiento.

Realizando algunos manejos matemáticos en esta última integral, dandose pequeñas variaciones en la coordenadas generalizadas, evaluando la diferencia entre dos integrales de Lagrange, además empleando las series de Taylor y la teoría de variaciones, podemos obtener la ecuación de Lagrange.
De tal manera que las ecuaciones de Lagrange toman la forma compacta. Ref 2

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{q}_i}\right) - \frac{\partial L}{\partial q_i} = F_{q_i} \qquad (2.3)$$

en donde q_i es una cualquiera de las coordenadas que aparecen en L y F_{qi} se denomina fuerza generalizada.

Para aclarar un poco esto, se tiene lo siguiente; las ecuaciones de movimiento para una masa libre se escribe.

$$F_x = m\ddot{x} \qquad (2.4)$$

$$F_y = m\ddot{y}$$

$$F_z = m\ddot{z}$$

2/ Dinamica de Sistemas, OGATA, 575 - 606.

Multiplicando las ecuaciones (2.4) por δx, δy, δz respectivamente y sumando se tiene que:

$$m\ (\ddot{x}\ \partial x + \ddot{y}\ \partial y + \ddot{z}\ \partial z) = F_x\ \partial x + F_y\ \partial y + F_z\ \partial z \qquad (2.5)$$

el miembro derecho de la ecuación (2.5) representa el trabajo hecho por la fuerza F sobre el cuerpo de masa m y, el miembro izquierdo puede interpretarse como el pequeño cambio correspondiente en la cinética de m. Esta ecuación se denomina ecuación de d´Alembert.

Al introducir las coordenadas generalizadas en (2.5) y realizar algunos manejos matemáticos, se obtienen las ecuaciones de Lagrange.

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{q}_i}\right) - \frac{\partial L}{\partial q_i} = F_x \frac{\partial x}{\partial q_i} + F_y \frac{\partial y}{\partial q_i} + F_z \frac{\partial z}{\partial q_i} \qquad (2.6)$$

por conveniencia se escribe:

$$F_x \frac{\partial x}{\partial q_i} + F_y \frac{\partial y}{\partial q_i} + F_z \frac{\partial z}{\partial q_i} = F_{q_i} \qquad (2.7)$$

En la ecuación (2.6) i = 1, 2, 3; en general, hay tantas ecuaciones del movimiento de Lagrange cuantos grados de libertad se tengan. Esto se refiere a si el cuerpo se mueve en un plano, superficie o espacio; ó en los circuitos eléctricos, a las ecuaciones de mallas independientes.

Como se indicó antes para el Lagrangiano, se puede realizar un mejor esclarecimiento, si se considera el circuito de la

figura 2.2.

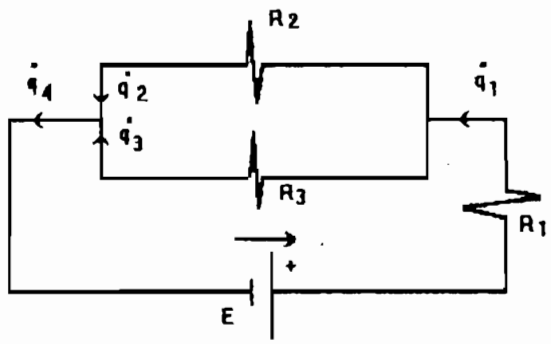


Figura 2.2.

Las cargas q_1, q_2, etc, que pasan por las diferentes ramas del circuito después de un instante determinado, por ejemplo t=0, constituyen "coordenadas" adecuadas. Entonces la corriente.

$$i = \frac{d\,q}{d\,t} = \dot{q}$$

corresponde a una "velocidad" e igualmente q representa una "aceleración". La asignación arbitraria de una dirección positiva para el flujo (dirección de la corriente), de cada una de las cargas corresponde a la dirección positiva de cada una de las coordenadas.

Energía Cinética. La energía magnética de una bobina de inductancia constante M, es: [3]

---

[3] /Dinamica de Lagrange. pag. 317 - 318

$$E = \frac{1}{2} M \dot{q}^2$$

Comparando este resultado con la energía cinética de una partícula:

$$E = \frac{1}{2} m v^2$$

M correspondería a la masa y q a la velocidad.

Energía Potencial. Puede considerarse como respuesta de dos partes: las fuentes de energía (baterías, generadores, etc.) y la energía almacenada en los condensadores.
Una fuente de voltaje terminal constante e, suministra una energía al sistema e q, en donde q es la carga "entregada por la fuente" en al dirección de e. Así, si se refiere la energía potencial al "punto" q = 0, se escribe $U_{fuente} = - e q$. Nótese la gran analogía de esto con la relación U = - m g y de la energía potencial de una masa m debida a la gravedad, en donde se toma como positiva la dirección vertical hacia arriba.

Fuerzas Generalizadas F_{qi}. Dentro de la categoria de fuerzas disipativas se catalogan todas aquellas que originan, disipación de energía del sistema durante el movimiento. Se recurre a una función de potencia para la determinación de las fuerzas generalizadas. Ref [4]

$$F_{q_i} = \frac{\partial P}{\partial \dot{q}_i} \qquad (2.8)$$

---

[4] /Dinamica de Lagrange. pag. 317 - 318

donde:

$$P = - \frac{1}{2} R \dot{q}^2 \qquad (2.9)$$

Se define como la rápidez con que la energía mecánica se transforma en térmica.

Generalizando las ecuaciones de Lagrange para los circuitos electromecánicos, tiene la forma siguiente:

$$\frac{d}{dt} \left(\frac{\partial La}{\partial \dot{q}_i} \right) - \frac{\partial La}{\partial q_i} = F_{q_i} \qquad (2.10)$$

donde la función de Lagrange es:

$$La = T_i - U_i \qquad (2.11)$$

2.2.2.- **Determinación de las ecuaciones no lineales del sistema.**

Si se toma como base lo descrito anteriormente, para poder definir el comportamiento no lineal del sistema en estudio, considerando un simil del circuito magnético, esto es en las figuras 2.3.a, 2.3.b. Suponiendo que el electroimán tiene una reluctancia R_1, y una reluctancia generada por el entrehierro que forma la esfera R_2, (la cual según la teoría electromagnética es igual a $R_2 = x /(\mu_o A)$, donde A es el área transversal del núcleo y x es la separación entre la esfera y el

electroimán). Ref [5]

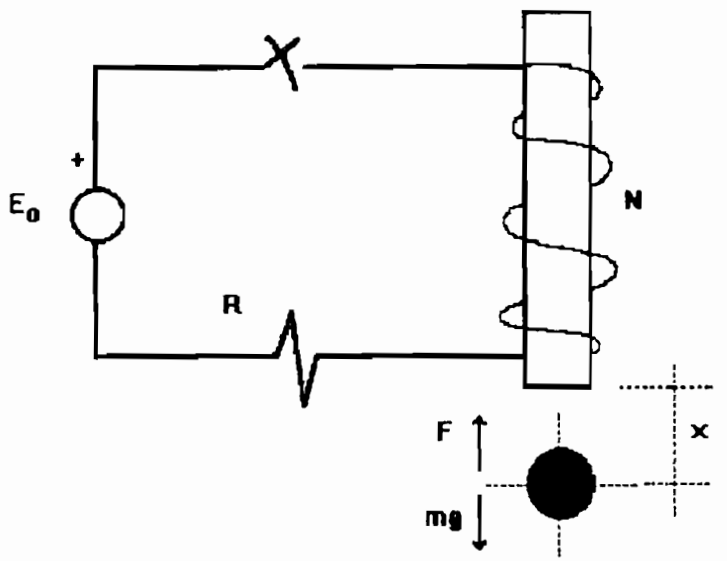


Figura 2.3.a.

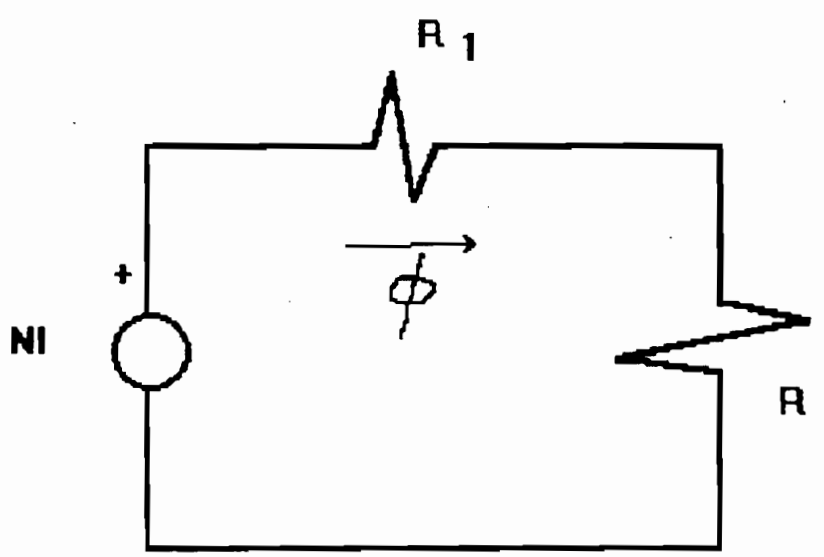


Figura 2.3.b.

---

[5] /Electromagnetismo. John D. Kraus pag. 245 – 246.

Las relaciones entre el flujo magnético (Φ), la fuerza magnetomotriz (N I) y la reluctancia (R) son idénticas a las existentes entre la corriente, el voltaje y la resistencia en un circuito eléctrico.

Entonces se tiene que el flujo magnético es:

$$\phi = \frac{N\,I}{R_1 + \frac{x}{\mu_0 A}}$$

El flujo concatenado será:

$$\psi = N\,\phi = L\,I$$

De donde la inductancia toma el valor:

$$L = \frac{N^2}{R_1 + \frac{x}{\mu_0 A}} \qquad (2.12)$$

Considerando coordenadas generalizadas, el trabajo realizado o energía cinética del sistema (T), la energía potencial (U) y, el Lagrangeano de la energía (La) vienen dados por:

$$T = \tfrac{1}{2}\,L\,\dot{q}^2 + \tfrac{1}{2}\,m\,\dot{x}^2$$

$$U = m\,g\,x - q\,e_0$$

de donde:

$$La = T - U = \tfrac{1}{2}\,L\,\dot{q}^2 + \tfrac{1}{2}\,m\dot{x}^2 - mgx + qe_0 \qquad (2.13)$$

reescribiendo la ecuación (2.10) tenemos:

$$\frac{d}{dt}\left(\frac{\partial La}{\partial \dot{q}}\right) - \frac{\partial La}{\partial q} = - \frac{\partial}{\partial \dot{q}}\left(\frac{1}{2} R \dot{q}^2\right) \qquad (2.14)$$

entonces:

$$\frac{\partial La}{\partial \dot{q}} = L \dot{q} \qquad (2.15)$$

$$\frac{\partial La}{\partial q} = e_0 \qquad (2.16)$$

$$\frac{\partial P}{\partial \dot{q}} = R \dot{q} \qquad (2.17)$$

reemplazando (2.15), (2.16), (2.17) en (2.14) se tiene:

$$\frac{d}{dt}(L \dot{q}) - e_0 = - R \dot{q}$$

donde:

$$e_0 = R \dot{q} + \frac{d}{dt}(L \dot{q}) = R i + \frac{d}{dt}(L i) \qquad (2.18)$$

Debido a la fuerza magnética que el imán ejerce sobre la masa, no podemos ya más considerar constante el valor de la inductancia (L) sino, como una fuerza F dependiente de variaciones pequeñas de x, i y t. Así, la fuerza magnética del electroimán vendrá dada por:

$$F_{mag} = \frac{1}{2} i^2 \frac{\partial L}{\partial x} \qquad (2.19)$$

reemplazando la ecuación (2.12) en (2.19)

$$F_{mag} = \frac{1}{2}\, i^2\, \frac{\partial}{\partial x}\, \frac{N^2}{R_1 + \frac{x}{\mu_0 A}}$$

$$F_{mag} = -\frac{N^2\, i^2}{2\, \mu_0\, A\, (R_1 + \frac{x}{\mu_0 A})^2} \qquad (2.20)$$

trabajando con la ecuación (2.18)

$$e_0 = R\, i + L\, \frac{di}{dt} + i\, \frac{dL}{dt}$$

$$e_0 = R\, i + L\, \frac{di}{dt} + i\, \frac{dL}{dt}\, \frac{dx}{dx}$$

$$e_0 = R\, i + L\, \frac{di}{dt} + i\, \frac{dL}{dx}\, \dot{x} \qquad (2.21)$$

reemplazando (2.20) en (2.21)

$$e_0 = R\, i + L\, \frac{di}{dt} - \frac{N^2\, i\, \dot{x}}{\mu_0 A\, (R_1 + \frac{x}{\mu_0 A})^2} \qquad (2.22)$$

trabajando nuevamente con la ecuación (2.13) tenemos:

$$\frac{\partial La}{\partial \dot{x}} = m\, \dot{x} \qquad (2.23)$$

$$\frac{\partial La}{\partial x} = -\, m\, g \qquad (2.24)$$

entonces:

$$m\ddot{x} = -mg + \frac{1}{2} i^2 \frac{\partial L}{\partial x}$$

de donde se tiene que:

$$m\ddot{x} = -mg - \frac{N^2 i^2}{2 \mu_0 A \left(R_1 + \frac{x}{\mu_0 A}\right)^2} \qquad (2.25)$$

Las ecuaciones (2.22) y (2.25) representan a un sistema no lineal y confirman que el efecto mecánico - eléctrico no es independiente. Estas ecuaciones describen la dinámica de nuestro sistema, determinandose que la fuerza debida al electroimán es una función de la distancia de atracción (x) y de la corriente(i) que circula por la bobina.

Resulta entonces que la ecuación dinámica viene dada por:

$$\sum F = m a$$

$$F_{mag} - mg = ma$$

es decir:

$$f_{(x,i)} - mg = m\ddot{x} \qquad (2.26)$$

2.3.- LINEALIZACION DE LAS ECUACIONES.

Con el objeto de tener el modelo matemático lineal de nuestro sistema, suponemos que las variables involucradas (x,i),

varían poco dentro de una condición normal de operación.

Por ello, para obtener las ecuaciones lineales que faciliten el proceso de control creemos necesario entonces trabajar con parámetros de un equipo existente, de esta manera tendremos una mejor visión de la realidad del sistema, que si haríamos una linealización matemática en la cual además nos faltarían ciertos parámetros que se los tendría estimativos y no reales. Con lo cual la respuesta del equipo existente nos da las verdaderas constantes de linealización.

Si consideramos como entradas del sistema las variables (x,i), como salida la fuerza F_{mag} que como ya se demostró es igual a f(x,i), y si la condición normal de operación corresponde a $\bar{x}$ e $\bar{i}$, entonces la ecuación de la fuerza puede ser desarrollada en una serie de Taylor alrededor de este punto, figura 2.5, de la manera siguiente:

$$F_{mag} = f(x, i)$$

$$F = f(\bar{x}, \bar{i}) + \left[\frac{df}{dx}(x-\bar{x}) + \frac{df}{di}(i-\bar{i})\right] + \frac{1}{2}\left[\frac{d^2f}{dx^2}(x-\bar{x}) \ldots .\right.$$

$$\left. \ldots\ldots + 2\frac{d^2f}{dxdi}(x-\bar{x})(i-\bar{i}) + \frac{d^2f}{di^2}(i-\bar{i})\right] + \ldots$$

Donde las derivadas partciales df/dx, d^2f/dx^2, df/di, d^2f/di^2,..., son evaluadas en $x = \bar{x}$ e $i = \bar{i}$. Si las variaciones $(x - \bar{x})$ e $(i - \bar{i})$ son pequeñas (cerca del punto normal de

operación), se pueden despreciar los términos de orden superior a uno, de las derivadas.

Procedemos a buscar un punto de trabajo estable, para ello sabemos que el modelo matemático lineal de la fuerza del electroimán alrededor de la condición normal de operación, viene dado por:

$$F - \bar{F} = K_1 (x - \bar{x}) + K_2 (i - \bar{i}) \qquad (2.27)$$

donde:

$$\bar{F} = f(\bar{x} - \bar{i})$$

$$K_1 = \frac{df}{dx} \Big|_{x=\bar{x}, i=\bar{I}} \qquad (2.28)$$

$$K_2 = \frac{df}{di} \Big|_{x=\bar{x}, i=\bar{I}} \qquad (2.29)$$

Donde $\bar{x}$ = Xo e $\bar{i}$ = Io son los valores de la posición y corriente en un punto normal de operación, punto de trabajo. [6]

Para obtener el punto normal de operación que permita conseguir las constantes de linealización K_1 y K_2, se ha tomado los valores que permiten obtener las curvas que relacionan la fuerza de atracción ejercida por el electroimán sobre la masa considerada y, la distancia de atracción. [6]

Para obtener estos valores se ha empleado esferas de

---

[6] /Prototipo de un Sistema de Control. Galarza I., TESIS, ESPE, 1987

material ferromagnético (tol), huecas, de 1.0, 2.6, y 4.0 gramos respectivamente. Para la determinación de la fuerza de atracción se ha utilizado una balanza CENT-O-GRAM (sensibilidad un décimo de gramo), trabajando con una fuente de alimentación variable. El proceso que se ha desarrollado consistió en acercar el electroimán a las esferas, y donde el peso de la balanza marque cero, se ha tenido un punto de equilibrio.
En la tabla 4.1 se tabulan los valores que permiten obtener el punto normal de operación, es decir se escogió los valores de masa, corriente y distancia, alrededor de los cuales se analiza el comportamiento del sistema.

masa	20 V	25 V	30 V	35 V
g	0.625 A	0.781 A	0.938 A	1.094 A
	distancia " d " (cm)			
1.0	2.5	3.0	3.5	4.0
2.6	2.0	2.6	3.2	3.6
4.0	2.0	2.5	3.0	3.5

Tabla 2.1 Valores de la distancia de atracción respecto de la masa de las esferas de influencia.

Si se define "d" como la distancia entre el electroimán y la masa, "x" la distancia del imán respecto del eje de referencia, según la figura 2.5, se tiene:

Esfera considerada:

masa = 2.6 grm

diametro de la esfera = 2.0 cm = ϕ

$d_{mínimo}$ = 1.0 cm $d_{máximo}$ = 3.0 cm

d_{medio} = 2.0 cm

X_o = 0 (Referencia)

I_o = 600 mA

V_{fuente} = 30 v

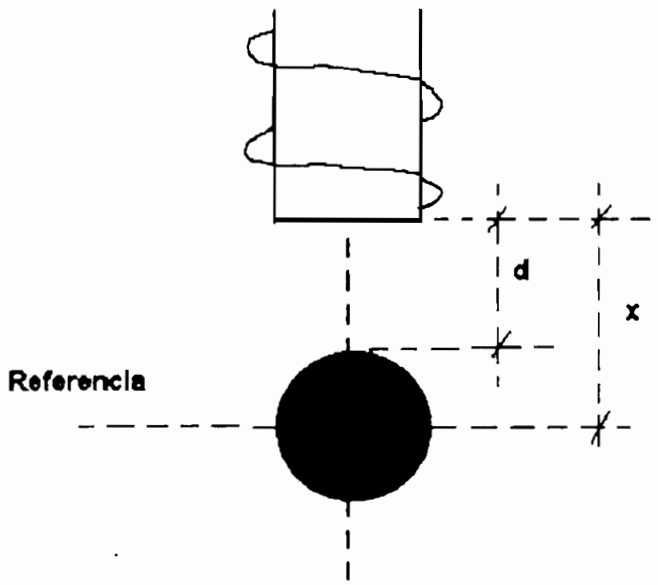


Figura 2.5 Punto de Trabajo considerado.

donde:

- d_{medio} es la distancia promedio a la cual la esfera se mantiene en suspensión
- $d_{mínimo}$ es la distancia mínima de flotación. Si d < $d_{mínimo}$ la esfera es atraida fuertemente hacia el imán.
- $d_{máximo}$ es la distancia máxima de flotación. Si d < $d_{máximo}$ la esfera cae.

Los valores de $d_{mínimo}$ y $d_{máximo}$ dan la zona de atracción de la esfera debido a la influencia que tiene la sombra de la misma sobre el fotodetector.

Una vez que se ha seleccionado el punto de trabajo, se procede a buscar las constantes de linealización encontrando para esto las pendientes de las curvas (de la fuerza de atracción del electroimán en función de la posición y de la corriente), correpondientes alrededor del punto de trabajo. Los datos experimentales que ayudan a la construcción de estas curvas se presentan en la Tabla 2.2. [6]

i (A)	F (N) d = 2 (cm)	F(N) d = 3 (cm)	F(N) d = 4 (cm)
0.000	0.00	0.00	0.00
0.150	0.55	0.30	0.10
0.300	1.00	0.60	0.20
0.450	1.75	1.00	0.30
0.500	2.00	1.20	0.40
0.600	2.60	1.40	0.45
0.750		1.80	0.65
0.800		2.00	0.75
0.900		2.30	1.00
0.950		2.45	1.11
1.000		2.60	1.20

Tabla 2.2 Valores correspondientes a las curvas $F = f(x,i)$ para la Linealización

Encontrado las pendientes de las curvas correspondientes a los valores de la tabla 2.2, alrededor del punto de trabajo, y aplicando las ecuaciones (2.28) y (2.29) y recordando que:

$$x = d_{media} - d$$

tenemos:

$$K_1 = \frac{df}{dx}\Big|_{i=I_0} \approx \frac{\Delta f}{\Delta x}\Big|_{i=I_0} = \frac{(2.6-1.4)\ 9.8\ 10^{-3}}{[0-(-1)\ 10^{-2}]} = 1.176\,\frac{N}{m}$$

$$K_2 = \frac{df}{dx}\Big|_{x=X_0} \approx \frac{\Delta f}{\Delta i}\Big|_{x=X_0} = \frac{(2.6-2.4)\ 9.8\ 10^{-3}}{[0.6\ -\ 0.5]} = 0.059\,\frac{N}{A}$$

Por lo tanto la ecuación de la fuerza ejercida por el electroimán puede representarse dentro de la región lineal de la manera siguiente:

$$F = 1.176\ x + 0.059\ i \qquad (2.30)$$

Donde x e i son los valores de las variables de posición y corriente con respecto a los valores referenciales de Xo e Io.

Para la ecuación (2.22) su expresión luego de ser linealizada queda de la siguiente manera:

$$e_0 = Ri + L\frac{di}{dt} + C\dot{x} \qquad (2.31)$$

Donde: L = 0.478 H (determinado experimentalmente)

R = 50 Ω

C = Constante de linealización del término no lineal

Al efectuar la linealización matemática de los términos no lineales de las ecuaciones (2.22) y (2.20) se obtiene que la constante que acompaña a la variable corriente en la ecuación (2.20) es idéntica a la constante que acompaña a la variable velocidad en la ecuación (2.22) por lo tanto:

$$C = K_2$$

con lo que la ecuación (2.22) puede expresarse de la manera siguiente:

$$e_0 = Ri + L\frac{di}{dt} + K_2\dot{x} \qquad (2.32)$$

2.4.- FUNCIONES DE TRANSFERENCIA Y DIAGRAMAS DE BLOQUE.

Reescribiendo la ecuación (2.26) que describe la dinámica el sistema, tenemos:

$$m\ddot{x} = -mg + f(x,i)$$

En el equilibrio, la fuerza magnetica compensa el peso de la masa, y se supone que la corriente en este estado es Io. Si se escribe la corriente como I = Io + i, y se expande f alrededor de x = 0 e I = Io, y despreciando términos de orden superior, se obtiene:

$$m\ddot{x} = K_1 x + K_2$$

Si se toman las transformadas de Laplace a ambos miembros, se puede llegar a determinar la función de transferencia del electroimán. Así.

$$m s^2 X(s) = K_1 X(s) + K_2 I(s)$$

de lo cual se obtiene:

$$\frac{X(s)}{I(s)} = \frac{K_2}{ms^2 - K_1} = \frac{\frac{K_2}{m}}{s^2 - \frac{K_1}{m}}$$

reemplazando los valores de las constantes K1, K2 y m se tiene la siguiente expresión:

$$\frac{X(s)}{I(s)} = \frac{22.7}{s^2 - 452.3} \qquad (2.33)$$

De la misma manera si se aplica la transformada de Laplace a la ecuación (2.32) permite obtener la siguiente igualdad:

$$E_0(s) = R I(s) + s L I(s) + s K_2 X(s)$$

expresión que reordenándola da:

$$E_0(s) - s K_2 X(s) = (R + s L) I(s)$$

si se coloca la expresión anterior en términos más manejables para una función de transferencia se llega a:

$$\frac{I(s)}{E_0(s) - s\,K_2\,X(s)} = \frac{\frac{1}{L}}{s + \frac{R}{L}} = \frac{2.09}{s + 104.6} \qquad (2.34)$$

Al combinar las ecuaciones (2.33) y (2.34) se puede llegar al siguiente diagrama de bloques.

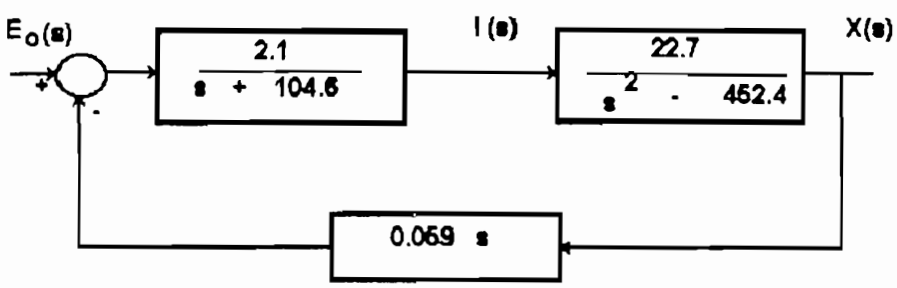


Figura 2.6 Diagrama de Bloques de la Planta.

A fin de obtener una representación matemática total del sistema que se trata, falta por encontrar la función de transferencia del fotosensor de posición, cuyo comportamiento lo representamos por los datos obtenidos experimentalmente.

Estos datos se los presenta en la tabla 2.3 a continuación:

V (Fotocelda) (v)	s (cm)
1.489	1.60
1.475	1.70
1.462	1.80
1.458	1.90
1.440	2.00
1.424	2.10
1.407	2.20

Tabla 2.3 Respuesta del Fotosensor.

Por regresión lineal, la pendiente de la línea recta que mejor se ajusta al conjunto de puntos dados es de - 13.2. Esto es la función de transferencia para el bloque de realimentación referida a la posición x (basados en la relación x y d) es:

$$H(s) = 13.2 \qquad (2.35)$$

el signo negativo es tomado encuenta en el signo de la realimentación.

Con esto ya se puede plantear el diagrama de bloques completo del Sistema de Control, que queda de la manera siguiente:

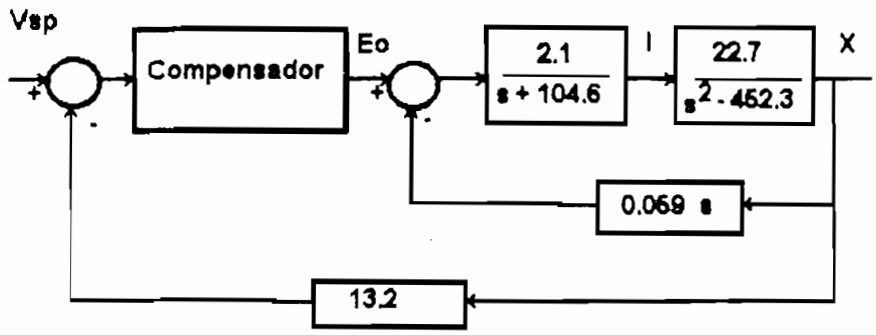


Figura 2.7 Diagrama de Bloques del Sistema de Control

2.5.- DESCRIPCION DEL SISTEMA A VARIABLES DE ESTADO.

Para obtener la Función de Transferencia de la Planta, debemos resolver el lazo interno de la figura 2.9, utilizando la reducción de bloques tenemos que:

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

$$\frac{X(s)}{E_0(s)} = \frac{47.7}{s^3 + 104.6\ s^2 - 449.6\ s - 47322} \qquad (2.36)$$

Por lo tanto tenemos que:

$$\dddot{X}(s) + 104.6\ \ddot{X}(s) - 449.6\dot{X}(s) - 47322\ X(s) = 47.7\ E_0(s)$$

Si tomamos como variables de estado a:

$$\dddot{X}(s) + a_2 \ddot{X}(s) + a_1 \dot{X}(s) + a_0 X(s) = b_0 E_0(s)$$

$x_1(t) = x(t) =$ espacio

$x_2(t) = x(t) =$ velocidad

$x_3(t) = x(t)$

Con las ecuaciones deducidas y con los estados elegidos se obtiene la descripción del sistema por variables de estado:

$$\dot{x}_1 = x_2$$

$$\dot{x}_2 = x_3$$

$$\dot{x}_3 = -a_0 x_1 + -a_1 x_2 + -a_2 x_3 + b_0 e_0$$

En base a estas tres ecuaciones se obtiene la descripción por variables de estado en forma matricial

La ecuación matricial es la siguiente:

$$\dot{\underline{x}}(t) = \underline{A}\,\underline{x}(t) + \underline{B}\,u(t)$$

en donde:

$\underline{x}(t)$ es el vector de estados,

$\dot{\underline{x}}(t)$ es el vector de las derivadas de los estados,

$\underline{A}$ es una matriz cuadrada 3x3

$\underline{B}$ es un vector de tres elementos

$u(t)$ es la entrada, en este caso es el voltaje eo.

$$\begin{vmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \end{vmatrix} = \begin{vmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ -a_0 & -a_1 & -a_2 \end{vmatrix} \begin{vmatrix} x_1 \\ x_2 \\ x_3 \end{vmatrix} + \begin{vmatrix} 0 \\ 0 \\ b_0 \end{vmatrix} e_0$$

CAPITULO III

COMPENSACIONES

3.1.- ANALISIS DEL SISTEMA SIN COMPENSAR.

En el presente capítulo se desarrollarán diferentes compensaciones para nuestro sistema; hay que recordar que "Compensación", es el ajuste de un sistema para satisfacer especificaciones dadas, que habitualmente se refieren a exactitud, estabilidad relativa y velocidad de respuesta. Para el presente sistema se dan especificaciones de respuesta transitoria con el tiempo de establecimiento, y de estabilidad relativa, con el máximo sobreimpulso.

Para la adecuada determinación de las Compensaciones, se emplea el lugar geométrico de las raices y se determina la respuesta de frecuencia del sistema sin compensar.

Recordando la función de trasferencia de la planta se tiene que las raices de la ecuación característica son:

$$\frac{X_{(s)}}{E_{0(s)}} = \frac{47.7}{s^3 + 104.6\,s^2 - 449.6\,s - 47322}$$

donde los polos de lazo abierto son:

$$p_1 = -21.3$$

$$p_2 = 21.3$$

$$p_3 = -104.6$$

En la figura 3.1, se indica el L.G.R. del sistema sin compensar; debido a la presencia de un polo positivo, existe un

ramal del L.G.R. en el semiplano derecho del plano "s", dando la inestabilidad del sistema, por tanto se debe modificar el L.G.R. para hacerlo estable, esto es hacia la izquierda.

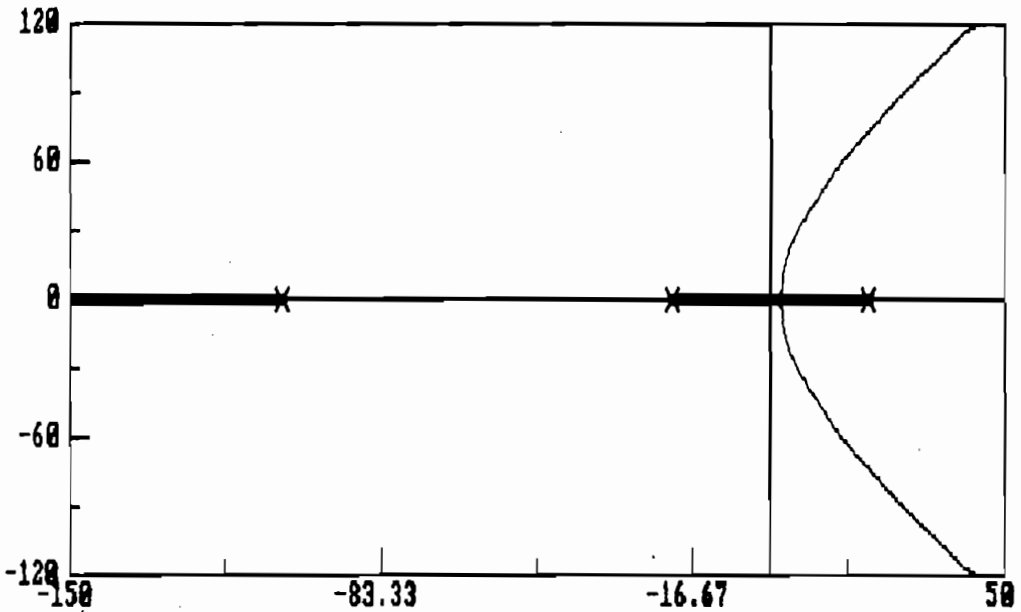


Figura 3.1 L.G.R. del Sistema no Compensado

Del análisis del lugar geomémtrico de las raíces, se observa que la red Compensadora debe ser una red de Adelanto de Fase. Las técnicas que se va a utilizar se basan en un sistema con sobretiro, por cuanto tales redes para su implementación requieren asumir polos dominantes complejos.

En el diagrama de Bode de la planta, figura 3.2.a y 3.2.b se demuestran las caracteríticas del sistema, donde la curva del módulo no corta al eje de frecuencia; en la figura 3.2.a se indica esta situación. Así mismo la curva de fase, figura 3.2.b tiene un valor de -180° para frecuencia cero. No se puede

analizar estabilidad con esta herramienta, pues el sistema no es de fase mínima.

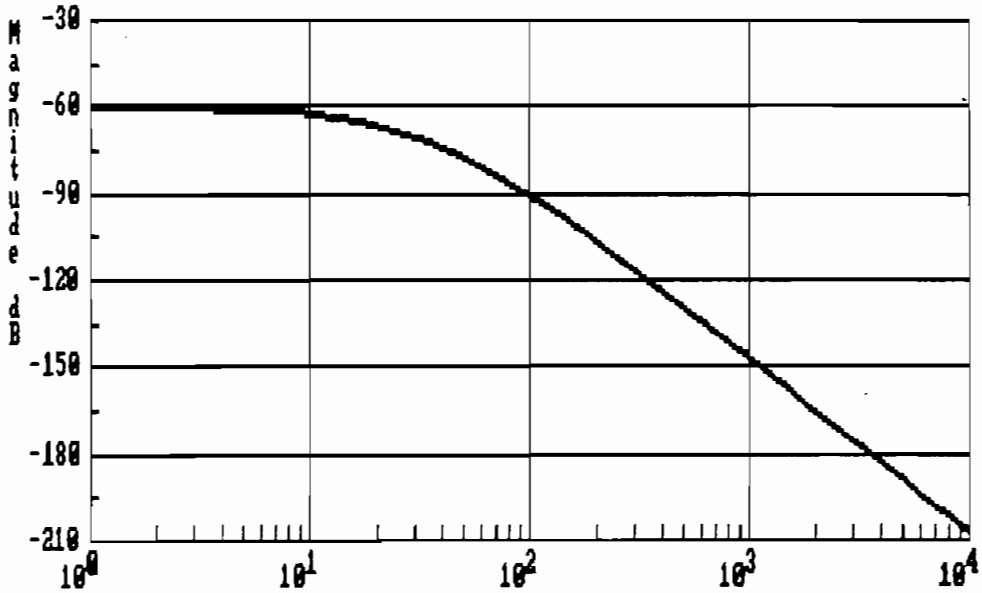


Figura 3.2.a Diagrama de Bode, Modulo.

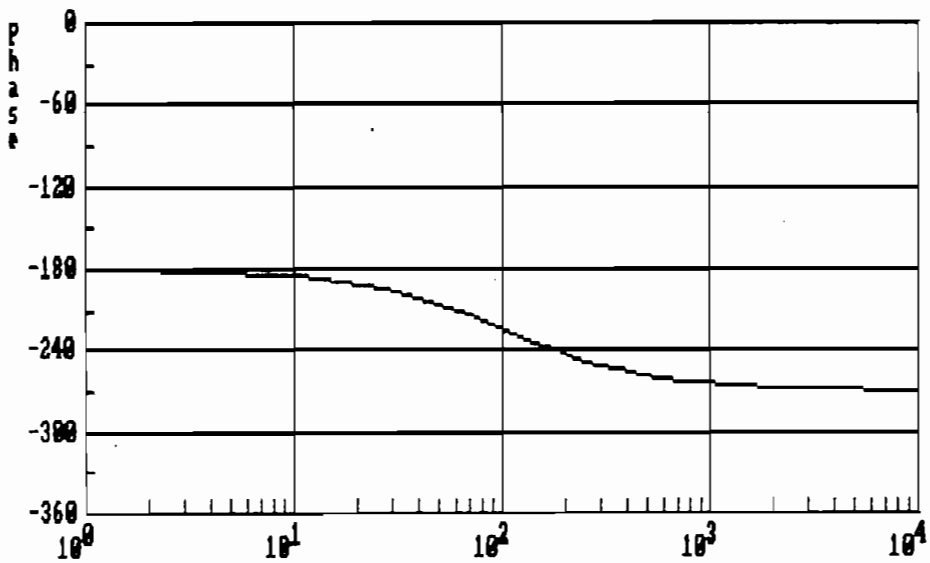


Figura 3.2.b Diagrama de Bode, Fase.

3.2.- REALIMENTACION DE ESTADO.

Para este tipo de control se dispone del siguiente diagrama de bloques, figura 3.3, en el cual la planta se la cambia por la descripción a variables de estado:

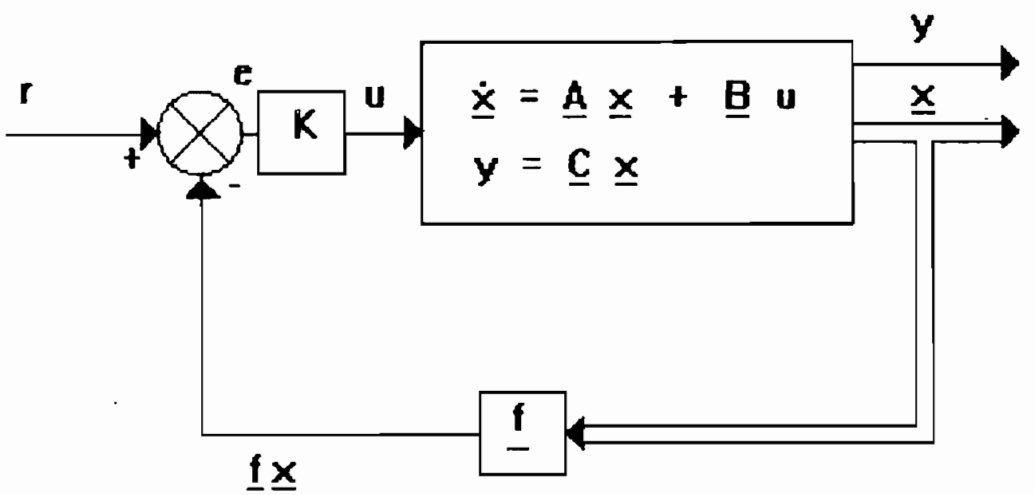


Figura 3.3 Realimentación de estado.

Para estabilizar al sistema el voltaje será una función de las variables de estado, de esta manera, será igual:

$$u = K\,e$$

$$e = r - \underline{f}\,\underline{x}$$

$$u = K\,(r - \underline{f}\,\underline{x})$$

donde: K es la ganancia.

$\underline{x}$ es el vector de estados.

u es el parametro de control

r es la referencia (posición)

$\underline{f}$ es el vector de realimentación f = [f1 f2 f3]

A continuación se prueba que el sistema es controlable para ello se chequea que el determinante de la matriz de controlabilidad U sea diferente de cero (o sea que los vectores columna sean linealmente independientes); para garantizar que los resultados que se obtendrán sean válidos. El concepto de controlabilidad permite conocer si existe una entrada **u** físicamente factible que permita llevar el vector de estados de un punto **x(0)** del espacio de estados a otro punto **x(1)** arbitrario.

Por definición se tiene:

$$U = [\ B \mid AB \mid A^2B\ ]$$

de donde se obtiene que:

$$U = \begin{bmatrix} 0 & 0 & b_o \\ 0 & b_o & -a_2 b_o \\ b_o & -a_2 b_o & -a_1 b_o + a_2^2 b_o \end{bmatrix}$$

El determinante de esta matriz es diferente de cero y es el siguiente:

$\det(U) = -b_o^3$

por lo tanto el sistema es controlable.

El diagrama de la figura 3.3 arriba indicado se lo puede desglosar de la siguiente manera, figura 3.4, con el fin de obtener las ecuaciones de control.

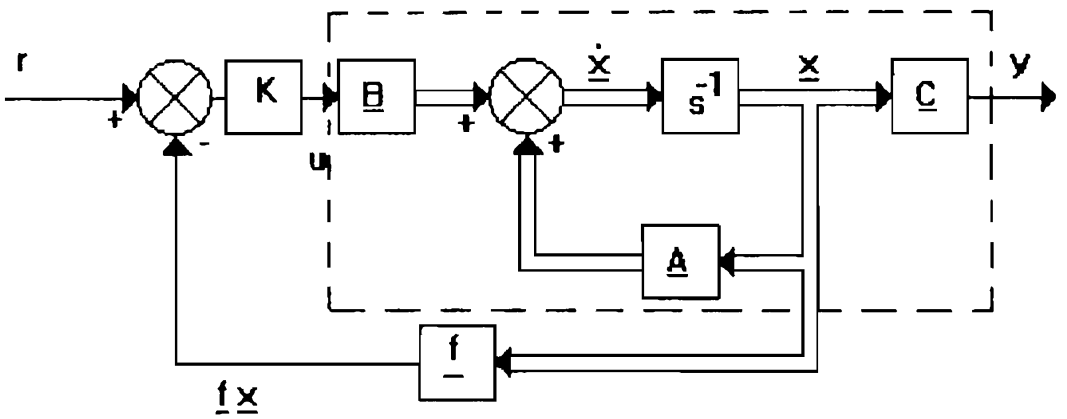


Figura 3.4 Descripción del Control con Realimentación de estado.

De donde podemos obtener que:

$$\dot{\underline{x}} = \underline{A}\,\underline{x} + \underline{B}\,u$$

$$\det(s\,\underline{I} - \underline{A}) = s^3 + s^2 a_2 + s\,a_1 + a_o$$

donde

$$u = K\,e = K\,(r - \underline{f}\,\underline{x})$$

$$\dot{\underline{x}} = (\underline{A} - K\,\underline{B}\,\underline{f})\,\underline{x} + K\,\underline{B}\,r$$

ademas se la ecuación de salida como:

$$y = C\,x$$

aplicando la transformada de Laplace para encontrar la solución a la ecuación se tiene que:

$$s\,X(s) - x(0) = (A - K\,B\,f)\,X(s) + K\,B\,R/s$$

despejando X(s) se tiene que:

$$X(s) = [s\,I - (A - K\,B\,f)]^{-1}\,[\,x(0) + K\,B\,R/s\,]$$

Esta ecuación es la solución general en el dominio de Laplace de la ecuación de estado.

Si reemplazamos

$$M = [s\,I - (A - K\,B\,f)]$$

$$N = [\,x(0) + K\,B\,R/s\,]$$

de donde: $X(s) = M^{-1}\,N$

$$[sI - (A + K\,B\,f)] = \begin{bmatrix} s & -1 & 0 \\ 0 & s & -1 \\ a_o + Kbof_1 & a_1 + Kbof_2 & s + (a_2 + Kbof_3) \end{bmatrix}$$

A esta matriz debemos invertirla y multiplicarla por el vector N el cual consta de la suma de las condiciones iniciales con la referencia, para obtener de ésta manera el vector de

estados en el dominio de Laplace. El resultado de estas operaciones es el siguiente:

$$X(s) = \frac{1}{\det} \begin{bmatrix} a11 & a12 & a13 \\ a21 & a22 & a23 \\ a31 & a32 & a33 \end{bmatrix} \begin{bmatrix} x1(0) \\ x2(0) \\ x3(0) + K\ bo\ R/s \end{bmatrix}$$

donde:

$$\det = \det [s \underline{I} - (\underline{A} - K \underline{B} \underline{f})]$$

$$\det = s^3 + s^2 (a_2 + K\ bo\ f_3) + s (a_1 + K\ bo\ f_2) + (a_o + K\ bo\ f_1)$$

$$a11 = s^2 + s (a_2 + K\ bo\ f_3) + (a_1 + K\ bo\ f_2)$$

$$a12 = s + (a_2 + K\ bo\ f_3)$$

$$a13 = 1$$

$$a21 = - (a_o + K\ bo\ f_1)$$

$$a22 = s^2 + s (a_2 + K\ bo\ f_3)$$

$$a23 = s$$

$$a31 = - s (a_o + K\ bo\ f_1)$$

$$a32 = - s (a_1 + K\ bo\ f_2) - (a_o + K\ bo\ f_1)$$

$$a33 = s^2$$

Realizando el producto de la matriz por el vector N de condiciones iniciales y referencia, se tiene cada estado en el dominio de Laplace:

$$X_1(s) = \{ \ s^3 X_1(0) + s^2 [X_1(0)(a_2 + K\ bo\ f_3) + X_2(0)] + s [X_1(0)(a_1 + K\ bo\ f_2) + X_2(0)(a_2 + K\ bo\ f_3) + X_3(0)] + K\ bo\ R \ \} / (\det s)$$

$$X_2(s) = \{ \; s^2 X_2(0) + s [X_2(0)(a_2 + K\, b_o f_3) + X_3(0)] + K\, b_o R - X_1(0)(a_o + K\, b_o f_1) \; \} / \det$$

$$X_3(s) = \{ \; s^2 X_3(0) + s [K\, b_o R - X_1(0)(a_o + K\, b_o f_1) - X_2(0)(a_1 + K\, b_o f_2)] - X_2(0)(a_o + K\, b_o f_1) \; \} / \det$$

Para estabilizar al sistema se deberá ubicar los raíces de lazo cerrado en el semiplano izquierdo del plano "**s**", éstos deben ser tres, si son complejos dos se ellos deberán ser conjugados.

Si los polos son p1, p2, p3, entonces de la multiplicación de éstos resulta el polinomio característico de la matriz

$[s\, I - (A - K\, B\, f)]$

$$(s - p1)(s - p2)(s - p3) = s^3 + a_{lc2} s^2 + a_{lc1} s + a_{lc0}$$

$$= s^3 + s^2 (a_2 + K\, b_o f_3) + s (a_1 + K\, b_o f_2) + (a_o + K\, b_o f_1)$$

En lazo cerrado se tendrá que la función de trnasferencia toma la siguiente forma:

$$T(s) = \frac{K\, b_o}{a_{lc0} + a_{lc1} s + a_{lc2} s^2 + s^3}$$

Donde se tiene que para que se tenga un error de posición nulo se debe cumplir que:

$$K\, b_o = a_{lc0}$$

de donde obtenermos el valor de la ganancia:

$$K = \frac{a_{lc0}}{b_o}$$

Luego se encuentra el vector de realimentación f = [f1, f2,

f3] despejando de la siguiente ecuación .

$$(s - p1)(s - p2)(s - p3) = s^3 + a_{1c2}s^2 + a_{1c1}s + a_{1c0}$$

$$= s^3 + s^2 (a_2 + K\, b_o\, f_3) + s (a_1 + K\, b_o\, f_2) + (a_o + K\, b_o f_1)$$

de donde se tiene que:

$$f_1 = \frac{a_{1c0} - a_o}{K\, b_o}$$

$$f_2 = \frac{a_{1c1} - a_1}{K\, b_o}$$

$$f_3 = \frac{a_{1c2} - a_2}{K\, b_o}$$

De esta manera una vez obtenidos los elementos del vector de realimentación f, y de la ganancia K; se los reemplaza en cada uno de los estados para posteriormente aplicar la transformada inversa de Laplace y obtener la respuesta en el tiempo.

Se debe mencionar que por la presencia de la referencia r la forma de los estados difiere, como se observa a continuación:

$$X_1(s) = \frac{k_{31}\, s^3 + k_{21}\, s^2 + k_{11}\, s + k_{01}}{s(\, s^3 + a_{1c2}\, s^2 + a_{1c1}\, s + a_{1c0}\,)}$$

$$X_2(s) = \frac{k_{22}\, s^2 + k_{12}\, s + k_{02}}{s^3 + a_{1c2}\, s^2 + a_{1c1}\, s + a_{1c0}}$$

$$X_3(s) = \frac{k_{23}\, s^2 + k_{13}\, s + k_{03}}{s^3 + a_{1c2}\, s^2 + a_{1c1}\, s + a_{1c0}}$$

donde las constantes k_{31}, k_{21}, k_{11}, k_{01}, k_{22}, k_{12}, k_{02} estan ya determinadas y dependen de las condiciones iniciales, de los polos de lazo abierto, y de la amplitud de la referencia que se desee.

Para que la obtención de la transformada inversa de Laplace sea fácil se divide a cada ecuación de estado en fracciones parciales. La forma de estas fracciones parciales dependerá de la ubicación de los polos de lazo cerrado que se ingresen, por lo tanto, como se deben ingresar tres polos de lazo cerrado, se tienen cuatro formas de ingresar estas raíces, con lo cual la respuestas en el tiempo del el espacio y velocidad de la esfera para cada caso son:

a) Todos los polos reales y diferentes:

Para el espacio se tiene que:

$$X_1(s) = \frac{A}{s} + \frac{B}{s - p_1} + \frac{C}{s - p_2} + \frac{D}{s - p_3}$$

de donde:

$$x_1(t) = A + B\, e^{p_1 t} + C\, e^{p_2 t} + D\, e^{p_3 t}$$

Para la velocidad se tiene:

$$X_2(s) = \frac{A}{s - p_1} + \frac{B}{s - p_2} + \frac{C}{s - p_3}$$

de donde:

$$x_2(t) = A\, e^{p_1 t} + B\, e^{p_2 t} + + C\, e^{p_3 t}$$

b) Todos los polos reales e iguales.

Para el espacio se tiene que:

$$X_1(s) = \frac{A}{s} + \frac{B}{s - p_1} + \frac{C}{(s - p_1)^2} + \frac{D}{(s - p_1)^3}$$

de donde:

$$x_1(t) = A + (B + C\, t + \frac{D}{2}\, t^2)\, e^{p_1 t}$$

Para la velocidad se tiene:

$$X_2(s) = \frac{A}{s - p_1} + \frac{B}{(s - p_1)^2} + \frac{C}{(s - p_1)^3}$$

se tiene:

$$x_2(t) = (A + B\, t + \frac{C}{2}\, t^2)\, e^{p_1 t}$$

c) Dos polos reales iguales y un polo real diferente.

Para el espacio se tiene que:

$$X_1(s) = \frac{A}{s} + \frac{B}{s - p_1} + \frac{C}{(s - p_1)^2} + \frac{D}{s - p_2}$$

es decir:

$$x_1(t) = A + (B + C\,t)\; e^{p_1 t} + D\, e^{p_2 t}$$

Para la velocidad se tiene:

$$X_2(s) = \frac{A}{s - p_1} + \frac{B}{(s - p_1)^2} + \frac{C}{s - p_2}$$

se tiene:

$$x_2(t) = (A + B\,t)\; e^{p_1 t} + C\, e^{p_2 t}$$

d) Dos polos complejos conjugados y un polo real.

Para el espacio se tiene que:

$$X_1(s) = \frac{A}{s} + \frac{B(s - \sigma_1)}{(s - \sigma_1)^2 + \beta_1^2} + \frac{C\,\beta_1}{(s - \sigma_1)^2 + \beta_1^2} + \frac{D}{s - p_2}$$

se tiene:

$$x_1(t) = A + B\, e^{\sigma_1 t}\cos(\beta_1 t) + C\, e^{\sigma_1 t}\sin(\beta_1 t) + D\, e^{p_2 t}$$

Para la velocidad se tiene que:

$$X_2(s) = \frac{A(s - \sigma_1)}{(s - \sigma_1)^2 + \beta_1^2} + \frac{B\beta_1}{(s - \sigma_1)^2 + \beta_1^2} + \frac{C}{s - p_2}$$

se tiene:

$$x_2(t) = A\, e^{\sigma_1 t} \cos(\beta_1 t) + B\, e^{\sigma_1 t} \sin(\beta_1 t) + C\, e^{p_2 t}$$

Con esto se ha analizado completamente el caso de la realimentación de estado.

3.3.- COMPENSACION POR RED DE ADELANTO DE FASE. METODO DE LA BISECTRIZ.

Para el diseño de este Compensador, y usando el método del L.G.R., como ya se mencionó la red de compensación debe ser una red de Adelanto de fase; para la determinación de los parámetros de dicha red se va a utilizar dos de los métodos más difundidos.

En primer lugar se debe encontrar cualés son las raíces dominantes deseadas para una adecuada compensación, tomando en consideración las especificaciones de respuesta transitoria y estabilidad relativa. A partir del conocimiento del sistema original, el tiempo mínimo en el que debe actuar el sistema de control a implementarse, debe ser menor al tiempo en el cual la esfera sale de la zona de influencia del electroimán, habiéndose encontrado por la experimentación del equipo que el tiempo de 1

segundo es el adecuado. De la modelación del Sistema se sabe que éste es de tercer orden, pero por otra parte el sistema de control que se va a implementar debe tener constante la parte real de las raíces deseadas.

La dominancia es importante con el fin de que el sistema se comporte como uno de segundo orden que cumpla con las especificaciones deseadas.

Para el sobreimpulso no es conveniente que éste supere el 40% ya que si se producen elevadas variaciones de amplitud en la entrada, la posición de la esfera puede sobrepasar los límites de la zona de influencia definida por la ubicación de la fotocelda y el electroimán.

La función de transferencia de la red de adelanto tiene la siguiente estructura:

$$G_{C(s)} = K_C \frac{s + c_C}{s + p_C}$$

A continuación en la figura 3.5 se indica el diagrama de bloques del sistema de control resultante:

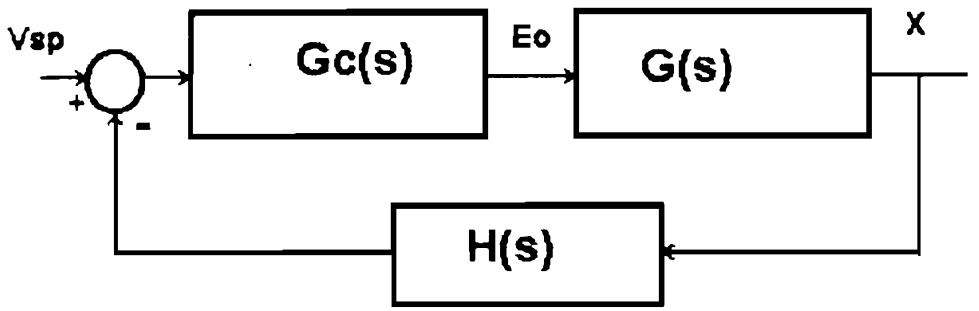


Figura 3.5 Diagrama de Bloques del Sistema de Control con la red de Adelanto de Fase Gc(s).

Para la red de adelanto el cero siempre está ubicado a la derecha del polo en el plano complejo.

Con las especificaciones de funcionamiento de ts = 0.4 seg y un M_p = 20 % se determina la posición deseada de las raíces dominantes de lazo cerrado, de la manera siguiente:

$$\epsilon = \sqrt{\frac{\ln Mp^2}{\pi^2 + \ln Mp^2}}$$

$$W_n = \frac{4}{\epsilon\, t_s}$$

$$p_{D_{1,2}} = -\epsilon W_n \pm W_n\sqrt{1 - \epsilon^2}$$

$$p_{D_1} = -10 + j19.5$$
$$p_{D_2} = -10 - j19.5$$

Se calcula el ángulo que debe dar la red de adelanto para que el L.G.R. pase por los puntos deseados,esto es reemplazando "s" por las raíces dominantes deseadas y sin considerar la fase del compensador se tiene:

$$\Phi \mid G_{COMPENSADOR}(s)\; G_{PLANTA}(s)\; H(s) \mid_{p_{D_{1,2}}} = -180°$$

$$\Phi_{PLANTA} + \Phi_{COMPENSADOR} = -180°$$

$$\Phi \mid \frac{47.7\;13.2}{(s + 21.3)(s - 21.3)(s + 104.6)} \mid_{p_{D_{1,2}}} = -219°$$

de donde:

$$\Phi_{COMPENSADOR} = 39°$$

Por lo que el compensador debe aportar con una fase de 39°. Si se utiliza el **Método de la Bisectriz** Ref 7 , para ubicar el polo y cero del compensador se modo que este aporte con la fase

---

7/ Ingeniería de Control Moderna. OGATA.

calculada se obtiene que:

$$P_{COMPENSADOR} = -\epsilon W_n - \frac{W_n\sqrt{1 - \epsilon^2}}{\tan((\Phi_{p_D} - \Phi_{COMPENSADOR}) / 2)}$$

$$C_{COMPENSADOR} = -\epsilon W_n - \frac{W_n\sqrt{1 - \epsilon^2}}{\tan((\Phi_{p_D} + \Phi_{COMPENSADOR}) / 2)}$$

Se determina entonces:

$$P_{COMPENSADOR} = -34$$
$$C_{COMPENSADOR} = -14.2$$

A continuación se indica en la figura 3.6 la forma gráfica del método de la bisectriz, para la ubicación del polo y cero de la red de adelanto de fase.

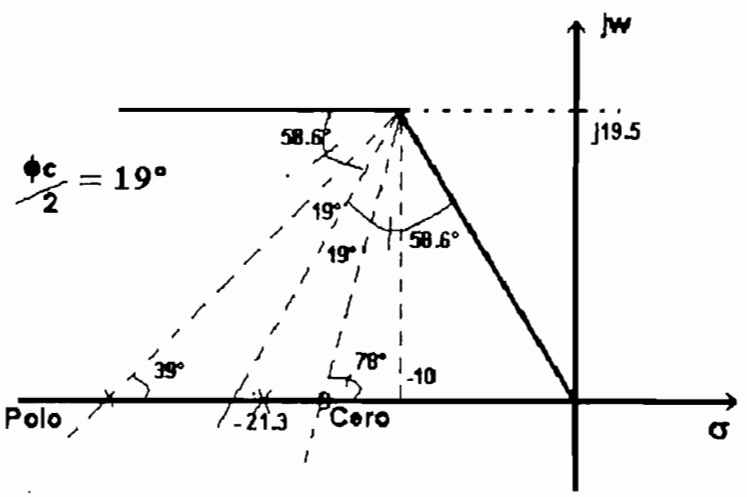


Figura 3.6 Método de la bisectriz para ubicar el polo y el cero del compensador.

Se determina la ganancia de lazo abierto del sistema compensado con la condición de módulo.

$$\left| G_{COMPENSADOR}(s)\ G_{PLANTA}(s)\ H(s) \right|_{p_{D_{1,2}}} = 1$$

$$\left| \frac{K_{COMPENSADOR}(s + 14.2)\ 47.7\ 13.2}{(s + 34)(s + 21.3)(s - 21.3)(s + 104.6)} \right|_{p_{D_{1,2}}} = 1$$

$$K_{COMPENSADOR} = 200$$

Por lo tanto la función de transferencia de la compensación buscada es:

$$G_{COMPENSADOR}(s) = 200\frac{(s + 14.2)}{(s + 34)}$$

Por lo que se debe incluir una ganancia de 200; la atenuación de la red es de 34/14.2 = 2.39

Esta red compensadora obliga a que el L.G.R. pase por las raíces deseadas, pero no determina la dominancia de las mismas lo cual observamos si obtenemos las raíces de lazo cerrado.

$$p_1 = -2.4$$
$$p_2 = -116.5$$
$$p_3 = -10+j19.5$$
$$p_4 = -10-j19.5$$

La raiz que está ubicada más cerca del eje "jw" del plano

"s" es la de mayor dominancia, por lo cual el sistema se comporta como uno de primer orden pero cuyas especificaciones son impredecibles.

Las raíces de lazo cerrado no dominantes modifican la respuesta obtenida por las raíces de lazo cerrado dominantes. El valor de esa modificación depende de la posición de estas raíces de lazo cerrado remanentes.

En la figura 3.7 se indica el L.G.R. de la planta compensada por este método.

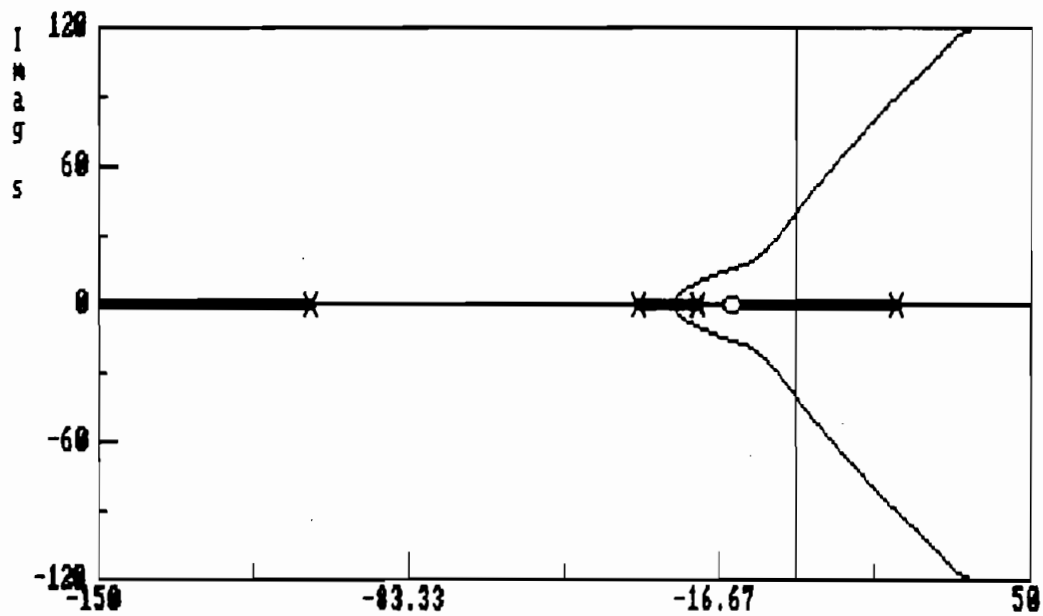


Figura 3.7 L.G.R. del Sistema compensado por el Método de la Bisectriz.

3.4.- COMPENSACION POR RED DE ADELANTO DE FASE. METODO DEL CERO ARBITRARIO.

Este tipo de compensación sirve para diseñar una red de adelanto de fase, siendo otra alternativa para obtener de mejor manera los parámetros de la red compensadora, básicamente consiste en colocar al cero de la función de trasferencia de la red de adelanto en un lugar conveniente, y luego se calcula la posición del polo de tal manera que se verifique la fase para el compensador. Este se lo conoce como el método del cero arbitrario. Ref [8]

Si se considera el mismo caso anterior, se puede asignar como un buen valor para la ubicación del cero el valor de -30. El compensador debe contribuir como se vió antes con una fase de 39° con lo cual se tiene:

$$\Phi \mid K_{COMPENSADOR}\frac{(s + c_{COMPENSADOR})}{(s + p_{COMPENSADOR})} \mid_{p_{D_{1,2}}} = 39°$$

es decir se tiene evaluando la expresión anterior:

---

[8] / Retroalimentación y Sistemas de Control. DiStefano, Stubberud, Williams.

$$\Phi \mid (-10+j19.5+30) - (-10+j19.5+p_{COMPENSADOR}) \mid = 39$$

$$44.7^\circ - \arctan\left(\frac{19.5}{p_{COMPENSADOR} - 10}\right) = 39^\circ$$

de donde se obtiene:

$$p_{COMPENSADOR} = -249$$

La ganancia del compensador se la calcula de manera análoga al método anterior, esto es por la condición del módulo. De esta manera se obtiene la siguiente función de transferencia para ésta manera de calcular los parámetros de la red de adelanto de fase.

$$G_{COMPENSADOR}(s) = 1097\ \frac{(s+30)}{(s+249)}$$

Donde se debe incluir una ganancia de 1097; la atenuación de la red es de 249/30. Con el compensador ya calculado se puede encontrar las raíces de lazo cerrado, los cuales se indican a continuación, se observa claramente la dominancia, por lo cual la respuesta a una entrada paso verificará los requerimientos de tiempo de establecimiento y de máximo sobreimpulso que se establecieron para el cálculo del compensador.

$$p_1 = -10+j19.5$$
$$p_2 = -10-j19.5$$
$$p_3 = -69.9$$
$$p_4 = -264.1$$

A continuación se indica en la figura 3.8 el L.G.R. del sistema completo, debido a la ubicación del cero a la izquierda de los dos polos de lazo abierto, se dispone ahora de un lugar geométrico de las raíces en el cual existe estabilidad del sistema para determinado rango de ganancias.

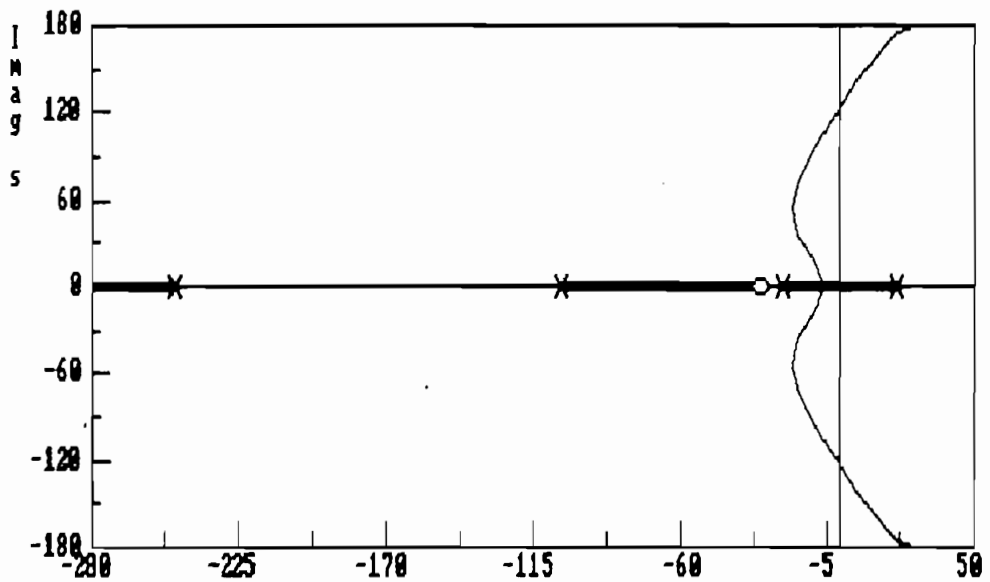


Figura 3.8 L.G.R. del Sistema compensado por el Método del Cero Arbitrario.

3.5.- COMPENSACIONES POR ACCIONES DE CONTROL.

Por la importancia que tiene la compensación con la utilización de las acciones de control se revisan las mejores alternativas a continuación:

Como se mencionó antes por la observación del L.G.R. del sistema no compensado, ningún valor de ajuste de ganancia hace que el sistema sea estable, por lo tanto la acción de control proporcional pura no es solución para el sistema de control que se trata implementar.

Utilizando el mismo razonamiento se puede observar en las figuras 3.9.a, y 3.9.b que las acciones de control puras tanto integral, como derivativa tampoco ayudan a estabilizar al sistema, ya que igual siguen manteniendo ramas del L.G.R. en el semiplano derecho del plano "s".

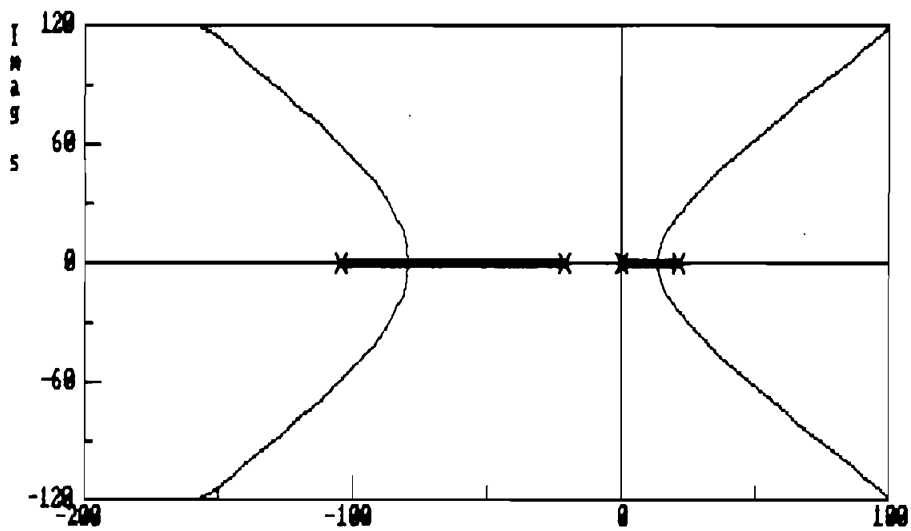


Figura 3.9.a L.G.R. del Sistema compensado por la Acción de Control Integral pura.

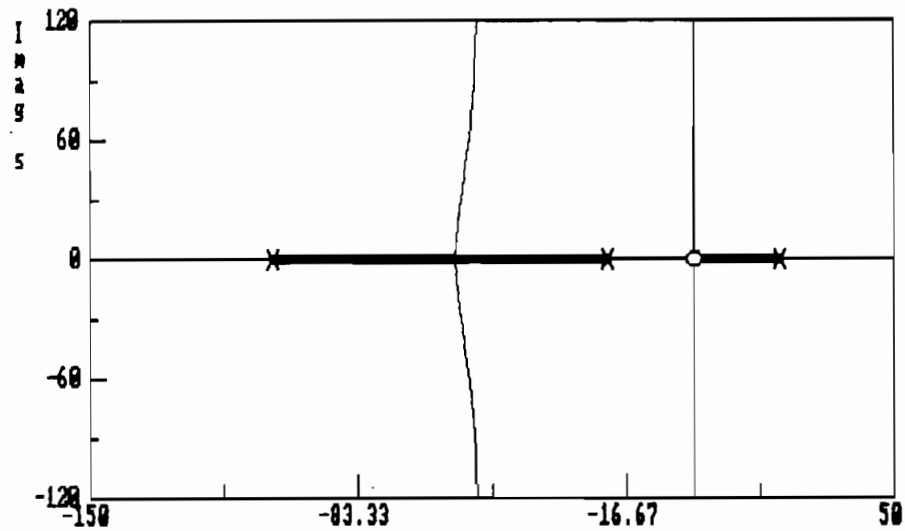


Figura 3.9.b L.G.R. del Sistema compensado por la Acción de Control Derivativa pura.

A continuación se analiza como las acciones de control combinadas afectan al comportamiento del sistema sin compensar.

3.5.1.- Acción de Control Proporcional Integral.

En la figura 3.10 se indica el diagrama de bloques del sistema de control total con la acción de control proporcional integral.

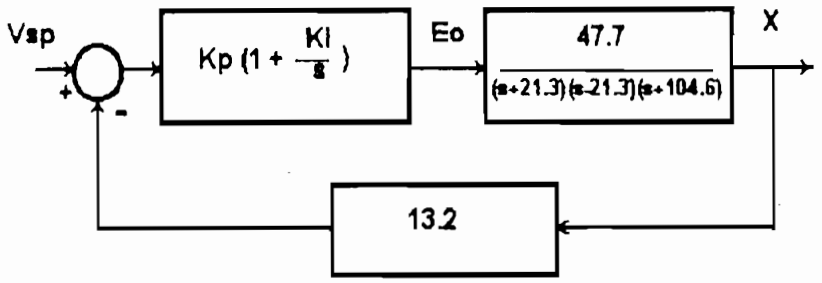


Figura 3.10 Diagrama de Bloques con
la Acción de Control Proporcional Integral.

Del diagrama anterior se observa que esta acción de control añade un polo en el eje "jw", sabiendo que el agregado de un polo a la función de transferencia de lazo abierto tiene el efecto de desplazar el lugar de las raices hacia la derecha tendiendo a hacer más lento el establecimiento de la respuesta.

de donde resulta que:

$$1 + G{\cdot}H(s) = 1 + 627\,K_p \frac{(s + K_I)}{s\,(s + 21.3)\,(s - 21.3)\,(s + 104.6)}$$

Si se aplican las reglas para la construcción del L.G.R. nos permite obtener la siguiente información:

De las asíntotas se sabe:

$$A_1 = 60\ ^\circ$$

$$A_2 = 180\ ^\circ$$

$$A_3 = 300\ ^\circ$$

De la intersección de las Asíntotas se tiene:

$$\sigma_A = (\Sigma\ Pi - \Sigma\ Zi)/(n - m)$$

$$\sigma_A = Ki/3 - 35.5$$

La ecuación de los puntos de corte del L.G.R. con el eje real:

$$3s^4 + (209.6+4K_1)s^3 + (313.8K_1-452.4)s^2 - 904.8K_1s - 47321K_1 = 0$$

Para la estabilidad interesa que el L.G.R. se mantenga en el semiplano izquierdo, se hace que la intersección de las asíntotas sea un punto en el semiplano izquierdo, esto es $\sigma_A < 0$, por lo cual resulta la siguiente condición:

$$Ki < 106.5$$

Si se analiza diferentes valores de Ki (que cumplan con la condición), se obtiene que el corte del L.G.R. con el eje real siempre se encuentra ubicado en el semiplano derecho del plano "s", por lo cual ésta acción de control no estabiliza al sistema. En las figuras 3.11.a y 3.11.b se indican los L.G.R. para diferentes valores de Ki que cumplen con la condición encontrada, determinandose que no es estable el sistema, con este tipo de compensación.

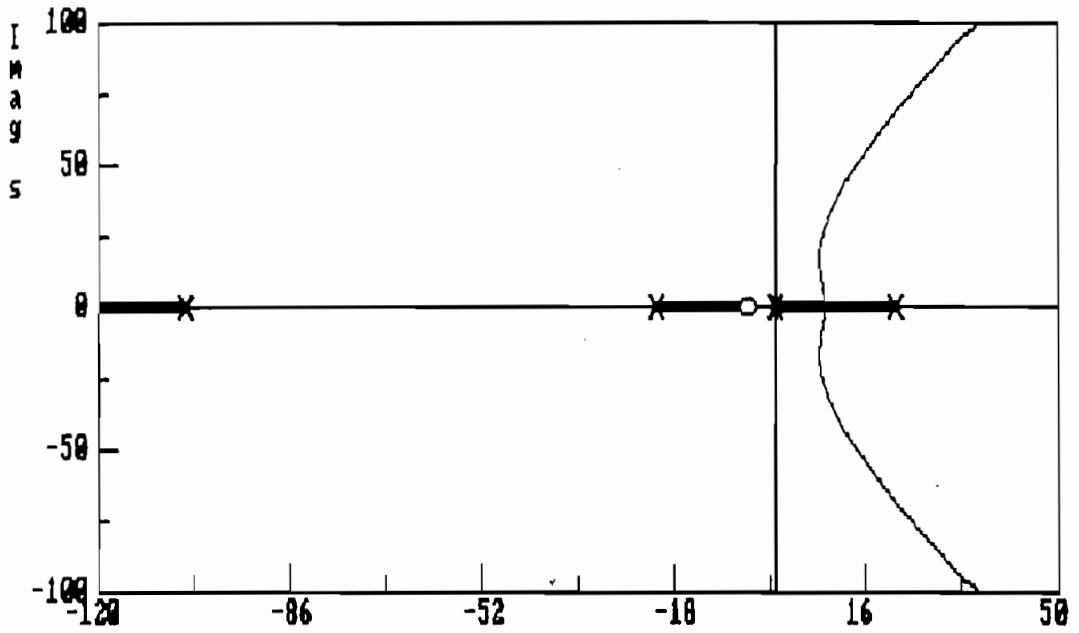


Figura 3.11.a Compensación Proporcional Integral. Ki = 5

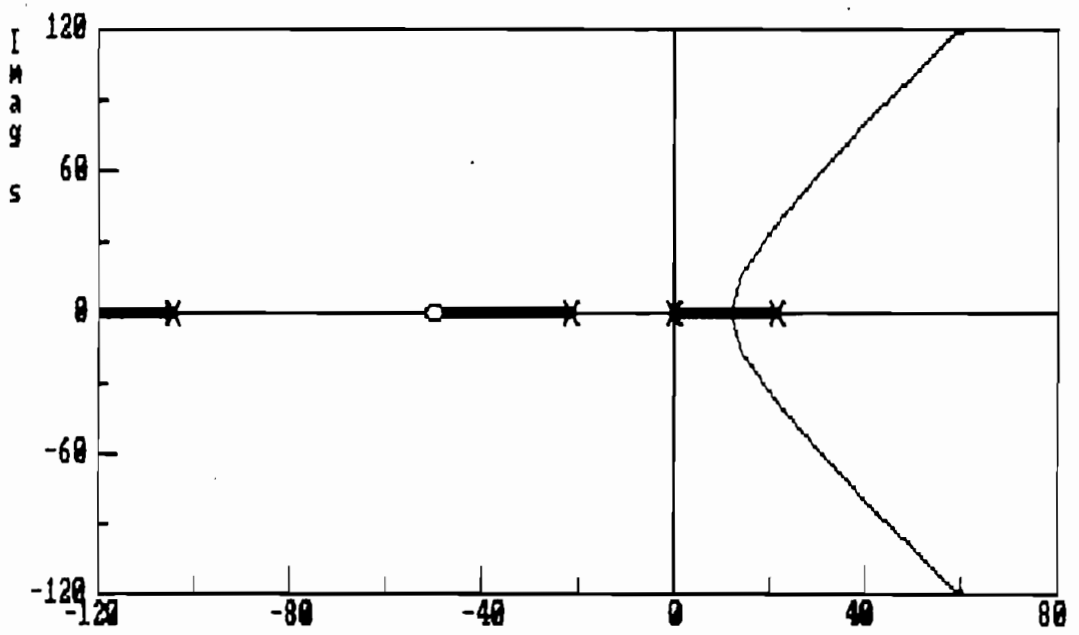


Figura 3.11.b Compensación Proporcional Integral. Ki = 50

Si se sobrepasa la condición para que la intersección de asíntotas se de en el semiplano izquierdo se tiene que se modifica el L.G.R. de la manera como se indica en la figura 3.12

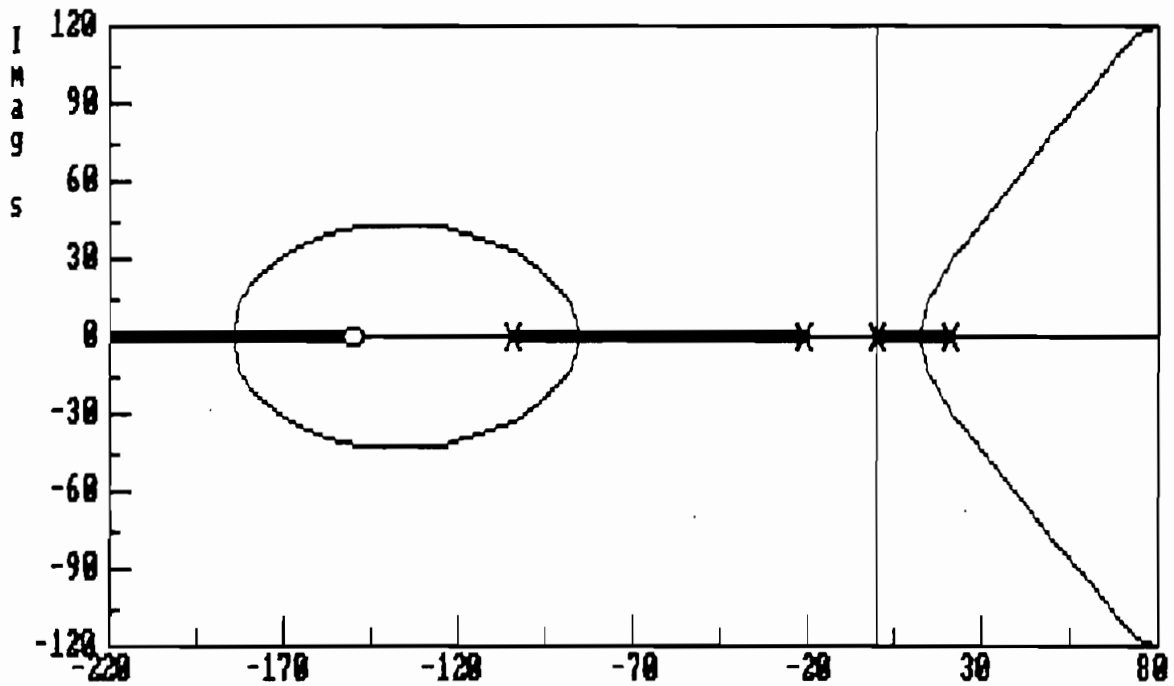


Figura 3.12 Compensación Proporcional Integral. Ki = 150

3.5.2.- Acción de Control Proporcional Derivativa.

A continuación se indica el diagrama de bloques de esta acción de control:

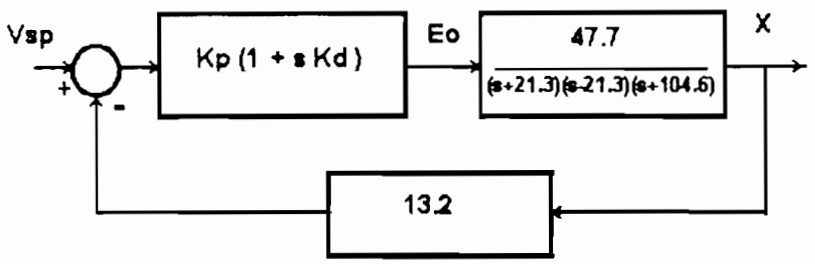


Figura 3.13 Diagrama de Bloques con
la Acción de Control Proporcional Derivativa.

Esta acción de control añade un cero al L.G.R.; el agregado de un cero a la función de transferencia tiene el efecto de desplazar el lugar de la raíz hacia la izquierda, tendiendose a hacer el sistema más estable y a acelerar el establecimiento de la respuesta, esto es cierto grado de anticipación al sistema, con aceleración de la respuesta transitoria.

De manera similar se emplea la técnica del L.G.R. para la determinación de los parámetros de esta acción de control. Por las reglas de construcción del L.G.R. se tiene:

$$1 + G{\cdot}H(s) = 627\,K_p K_d \frac{(s + \frac{1}{K_d})}{(s + 21.3)(s - 21.3)(s + 104.6)}$$

de lo cual se puede obtener la información de las asíntotas de la manera siguiente:

Asíntotas:

$$A_1 = 90\ °$$

$$A_2 = 270\ °$$

Intersección de las asíntotas:

$$\sigma_A = (\Sigma\ Pi - \Sigma\ Zi)/(n - m)$$

$$\sigma_A = (1/(2Kd)) - 52.3$$

Ecuación de los puntos de corte del L.G.R. con el eje real:

$$2s^3 + (104.6 + \frac{3}{K_d})s^2 + (\frac{209.2}{K_d})s + (47322 - \frac{452.4}{K_d}) = 0$$

Interesa que el L.G.R. se desplace hacia la izquierda del plano "s", esto es el corte de las asíntotas se de en el semiplano izquierdo:

$$\sigma_A < 0$$

Con lo que:

$$1/\ Kd < 104.6$$

Si se reemplaza en la ecuación anterior valores de 1/ Kd que cumplan la condición impuesta se obtienen L.G.R. que demuestran la factibilidad de estabilizar al sistema. En las figuras 3.14.a y 3.14.b se indican dos valores para la condición impuesta.

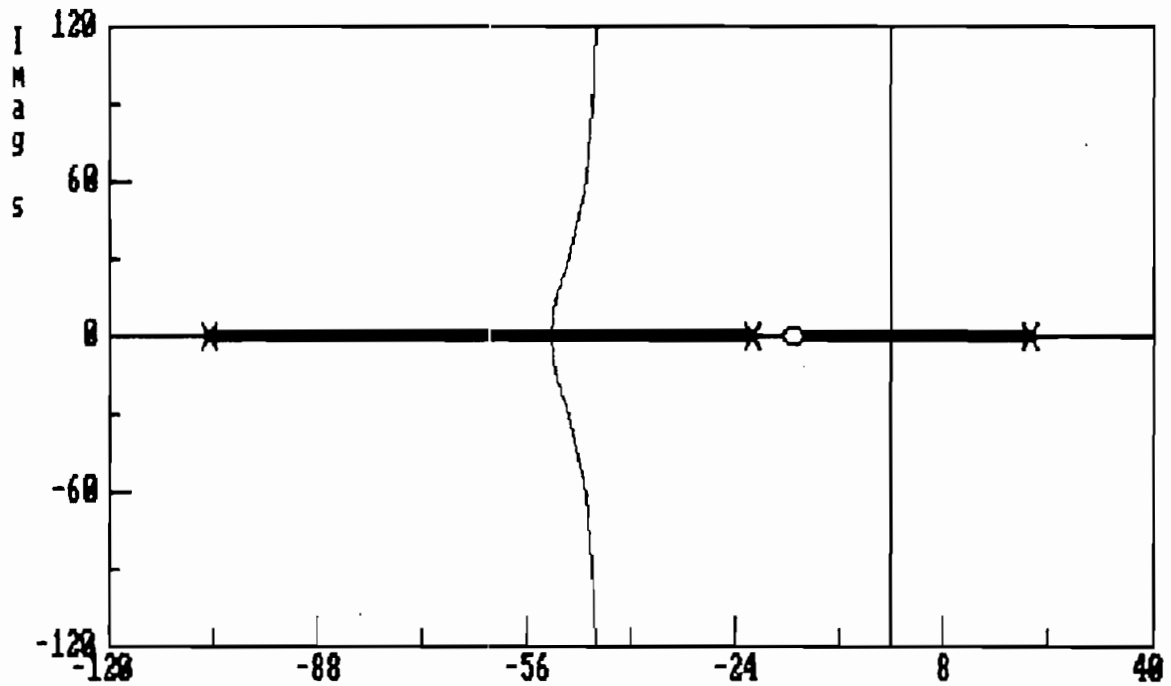


Figura 3.14.a Compensación Proporcional Derivativa. (1/Kd) = 15

Cuando se toma el valor de (1/Kd) = 15 es posible obtener una respuesta del sistema sin sobreimpulso y de la rapidez requerida.

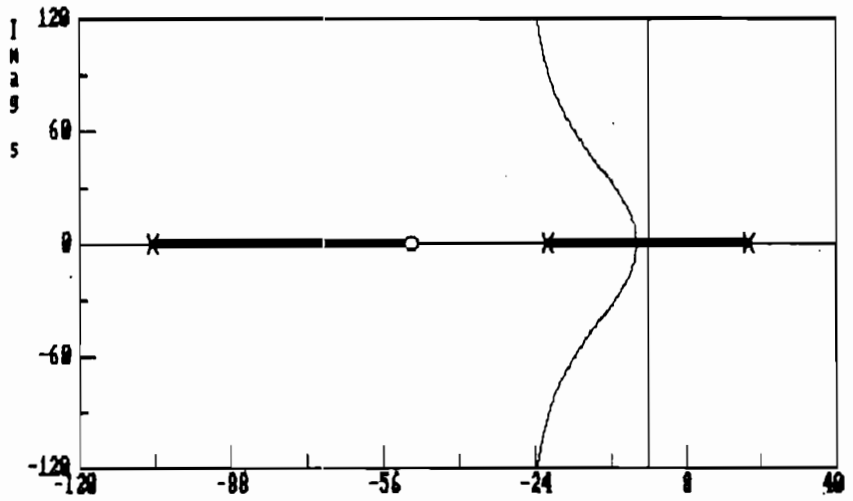


Figura 3.14.b Compensación Proporcional Derivativa. (1/Kd) = 50

Si se sobrepasa la condición para la intersección de las asíntotas en el semiplano izquierdo del plano "s" se obtiene el siguiente L.G.R. que se indica en la figura 3.15.

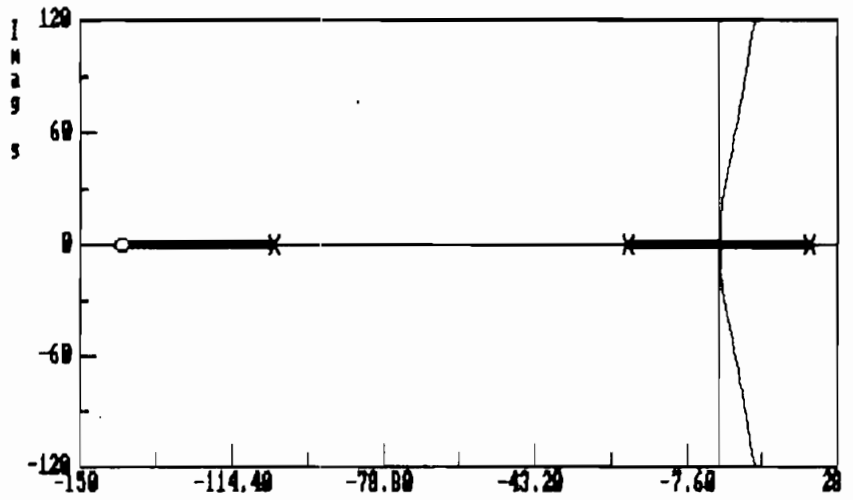


Figura 3.15 Compensación Proporcional Derivativa. (1 / Kd) = 140

A continuación se va a analizar los resultados que se obtienen cuando deseamos obtener raíces de lazo cerrado dominantes. Si tomamos las raíces dominantes iguales para la red de adelanto de fase que se analizó anteriormente:

$$p_1 = -10+j19.5$$

$$p_2 = -10-j19.5$$

Se tiene los siguientes resultados:

Por la condición de fase que se debe cumplir se tiene que:

$$\Phi \left| \frac{627 K_p K_d (s + \frac{1}{K_d})}{(s + 21.3)(s - 21.3)(s + 104.6)} \right|_{p_{D_{1,2}}} = -180°$$

en los polos deseados se debe cumplir.

$$\Phi|(s+\tfrac{1}{K_d})| - \Phi|(s+21.3)| - \Phi|(s-21.3)| - \Phi|(s+104.6)| = -180°$$

de aqui se puede obtener el parámetro de la parte derivativa 1/Kd del compensador, por lo que:

$$\Phi \left|(-10+\tfrac{1}{K_d}) + j19.5\right| - 59.9° - 148.1° - 11.7° = -180°$$

$$\Phi \left|(-10+\tfrac{1}{K_d}) + j19.5\right| = 39.7°$$

$$\frac{1}{K_d} = 33.5$$

de la condición de módulo se obtiene la constante proporcional

Kp de la siguiente manera:

$$\frac{627\ K_p\ K_d\ |235.5+j19.5|}{|11.3+j19.5|\ |-31.3+j19.5|\ |94.6+j19.5|} = 1$$

de donde:

$$K_p\ K_d = 4.2$$

como antes se obtuvo. 1/Kd = 33.5, se tiene entonces que:

$$K_p = 140.3$$

Con estos valores de Kp, y de Kd se obtienen los resultados deseados, como se observa en el L.G.R. de la figura 3.16.

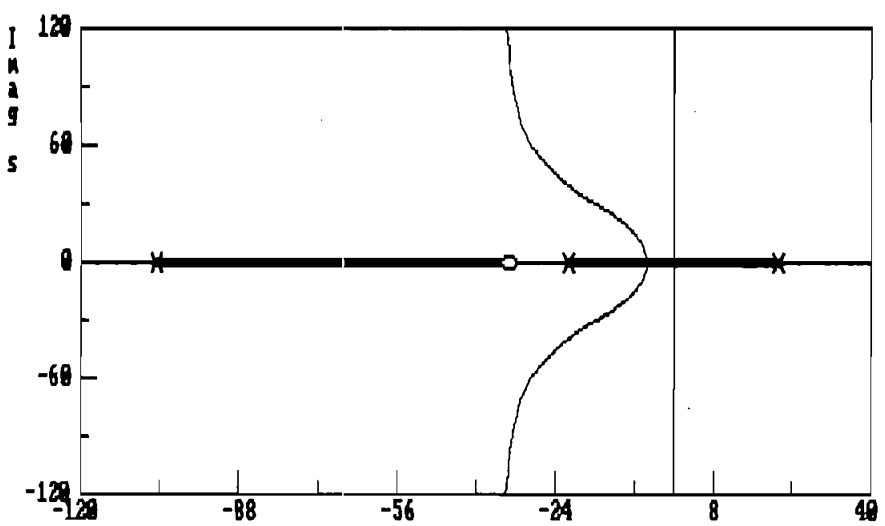


Figura 3.16 Compensación Proporcional derivativa con Kp =140.3 y 1/Kd = 33.5

3.5.3.- Acción de Control Proporcional Integral Derivativa.

A continuación se indica el diagrama de bloques de esta acción de control:

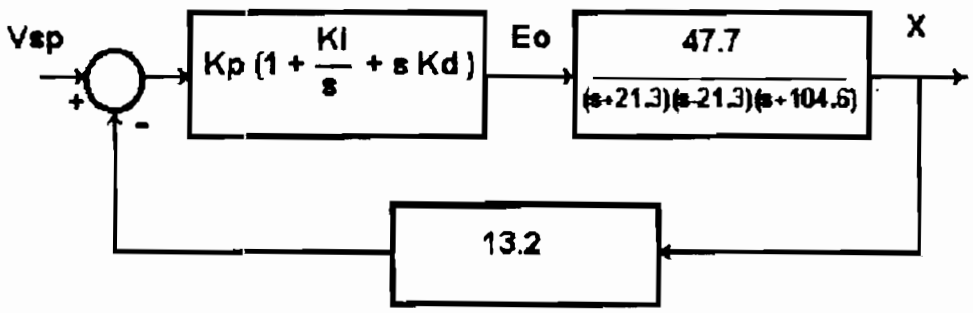


Figura 3.17 Diagrama de Bloques con la
Acción de Control Proporcional Integral Derivativa.

Como se analizó la acción proporcional integral no es una buena alternativa de control, debido a que este controlador tiene dos ceros y un polo, se puede decir que sería similar a un compensador por atraso adelanto, pero posee la desventaja de un polo en el origen, esto es un integrador, con lo cual el sistema difícilmente se estabilizará.

Existe una gran variedad en la metodología para la obtención de un compensador Proporcional Integral Derivativo,

por lo cual no se lo abordará, simplemente podemos decir que en el programa se pueden ingresar cualquier juego de valores para las contantes proporcional, integral, y derivativa, calculando en base a estos valores la respuesta del sistema con la posibilidad de partir con condiciones a la entrada, o variar la referencia de la esfera.

CAPITULO IV

SIMULACION DINAMICA

En este capítulo se revisan los algorítmos para la implementación de las diferentes compensaciones utilizadas para la estabilización del sistema y la simulación dinámica del problema, con el fin de entender el funcionamiento del programa. Además se indica el menú principal del programa desde el cual el usuario puede acceder fácilmente para el ingreso de datos de las compensaciones, condiciones iniciales, set point, realizar cálculos, animación, obtención de los gráficos de espacio, velocidad, impresión, ayudas.

4.1.- ALGORITMOS PARA EL CALCULO.

Cálculo del Compensador.- Recordando la función de transferencia de la planta y la del sensor del sistema, obtenidas en el capítulo 2.

$$G_{PLANTA}(s) = \frac{47.7}{(s + 21.3)(s - 21.3)(s + 104.6)}$$

$$H(s) = k_{sensor} = 13.2 \frac{Volt}{m}$$

Como ya se vió, se puede efectuar la compensación del sistema utilizando redes de compensación, ó acciones de control, ó realimentación de estado.

4.1.1. Redes de Compensación.

En el capitulo 2 se indicó la necesidad de que la red sea

una red de adelanto de fase, la cual tiene la siguiente función de transferencia:

$$G_{COMPENSADOR} = K_{COMP} \frac{(s - CEROC)}{(s - POLOC)} \qquad (4.1)$$

donde:

GCOMPENSADOR(s) Es la funcion de transferencia del compensador

KCOMP es la ganancia del compensador en el plano "s"

CEROC es la ubicacion del cero del compensador en el plano "s"

POLOC es la posición del polo del compensador en el plano "s"

Para los límites de tiempo de establecimiento y del sobreimpulso, la region del plano "s" apta para ubicar las raices de lazo cerrado dominantes, se indica a continuación en la figura 4.1:

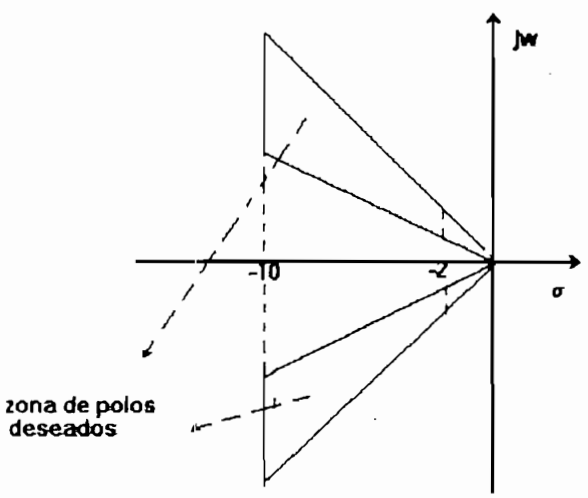


Figura 4.1 Zona apta para ubicar los polos de lazo cerrado

Dentro de las alternativas para encontrar los parámetros del compensador, se analiza las dos descritas en el capítulo 2 para entender el funcionamiento del programa.

Los dos métodos se diferencian sólo en la ubicación del cero y del polo del compensador, explicados a continuación:

Método de la Bisectriz. Según la referencia 6, el método de la bisectriz se aplica cuando se requiere encontrar los parámetros de un compensador de adelanto de fase, y consiste en encontrar geométricamente la ubicación del polo y cero de modo que se genere la fase a compensar.

Para este método, el programa necesita que se ingresen los valores de tiempo de establecimiento y de máximo sobreimpulso, para el cálculo de las raíces dominantes como se explicó en el capítulo 2.

Con la ayuda de la figura siguiente se puede encontrar las expresiones para la ubicación del cero y del polo del compensador.

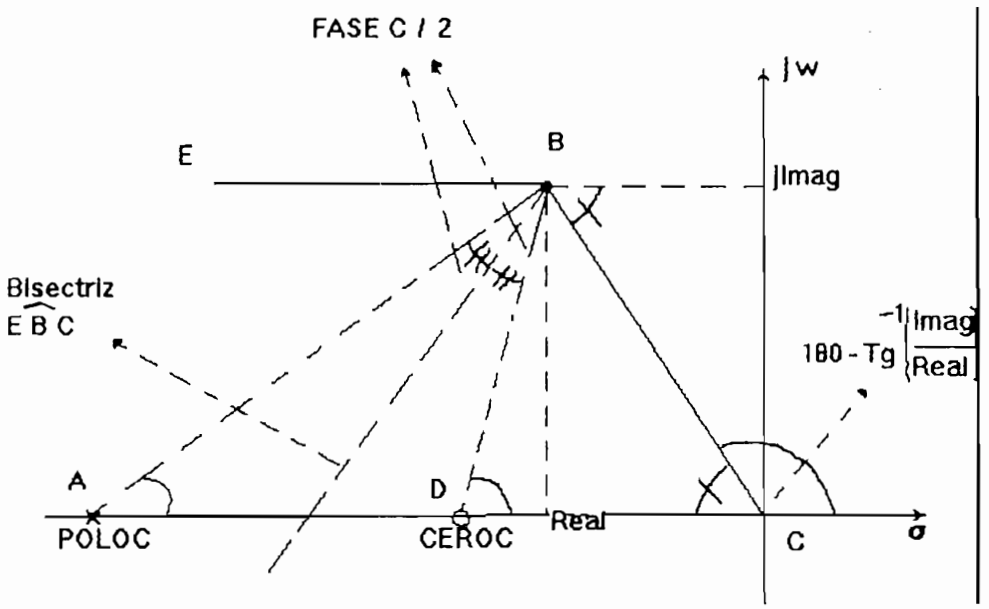


Figura 4.2 Metodo de la Bisectriz

De la figura las definiciones de las variables son las siguientes:

Real: la parte real de una de las raíces dominantes del lazo cerrado por donde deseamos que pase el L.G.R. del sistema compensado.

Imag: la parte imaginaria de una de las raíces dominantes del lazo cerrado por donde deseamos que pase el L.G.R. del sistema compensado.

CEROC: la ubicación del cero del compensador en el plano "s"

POLOC: la ubicación del polo del compensador en el plano "s"

FASEC: el ángulo que debe proporcionar el compensador.

De la figura anterior se puede encontrar la ubicación del CEROC, y del POLOC, en el plano complejo "s" de la manera siguiente:

Por la condición de fase se tiene que:

$$FASEC + FASE = -180^{\circ} \qquad (4.2)$$

donde FASE, es la fase que tiene el sistema sin compensar, en los polos deseados.

Por la construcción de la bisectriz en el triángulo ABC se tiene que, para la ubicación del polo del compensador.

$$POLOC = Real - \frac{Imag}{\tan \prec BAC} \qquad (4.3)$$

pero:

$$\prec BAC = \frac{1}{2}\left[180^{\circ} - \arctan\left(\frac{|Imag|}{|Real|}\right)\right] - \frac{1}{2} FASEC \qquad (4.4)$$

por lo tanto:

$$POLOC = Real - \frac{|Imag|}{\left[\frac{1}{2}\left[180^{\circ} - \arctan\left(\frac{|Imag|}{|Real|}\right)\right] - \frac{1}{2} FASEC\right]}$$

Para ubicar el cero del compensador en el triángulo BCD se tiene la siguiente relación:

$$CEROC = Real - \frac{Imag}{\tan(\prec BDC)} \qquad (4.6)$$

de manera similar al caso del polo se cumple que:

$$\prec BDC = \frac{1}{2}\left[180^{\circ} - \arctan\left(\frac{|Imag|}{|Real|}\right)\right] + \frac{1}{2} FASEC \qquad (4.7)$$

Con lo que queda determinada la posicion del cero del compensador mediante la siguiente ecuacion:

$$CEROC = Real - \frac{|Imag|}{[\frac{1}{2}[180° - \arctan(\frac{|Imag|}{|Real|})] + \frac{1}{2} FASEC]}$$

Queda por determinar en las ecuaciones anteriores el valor de FASEC (fase que debe proporcionar el compensador), se lo calcula a través de una rutina llamada "CalcFase", la cual se explica adelante en la parte de diagramas de flujo.

Método de la ubicación arbitraria del cero. Esta es la otra alternativa utilizada en el programa para encontrar los parámetros del compensador con adelanto de fase; basicamente consiste en colocar en un lugar adecuado el cero del compensador (para que se cumpla la dominancia de los polos deseados), para luego encontrar la ubicación del polo del compensador en el plano "s", satisfaciendo la fase requerida por el sistema a compensar.

Para este método el programa necesita que se ingresen los valores de las especificaciones deseadas, así como la ubicación que se considere adecuada (para obtener la dominancia de los polos deseados), del cero del compensador.

En la figura 4.3 se indican las tres regiones posibles para la ubicación del cero del compensador.

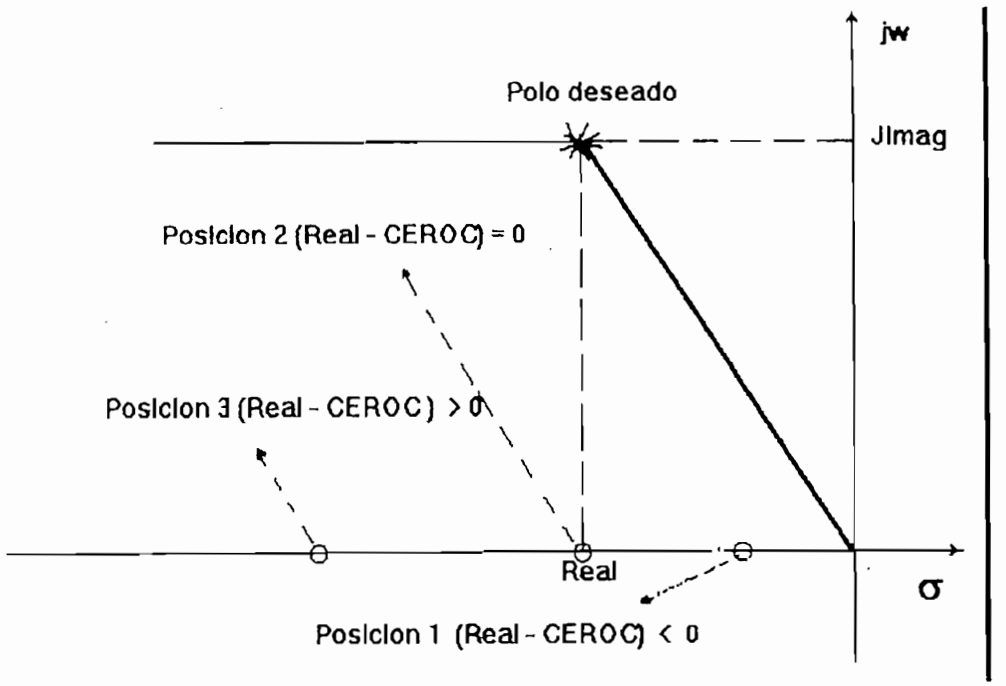


Figura 4.3 Posibles regiones para la ubicacion del Cero

Los valores de (Real +/- J Imag) se los obtiene de las especificaciones deseadas.

Si se substituye en la ecuación 4.9 el valor de s por (Real + jImag); y se calcula la fase para aquel valor de "s", se obtiene la fase del compensador en uno de los polos deseados, es decir:

$$Fase\ [K_{COMP}\ \frac{s\ -\ CEROC}{s\ -\ POLOC}]\ \Big|_{s\ =\ Real\ +\ jImag}\ =\ FASEC \qquad (4.9)$$

por lo que se tiene:

$$\tan(Real+jImag - CEROC) - \tan(Real+jImag - POLOC) = \ldots\tan(FASEC) \qquad (4.10)$$

ecuacion que reordenándola da:

$$\tan[\arctan(\frac{|Imag|}{Real - CEROC}) - FASEC] = [\frac{|Imag|}{Real - POLOC}]$$

Considerando las tres regiones posibles para la ubicación del Cero C, se puede obtener el valor de la ubicación del Polo C en el plano complejo "s"; el término (Real - Cero C) cambia en cada región:

a) Cero C ubicado a la izquierda de Real

$$POLOC = Real - \frac{|Imag|}{\tan[\arctan(\frac{|Imag|}{Real - CEROC}) - FASEC]}$$

b) Cero C ubicado justo en la parte Real

$$POLOC = Real - \frac{|Imag|}{\tan(90° - FASEC)}$$

c) Cero C ubicado a la derecha de Real

$$POLOC=Real-\frac{|Imag|}{\tan[180°-\arctan(\frac{|Imag|}{Real-CEROC})-FASEC]}$$

De las tres relaciones anteriores los valores de CEROC y POLOC se refieren a la ubicación del cero y del polo del compensador en el eje real negativo del plano "s".

Para los dos métodos descritos anteriormente hace falta conocer el valor de FASEC, que es el valor de la fase que debe introducir el compensador en la red de adelanto de fase. Esto se lo hace a través de la subrutina CalcFase, el diagrama de flujo de esta subrutina se explica en este mismo capítulo en la parte de diagramas de flujo. Esto se lo hace con el fin de no extender la explicación de la programación.

Para la determinación total de la compensación se necesita conocer el valor de la ganancia adicional para cumplir la condición de módulo que se explicó en el capítulo 2. El valor de esta ganancia Kcomp. es indispensable para asegurar que los polos deseados formen parte del L.G.R. del sistema compensado. El diagrama de flujo para este cálculo se lo presenta en la parte de Diagramas de Flujo.

4.1.2. Acciones de Control.

En el programa se obtiene los resultados que se producen al aplicar cualquiera de las tres acciones de control como son:

Acción de Control Proporcional Derivativa:

$$G_c(s) = K_p \ (\ 1 + K_d \ s \ ) \qquad (4.15)$$

Acción de Control Proporcional Integral:

$$G_C(s) = K_P \left(1 + \frac{K_i}{s} \right) \qquad (4.16)$$

Acción de Control Proporcional Integral Derivativa

$$G_C(s) = K_P \left(1 + \frac{K_i}{s} + K_d\, s \right) \qquad (4.17)$$

Para la obtención de la función de transferencia de la acción de control se puede emplear cualquiera de los métodos conocidos (Respuesta de frecuencia, sustitución de polos, método de Ziegler Nichols, etc), se ha optado por dejar ese cálculo en manos del estudiante que utilize el programa. Por lo que el programa necesita que se ingresen los valores de los parámetros de las acciones de control, aunque en el capítulo 3 se mostró el método de diseño con un ejemplo.

Se mantiene la estructura de cada una de las Acciones de Control en la implementación de los algorítmos, entonces al ingresar las constantes para una determinada acción de control, el programa posee todos los parámetros de las funciones de transferencia del sistema de control, este se encarga de obtener la función de transferencia en lazo cerrado para luego obtener las fracciones parciales de ella, a continuación obtiene la transformada inversa de Laplace, con lo que se obtienen las ecuaciones en el tiempo de la posición y velocidad de la esfera. Partiendo de las expresiones arriba indicadas, para cada una de las acciones de control el programa calcula la respuesta en el tiempo y emula al sistema físico en movimiento bajo la ley de

control que se seleccione.

4.1.3. Realimentación de estado.

En el capítulo 3 se desarrollan las expresiones que permiten encontrar los elementos del vector de realimentación, por lo cual la subrutina que encuentra esos valores se limita a aplicarlas, verificándose antes la estabilidad y consistencia de los polos ingresados de acuerdo a las características de la planta. Para realimentación de estados solamente se puede ingresar raíces que se ubiquen en el semiplano izquierdo del plano "s", ya que sólo estos conseguirán la estabilización del sistema.

Con estas raíces se calcula el polinomio característico en lazo cerrado utilizando para esto una rutina de multiplicación de polinomios. A continuación se recuerda los elementos del vector de realimentación:

$$f = [\ f_1 \quad f_2 \quad f_3\ ]$$

$$f_1 = \frac{a_{lc\,0} - a_0}{b_0\ K}$$

$$f_2 = \frac{a_{lc\,1} - a_1}{b_0\ K}$$

$$f_3 = \frac{a_{lc\,2} - a_2}{b_0\ K} \qquad (4.18)$$

donde a_{lci} son los coeficientes del polimonio caracteristico de lazo cerrado, funcion de los polos ingresados y que se los

calcula utilizando una rutina de multiplicacion de polinomios, los a_i son los coeficientes del polinomio de lazo abierto, y K es la ganancia del control con realimentacion de estados.

Polos multiples de lazo cerrado.- Debido a que la realimentación de estado permite que se ubiquen los polos de lazo cerrado en cualquier posición, se han desarrollado rutinas que permitan considerar de que caso se trata. Estas rutinas se encargan fundamentalmente de encontrar la expansión en fracciones parciales para los estados de interés, y luego en base a ésta expansión encontrar su transformada inversa de Laplace. En este capítulo se indica a partir de los diagramas de flujo.

Para explicar las mencionadas rutinas, se presentan las expresiones que son motivos de nuestro análisis:

$$X_1(s) = \frac{k_{31}\, s^3 + k_{21}\, s^2 + k_{11}\, s + k_{10}}{s\,(\, s^3 + a_{lc2}\, s^2 + a_{lc1}\, s + a_{lc0}\,)} \qquad (4.19)$$

donde las constantes ki1 dependen de los valores de las condicones iniciales ingresadas, de los coeficientes de la funcion de transferencia de lazo abierto de la planta.

$$X_2(s) = \frac{k_{22}\, s^2 + k_{21}\, s + k_{20}}{s^3 + a_{lc\,2}\, s^2 + a_{lc\,1}\, s + a_{lc\,0}} \qquad (4.20)$$

Las raíces múltiples del polinomio característico pueden presentarse de cuatro maneras diferentes:

a) Los tres polos reales y diferentes.

b) Los tres polos reales e iguales.

c) Dos polos reales iguales, y un polo real diferente.

d) Dos polos complejos conjugados, y un polo real

Las expresiones implementadas en el programa y que permiten trabajar con estos casos parten del hecho de que la fracción compuesta a expandir puede acomodarse de la manera siguiente:

$$X_{(s)} = \frac{f_{(s)}}{(s - p)^n} \qquad (4.21)$$

donde:f(s) es la fracción compuesta que se forma al eliminar de X(s) los polos ubicados en s = p

p es el polo cuyo coeficiente deseamos encontrar.

n es la multiplicidad.

Entonces el coeficiente que acompaña al polo en s = p está dado por la siguiente igualdad:

$$R = \frac{1}{(n - 1)!} \frac{d^{n-1}}{d s^{n-1}} \left(f(s) \right)_{s = p} \qquad (4.22)$$

Para encontrar las respuestas en el tiempo, se han

implementado las transformadas inversas de Laplace, según el caso de multiplicidad, tomando como dato los coeficientes antes calculados.

4.2.- SELECCION DEL COMPILADOR.

El programa que se ha desarrollado trabaja en el entorno gráfico de Windows. Microsoft Windows es un entorno operativo que funciona con MS-DOS, es un entorno de ventanas multitarea basado en gráficos, ofrece una pasarela hacia la siguiente generación de aplicaciones gráficas. Windows 3.1. libera al desarrollador de software de las restricciones de memoria del DOS. Windows ofrece abundantes subrutinas incorporadas que permiten la fácil implementación de menús instantáneos, barras de desplazamiento, cuadros de diálogo, iconos y otras muchas características de un interfaz gráfico de fácil manejo para el usuario. Para el desarrollo del programa se ha aplicado el compilador de BORLAND C++ versión 3.1, trabajando totalmente bajo el entorno de Windows (Turbo C++ ver. 3.1)., se empleó la técnica de programación orientada a objetos. Con la Programación Orientada a Objetos aplicada adecuadamente, hasta cierto punto se pueden construir aplicaciones a partir de componentes que se encuentran disponibles; estos componentes usualmente se encuentran agrupados en bibliotecas de objetos, o bien, utilizando la terminología de C++, en bibliotecas de clases.

El programa implementado consta de un menu principal, al

cual se puede acceder a través del teclado, ó por el ratón. Por el menu se pueden acceder a varias funciones del programa como:

"Archivo", el cual posee las siguientes funciones:

"Limpiar la pantalla", inicializa cualquier grafico o texto borrandolo de la ventana principal.

"Imprimir", permite imprimir un gráfico, o texto.

"Salir", permite salir del programa.

"Edición", permite gestión de gráficos o texto hacia el portapapeles de Windows.

"Tipos de Control".

- "Red de Adelanto de fase", a través de dos métodos, el método de la Bisectriz, en el cual se deben ingresar el máximo sobreimpulso, y el tiempo de establecimiento; y el método de la ubicación arbitraria del cero del compensador, en el cual se debe ingresar el máximo sobreimpulso, el tiempo de establecimiento, y la ubicación adecuada del cero.
- "Acciones de Control", proporcional integral, proporcional derivativa, proporcional integral derivativa, en estos se debe ingresar las contantes Kp, Ki, Kd, según que acción de control se elija.
- "Realimentación de Estados", se deben ingresar las raíces del polinomio característico de lazo cerrado.
- "Condiciones iniciales", de espacio y velocidad de la esfera.
- "Set point", referencia para la ubicación de la esfera en el espacio.

"**Simulación**", que permite realizar:

- "Resultados", presenta los cálculos realizados de las raíces de lazo cerrado.
- "Inicio de Cálculos", realiza todos los cálculos.
- "Ejecutar Animación", realiza la animación del sistema.

"**Graficos**", presenta los gráficos del espacio, velocidad, error, versus tiempo y el gráfico de velocidad versus espacio.

"**Herramientas**", permite tener, el reloj, calculadora, y el administrador de archivos, que proporciona Windows.

"**Ayuda**", nos sirve de ayuda al usuario para el correcto uso del programa.

4.3.- SIMULACION DINAMICA.

La simulación dinámica se realiza mediante la función Simulación que ubica a la esfera en suspensión en la pantalla del computador. Para la graficación de la esfera se utilizan vectores, previamente almacenados con los valores que se obtienen al aplicar las diferentes compensaciones al sistema, y escogiendo del menu pricipal la opción de realizar cálculos.

La simulación dinámica del sistema se realiza por medio de la función animación; ésta consta de dos partes, la simulación del sistema de control, electroimán, fotocelda, fuente de luz se la realiza por medio de un mapa de bits, el cual se presenta en

todo momento en la pantalla del computador.

La otra parte la constituye la simulación de la esfera que se realiza en el espacio determinado previamente para el movimiento de la esfera, llamado celda. En este método se utiliza una memoria intermedia llamada página oculta, en la cual se copia la nueva posición de la esfera, para luego graficarla en la pantalla del computador, con esto se consigue evitar el efecto de parpadeo que se produciría al realizar la animación de la esfera por medio de la técnica tradicional, que graficaría la posición de la esfera en la pantalla, luego la borraría y por último copiaría la nueva posición de la esfera en la pantalla del computador.

La técnica empleada en esta tesis ofrece una mejor visualización de la animación de la esfera (la página oculta en realidad no es una página sino un segmento de memoria donde se almacena temporalmente los valores que permiten posicionar la esfera en la celda), debido a que en la página oculta se borra la posición anterior de la esfera y luego se copia la nueva posición, inmediatamente la página oculta se copia en la ventana principal del programa, superponiéndose a la imagen anterior de la ventana. Así los puntos que coinciden con la posición anterior y que son del mismo color no se van a presentar variaciónes, sino solamente aquellos puntos que tienen una nueva posición dentro de la celda de movimiento permitido para la esfera.

4.4.- DIAGRAMAS DE FLUJO.

En los diagramas de flujo que se presentan se emplean dos tipos de variables, unas que van relacionadas directamente con el programa principal y otras que van relacionadas con cada rutina. Para no extenderse demasiado en la construcción de flujogramas se trata de sintetizar al máximo las ideas que se desea transmitir.

Los diagramas de flujo que se presentan son de las funciones principales que realiza el programa, en estos se deben realizar bifurcaciones y tomar decisiones dependiendo de algún parámetro.

1.- A continuación en la figura 4.4, se presenta el diagrama de flujo de la animación del sistema, en la ventana principal del programa. Con el fin de obtener un gráfico que represente la realidad física del problema, se ha diseñado una rutina, que primero copia el mapa de bits (que representa los elementos estáticos de que se halla conformado el sistema, electroimán, bloque de control, fotodetector de posición, fuente de luz), en la ventana principal de la aplicación, (previamente el mapa de bits ha sido cargado al inicio del programa), luego grafica la esfera en las posiciones que tiene el vector de espacio (luego de realizar todos los cálculos), por último se verifica si la esfera a sobrepasado los límites del movimiento. El programa tiene la opción de escalar toda la ventana, sin perder

los elementos constitutivos del sistema.

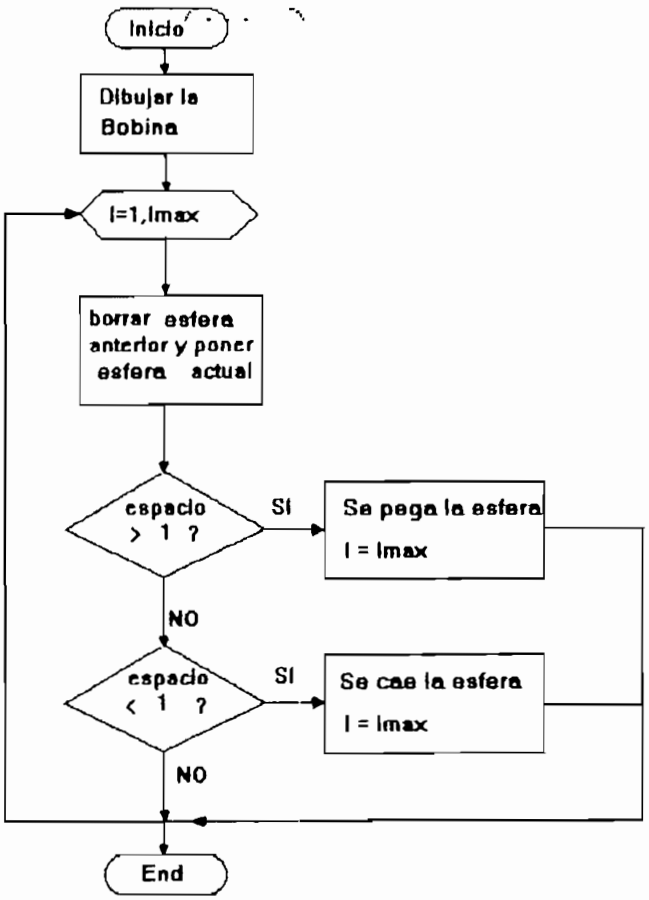


Figura 4.4 Diagrama de flujo de la rutina que realiza la animación del sistema.

2.- Para el cálculo de la fase del compensador de la red de adelanto de fase se tiene el siguiente diagrama de flujo, el cual se indica en la figura 4.5. Se parte de la función de transferencia de la planta, como también de las especificaciones deseadas de respuesta; y se basa en la ecuación 4.2, la cual permite obtener la fase del compensador para las raíces deseadas.

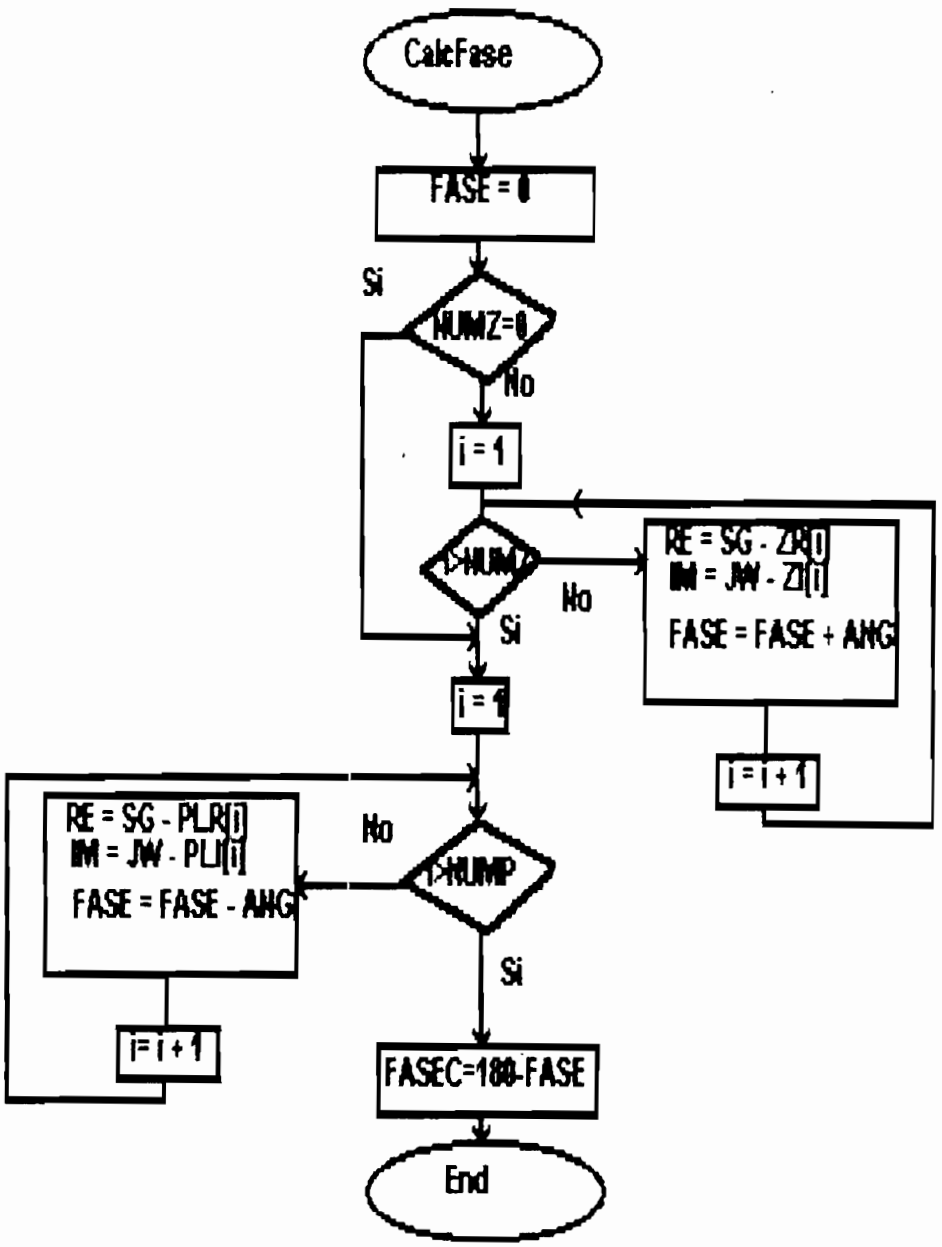


Figura 4.5 Flujograma de la rutina que calcula la fase que debe introducir el compensador en la red de adelanto de fase.

donde:

NUMZ = # de ceros de la F.T. cuya fase se busca.

NUMP = # de polos de la F.T. cuya fase se busca.

SG = Parte real de las raíces deseadas

JW = Parte imaginaria de una de las raíces deseadas

ANG = Angulo de una de las raíces deseadas.

FASE = Fase de la F.T. en las raíces deseadas.

ZR[i] = Vector de la parte real de los ceros de la F.T.

ZI[i] = Vector de la parte imaginaria de los ceros de la F.T.

PLR[i] = Vector de la parte real de los polos de la F.T.

PLI[i] = Vector de la parte imaginaria de los polos de la F.T.

3.- Para el cálculo de la ganancia del compensador Kcomp, con el fin de conseguir los polos deseados; se dispone del siguiente diagrama de flujo, figura 4.6. El algorítmo se basa en ir encontrando cada uno de los módulos de los complejos que se forman en el número rador de la función de transferencia Gc(s) G(s) H (s), al substituir "s" por uno de los polos deseados y multilplicarlos entre si, para luego a éste resultado dividirlo por el módulo de cada uno de los complejos que se forman en el denominador de dicha función de transferencia.

Las variables importantes que intervienen en la rutina que se presenta en la figura 4.6 son:

K_{planta} = Es la ganancia característica de la planta sin compensación.

K_{sensor} = Es la ganancia física del fotosensor, y que es igual a 13.2 V/m.

K_{comp} = Es la ganancia del compensador, necesariapara que los polos deseados formen parte del L.G.R. del sistema compensado.

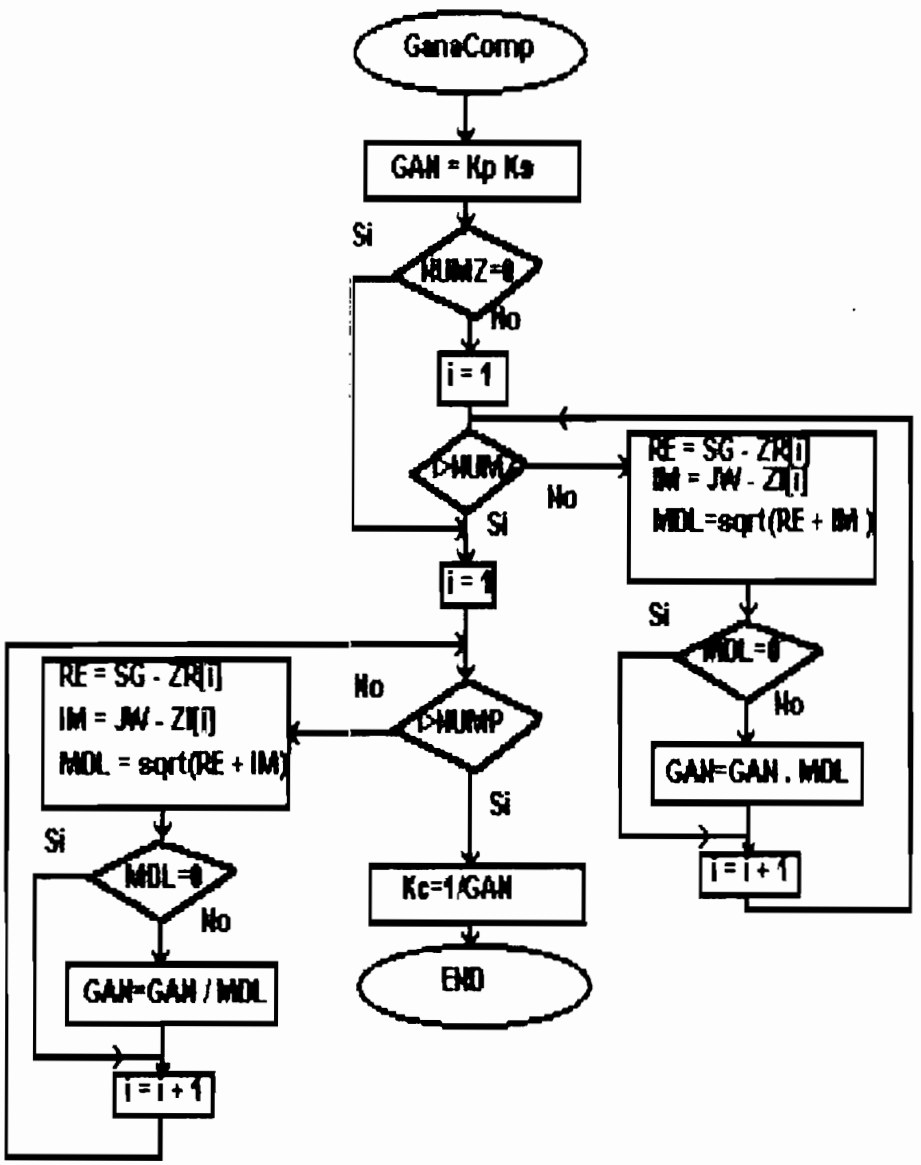


Figura 4.6 Flujograma de la rutina que calcula la ganancia introducir el compensador en la red de adelanto de fase.

4.- Para encontrar la respuesta en el tiempo, para los estados de interés (espacio, y velocidad de la esfera), a partir de los polos múltiples, se indica el siguiente diagrama de flujo, figura 4.7. Estas rutinas se utilizan en el caso del control por realimentación de estado.

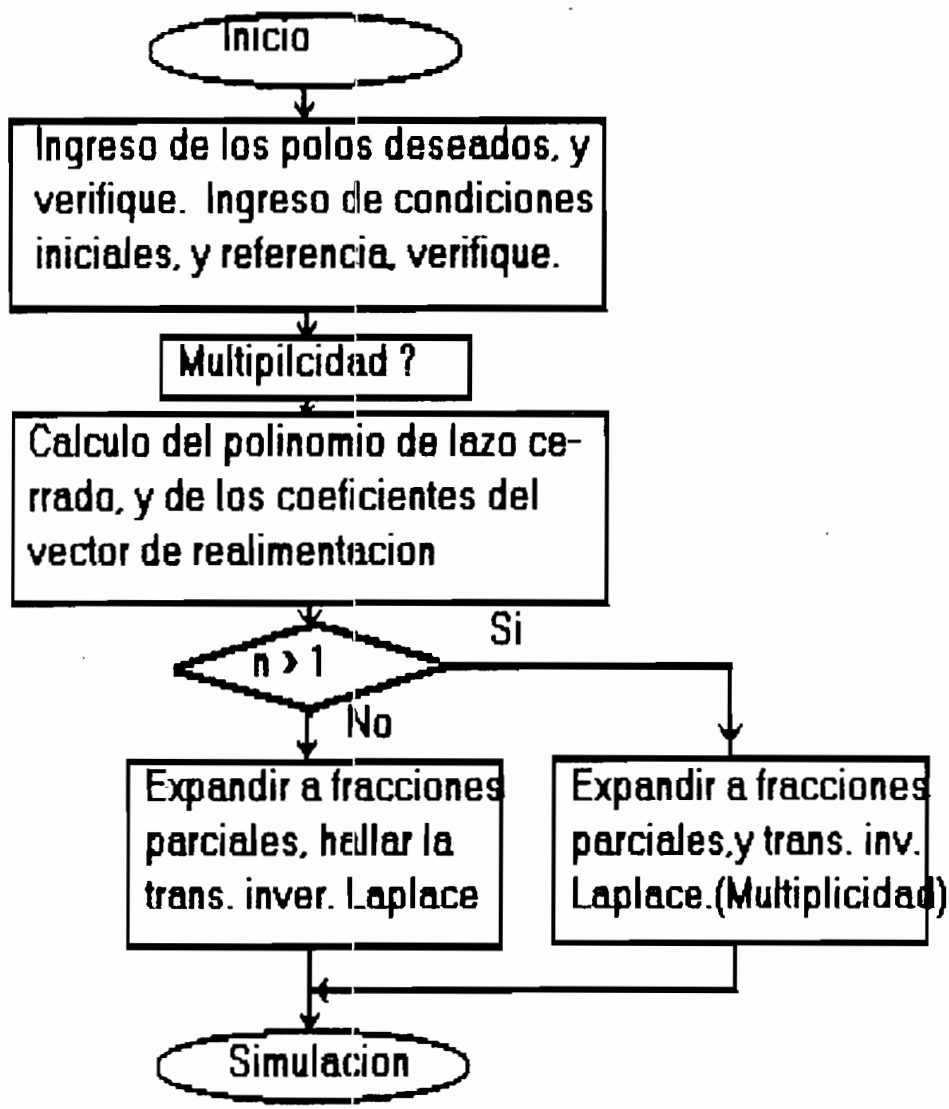
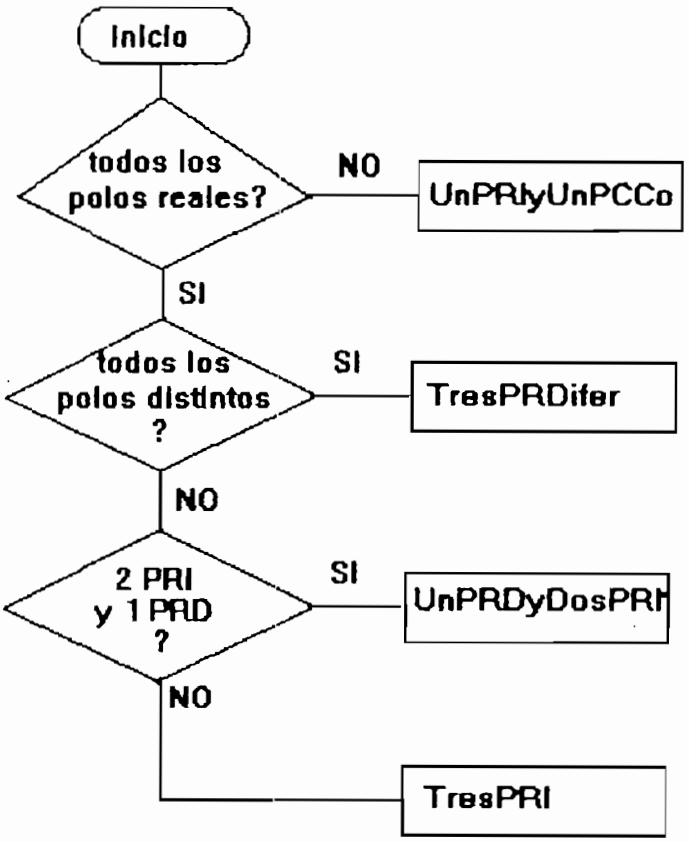


Figura 4.7 Flujograma para encontrar la respuesta en el tiempo para Realimentación de estados.

A continuación se indica la rutina que permite seleccionar el caso que se trate, cuando ingresamos los polos de lazao cerrado, en el control por relimentación de estado.



5.- Aparte de utilizar rutinas conocidas como son las de suma, multiplicación y obtención de las raíces de polinomios, se han desarrollado dos rutinas determinantes en la solución del problema.

Una de ellas es la rutina que se emplea para expandir en fracciones parciales una fracción compuesta sin multiplicidad de polos, figura 4.8.

Esta se basa en suponer que la expresión a expandirse en fracciones parciales presenta la siguiente estructura:

$$X(s) = \frac{\prod (s - ZR[i] - jZI[i])}{\prod (s - XR[i] - jXI[i])}$$

$$= \sum \frac{(COR[i] + jCOI[i])}{(s - XR[i] - jXI[i])}$$

donde:

ZR[i] + j ZI[i] = son los ceros.

XR[i] + j XI[i] = son los polos.

COR[i] + j COI[i] = son los residuos de la expansión.

Para la determinación de los valores de COR[i] y COI[i] falta sustituir s por el polo cuyo residuo se desea encontrar, y evaluar la expresión eliminando antes el término que se hace cero. El diagrama de flujo que se presenta a continuación permite realizar éste proceso.

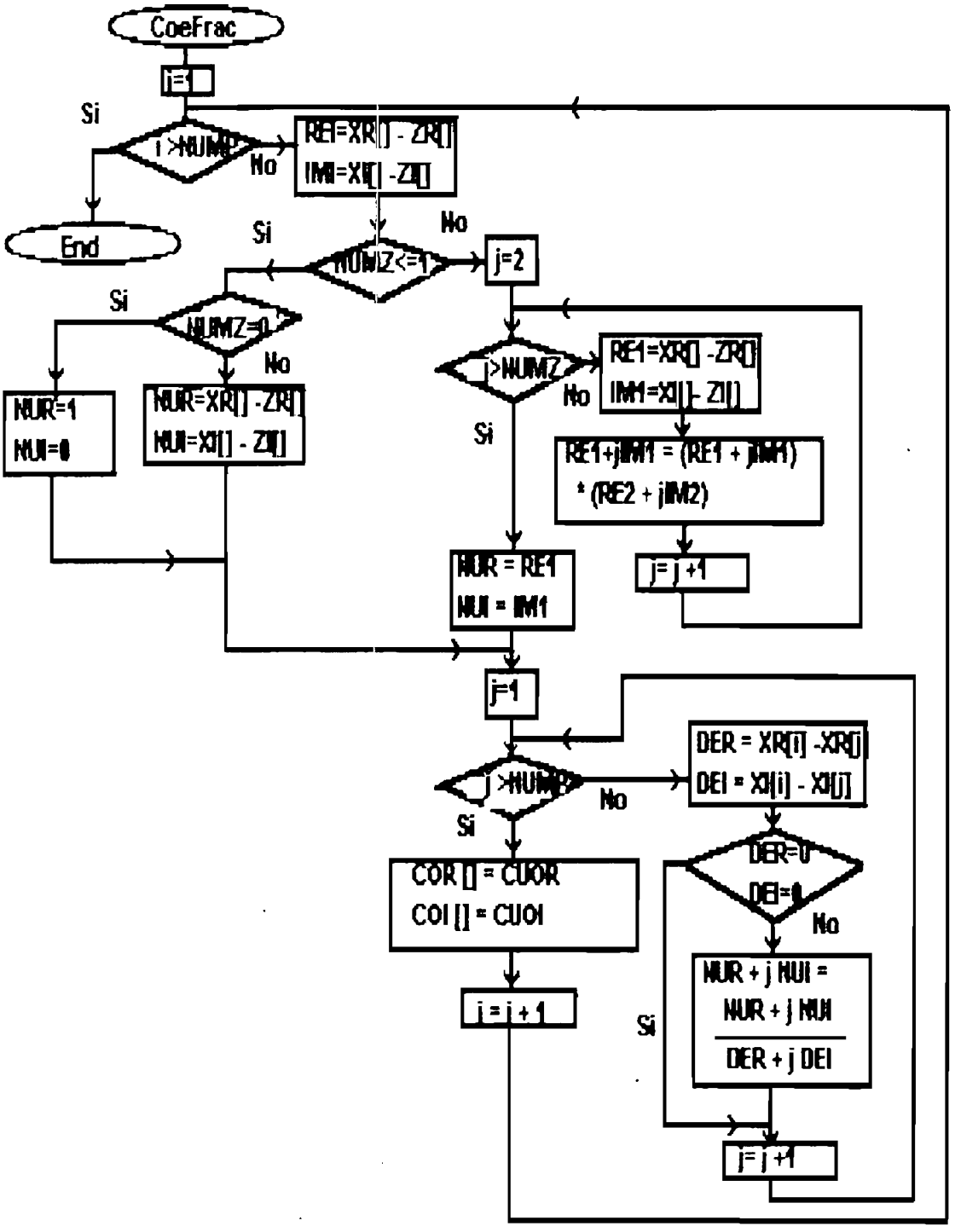


Figura 4.8 Flujograma de la rutina que expande en fracciones parciales una fracción compuesta.

La otra rutina importante que se utiliza, es la que permite encontrar la respuesta en el tiempo de la expansión en fracciones parciales, figura 4.9. En esta rutina se almacena un vector que representa la respuesta en el tiempo del sistema, la rutina evalúa dentro de un lazo a, una expresión exponencial si el polo que se manipula no posee parte imaginaria, y a una expresión en senos y cosenos si existe parte imaginaria. A continuación se indica el diagrama de flujo:

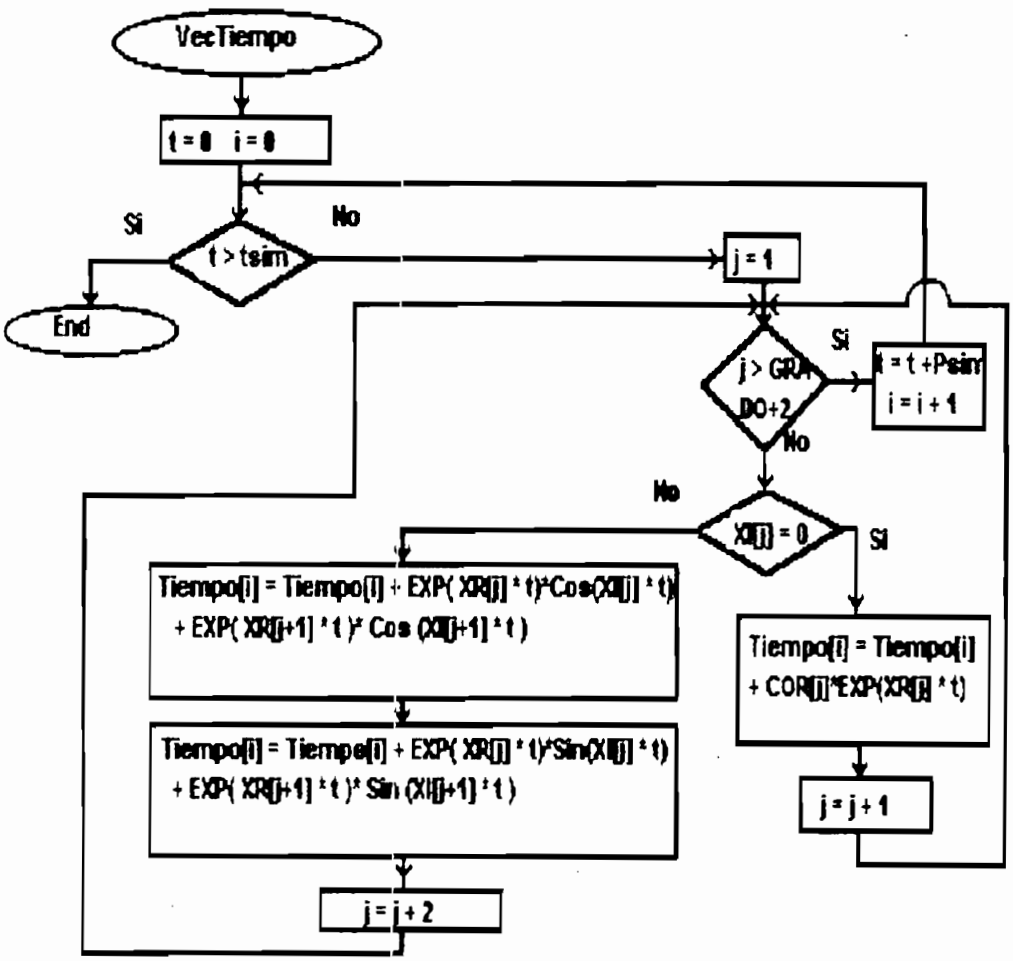


Figura 4.9 Flujograma de la rutina que encuentra la transformada inversa de Laplace de una expresión en fracciones parciales.

donde: t es el tiempo en el cual se evalúa la expresión, Tiempo[i] es el vector que contiene los puntos calculados, y Psim es el intervalo de tiempo entre los puntos.

Las rutinas que se ha presentado no son todas las rutinas de las cuales se sirve el programa para la solución del problema planteado, pero son las importantes para entender el funcionamiento global del programa.

4.5.- FUNCIONES UTILIZADAS.

Existen en el programa otras funciones, las cuales se las explicará brevemente a continuación.

Para las diferentes compensaciones existen varias funciones que consiguen los datos que se ingresan al computador a través de un cuadro de diálogo y los copia en una cadena de caracteres añadiendo las unidades, para luego mostrar un cuadro de diálogo, para su respectiva confirmacion por el usuario de los valores ingresados, estos valores son colocados en el módulo principal del programa como variables globales.

Hay una función que selecciona la compensación a ejecutar, dependiendo del último dato ingresado, por lo tanto que cálculos se van a realizar, tales como: red de adelanto de fase, acciones de control, o realimentación de estado.

La función ClearMenu realiza un reseteo de todas las banderas de gráficas, además la última de las pantallas presentadas las inicializa borrándolas.

Existen funciones que realizan gráficos de espacio, velocidad, error de la esfera, como una función del tiempo, para cada gráfico, dependiendo del máximo valor a graficar y del tamaño de la ventana principal, se escala adecuadamente, y en cada ventana de graficación aparecen las coordenadas del gráfico, las cuales varian con el movimiento del ratón del computador, para seleccionar los puntos de interés. Para esto se resetean todas las banderas de gráficos activos y se setean la bandera del gráfico que se seleccionó en el menú principal del programa, luego se llama a la función para limpiar la pantalla, generando un mensaje WM_PAINT que llama a la función Paint donde de acuerdo a la bandera llamarán a la función de gráfico seleccionado. Existe además para cada grafico la posibilidad de un redimensionamiento de cualquier ventana del programa.

Otra de las funciones importantes que se han incorporado en el programa permite la impresión de cualquier gráfico que se encuentre en la ventana principal del programa, se lo puede hacer gracias a la función CMimprimirPant, que inicializa una estructura PRINTDLG, para luego mostrar el cuadro del diálogo de impresión. Para realizar ésto, se crea un mapa de bits compatible con la ventana donde se la copia, para luego enviar

el mapa de bits a la impresora.

Así mismo como se puede enviar cualquier mapa de bits a la impresora, también se lo puede enviar al portapapeles, es decir, copiar cualquier gráfico o texto al portapapeles mediante la selección copiar del menú del programa. La función que realiza éste trabajo es CMcopyClipboard, en ésta primero se crea un mapa de bits compatible con la ventana principal y se la copia; luego se borra el contenido del portapapeles, para posteriormente copiar el mapa de bits en el portapapeles.

En el menu del programa existe una opción de ayuda al usuario, en la cual se indican las diferentes compensaciones utilizadas por el programa para resolver el problema planteado, se indican además los datos que se requieren para las diferentes simulaciones realizadas.

Por último podemos mencionar que se dispone en el menú principal del programa, los programas proporcionados por Windows como el administrador de archivos, la calculadora y el reloj, los cuales pueden ser útiles para gestión de archivos, o para selección de los parámetros que se deben ingresar como datos al programa.

CAPITULO V

RESULTADOS Y CONCLUSIONES

5.1.- RESULTADOS.

En esta parte se presentarán los resultados que se obtiene con el programa, cuando se aplican los algorítmos para las diferentes alternativas de control (desarrollados en el capítulo 4), para ello se utlizarán los gráficos que se obtienen con la utilización del programa.

Analizando las tres compensaciones desarrolladas para solucionar el problema en estudio, se puede obtener:

Método de la Bisectriz.- En el capítulo 2, tomando el modelo físico del sistema se planteó ciertos límites para los requerimientos de máximo sobreimpulso (Mp %), y de tiempo de establecimiento (ts), es por ello que se seleccionó como raíces dominantes las siguientes:

$$p_1 = -10 + j\,19.5$$

$$p_2 = -10 - j\,19.5$$

valores que son la representación de estabilidad relativa Mp% = 20, y de tiempo de establecimiento de ts = 0.4 seg. (tomando el criterio del 2%).

Si se ingresan estos valores como especificaciones deseadas al programa, este luego de aplicar el algorítmo (presentado en el capítulo 4), da como resultado la siguiente función de transferencia para el compensador:

$$G_{COMP} = 200\frac{(s + 14)}{(s + 34.3)}$$

Este resultado obtenido por el programa, es similar al

obtenido en el capítulo 2, obtenido mediante el procedimiento convencional para el diseño del compensador de adelanto de fase encontrado por el método de la bisectriz.

Si se obtiene la respuesta en el tiempo del espacio, para una entrada escalón, (partiendo de condiciones iniciales nulas) por medio del programa se obtiene la figura 5.1.

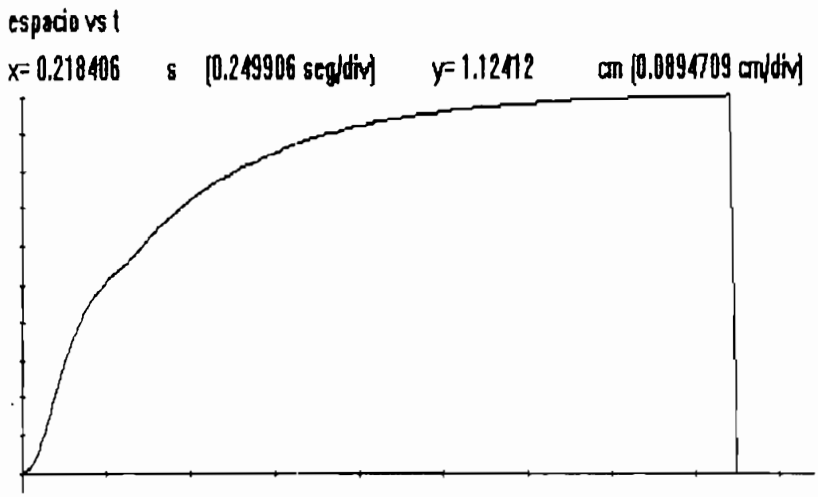


Figura 5.1. Respuesta del espacio vs tiempo para una entrada escalón, por el método de la bisectriz

Si se observa en la figura 5.1, se nota el comportamiento del sistema, como uno de primer orden, esto se produce debido a que en el lazo cerrado existe la presencia de una raiz ubicada en el plano "s", en - 2.4, ejerciendo una mayor dominancia que la que ejercerían las raíces (-10 +/- j 19.5). Con estos resultados podemos concluir que, en el diseño de una red de adelanto de fase por el método de la bisectriz para el problema en estudio, aunque permite minimizar la ganancia del compensador, produce resultados inesperados, es decir se obtiene

una respuesta de especificaciones no esperadas.

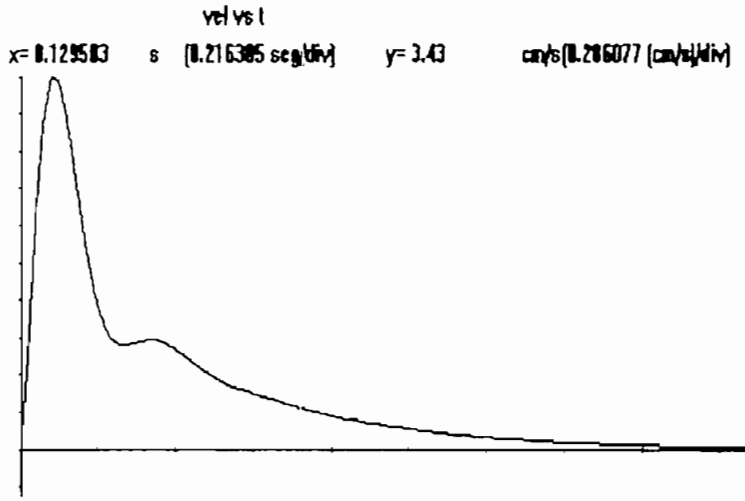


Figura 5.2 Respuesta de la velocidad vs tiempo para una entrada escalón, por el método de la bisectriz.

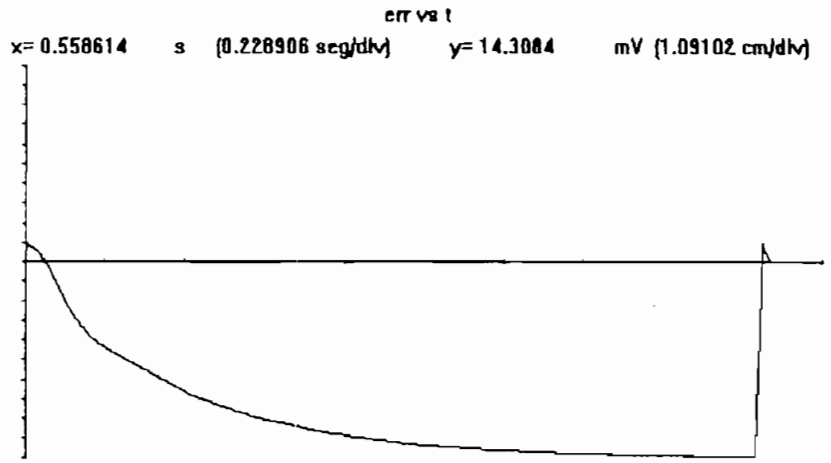


Figura 5.3 Respuesta del error en el tiempo para una entrada escalón, por el metodo de la bisectriz.

Observando las figuras 5.2 y 5.3, se nota la presencia de un error de estado estacionario, esto se produce debido a que el sistema luego de ser compensado mantiene la característica de

ser un sistema tipo cero. La curva de velocidad nos indica la posición de la esfera en el espacio.

En el programa también podemos obtener la posición de la esfera relacionada con su variación temporal (velocidad), es decir el diagrama de fase. En la figura 5.4. podemos observar el diagrama de fase cuando se tiene una entrada escalón, en ésta figura no se observa un comportamiento amortiguado en la respuesta del sistema, debido a que durante toda la trayectoria la velocidad posee una sola dirección, hasta que la esfera se ubica en el nuevo punto de suspensión. La posición final a la que llega la esfera es diferente de la referencia (x = 0, punto medio de la zona de atracción), debido a que se ha modificado la entrada del sistema (set point), con una condición de reposo, entonces se podría decir que la esfera se ha ubicado a 0.9 cm de la referencia, (hay que tomar encuenta el error de estado estable).

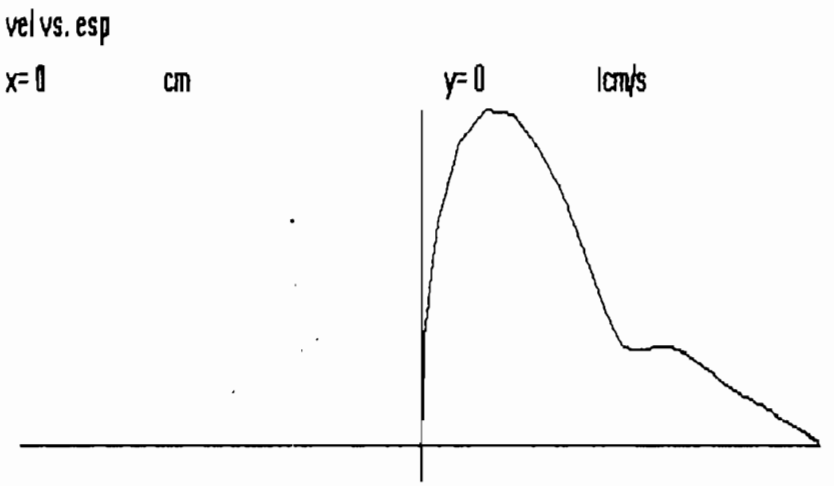


Figura 5.4 Diagrama de fase para una entrada escalón con condición de reposo, por el método de la bisectriz.

Si se ubica la entrada del sistema en la referencia, y se parte con condicion inicial de posición, la esfera se ubica en la zona media de atracción. Esto se observa en las figuras 5.5.a y 5.5.b que corresponden a una posición inicial bajo la referencia de 0.9 cm y velocidad nula.

En la figura del espacio se observa la tendencia de la esfera a regresar a la referencia, por otro lado la trayectoria en el diagrama de fase tiende hacia el origen, indicando que parte con velocidad nula y tiende hacia el punto medio de la zona de atracción de la esfera.

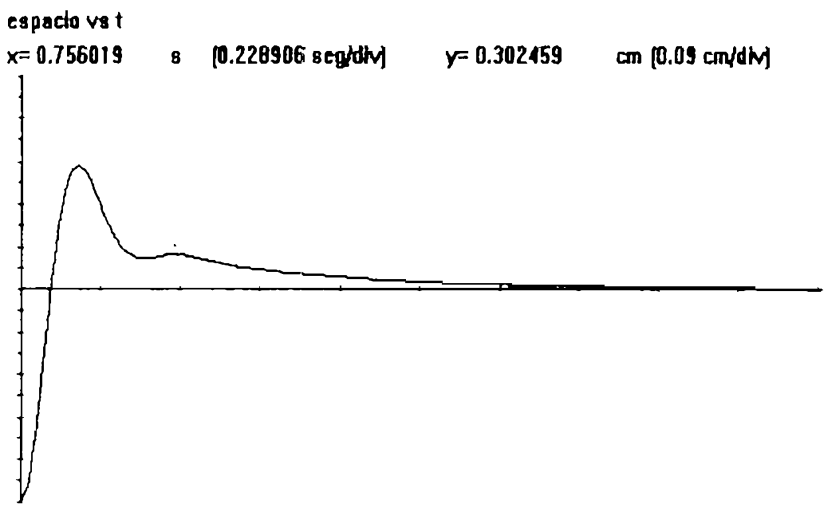


Figura 5.5.a Respuesta del espacio vs tiempo para una condición inicial de -0.9 cm, por el método de la bisectriz.

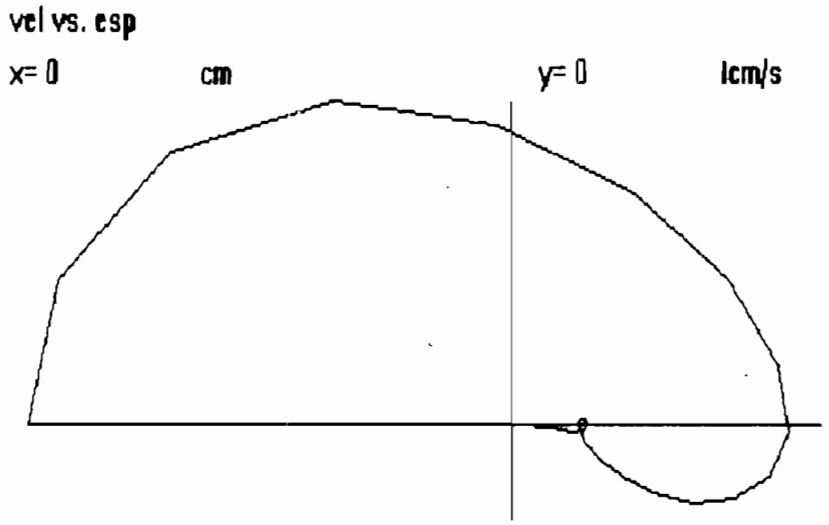


Figura 5.5.b Diagrama de fase para una condición inicial de -0.9 cm y velocidad nula, por el método de la bisectriz.

Método de la Ubicación Arbitraria del Cero. Así mismo aplicando los algorítmos (capítulo 4), para este método, con especificaciones de máximo sobreimpulso Mp% = 20, tiempo de establecimiento ts = 0.4 s, y una ubicación del cero de -30 se obtiene la siguiente red de adelanto de fase:

$$G_{COMP} = 1097 \frac{(s + 30)}{(s + 249)}$$

Resultados muy parecidos al los obtenidos por el procedimiento convencional aplicado en el capítulo 2.

Al aplicar este compensador a la planta se obtiene la respuesta del sistema con un sobretiro de 23 %, y un ts de 0.4 segundos, esto se puede observar en la figura 5.6.

Con lo que la dominancia de las raíces deseadas se consigue aplicándo este método.

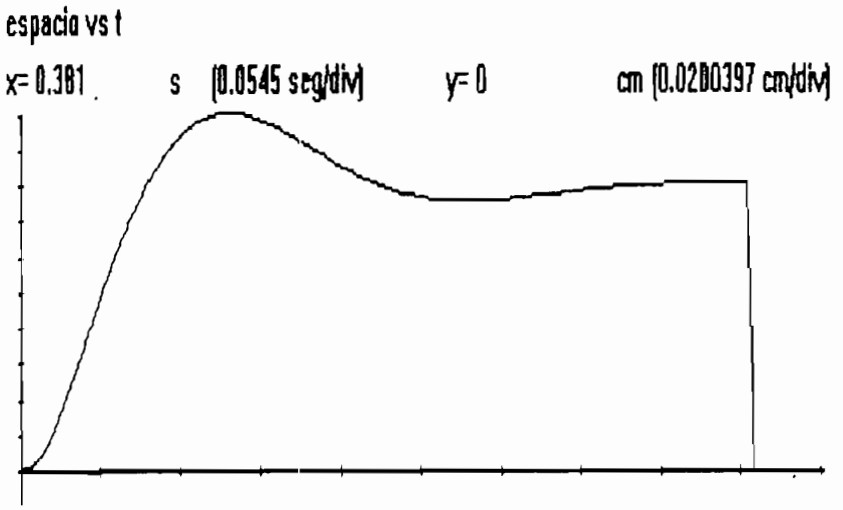


Figura 5.6 Respuesta del espacio vs tiempo para una entrada escalón, por el método del cero arbitrario.

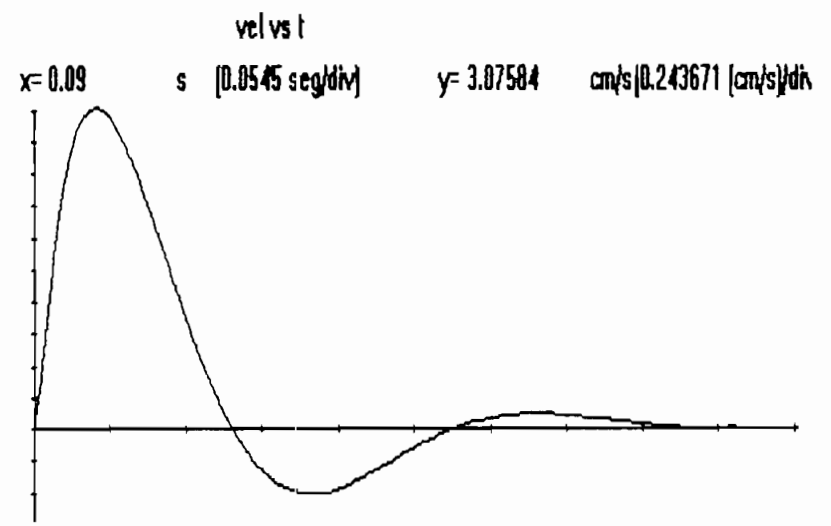


Figura 5.7 Respuesta de la velocidad vs tiempo para una entrada escalón, por el método del cero arbitrario.

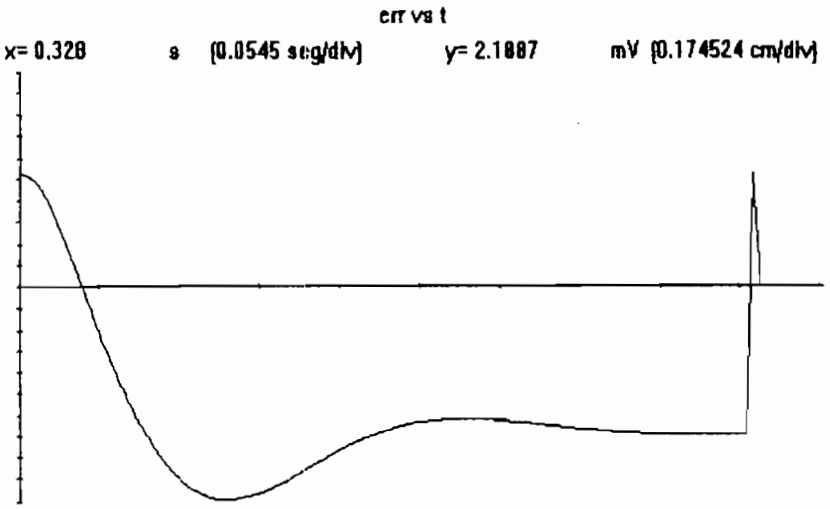


Figura 5.8 Error vs tiempo para una entrada escalón por el método del cero arbitrario.

Con este método para encontrar los parámetros de una red de adelanto de fase, se puede conseguir la dominancia de las raíces deseadas, el inconveniente se muestra en la ganancia que se debe dar para que esto suceda, con lo cual se debe colocar un amplificador al sistema, pero se consigue que el sistema se comporte como uno de segundo orden (figura 5.7). El error de estado estable no es nulo, esto se lo puede observar en la figura 5.8, no se lo puede correguir ya que con ésta compensación no se cambia el tipo del sistema.

En el diagrama de fase correspondiente (figura 5.9), para esta compensación se puede observar el cambio de dirección que sufre la velocidad de la esfera hasta llegar a la nueva referencia que se le impone.

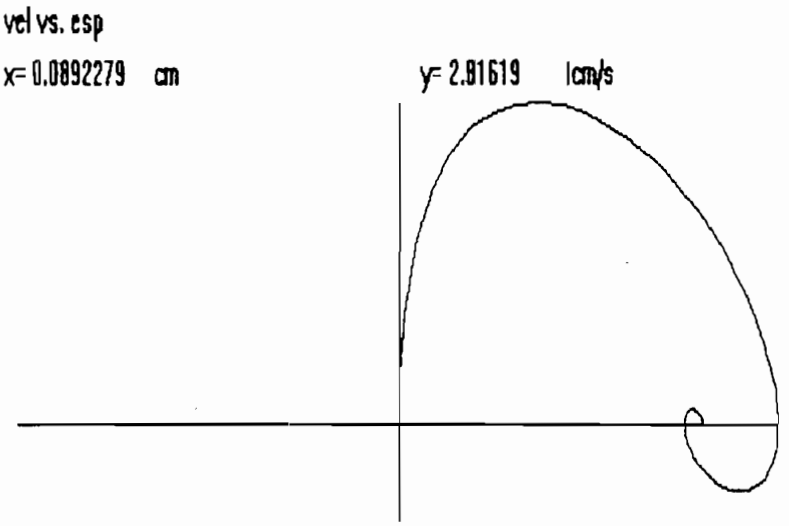


Figura 5.9 Diagrama de fase para una entrada escalón, por el método del cero arbitrario.

Si solo se escogen condiciones iniciales de - 0.9 cm y velocidad nula, se puede observar que el sistema con esta compensación tiende hacia la referencia (punto medio de la zona de atracción), con amortiguación. Esto se observa en las figuras 5.10.a y b, que corresponden a una posición inicial bajo la referencia de 0.9 cm y velocidad nula. En la figura del espacio se observa la tendencia de la esfera a regresar a la referencia, como un sistema de segundo orden, lo que se ratifica en la trayectoria del diagrama de fase, el cual tiende hacia el origen.

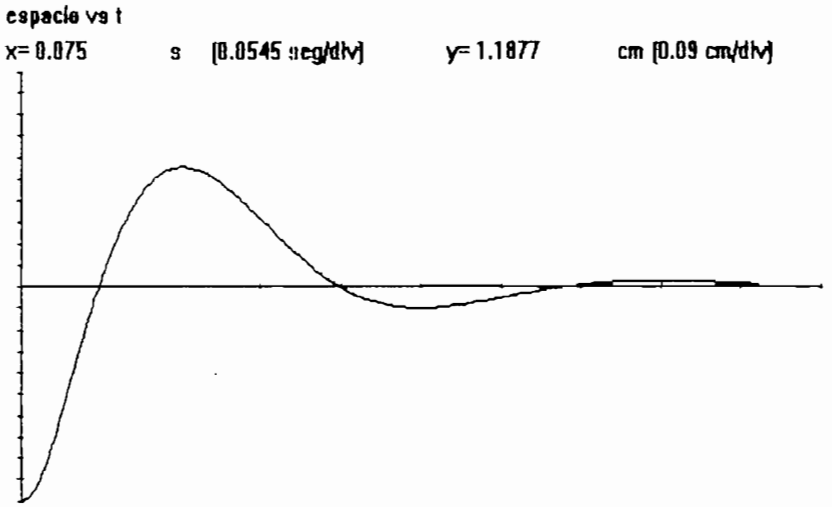


Figura 5.10.a Respuesta del espacio vs tiempo para una condición inicial de -0.9 cm, por el método del cero.

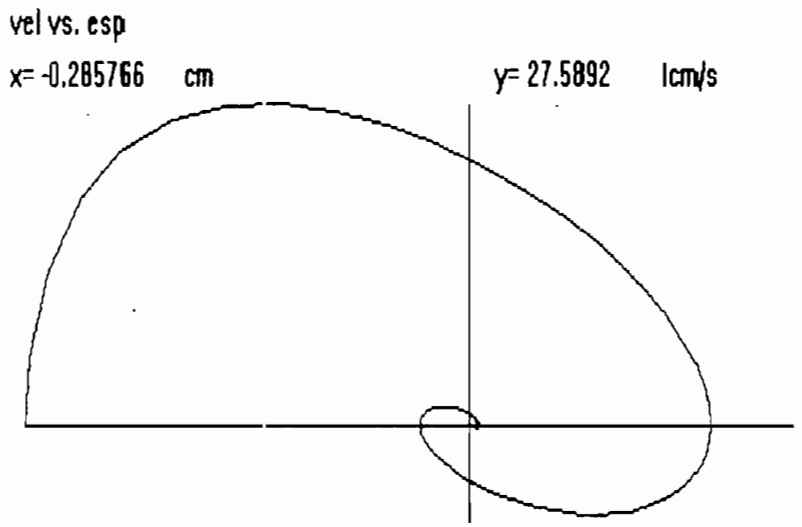


Figura 5.10.b Diagrama de fase para una condición inicial de -0.9 cm y velocidad nula, por el método del cero.

Acciones de Control.- En el capítulo 2 se calculó un

$$G_{COMP} = 140.3\ (1 + 0.03\ s)$$

compensador proporcional derivativo, de forma que las raíces dominantes sean las mismas que en el caso anterior, resultando que los parámetros de control para esta compensación sean de Kp = 140.3 y Kd = 0.03. Al ingresar estos datos al programa nos produce un compensador que hace que el sistema tenga los siguientes polos de lazo cerrado:

. $p_1 = -10 + j\,19.5$

$p_2 = -10 - j\,19.5$

$p_3 = -84.3$

Donde se observa la dominancia de las raíces deseadas, por lo tanto la respuesta que se esperaba es cercana a la que se obtiene con este compensador, ya que observando la figura 5.11 se determina que las especificaciones que se obtienen son de 22.8 % para el máximo sobre impulso, y de 0.42 segundos para el tiempo de establecimiento.

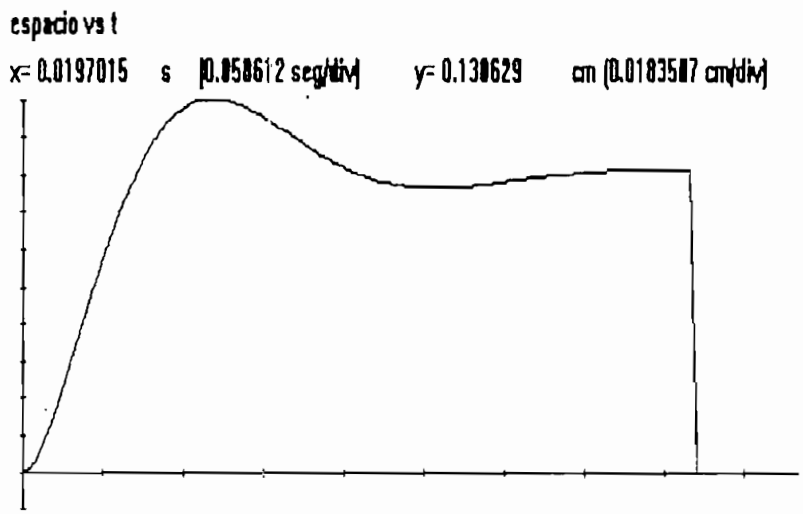


Figura 5.11 Respuesta del espacio vs tiempo para una entrada escalón, con compensación PD.

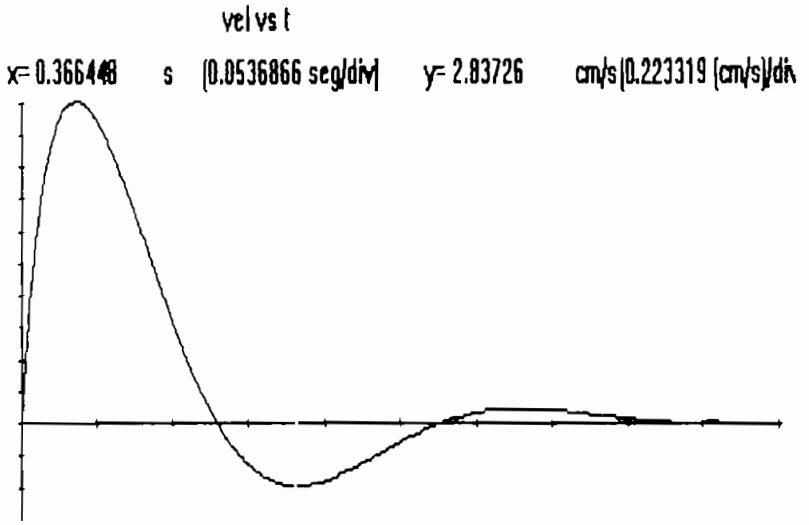


Figura 5.12 Respuesta de la velocidad vs tiempo para una entrada escalón, con compensación PD.

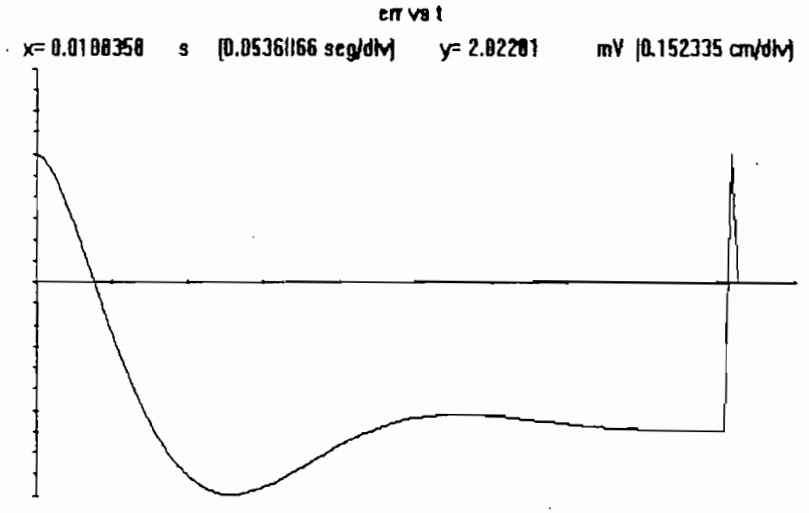


Figura 5.13 Error en el tiempo para una entrada escalón, con compensación PD.

En la figura 5.13 se observa la curva que representa el error del sistema, de esto se desprende que el tipo del sistema no se modifica con esta compensación, debido a que su error en

estado estable sigue siendo diferente de cero.

El diagrama de fase para este tipo de compensación se la observa en la figura 5.14, así mismo el sistema tiene un comportamiento subamortiguado, que brindan las raíces dominantes, es decir el sistema se comporta como uno de segundo orden, en ésta figura se observa el cambio de dirección que sufre la velocidad de la esfera para llegar al nuevo punto de reposo.

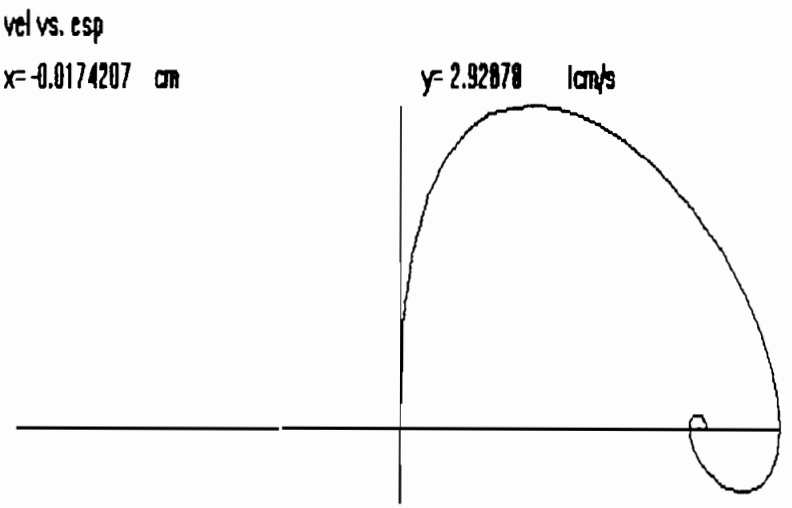


Figura 5.14 Diagrama de fase para una entrada escalón, con compensación PD.

Si aplicamos la compensación proporcional integral derivativa, con el fin de estabilizar al sistema, con un error de posición nulo, se tendrá lo siguiente:

Si se diseña un compensador PID, que trate de mantener un margen de fase de 18.5° y un margén de ganancia de 7 db, utilizando para el efecto el método de diseño de Ziegler Nichols, se obtienen los siguientes parámetros del compensador.

Kp = 127, Ki = 5.93, y Kd = 0.0421. El disenó de éste compensador se lo deja al estudiante, por lo tanto no se profundizará en el procedimiento seguido para el diseño de este compensador.

Al ingresar estos valores como parámetros del compensador PID, al programa, se obtienen los siguientes resultados:

$p_1 = -2.5 + j\ 14.5$

$p_2 = -2.5 - j\ 14.5$

$p_3 = -33$

$p_4 = -66.5$

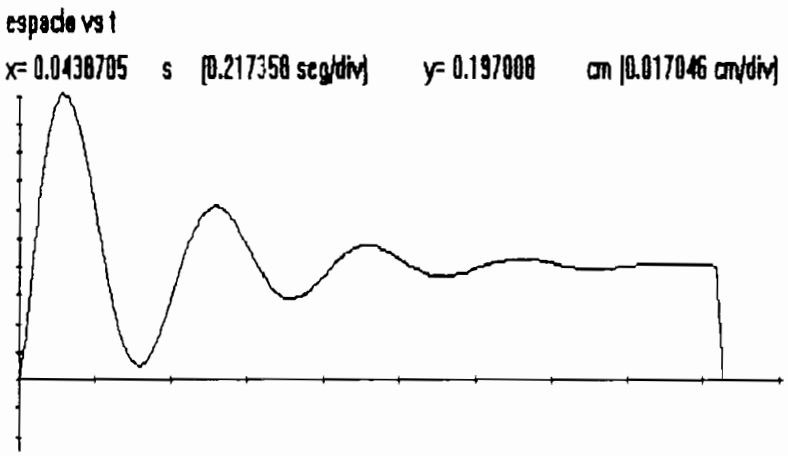


Figura 5.15 Respuesta de espacio en el tiempo para una compensación PID

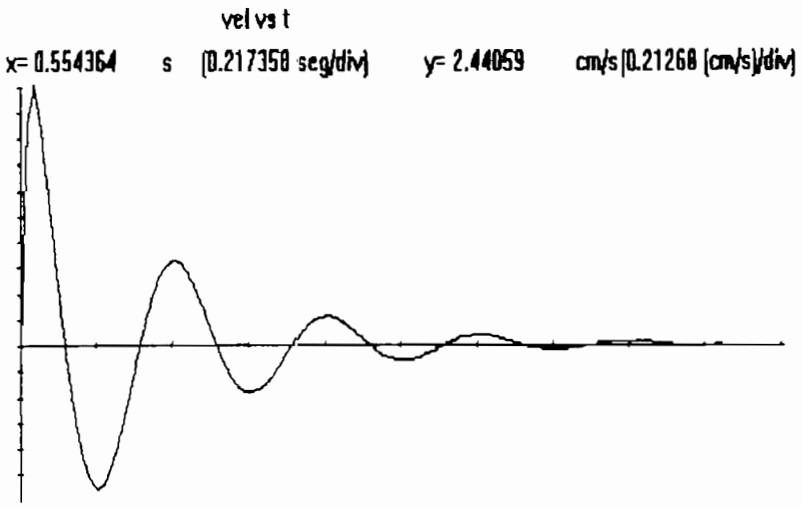


Figura 5.16 Velocidad en el tiempo
para una compensación PID

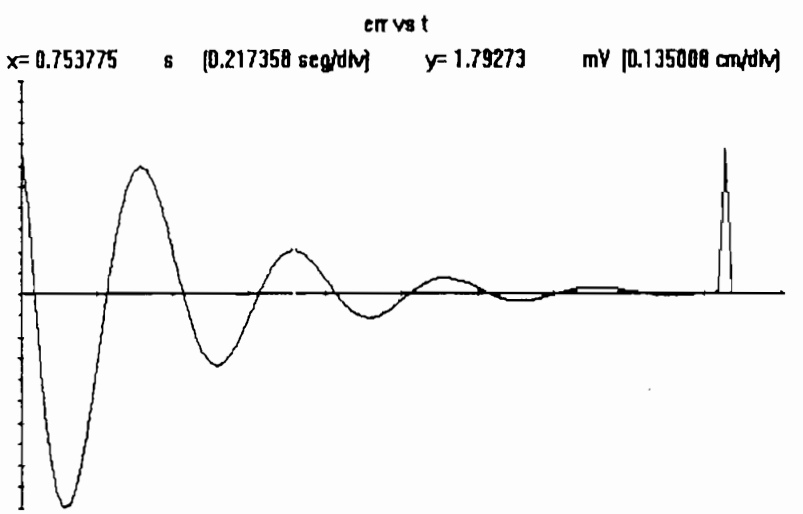


Figura 5.17 Error en el tiempo
para una compensación PID

De la figura 5.15 del espacio, se desprende el efecto oscilatorio que se produce en el sistema, esto se debe al margen de fase que se le ha impuesto, el cual se relaciona con un coeficiente de amortiguamiento bajo. La figura 5.16 representa

la velocidad que posee la esfera con esta compensación, la cual termina por hacerce nula en el equilibrio impuesto. Lo que se deseaba con esta compensación se lo observa en la figura 5.17, la cual representa al error, el cual tiende a cero debido a que el sistema con esta compensación dispone de aumento en el tipo del sistema, es decir se vuelve de tipo uno, en el cual el error de posición es nulo como se lo verifica con la curva del error.

En la figura 5.18 se indica el diagrama de fase del sistema con esta compensación, en el cual se observa el comportamiento amortiguado del sistema por la presencia del bajo coeficiente de amortiguamiento de los polos dominantes de lazo cerrado.

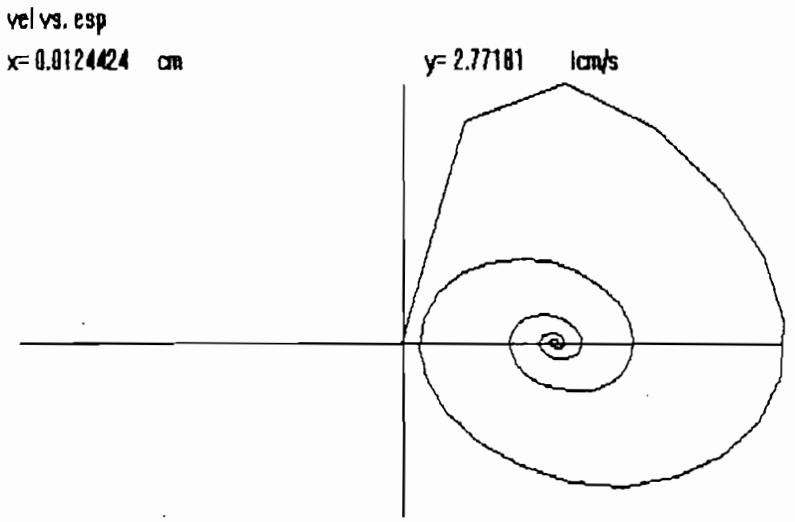
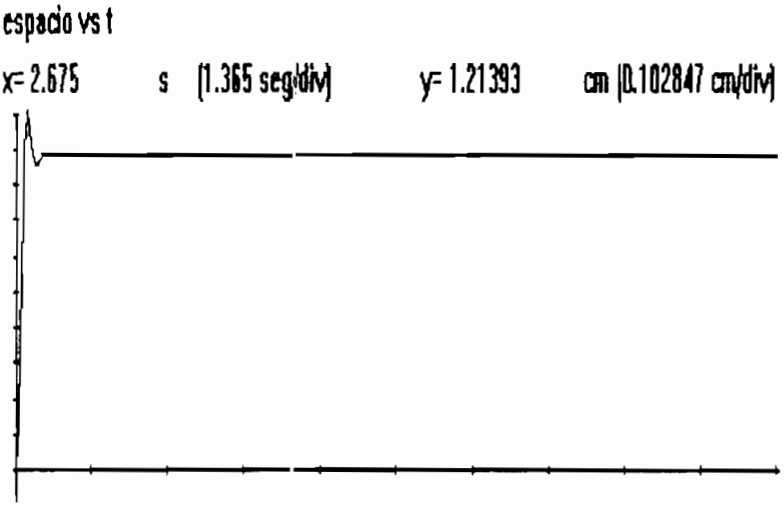


Figura 5.18 Diagrama de fase para una compensación PID, con entrada escalón

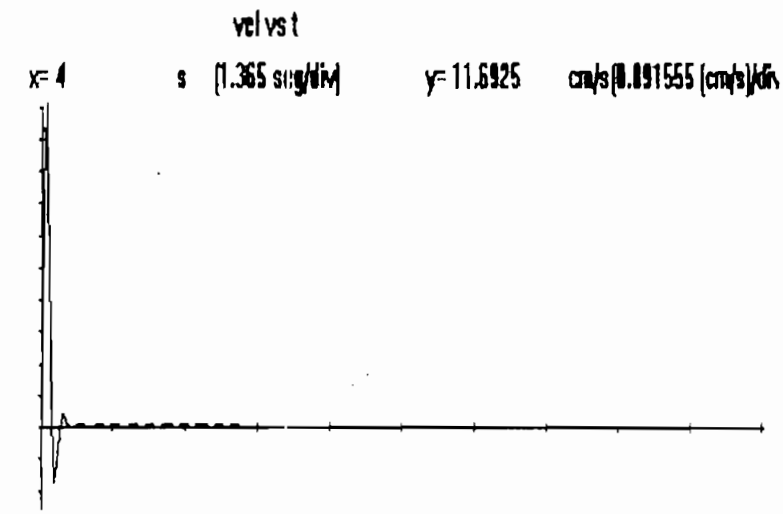
Realimentación de Estado.- Para este caso se aplica control moderno al sistema, en el cual se realiza el cálculo del vector de realimentación.

Si se ingresan los polos de lazo cerrado para el caso

similar de dominancia, y con especificaiones de maximo sobreimpulso de 20% y con un tiempo de establecimiento de 0.4 seg., se obtienen los siguientes resultados:



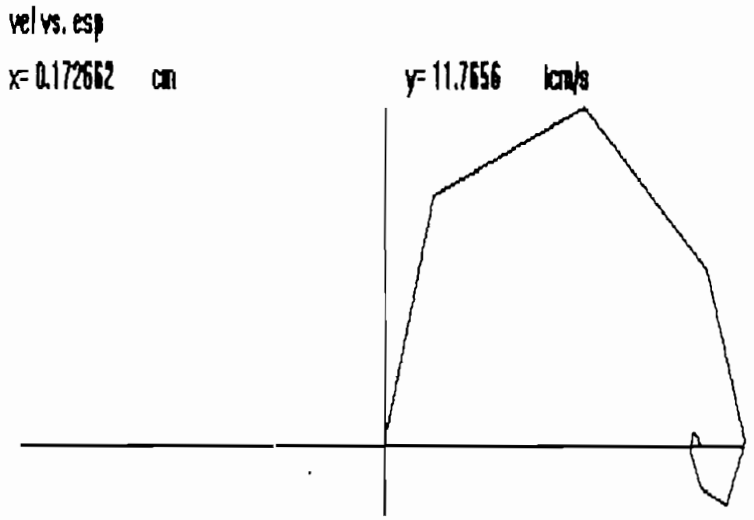
5.19 Respuesta en el tiempo del espacio para el caso de realimentación de estado



5.20 Velocidad versus el tiempo para el caso de realimentación de estado

De las figuras 5.19, 20 y 21, se puede desprender que el

sistema se comporta, como uno de segundo orden, debido a las especificaciones que se ingresaron para esta compensación. El error que se obtiene en estado estable para esta compensación es nulo, debido a que en el diseño del estimador de estado, y de la ganancia del compensador se puso esta condición para el cálculo del sistema en lazo cerrado. En el diagrama de fase se observa que la esfera se estabiliza en la posición que se desea, lo hace de manera subamortiguada confirmando de esta manera el funcionamiento del sistema como uno de segundo orden por el ingreso de las raíces dominantes.



5.21 Diagrama de fase para el caso de realimentación de estado

5.2.- CONCLUSIONES.

El programa se utiliza bajo la plataforma de Windows ver 3.1, y gracias a esto se puede obtener las facilidades, de

multiuso y multitarea que proporciona este entorno operativo. Es decir la facilidad de uso del programa en este entorno, nos permite que el funcionamiento del programa sea "amigable" con el usuario, además existe la posibilidad de enlace e incrustación de este programa con los programas desarrollados para Windows.

Aunque el programa ha sido desarrollado para Windows ver 3.1, puede correrse en el sistem operativo Windows N.T., el cual es un sistema operativo puro, esto es posible debido a la técnica de programación orientada a objetos. Como el programa ha sido implementado para Windows, se necesita que este sea corrido en un computador que soporte este entorno operativo, y con la velocidad de procesamiento necesaria para que la animación no sea lenta.

El empleo del compilador Turbo C++ de la Borland, ha permitido que el desarrollo del programa sea versátil, y también que la implementación de la programación orientada a objetos sea factible. Es uno de los compiladores que facilitan obtener el mayor provecho de los computadores que funcionan con palabras de 32 bits.

Cabe mencionar que el modelo, que se emplea para la simulación de un sistema, depende de los parámetros relativos al experimento que se considere (al efectuar la modelación matemática) y son estos parámetros los que determinan la complejidad del modelo a utilizarse. El modelo matemático que

se emplee depende además de las restriciones propias que se olvidan en la implementación del sistema físico, es decir factores que serían de dificil implementación en el computador, como condiciones del ambiente, criterios de construcción, empleo de materiales para su construcción, etc.

El Programa SDBS.EXE es una introducción a la simulación dinámica del sistema de la bola suspendida, en ningún momento pretende cubrir todos los aspectos relativos a la misma. Sin embargo deja la posibilidad de ensayar varios compensadores de suerte que el estudiante analice el comportamiento físico del sistema ante las diferentes alternativas de control. Existe una gran variedad de sistemas que por las facilidades que presentan, permiten emular físicamente en la pantalla del computador su comportamiento dinámico; con la facilidad de que los algoritmos desarrollados para representarlos bajo una ley de control son particulares y no genéricos como lo son los algoritmos de la mayoría de programas existentes para el análisis y diseño de sistemas de control para enseñanza superior.

Luego de haber finalizado el estudio del problema de la bola suspendida, se puede concluir que la simulación dinámica que se ha obtenido, permite realizar de una manera fácil el estudio del sistema con diferentes compensaciones, a través de un computador. Es decir se puede simular en el computador experimentos que en la práctica poseen varias limitaciones para su ejecución en un sistema físico; estas limitaciones tienen que

ver con el costo que produciría el diseño y construcción en un prototipo, así como el tiempo que se ocuparía en las diferentes implementaciones.

Una desventaja en la simulación tiene que ver respecto de la presentación de los gráficos, ya que si bien para determinados casos las curvas indican estabilidad para el sistema, en la animación del sistema no se cumple esto porque la esfera ha sobrepasado los límites impuestos para la zona de atracción, es decir la esfera se adhiere al electroimán o cae atraida por la gravedad según la posición supere esta zona por la parte superior o por la parte inferrior de la celda.

Los algorítmos que se utilizan en el diseño de la animación de las diferentes compensaciones del sistema, no son complejos, sino más bien son estructuras que trabajan bien dentro de los requerimientos del problema planteado.

Para el diseño de una red de adelanto, se puede decir que utilizando el método de la ubicación arbitraria del cero del compensador, se obtiene la dominancia requerida para que el sistema se comporte como uno de segundo orden, pero se requiere que la red activa posea una ganancia alta, es decir que exista un amplificador para que el sistema sea compensado adecuadamente. En cuanto al error de posición del sistema se puede decir que éste no es nulo, ya que el sistema compensado sigue siendo de tipo cero.

De las acciones que se pueden implementar para el sistema, podemos decir que la acción de control proporcional derivativa, ofrece la posibilidad de obtener la dominancia deseada para que el sistema se comporte como uno de segundo orden, esto se produce debido a que con esta compensación se añade un cero al sistema, pero así mismo existe la presencia de error de estado estable, que no se lo puede anular con esta compensación. Con la compensación proporcional integral derivativa, se puede conseguir que el tipo del sistema compensado sea uno, con lo cual el error de estado estable se hace nulo.

Aplicando el control moderno, al sistema, es decir la realimentación de estados, se obtienen las mejores condiciones, debido a que se pueden conseguir las especificaciones deseadas, además de poseer un error de estado estable nulo, por la construcción del estimador de estado y por la consideración de ganancia de esta compensación.

Los cálculos que no se han presentado de manera detallada en este trabajo, se debe a que para ciertos casos se ha considerado dejar los mismos en manos del estudiante para que pueda desarrollar la inventiva en la preparación de la teoría de control. Con la utilización del programa el estudiante podrá obtener criterios, para al diseño de compensaciones, que puede ser adaptado al Laboratorio, como parte de ensayos.

Finalmente con lo mencionado anteriormente, se cree haber

cumplido con todos los objetivos propuestos al iniciar el presente trabajo.

BIBLIOGRAFIA

BIBLIOGRAFIA

- OGATA K., " Ingeniería de Control Moderna ", Ed. Prentice Hall, México, 1985.

- DISTEFANO J, STUBBERUD A, WILLIAMS I., " Retroalimentación y Sistemas de Control ", Ed. McGraw Hill, Santafé de Bogotá, 1992.

- OGATA K., " Dinámica de Sistemas ", Ed. Prentice Hall, México, 1987.

- DORF R., " Sistemas Automáticos de Control ", Ed. Prentice Hall Internacional, 1978.

- KUO B., " Automatic Control System ", Ed. Prentice Hall, 1967.

- WELSTEAD P., " Intoduction to Physican System Modeling ", Ed. Prentice Hall Internacional, 1978.

- KEETER R., " Modern Methods of Engeneering Computation ", Ed. McGraw Hill, 1979.

- GALARZA I., " Prototipo de un Sistema de Control para una bola suspendida ", Tesis de Grado, ESPE, Quito, 1987.

- SHELDON T., " Windows 3.1 Manual de Referencia ", Ed. Osborne/McGraw Hill, Madrid, 1993.

- MURRAY W., PAPPAS Ch., " Programación en Windows 3.1 ", Ed. Osborne/McGraw Hill, Madrid, 1993.

- PAPPAS Ch., MURRAY W., " Manual de Borland C++ ", Ed. Osborne/McGraw Hill, Madrid, 1993.

- ADAMS L., " Programación avanzada de gráficos en C para Windows ", Ed. Windcrest/McGraw Hill, Madrid, 1993.

- PORTER A., " Programación en C++ para Windows ", Ed. Osborne/McGraw Hill, Madrid, 1994.

ANEXOS

ANEXOS.

MANUAL DEL USUARIO.

En este punto se trata de dar una guía de como utilizar de mejor manera el programa motivo de la presente tesis.

El programa para la *simulación dinamica* del problema de la *bola suspendida* es un archivo ejecutable bajo Windows, **SDBS.EXE** el cual es formado por los siguientes archivos fuente para su compilación en Turbo C++ ver. 3.1

SDBS.PRJ desarrollo del proyecto para la aplicación.

SDBS.CPP contiene el codigo fuente del programa, punto de entrada de la aplicación, bucle de mensajes, rutinas de ventanas y funciones propias de la aplicación.

SDBS.H definición de las constantes de las funciones.

SDBSM:H definición de los prototipos de las funciones.

OWL.DEF definición de módulo.

SDBS.RC declaraciones de los menus, las abreviaturas de teclado y las cadenas.

SDBS.BMP mapa de bits de la animación.

BOLA:ICO mapa de bits del icono que representa la aplicación.

SDBS.HLP archivo de ayuda para Windows de la aplicación.

SDBS.HPJ archivo de proyecto de la ayuda.

SDBS.RTF archivo con formato de enriquecimiento de texto para la ayuda.

SDBS.SYM contiene todas las bibliotecas necesarias para que se

ejecute la aplicación.

Es de suma importancia que se encuentre el archivo BWCC.DLL en el subdirectorio C:\WINDOWS\SYSTEM, directorio del sistema de Windows. Este es un archivo que se encuentra en el compilador Turbo C++ ver 3.1, y sirve para ejecutar todos los mensajes, cuadros de dialogo, ventanas, con la programación orientada a objetos, es decir permite que se ejecute la aplicación. Esto se realiza sino se dispone del archivo SDBS.SYM el cual es muy extenso. Es decir se puede ejecutar la aplicación si se dispone en el mismo directorio del programa ejecutable del archivo SBDS.SYM

El programa puede ser ejecutado directamente si se ingresa desde el inductor del DOS.

C:\>WINDOWS\WIN SDBS

Otra posibilidad de ingreso rápido al programa es ubicando el icono que representa al programa (BOLA.ICO), en el grupo de inicio de Windows.

Por último se puede ingresar al programa a través del administrador de programas de Windows, realizando doble pulsación con ratón en el icono que representa al programa. O en su defecto a través del administardor de archivos con la doble pulsación sobre el nombre del archivo ejecutable del programa.

Inmediatamente después del ingreso al programa aparece una

ventana de presentación del programa, presionando la tecla OK que aparece en dicho mensaje el programa queda listo para ser utilizado.

El programa dispone de un menu principal el cual posee 7 menús desplegables, al accionarlos a través del teclado por las teclas de acceso rápido (las teclas que se encuentran subrayadas), ó por la pulsación del botón izquierdo del ratón.
A continuación se presenta una descripción de cada uno de estos menús.

(Archivo)

Mediante esta opción se dispone de un submenú con tres opciones.

Limpiar Pantalla Borra cualquier gráfico o texto que se encuentre en la ventana de trabajo.
Imprimir Permite imprimir un gráfico, o texto.
Salir Permite salir del programa.

(Edición)

Permite gestión de gráficos o textos hacia el portapapeles de Windows, este menú dispone de dos posibilidades.

Copy Ctrl+Ins Permite copiar cualquier gráfico o texto hacia el portapapeles de Windows.
Paste Shift+Ins Con esta opción se puede incrustar lo que se

encuentre en el portapapeles de Windows.

(Tipos de Control)

Con este menú se puede acceder a las diferentes tipos de compensaciones existentes en el programa, además de ingreso de condiciones iniciales al sistema, así como también el ingreso de la señal de entrada al sistema. Con esta opción se puede acceder a cinco cuadros de diálogos.

Red de Adelanto de Fase Con esta opción se elige la compensación por una red de adelanto de fase, el cálculo de los parámetros del compensador se lo puede realizar mediante dos métodos.

Método de la Bisectriz Con esta opción se indica un cuadro de diálogo en el cual se deben ingresar el máximo sobreimpulso, y el tiempo de establecimiento que se desea que el sistema compensado posea.

Ubicación del Cero Arbitrario. Con esta opción se indica un cuadro de diálogo en el cual se debe ingresar el máximo sobreimpulso, el tiempo de establecimiento, que se desea que el sistema compensado tenga, además la ubicación adecuada del cero del compensador para que se verifique la dominancia de las raíces que se obtienen con los dos datos anteriores.

Acciones de Control Con esta opción se puede realizar la compensación del sistema mediante acciones de control proporcional integral, proporcional derivativa, y proporcional

integral derivativa, entonces al accionar este menú aparecen tres submenús.

Acción PI Con esta opción aparece un cuadro de diálogo, en el cual se debe ingresar los parámetros Kp y Ki.

Acción PD Con esta opción aparece un cuadro de diálogo, en el cual se debe ingresar los parámetros Kp y Kd.

Acción PID Con esta opción aparece un cuadro de diálogo, en el cual se debe ingresar los parámetros Kp, Ki y Kd.

Realimentación de Estado Esta es la opción de compensación al sistema empleando las técnicas de control moderno, al acceder a esta opción aparece un cuadro de diálogo, en el cual se debe ingresar las raices del polinomio característico de lazo cerrado.

Condiciones Iniciales Con esta opción se ingresan condiciones iniciales para la posición y/o para la velocidad de la esfera en la zona de atracción.

Set Point Con esta opción se puede ingresar la señal de entrada al sistema, es decir la referencia para la ubicación de la esfera en el espacio.

(Simulación)

Al seleccionar esta opción se dispone de tres opciones que realizan lo siguiente:

Resultados Presenta los cálculos realizados para las compensaciones, además de los polos de lazo cerrado del sistema.

Inicio de Cálculos Realiza todos los cálculos.

Ejecutar Animación Realiza la animación del sistema.

(Graficos)

Con esta opción se puede observar la respuesta del sistema en el tiempo, se dispone de cuatro curvas que son de gran ayuda en la comprención del sistema.

Espacio vs. t Con esta opción se puede graficar la respuesta del sistema en el tiempo, del espacio que recorre la esfera, se dispone de coordenadas de los puntos de la curva mediante el ratón.

Velocidad vs. t Con esta opción se presenta la velocidad que se le imprime a la esfera con la compensación del sistema.

Error vs. t Esta es la opción que permite observar el error de posición del sistema.

Todos Con esta opción se observan las tres curvas anteriores, sin las coordenadas de los puntos de las mismas.

Velocidad vs. Espacio Con esta opción se dispone del diagrama de fase del sistema.

(Herramientas)

Con este menú se puede acceder a tres diferentes aplicaciones de Windows.

Reloj Con esta opción se activa el reloj de Windows.

Calculadora Con esta opción se activa la calculadora de Windows.

Administrador de Archivos Con esta opción se puede acceder rápidamente al administrador de archivos de Windows, para gestión de los mismos.

(Ayuda)

Este menú nos permite acceder a las ayudas de Windows, y del programa.

Contenido F1 Con esta opción se presenta una ayuda del programa que se ha desarrollado para el correcto funcionamiento de la aplicación.

Usar Ayuda Con esta opción se puede acceder a la ayuda que proporciona Windows.

Acerca de ... Al accionar esta opción se puede visualizar el cuadro de presentación del programa.

LISTADO DEL PROGRAMA

Aechivo: SDBS.CPP

```
// ObjectWindows - (C) Copyright 1992 by Borland International

#include <owl.h>
#include <radiobut.h>
#include <edit.h>
#include <string.h>
#include "sdbs.h"
#include <stdlib.h>
#include <math.h>
#include <errno.h>
#include <commdlg.h>
#include "sdbsm.h"
#define PI 3.141592654

double ac[4][4], b[4], x[4];
double k1, k2, k3, p1, p2, p3;
double Pr[3], Pimag;
int pxmax, incX, jncX;
char incTiemp[25],incEspac[25],incVeloc[25];
char incError[25];

double E, CAM;
double Kp1=47.7, Ksn=13.2;
double Po[7],Qo[7];
double PROD1[7],PartReal[10],PartImag[10];
double PROD2[7];
double PLC[7];
double COR1[7],COI1[7];
double COR2[7],COI2[7];
double ZR[3], ZI[3], PLR[5], PLI[5];
double Tmp1[350], Tmp2[350], TMT[350];
double RealInic[7],ImagInic[7];
double CEROC,NUMINT;
double SG,JW,FASEC,Kcp;
double Mp, ts;
double Kp,Ki,Kd;
double Ampl;

int FLAGG = 0;
int ESTAB = 0;
int MaxI;
int nEc; // orden del sistema de ecuaciones para fracciones parciales

double Pos0,VEL0,Ace0=0.0;
double Psim;
double ResMp,Rests,ResSG,ResJW,ResPOLOC,ResCEROC,ResFASEC,ResKcp;
double ResNUMP;
double Wn,Si;
double Lp=0.842, g=9.81, M=1.0;
double an[1000],an_d[1000],es[1000],es_d[1000];
double MaxEsp=0.0, MaxVel=0.0, MaxErr=0.0;
```

```
HCURSOR hPrevCursor;                    // guardará el cursor anterior
HCURSOR hHourGlass;                  // gestor del cursor reloj de arena
HCURSOR hCrossHair;                  // gestor del cursor de cruz

// variables para el control de la animación en pantalla
HDC hMapaDC; // contexto visual. en memoria para mapas bits actores
HDC hPaginaOcultaDC;  // contexto visual. en memoria para página oculta
HBITMAP hMapa;                        // gestor del mapa de bits actor
HBITMAP hPaginaOculta;   // gestor del mapa de bits de la página oculta
HBITMAP SDBSBmp,hPrevBitmap;
BITMAP bmPaste;        // contendrá la información del mapa de bits

int anchoVentx;                         // ancho de la ventana principal
int altoVenty;                         // altura de la ventana principal
int coordenadaX;                              // coordenada x del cursor
int coordenadaY;                              // coordenada y del cursor
int UbicEjex;
int AnchoCelda;                       // anchura de la página oculta
int AlturaCelda;                        // altura de la página oculta
HBITMAP hPrevBitmap1;           // gestor del mapa de bits por defecto
HBITMAP hPrevBitmap2;           // gestor del mapa de bits por defecto

BOOL Comienzo=TRUE;
BOOL bCeldasListas= FALSE; // TRUE si los mapas de bits están preparados
BOOL bPause= TRUE;         // FALSE si se está ejecutando la animación
BOOL GrafiAnimaci= FALSE;
BOOL GrafiEspacio= FALSE;
BOOL GrafiVelocid= FALSE;
BOOL GrafiErrPosi= FALSE;
BOOL GraficarTodo= FALSE;
BOOL GrafiVeloEsp= FALSE;
BOOL Resultados= FALSE;
BOOL Calculos=FALSE;
BOOL CNTESF_A= FALSE;
BOOL CNTESF_B= FALSE;
BOOL CNTESF_C= FALSE;
BOOL CNTESF_D= FALSE;
BOOL CNTESF_E= FALSE;
BOOL CNTESF_F= FALSE;
BOOL RESP2_C = FALSE;
BOOL RESP2_R = FALSE;

int UbicCeldaX, UbicCeldaY;   // posici≤n vi±eta de visualizaci≤n
int CoorEsqDerX, CoorEsqDerY;      // esq. inferior derecha mapa de bits
int AnchoMapa;                          // anchura del mapa de bits
int AlturaMapa;                           // altura del mapa de bits
int EsqIzqBolaX;
int EsqDerBolaX;
int EsqIzqBolaY;
int EsqDerBolaY;
int OrigenCeldaY=0, OrigenCeldaX=0;              // coordenadas iniciales
int DeltaY;

struct TCoIniStruct {
char EspIniDato[10];
char VelIniDato[10];
```

```
char AceIniDato[10];
};

struct TSetPointStruct {
char SetPointDato[10];
};

struct TRedAdelStruct {
char MpDato[10];
char tsDato[10];
char CarDato[10];
};

struct TPIStruct {
char PDato[10];
char IDato[10];
};

struct TPDStruct {
char PPDato[10];
char DDDato[10];
};

struct TPIDStruct {
char PPPDato[10];
char IIIDato[10];
char DDDDato[10];
};

struct TRealEstStruct {
char p1String[10];
char p2String[10];
char p3String[10];
char pi1String[10];
};

class TTransferApp : public TApplication {
public:
  TTransferApp(LPSTR AName, HINSTANCE hInstance, HINSTANCE hPrevInstance,
    LPSTR lpCmdLine, int nCmdShow)
    : TApplication(AName, hInstance, hPrevInstance, lpCmdLine, nCmdShow) {};
  virtual void InitInstance();
  virtual void InitMainWindow();
};

class TTransferWindow : public TWindow {
public:
  TCoIniStruct CoIniStruct;
  TSetPointStruct SetPointStruct;
  TRedAdelStruct RedAdelStruct;
  TPIStruct PIStruct;
  TPDStruct PDStruct;
  TPIDStruct PIDStruct;
  TRealEstStruct RealEstStruct;

  TTransferWindow(PTWindowsObject AParent, LPSTR ATitle);
```

```
  virtual void GetWindowClass( WNDCLASS& WndClass);
  virtual LPSTR GetClassName();
  virtual void SetupWindow()
  {
      TWindow::SetupWindow();
      SDBSBmp = LoadBitmap(GetApplication()->hInstance, "SDBSBitmap");
  }

  virtual void Paint( HDC hdc, PAINTSTRUCT& PaintInfo);
  virtual void ClearMenu(RTMessage) = [CM_FIRST + CM_U_ClrScr];
  virtual void CMimprimirPant(RTMessage) = [CM_FIRST + CM_U_Imprimir];
  virtual void CMcopyClipboard(RTMessage) = [CM_FIRST + CM_U_Copiar];
  virtual void CMdatosBisectriz(TMessage& Msg) = [CM_FIRST + CM_BISECTRIZ];
  virtual void CMdatosCeroArbit(TMessage& Msg) = [CM_FIRST + CM_CEROARBIT];
  virtual void CMdatosPI(TMessage& Msg) = [CM_FIRST + CM_PI];
  virtual void CMdatosPD(TMessage& Msg) = [CM_FIRST + CM_PD];
  virtual void CMdatosPID(TMessage& Msg) = [CM_FIRST + CM_PID];
  virtual void CM_datosRealEst(TMessage& Msg) = [CM_FIRST + CM_REALEST];
  virtual void CMCondInic(TMessage& Msg) = [CM_FIRST + CM_CONDINIC];
  virtual void CMSetPoint(TMessage& Msg) = [CM_FIRST + CM_SETPOINT];
  virtual void WMEnterIdle( RTMessage ) = [WM_FIRST + WM_ENTERIDLE];
  virtual void CMExit( RTMessage ) = [CM_FIRST + CM_EXIT];
  virtual void CMUHelpIndex( RTMessage ) = [CM_FIRST + CM_U_HELPINDEX];
  virtual void CMUHelpHelp( RTMessage ) = [CM_FIRST + CM_U_HELPHELP];
  virtual void CMUHelpAbout( RTMessage ) = [CM_FIRST + CM_U_HELPABOUT];
  virtual void CMReloj(RTMessage Msg) = [CM_FIRST + CM_Reloj];
  virtual void CMCalculadora(RTMessage Msg) = [CM_FIRST + CM_Calculadora];
  virtual void CMArchivos(RTMessage Msg) = [CM_FIRST + CM_Archivos];
  virtual void ResultadosMenu(RTMessage) = [CM_FIRST +CM_U_Resultados];
  virtual void CalculosMenu(RTMessage) = [CM_FIRST + CM_U_BuildCels];
  virtual void AnimacionMenu(RTMessage) = [CM_FIRST + CM_U_Animate];
  virtual void GrafEspMenu(RTMessage) = [CM_FIRST + CM_U_Espacio];
  virtual void GrafVelMenu(RTMessage) = [CM_FIRST + CM_U_Velocidad];
  virtual void GrafErrMenu(RTMessage) = [CM_FIRST + CM_U_Error];
  virtual void GrafTodMenu(RTMessage) = [CM_FIRST + CM_U_GrafTodo];
  virtual void GrafFasMenu(RTMessage) = [CM_FIRST + CM_U_Fase];
  virtual void WMSize(TMessage& Message)=[WM_FIRST + WM_SIZE];
  virtual void WMRaton(TMessage& Message)=[WM_FIRST + WM_MOUSEMOVE];

  void Fondo();
  void RealimEstad();
  void PolosRealesDif();
  void DosPRIyUnPRD();
  void TresPRI();
  void UnrealyUnComp();
  void Principal();
  void   CalcFase(double  NUMZ,double   NUMP,double  ZR[],double   ZI[],double
PLR[],double PLI[],double SG,double JW,double &FASE,double &FASEC);
  void Resultado(double NUMP1);
  void Cuad(double RE1,double IMS,double &ANG1);
  void  Bicectriz(double  SG,double  JW,double  FASEC,  double   &CEROC,double
&POLOC);
  void CeroArb(double SG,double JW, double FASEC,double CEROC,double &POLOC);
  void  GanaComp(double  NUMZ,  double  NUMP,double  ZR[],double   ZI[],double
PLR[], double PLI[],double SG,double JW,double GAN, double &Kcp);
  void InfEsfera(double  Mp, double ts,double SG,double JW,double POLOC,double
```

```
CEROC,double FASEC,double Kcp);
  void RUTINA(double U1R,double V1R,double  &R1RR,double &R2RR, double  &R1IR,
double &R2IR);
  void SumPoli(double GRADO,double PROD1[],double PROD2[],double PLC[]);
  void RootPoli(double GRADO,double POLI[],double PtRe[],double PtIm[]);
  void MultPoli(double GP,double GQ,double Po[],double Qo[],double PROD[]);
  void CoeFrac (double NUMZ,double NUMP, double  ZR[],double ZI[],double XR[],
double XI[], double COR[], double COI[]);
  void  VecTiempo  (int  ESTAB,double  Cond,double  Psim,double  Tsim,double
GRADO,double    COR[],double     COI[],double    PTR[],double     PTI[],double
Tempo[],double &NUMINT);    //ojo Psim por Pim
  void SisEcuaciones();
  void ConstrCeldas();
  void Animacion();
  void LimpiarPantalla();

  void GraficaEjes();
  void GrafEspacio();
  void GrafVelocidad();
  void GrafError();
  void GrafFase();
  void CoorEspacio();
  void CoorVelocid();
  void CoorError();
  void CoorFase();

    BOOL tfF1Pressed;
};

class TBisectrizDialog : public TDialog {
public:
  TBisectrizDialog(PTWindowsObject AParent, int ResourceId);
};

void TTransferWindow::CMdatosBisectriz(TMessage&)
{
  char ALabel[255];
  char Mpsal[]="Mp = ";
  char Mpsss[]="  %";
  char Tssal[]="ts = ";
  char Tssll[]="  seg";

  if  ( GetModule()->ExecDialog(new  TBisectrizDialog(this, ID_BISECTRIZ))  ==
IDOK )
  {
    strcpy(ALabel, "Método de la Bisectriz:\n\n");
    strcat(ALabel, Mpsal);
    strcat(ALabel, RedAdelStruct.MpDato);
    strcat(ALabel, Mpsss);
    strcat(ALabel, "\n");
    strcat(ALabel, Tssal);
    strcat(ALabel, RedAdelStruct.tsDato);
    strcat(ALabel, Tssll);
    MessageBox(HWindow, ALabel, "datos ingresados", MB_OK);

    CNTESF_A= TRUE;
```

```
    CNTESF_B= FALSE;
    CNTESF_C= FALSE;
    CNTESF_D= FALSE;
    CNTESF_E= FALSE;
    CNTESF_F= FALSE;

    Mp=atof(RedAdelStruct.MpDato);
    ts=atof(RedAdelStruct.tsDato);

    if ((Mp>40.0) | (ts>2.0) | (Mp<5) | (ts<0.2))
      {
      MessageBox(HWindow,
      "Los valores ingresados no estβn\n"
      "dentro de los rangos establecidos\n"
      "ingrese otros datos\n",
      "Advertencia",
      MB_ICONEXCLAMATION | MB_OK);
      CNTESF_A= FALSE;
      }
  }
}

TBisectrizDialog::TBisectrizDialog(PTWindowsObject AParent, int ResourceId)
            : TDialog(AParent, ResourceId)
{
  new TEdit(this, ID_Mp,
          sizeof(((TTransferWindow *)Parent)->RedAdelStruct.MpDato));
  new TEdit(this, ID_Ts,
          sizeof(((TTransferWindow *)Parent)->RedAdelStruct.tsDato));
  TransferBuffer = (void far*)&(((TTransferWindow *)Parent)->RedAdelStruct);
}

class TCeroArbitDialog : public TDialog {
public:
  TCeroArbitDialog(PTWindowsObject AParent, int ResourceId);
};

void TTransferWindow::CMdatosCeroArbit(TMessage&)
{
  char ALabel[255];
  char Mpsal[]="Mp = ";
  char Mpsss[]="  %";
  char Tssal[]="ts = ";
  char Tssll[]="  seg";
  char Cersa[]="Cero = ";

  if  ( GetModule()->ExecDialog(new  TCeroArbitDialog(this, ID_CEROARBIT))  ==
IDOK )
  {
    strcpy(ALabel, "Método del Cero Arbitrario:\n\n");
    strcat(ALabel, Mpsal);
    strcat(ALabel, RedAdelStruct.MpDato);
    strcat(ALabel, Mpsss);
    strcat(ALabel, "\n");
    strcat(ALabel, Tssal);
    strcat(ALabel, RedAdelStruct.tsDato);
```

```
    strcat(ALabel, Tssll);
    strcat(ALabel, "\n");
    strcat(ALabel, Cersa);
    strcat(ALabel, RedAdelStruct.CarDato);
    MessageBox(HWindow, ALabel, "datos ingresados", MB_OK);

    CNTESF_A= FALSE;
    CNTESF_B= TRUE;
    CNTESF_C= FALSE;
    CNTESF_D= FALSE;
    CNTESF_E= FALSE;
    CNTESF_F= FALSE;

    Mp=atof(RedAdelStruct.MpDato);
    ts=atof(RedAdelStruct.tsDato);
    CEROC=atof(RedAdelStruct.CarDato);

    if ((Mp>40.0) | (ts>2.0) | (Mp<5) | (ts<0.2) |(CEROC>=0))
      {
      MessageBox(HWindow,
      "Los valores ingresados no estßn\n"
      "dentro de los rangos establecidos\n"
      "ingrese otros datos\n",
      "Advertencia",
      MB_ICONEXCLAMATION | MB_OK);
      CNTESF_B= FALSE;
      }
  }
}

TCeroArbitDialog::TCeroArbitDialog(PTWindowsObject AParent, int ResourceId)
            : TDialog(AParent, ResourceId)
{
  new TEdit(this, ID_Mp,
          sizeof(((TTransferWindow *)Parent)->RedAdelStruct.MpDato));
  new TEdit(this, ID_Ts,
          sizeof(((TTransferWindow *)Parent)->RedAdelStruct.tsDato));
  new TEdit(this, ID_Car,
          sizeof(((TTransferWindow *)Parent)->RedAdelStruct.CarDato));
  TransferBuffer = (void far*)&(((TTransferWindow *)Parent)->RedAdelStruct);
}

class TPIDialog : public TDialog {
public:
  TPIDialog(PTWindowsObject AParent, int ResourceId);
};

void TTransferWindow::CMdatosPI(TMessage&)
{
  char ALabel[255];
  char Kpsal[]="Kp = ";
  char Kisal[]="Ki = ";

  if ( GetModule()->ExecDialog(new TPIDialog(this, ID_PI)) == IDOK )
  {
    strcpy(ALabel, "Proporcional Integral:\n\n");
```

```
    strcat(ALabel, Kpsal);
    strcat(ALabel, PIStruct.PDato);
    strcat(ALabel, "\n");
    strcat(ALabel, Kisal);
    strcat(ALabel, PIStruct.IDato);
    MessageBox(HWindow, ALabel, "datos ingresados", MB_OK);

    CNTESF_A= FALSE;
    CNTESF_B= FALSE;
    CNTESF_C= TRUE;
    CNTESF_D= FALSE;
    CNTESF_E= FALSE;
    CNTESF_F= FALSE;

    Kp=atof(PIStruct.PDato);
    Ki=atof(PIStruct.IDato);

    if ((Kp<0.001) | (Ki<=0.001))
      {
      MessageBox(HWindow,
      "Los valores ingresados no estßn\n"
      "dentro de los rangos establecidos\n"
      "ingrese otros datos\n",
      "Advertencia",
      MB_ICONEXCLAMATION | MB_OK);
      CNTESF_C= FALSE;
      }
  }
}

TPIDialog::TPIDialog(PTWindowsObject AParent, int ResourceId)
            : TDialog(AParent, ResourceId)
{
  new TEdit(this, ID_P,
          sizeof(((TTransferWindow *)Parent)->PIStruct.PDato));
  new TEdit(this, ID_I,
          sizeof(((TTransferWindow *)Parent)->PIStruct.PDato));
  TransferBuffer = (void far*)&(((TTransferWindow *)Parent)->PIStruct);
}

class TPDDialog : public TDialog {
public:
  TPDDialog(PTWindowsObject AParent, int ResourceId);
};

void TTransferWindow::CMdatosPD(TMessage&)
{
  char ALabel[255];
  char Kpsal[]="Kp = ";
  char Kdsal[]="Kd = ";

  if ( GetModule()->ExecDialog(new TPDDialog(this, ID_PD)) == IDOK )
  {
    strcpy(ALabel, "Proporcional Derivativo:\n\n");
    strcat(ALabel, Kpsal);
    strcat(ALabel, PDStruct.PPDato);
```

```
    strcat(ALabel, "\n");
    strcat(ALabel, Kdsal);
    strcat(ALabel, PDStruct.DDDato);
    MessageBox(HWindow, ALabel, "datos ingresados", MB_OK);
    CNTESF_A= FALSE;
    CNTESF_B= FALSE;
    CNTESF_C= FALSE;
    CNTESF_D= TRUE;
    CNTESF_E= FALSE;
    CNTESF_F= FALSE;

    Kp=atof(PDStruct.PPDato);
    Kd=atof(PDStruct.DDDato);

    if ((Kp<0.001) | (Kd<=0.001))
      {
      MessageBox(HWindow,
      "Los valores ingresados no estßn\n"
      "dentro de los rangos establecidos\n"
      "ingrese otros datos\n",
      "Advertencia",
      MB_ICONEXCLAMATION | MB_OK);
      CNTESF_D= FALSE;
      }
  }
}

TPDDialog::TPDDialog(PTWindowsObject AParent, int ResourceId)
            : TDialog(AParent, ResourceId)
{
  new TEdit(this, ID_PP,
          sizeof(((TTransferWindow *)Parent)->PDStruct.PPDato));
  new TEdit(this, ID_DD,
          sizeof(((TTransferWindow *)Parent)->PDStruct.DDDato));
  TransferBuffer = (void far*)&(((TTransferWindow *)Parent)->PDStruct);
}

class TPIDDialog : public TDialog {
public:
  TPIDDialog(PTWindowsObject AParent, int ResourceId);
};

void TTransferWindow::CMdatosPID(TMessage&)
{
  char ALabel[255];
  char Kpsal[]="Kp = ";
  char Kisal[]="Ki = ";
  char Kdsal[]="Kd = ";

  if ( GetModule()->ExecDialog(new TPIDDialog(this, ID_PID)) == IDOK )
  {
    strcpy(ALabel, "Proporcional Integral Derivativo:\n\n");
    strcat(ALabel, Kpsal);
    strcat(ALabel, PIDStruct.PPPDato);
    strcat(ALabel, "\n");
    strcat(ALabel, Kisal);
```

```
    strcat(ALabel, PIDStruct.IIIDato);
    strcat(ALabel, "\n");
    strcat(ALabel, Kdsal);
    strcat(ALabel, PIDStruct.DDDDato);
    MessageBox(HWindow, ALabel, "datos ingresados", MB_OK);
    CNTESF_A= FALSE;
    CNTESF_B= FALSE;
    CNTESF_C= FALSE;
    CNTESF_D= FALSE;
    CNTESF_E= TRUE;
    CNTESF_F= FALSE;

    Kp=atof(PIDStruct.PPPDato);
    Ki=atof(PIDStruct.IIIDato);
    Kd=atof(PIDStruct.DDDDato);

    if ((Kp<0.001) | (Ki<=0.001) | (Kd<0.0001))
      {
      MessageBox(HWindow,
      "Los valores ingresados no estßn\n"
      "dentro de los rangos establecidos\n"
      "ingrese otros datos\n",
      "Advertencia",
      MB_ICONEXCLAMATION | MB_OK);
      CNTESF_E= FALSE;
      }
  }
}

TPIDDialog::TPIDDialog(PTWindowsObject AParent, int ResourceId)
            : TDialog(AParent, ResourceId)
{
  new TEdit(this, ID_PPP,
          sizeof(((TTransferWindow *)Parent)->PIDStruct.PPPDato));
  new TEdit(this, ID_III,
          sizeof(((TTransferWindow *)Parent)->PIDStruct.IIIDato));
  new TEdit(this, ID_DDD,
          sizeof(((TTransferWindow *)Parent)->PIDStruct.DDDDato));
  TransferBuffer = (void far*)&(((TTransferWindow *)Parent)->PIDStruct);
}

class TTransferDialog : public TDialog {
public:
  TTransferDialog(PTWindowsObject AParent, int ResourceId);
};

void TTransferWindow::CM_datosRealEst(TMessage&)
{
  double PoloImag;
  char str[25];
  int sig = 8; // significant digits

  char ALabel[255];
  char Polos[]="          Parte Real   Parte Imaginaria";
  char p1ing[]="P1 =   ";
  char p2ing[]="P2 =   ";
```

```
  char p3ing[]="P3 =    ";
  char falso[]="";

  if  (  GetModule()->ExecDialog(new  TTransferDialog(this,  ID_REAL_EST))  ==
IDOK )
  {
    PoloImag=-1*atof(RealEstStruct.pi1String);
    gcvt(PoloImag, sig, str);
    Pr[0]=atof(RealEstStruct.p1String);
    Pr[1]=atof(RealEstStruct.p2String);
    Pr[2]=atof(RealEstStruct.p3String);
    Pimag=atof(RealEstStruct.pi1String);

    strcpy(ALabel, "Realimentacion de estado:\n\n");
    strcat(ALabel, Polos);
    strcat(ALabel, "\n");
    strcat(ALabel, p1ing);
    strcat(ALabel, RealEstStruct.p1String);
    strcat(ALabel, "          ");
    strcat(ALabel, RealEstStruct.pi1String);
    strcat(ALabel, "\n");

       if(fabs(Pimag)>0)
        {
        strcat(ALabel, p2ing);
        strcat(ALabel, RealEstStruct.p1String);
        strcat(ALabel, "          ");
        strcat(ALabel, str);
        *RealEstStruct.p2String=*falso;
      }
       else
        {
        strcat(ALabel, p2ing);
          strcat(ALabel, RealEstStruct.p2String);
        }

    strcat(ALabel, "\n");
    strcat(ALabel, p3ing);
    strcat(ALabel, RealEstStruct.p3String);
    MessageBox(HWindow, ALabel, "datos ingresados", MB_OK);

    CNTESF_A= FALSE;
    CNTESF_B= FALSE;
    CNTESF_C= FALSE;
    CNTESF_D= FALSE;
    CNTESF_E= FALSE;
    CNTESF_F= TRUE;

    if(Pimag==0)
     {
      if((Pr[0]>=0)|(Pr[1]>=0)|(Pr[2]>=0))
        {
         MessageBox(HWindow,
         "Los polos de lazo cerrado no pueden\n"
         "tener parte real cero o positiva",
         "Advertencia",
```

```
          MB_ICONEXCLAMATION | MB_OK);
          CNTESF_F= FALSE;
         }
      }
     else
      {
       if((Pr[0]>=0)|(Pr[2]>=0))
       {
          MessageBox(HWindow,
          "Los polos de lazo cerrado no pueden\n"
          "tener parte real cero o positiva",
          "Advertencia",
          MB_ICONEXCLAMATION | MB_OK);
          CNTESF_F= FALSE;
         }
      }

  }
}

TTransferDialog::TTransferDialog(PTWindowsObject AParent, int ResourceId)
             : TDialog(AParent, ResourceId)
{
  new TEdit(this, ID_P1,
          sizeof(((TTransferWindow *)Parent)->RealEstStruct.p1String));
  new TEdit(this, ID_P2,
          sizeof(((TTransferWindow *)Parent)->RealEstStruct.p2String));
  new TEdit(this, ID_P3,
          sizeof(((TTransferWindow *)Parent)->RealEstStruct.p3String));
  new TEdit(this, ID_Pi1,
          sizeof(((TTransferWindow *)Parent)->RealEstStruct.pi1String));
  TransferBuffer = (void far*)&(((TTransferWindow *)Parent)->RealEstStruct);
}

class TTransDialog : public TDialog {
public:
  TTransDialog(PTWindowsObject AParent, int ResourceId);
};

void TTransferWindow::CMCondInic(TMessage&)
{
  char ALabel[255];
  char EspacInic[]="posici≤n  inicial =";
  char VelocInic[]="velocidad inicial =";
  char unidEspac[]="  cm";
  char unidVeloc[]="  cm/seg";

  if (  GetModule()->ExecDialog(new TTransDialog(this,  ID_COND_INIC)) == IDOK
)
  {
    strcpy(ALabel, "Condiciones Iniciales:\n\n");
    strcat(ALabel, EspacInic);
    strcat(ALabel, CoIniStruct.EspIniDato);
    strcat(ALabel, unidEspac);
    strcat(ALabel, "\n");
    strcat(ALabel, VelocInic);
```

```
    strcat(ALabel, CoIniStruct.VelIniDato);
    strcat(ALabel, unidVeloc);
    strcat(ALabel, "\n");

    MessageBox(HWindow, ALabel, "datos ingresados", MB_OK);
    RESP2_C = TRUE;
    RESP2_R = FALSE;

    Pos0=atof(CoIniStruct.EspIniDato);
    VEL0=atof(CoIniStruct.VelIniDato);

    if ((Pos0>1.0) | (Pos0<-1.0) | (fabs(Pos0)<=0.01))
      {
      MessageBox(HWindow,
      "Los valores ingresados no estßn\n"
      "dentro de los rangos establecidos\n"
      "ingrese otros datos\n",
      "Advertencia",
      MB_ICONEXCLAMATION | MB_OK);
      RESP2_C = FALSE;
      }
  }
}

TTransDialog::TTransDialog(PTWindowsObject AParent, int ResourceId)
                : TDialog(AParent, ResourceId)
{
  new TEdit(this, ID_EspI,
          sizeof(((TTransferWindow *)Parent)->CoIniStruct.EspIniDato));
  new TEdit(this, ID_VelI,
          sizeof(((TTransferWindow *)Parent)->CoIniStruct.VelIniDato));
  TransferBuffer = (void far*)&(((TTransferWindow *)Parent)->CoIniStruct);
}

class TSetPointDialog : public TDialog {
public:
  TSetPointDialog(PTWindowsObject AParent, int ResourceId);
};

void TTransferWindow::CMSetPoint(TMessage&)
{
  char ALabel[255];
  char datoSetP[]="  milivoltios";
  if  (  GetModule()->ExecDialog(new  TSetPointDialog(this,  ID_SETPOINT))  ==
IDOK )
  {
    Ampl=atof(SetPointStruct.SetPointDato);
    Ampl=Ampl/10;
    strcpy(ALabel, "Respuesta a una\nEntrada Escal≤n:\n");
    strcat(ALabel, SetPointStruct.SetPointDato);
    strcat(ALabel, datoSetP);
    strcat(ALabel, "\n");
    MessageBox(HWindow, ALabel, "Set Point", MB_OK);

    RESP2_C = FALSE;
    RESP2_R = TRUE;
```

```
  }
}

TSetPointDialog::TSetPointDialog(PTWindowsObject AParent, int ResourceId)
                : TDialog(AParent, ResourceId)
{
  new TEdit(this, ID_SetP,
          sizeof(((TTransferWindow *)Parent)->SetPointStruct.SetPointDato));
  TransferBuffer = (void far*)&(((TTransferWindow *)Parent)->SetPointStruct);
}

TTransferWindow::TTransferWindow(PTWindowsObject AParent, LPSTR ATitle) :
  TWindow(AParent, ATitle)
{
  AssignMenu(ID_MENU);
  memset(&CoIniStruct, 0x0, sizeof CoIniStruct);
  memset(&SetPointStruct, 0x0, sizeof SetPointStruct);
  memset(&RedAdelStruct, 0x0, sizeof RedAdelStruct);
  memset(&PIStruct, 0x0, sizeof PIStruct);
  memset(&PDStruct, 0x0, sizeof PDStruct);
  memset(&PIDStruct, 0x0, sizeof PIDStruct);
  memset(&RealEstStruct, 0x0, sizeof RealEstStruct);
}

void TTransferApp::InitInstance()
{
      TApplication::InitInstance();
      HAccTable =  LoadAccelerators( hInstance, MAKEINTRESOURCE( SDBSAPACCEL )
);
      hHourGlass= LoadCursor(NULL,IDC_WAIT);
//    hMiBitMap = LoadBitMap(hInstance,MAKEINTRESOURCE(MiBitMap));
}

void TTransferWindow::Paint( HDC hdc, PAINTSTRUCT _FAR &)
{

// Funciones utilizadas

if(Comienzo==TRUE)
{
 Comienzo=FALSE;
 GetApplication()->ExecDialog(new TDialog(this,"ABOUT"));
}

if(GrafiAnimaci== TRUE)
Fondo();

if(Resultados==TRUE)
{
if((CNTESF_A==TRUE) | (CNTESF_B==TRUE))
InfEsfera(ResMp,Rests, ResSG, ResJW, ResPOLOC, ResCEROC, ResFASEC, ResKcp);
if((CNTESF_A==TRUE) | (CNTESF_B==TRUE) |  (CNTESF_C==TRUE) |  (CNTESF_D==TRUE)
| (CNTESF_E==TRUE))
Resultado(ResNUMP);
}
```

```
if( GrafiEspacio ¦ GrafiVelocid
    ¦GrafiErrPosi)
GraficaEjes();

if(GrafiEspacio==TRUE)
GrafEspacio();

if(GrafiVelocid== TRUE)
GrafVelocidad();

if(GrafiErrPosi== TRUE)
GrafError();

if(GraficarTodo==TRUE)
{
GraficaEjes();
GrafEspacio();
GrafVelocidad();
GrafError();
}

if(GrafiVeloEsp== TRUE)
GrafFase();

      SetBkMode( hdc, TRANSPARENT );
}

void TTransferWindow::GetWindowClass(WNDCLASS& WndClass )
{
      TWindow::GetWindowClass(WndClass);
      WndClass.hCursor=LoadCursor(NULL,IDC_CROSS);
      WndClass.hIcon=LoadIcon(GetApplication()->hInstance,"Icon_SDBS");
}

LPSTR TTransferWindow::GetClassName()
{
      return "TTransferWindow";
}

void TTransferWindow::WMEnterIdle( RTMessage Msg )
{
      if( (Msg.WParam  == MSGF_MENU ) &&  ((GetKeyState( VK_F1 ) &  0x8000) !=
0) )
            // if the high bit is set, then the key is pressed
      {
            tfF1Pressed = TRUE;
            PostMessage( HWindow, WM_KEYDOWN, VK_RETURN, 0L );
      }
}

void TTransferWindow::CMExit( RTMessage Msg )
{
            TWindow::CMExit( Msg );
}

void TTransferWindow::CMUHelpIndex( RTMessage )
```

```
{
      WinHelp( HWindow, "SDBS.HLP", HELP_INDEX, 0L );
}

void TTransferWindow::CMUHelpHelp( RTMessage )
{
      WinHelp( HWindow, "WINHELP.HLP", HELP_HELPONHELP, 0L );
}

void TTransferWindow::CMUHelpAbout( RTMessage )
{

 GetApplication()->ExecDialog(new TDialog(this,"ABOUT"));

}

void TTransferApp::InitMainWindow()
{
  MainWindow = new TTransferWindow(NULL, "SDBS");
}

// Main program
int PASCAL WinMain(HINSTANCE hInstance, HINSTANCE hPrevInstance,
  LPSTR lpCmdLine, int nCmdShow)
{
      HINSTANCE hBorLibrary;
      hBorLibrary=LoadLibrary("bwcc.dll");
      if((int) hBorLibrary<=32)
         MessageBox(NULL, "No hay como abrir la libreria","error",MB_OK);

  TTransferApp TransferApp("TransferTest", hInstance, hPrevInstance,
    lpCmdLine, nCmdShow);

  TransferApp.Run();
  return TransferApp.Status;
}

//      LLamadas a funciones desde el menu

void TTransferWindow::CMimprimirPant(RTMessage)
{
  HDC hDisplayDC;
  HDC hMemoryDC; // contexto de dispositivo en memoria para mapa bits
  HBITMAP hBitmap, hPrevBitmap3;      // gestores de los mapas de bits

PRINTDLG pd;

hDisplayDC=GetDC(HWindow);

hMemoryDC=  // establecer contexto disp. en memoria para mapa de bits
           CreateCompatibleDC(hDisplayDC);
hBitmap= CreateCompatibleBitmap(              // crear mapa de bits...
                   hDisplayDC,          // compatible con la pantalla
               anchoVentx,                      // anchura
               altoVenty);                     // altura
hPrevBitmap3= (HBITMAP)SelectObject(hMemoryDC,hBitmap); //...y seleccionarlo
```

```
BitBlt(hMemoryDC,     // copiar imagen de la pantalla al mapa de bits
       0,0,       // coords de la esq. superior izquierda del destino
       anchoVentx,                                  // anchura
       altoVenty,                                  // altura
       hDisplayDC,                               // mapa de bits origen
       0,0,       // coords de la esq. superior izquierda del origen
       SRCCOPY);                          // operación gráfica : copiar


// Set all structure members to zero.
memset(&pd, 0, sizeof(PRINTDLG));

// Initialize the necessary PRINTDLG structure members.

pd.lStructSize = sizeof(PRINTDLG);
pd.hwndOwner = HWindow;
pd.Flags = PD_RETURNDC;

// Print a test page if successful
if (PrintDlg(&pd) != 0) {
    Escape(pd.hDC, STARTDOC, 8, "Test-Doc", NULL);

//  copiar mapa bits al contexto disp. en memoria para impres.
StretchBlt(pd.hDC,  // destino
           0,0,  // esq. superior izquierda del rectángulo de destino
         anchoVentx,   // anchura del destino
         altoVenty,   // altura del destino
           hMemoryDC,
           0,0,   // esq. superior izquierda del rectángulo de origen
         anchoVentx,    // anchura del origen
         altoVenty,    // altura del origen
         SRCCOPY);    // operación gráfica = copiar

//    Print text and rectangle
//    TextOut(pd.hDC, 50, 50, "Common Dialog Test Page", 23);
//    Rectangle(pd.hDC, 50, 90, 625, 105);

    Escape(pd.hDC, NEWFRAME, 0, NULL, NULL);
    Escape(pd.hDC, ENDDOC, 0, NULL, NULL);
    DeleteDC(pd.hDC);
    if (pd.hDevMode != NULL)
       GlobalFree(pd.hDevMode);
    if (pd.hDevNames != NULL)
       GlobalFree(pd.hDevNames);
}
else
;//    ErrorHandler();

SelectObject(hMemoryDC,hPrevBitmap3); // deseleccionar el mapa de bits
DeleteObject(hBitmap);   // borrar el mapa de bits
DeleteDC(hMemoryDC);  // borrar contexto disp. en mem. para mapa bits
ReleaseDC(HWindow,hDisplayDC);
}

void TTransferWindow::CMcopyClipboard(RTMessage)
{
```

```
  HDC hDC, hMemDC;
  HBITMAP hBitmap, hPrevBitmap3;
  int xwidth, ydepth;
xwidth= anchoVentx;
ydepth= altoVenty;
hDC= GetDC(HWindow);    // establecer el contexto de visualización
hMemDC= CreateCompatibleDC(hDC); // crear contexto visual. en memoria
hBitmap=                         // crear mapa de bits
         CreateCompatibleBitmap(hDC, xwidth, ydepth);
if (hBitmap)                 // si el mapa de bits se creó bien...
  {
  hPrevBitmap3=   (HBITMAP)SelectObject(hMemDC,  hBitmap);//seleccionar   mapa
bits
  BitBlt(hMemDC,0,0,xwidth,ydepth,hDC,0,0,SRCCOPY); // cargarlo
  OpenClipboard(HWindow);              // abrir el portapapeles
  EmptyClipboard();                  // limpiar el portapapeles
  SetClipboardData(CF_BITMAP, hBitmap);  // pasarle el gestor
  CloseClipboard();           // cerrar el portapapeles
  SelectObject(hMemDC, hPrevBitmap3); // deseleccionar el mapa de bits
  MessageBox(HWindow,
      "el grßfico actual se copi≤\n"
      "al portapapeles",
      "Clipboard",MB_OK);
  } // NOTA: No libere el gestor del mapa de bits después de pasarlo.
else
  {                // si existe algún error, mostrar un mensaje...
  MessageBeep(0);
  MessageBox(HWindow,"No se pudo copiar al portapapeles","error",MB_OK);
  }
DeleteDC(hMemDC);  // borrar el contexto de visualización en memoria
ReleaseDC(HWindow, hDC);   // liberar el contexto de visualización

}
void TTransferWindow::CMReloj(RTMessage)
{
 WinExec("clock.exe",SW_SHOWNORMAL);
}
void TTransferWindow::CMCalculadora(RTMessage)
{
 WinExec("calc.exe",SW_SHOWNORMAL);
}
void TTransferWindow::CMArchivos(RTMessage)
{
 WinExec("winfile.exe",SW_SHOWNORMAL);
}

void TTransferWindow::ClearMenu(RTMessage)
{
GrafiAnimaci= FALSE;
GrafiEspacio= FALSE;
GrafiVelocid= FALSE;
GrafiErrPosi= FALSE;
GraficarTodo= FALSE;
GrafiVeloEsp= FALSE;
Resultados=FALSE;
LimpiarPantalla();
```

```
}

void TTransferWindow::ResultadosMenu(RTMessage)
{
GrafiAnimaci= FALSE;
GrafiEspacio= FALSE;
GrafiVelocid= FALSE;
GrafiErrPosi= FALSE;
GraficarTodo= FALSE;
GrafiVeloEsp= FALSE;
Resultados=TRUE;
LimpiarPantalla();
}

void TTransferWindow::CalculosMenu(RTMessage)
{
GrafiAnimaci= FALSE;
GrafiEspacio= FALSE;
GrafiVelocid= FALSE;
GrafiErrPosi= FALSE;
GraficarTodo= FALSE;
GrafiVeloEsp= FALSE;
Resultados=FALSE;
LimpiarPantalla();

if((CNTESF_A==TRUE) ¦  (CNTESF_B==TRUE) ¦ (CNTESF_C==TRUE) ¦  (CNTESF_D==TRUE)
¦ (CNTESF_E==TRUE))
Principal();
if(CNTESF_F== TRUE)
RealimEstad();

}

void TTransferWindow::AnimacionMenu(RTMessage)
{
int bandera=0;
if(GrafiAnimaci== TRUE)
bandera=1;

GrafiAnimaci= TRUE;
GrafiEspacio= FALSE;
GrafiVelocid= FALSE;
GrafiErrPosi= FALSE;
GraficarTodo= FALSE;
GrafiVeloEsp= FALSE;
Resultados=FALSE;

if(bandera==0)
LimpiarPantalla();
Animacion();
}

void TTransferWindow::GrafEspMenu(RTMessage)
{

if(Calculos==FALSE)
```

```
 {
  MessageBox(HWindow,
  "Todavφa no se han hecho cβlculos",
  "Advertencia",
  MB_ICONEXCLAMATION | MB_OK);
  return;
 }

GrafiAnimaci= FALSE;
GrafiEspacio= TRUE;
GrafiVelocid= FALSE;
GrafiErrPosi= FALSE;
GraficarTodo= FALSE;
GrafiVeloEsp= FALSE;
Resultados=FALSE;
LimpiarPantalla();
}

void TTransferWindow::GrafVelMenu(RTMessage)
{

if(Calculos==FALSE)
 {
  MessageBox(HWindow,
  "Todavφa no se han hecho cβlculos",
  "Advertencia",
  MB_ICONEXCLAMATION | MB_OK);
  return;
 }

GrafiAnimaci= FALSE;
GrafiEspacio= FALSE;
GrafiVelocid= TRUE;
GrafiErrPosi= FALSE;
GraficarTodo= FALSE;
GrafiVeloEsp= FALSE;
Resultados=FALSE;
LimpiarPantalla();
}

void TTransferWindow::GrafErrMenu(RTMessage)
{

if(Calculos==FALSE)
 {
  MessageBox(HWindow,
  "Todavφa no se han hecho cβlculos",
  "Advertencia",
  MB_ICONEXCLAMATION | MB_OK);
  return;
 }

GrafiAnimaci= FALSE;
GrafiEspacio= FALSE;
GrafiVelocid= FALSE;
GrafiErrPosi= TRUE;
```

```
GraficarTodo= FALSE;
GrafiVeloEsp= FALSE;
Resultados=FALSE;
LimpiarPantalla();
}

void TTransferWindow::GrafTodMenu(RTMessage)
{

if(Calculos==FALSE)
 {
  MessageBox(HWindow,
  "Todavφa no se han hecho cßlculos",
  "Advertencia",
  MB_ICONEXCLAMATION | MB_OK);
  return;
 }

GrafiAnimaci= FALSE;
GrafiEspacio= FALSE;
GrafiVelocid= FALSE;
GrafiErrPosi= FALSE;
GraficarTodo= TRUE;
GrafiVeloEsp= FALSE;
Resultados=FALSE;
LimpiarPantalla();
}

void TTransferWindow::GrafFasMenu(RTMessage)
{

if(Calculos==FALSE)
 {
  MessageBox(HWindow,
  "Todavφa no se han hecho cßlculos",
  "Advertencia",
  MB_ICONEXCLAMATION | MB_OK);
  return;
 }

GrafiAnimaci= FALSE;
GrafiEspacio= FALSE;
GrafiVelocid= FALSE;
GrafiErrPosi= FALSE;
GraficarTodo= FALSE;
GrafiVeloEsp= TRUE;
Resultados=FALSE;
LimpiarPantalla();
}

void TTransferWindow::WMSize(TMessage& Message)
{
anchoVentx=Message.LP.Lo;
altoVenty=Message.LP.Hi;
UbicEjex=(altoVenty-50)/2+40;
```

```
if(bCeldasListas == TRUE)
  {
  SelectObject(hPaginaOcultaDC,hPrevBitmap2);
  DeleteObject(hPaginaOculta);
  DeleteDC(hPaginaOcultaDC);
  SelectObject(hMapaDC,hPrevBitmap1);
  DeleteObject(hMapa);
  DeleteDC(hMapaDC);
  }
ConstrCeldas();

}

// funcion para el fondo de pantalla
void TTransferWindow::Fondo(void)
{
 HDC hDC,hMemDC;
 hDC = GetDC(HWindow);

hMemDC= CreateCompatibleDC(hDC);  // crea contexto visual. en memoria
  GetObject(SDBSBmp, // obtener información de cabecera del mapa bits
          sizeof(BITMAP),(LPSTR) &bmPaste);

hPrevBitmap=(HBITMAP)SelectObject(hMemDC, SDBSBmp);// seleccionar mapa bits

StretchBlt(hDC,0,0,anchoVentx,altoVenty,hMemDC,0,0,bmPaste.bmWidth,bmPaste.bm
Height,SRCCOPY);

SelectObject(hMemDC,hPrevBitmap);  // deseleccionar el mapa de bits
DeleteDC(hMemDC);   // borrar el contexto de visualización compatible

ReleaseDC(HWindow,hDC);   // liberar el contexto de visualizacion
}

void TTransferWindow::WMRaton(TMessage& Message)
{
coordenadaX=Message.LP.Lo;
coordenadaY=Message.LP.Hi;
if(GrafiEspacio== TRUE)
CoorEspacio();
if(GrafiVelocid== TRUE)
CoorVelocid();
if(GrafiErrPosi== TRUE)
CoorError();
if(GrafiVeloEsp== TRUE)
CoorFase();
}

// CODIGO DE LAS FUNCIONES UTILIZADAS
// Funci≤n para realizar calculos de la Bisectriz y Cero Arbitrario

void TTransferWindow::Principal(void)
{
double SG,JW,FASE,FASEC, POLOC, NUMP, NUMZ, GAN, Kcp, Kft, GP,CAM;
double GQ,MENOR,tempTMT,tempTmp1,tempTmp2;
double Tsim,Cond;
```

```
int i,m;

if(((RESP2_C==TRUE)|(RESP2_R==TRUE))==0)
 {
  MessageBox(HWindow,
  "Falta ingresar datos",
  "Advertencia",
  MB_ICONEXCLAMATION | MB_OK);
  return;
 }

for(i=1;i<=6;i++)
      {
       Po[i]=0;
       Qo[i]=0;
       PROD1[i]=0;
       PartReal[i]=0;
       PartImag[i]=0;
       PROD2[i]=0;
       PLC[i]=0;
       COR1[i]=0;
       COI1[i]=0;
       COR2[i]=0;
       COI2[i]=0;
       RealInic[i]=0;
       ImagInic[i]=0;
        }

Kpl=47.7;
Ksn=13.2;

if((CNTESF_A==TRUE)|(CNTESF_B==TRUE))
goto PROS1;
if((CNTESF_C==TRUE)|(CNTESF_D==TRUE)|(CNTESF_E==TRUE))
goto ACCIONES;

Mp = Mp/100;
CAM = sqrt(log(Mp)*log(Mp) / (PI*PI + log(Mp)*log(Mp)));
Wn = 4 / (CAM * ts);
SG = -CAM * Wn;
JW = Wn * sqrt(1 - CAM*CAM);

PLR[1] = 21.3;
PLR[2] = -21.3;
PLR[3] = -104.6;

CalcFase(0,3,ZR,ZI,PLR,PLI,SG,JW,FASE, FASEC);

if(CNTESF_A==TRUE)
goto MTDBIC;

if(CNTESF_B==TRUE)
goto MTDCER;

//   Proceso para el metodo de la Bisectriz
MTDBIC:
```

```
        Bicectriz(SG,JW,FASEC,CEROC,POLOC);
        goto BICCER;

//  Proceso para el metodo del cero arbitrario
MTDCER:
      CeroArb(SG,JW,FASEC,CEROC,POLOC);
      if(POLOC>=CEROC)
          {
           MessageBox(NULL,
         "La ubicaci≤n propuesta para el cero\n"
           "del compensador impide la construcci≤n\n\n"
         "de la red de adelanto",
         "Error",MB_OK);
           return;
        }

// Proceso común a los dos metodos
BICCER:

//    Polos de la F.T. en lazo abierto:
        NUMP = 4;
      PLR[1] = POLOC;
      PLR[2] = 21.3;
      PLR[3] = -21.3;
      PLR[4] = -104.6;

      for (i = 1; i<=4;i++)
      PLI[i] = 0;

//      Ceros de la F.T.:
        NUMZ = 1;
      ZR[1] = CEROC;
      ZI[1] = 0;

//      Ganancias:
      GAN = Kpl * Ksn;

// Se encuentra la ganancia que requiere el compensador para que el L.G.R
// pase por los polos deseados
      GanaComp(NUMZ,NUMP,ZR,ZI,PLR,PLI,SG,JW,GAN,Kcp);
        Kft = Kcp * Kpl;

//Aqui presento resultados solo de INICIO DE CALCULOS
ResMp=Mp;Rests=ts;ResSG=SG;ResJW=JW;ResPOLOC=POLOC;ResCEROC=CEROC;ResFASEC=FA
SEC;ResKcp=Kcp;

// Resultados de los analisis teoricos

//Encuentro el polinomio en lazo cerrado
//para el metodo de la red de adelanto

//        Polinomio GH(s) de la F.T.:
      PROD2[1] = Kcp * Kpl * Ksn;
      PROD2[0] = -CEROC * PROD2[1];

//        Polinomios a multiplicar:
```

```
        GP = 2;
        Po[2] = 1;
        Po[1] = -(PLR[1] + PLR[2]);
        Po[0] = PLR[1] * PLR[2];
        GQ = 2;
        Qo[2] = 1;
        Qo[1] = -(PLR[3] + PLR[4]);
        Qo[0] = PLR[3] * PLR[4];
          goto COMUN;

ACCIONES:

//  Ingreso las variables de la subrutina que calcula el polinomio
//  caracteristico en lazo cerrado

       if(CNTESF_C==TRUE)
                {
                 NUMZ = 1;
                 NUMP = 4;
                    //Polos de la F.T. en lazo abierto:
                 PLR[1] = 21.3;
                 PLR[2] = -21.3;
                 PLR[3] = -104.6;
                 PLR[4] = 0;
                    //Polinomio GH(s) de la F.T.:
                 PROD2[1] = Kp * Kpl * Ksn;
                 PROD2[0] = Ki * PROD2[1];
                    //Ganancias :
                 Kft = Kp * Kpl;
                    //Polinomios a multiplicar:
                 GP = 2;
                 Po[2] = 1;
                 Po[1] = -PLR[1];
                 Po[0] = 0;

                 GQ = 2;
                 Qo[2] = 1;
                 Qo[1] = -(PLR[2] + PLR[3]);
                 Qo[0] = PLR[2] * PLR[3];
           }

           if(CNTESF_D==TRUE)
                {
                NUMZ = 1;
                NUMP = 3;
                   //Polos de la F.T. en lazo abierto:
                PLR[1] = 21.3;
                PLR[2] = -21.3;
                PLR[3] = -104.6;
                   //Polinomio GH(s) de la F.T.:
                PROD2[1] = Kp * Kd * Kpl * Ksn;
                PROD2[0] = PROD2[1] / Kd;
                   //ganancia:
                    Kft = Kp * Kd * Kpl;
                   //Polinomios a multiplicar:
                GP = 1;
```

```
                Po[1] = 1;
                Po[0] = -PLR[3];

                GQ = 2;
                Qo[2] = 1;
                Qo[1] = -(PLR[1] + PLR[2]);
                Qo[0] = PLR[1] * PLR[2];
            }

            if(CNTESF_E==TRUE)
                {
                NUMZ = 2;
                NUMP = 4;
                   //Polos de la F.T. en lazo abierto:
                PLR[1] = 21.3;
                PLR[2] = -21.3;
                PLR[3] = -104.6;
                PLR[4] = 0;
                   //Polinomio GH(s) de la F.T.:
                PROD2[2] = Kp * Kd * Kpl * Ksn;
                PROD2[1] = PROD2[2] * (1 / Kd);
                PROD2[0] = PROD2[2] * (Ki / Kd);
                   //ganancias :
                    Kft = Kp * Kd * Kpl;
                   //Polinomios a multiplicar:
                GP = 2;
                Po[2] = 1;
                Po[1] = -PLR[1];
                Po[0] = 0;
                GQ = 2;
                Qo[2] = 1;
                Qo[1] = -(PLR[2] + PLR[3]);
                Qo[0] = PLR[2] * PLR[3];
                  }

COMUN:
//Encuentro el polinomio caracteristico en lazo cerrado

      MultPoli(GP,GQ,Po,Qo,PROD1);

//GRADO=GP+GQ;

      SumPoli(GP+GQ,PROD1,PROD2,PLC);

//Encuentro las raices del polinomio caracteristico en lazo cerrado

      RootPoli(GP+GQ,PLC,PartReal,PartImag);

//    RESPUESTA EN EL TIEMPO

ResNUMP=NUMP;

//Aqui presento RESULTADOS DE LOS POLOS

      FLAGG = 0;
      for(m = 1;m<=NUMP;m++)
```

```
        {
        if(PartReal[m] >= 0)
        FLAGG = 1;
        ESTAB = 0;
       }
       if(FLAGG== 1)
         {
        ESTAB = 1;
         }

if(RESP2_R==TRUE)
  {
       if((CNTESF_A==TRUE) | (CNTESF_B==TRUE))
         {
          NUMZ=1;
        NUMP=4;
        ZR[1]=CEROC;
         }

       else if(CNTESF_C==TRUE)
         {
          NUMZ=1;
        NUMP=4;
        ZR[1]=-Ki;
         }

       else if(CNTESF_D==TRUE)
         {
          NUMZ=1;
        NUMP=3;
        ZR[1]=-1/Kd;
         }

       else
       {
          NUMZ=2;
        NUMP=4;
        PLC[2]=1;
        PLC[1]=1/Kd;
          PLC[0]=Ki/Kd;
          RootPoli(2,PLC,ZR,ZI);
         }

       CoeFrac(NUMZ,NUMP+1,ZR,ZI,PartReal,PartImag,COR1,COI1);

       for(i=1;i<=(NUMP+1);i++)
         {
             COR1[i]=COR1[i]*Kft*Ampl;
       COI1[i]=COI1[i]*Kft*Ampl;

       if(fabs(COI1[i])<0.0001)
       COI1[i]=0;
         }
       VELO=0;

       goto SIMULAC;
```

```
  }
else if (RESP2_C==TRUE)
  {
   CntCndIni:

      PLC[2]=1;
      PLC[1]=104.6+(VELO / Pos0);
      PLC[0]=-452.3+104.6*(VELO/Pos0);

      RootPoli(2,PLC,RealInic,ImagInic);

      ZR[1]=RealInic[1];
      ZI[1]=ImagInic[1];
      ZR[2]=RealInic[2];
      ZI[2]=ImagInic[2];

      if((CNTESF_A==TRUE)|(CNTESF_B==TRUE))
      {
       ZR[3]=POLOC;
       ZI[3]=0;
       NUMZ=3;
      }
      else if((CNTESF_C==TRUE)|(CNTESF_E==TRUE))
      {
       ZR[3]=0;
       ZI[3]=0;
       NUMZ=3;
      }
      else
      {
       NUMZ=2;
        }

      CoeFrac(NUMZ,NUMP,ZR,ZI,PartReal,PartImag,COR2,COI2);

      for(i=1;i<=NUMP;i++)
      {
       COR2[i]=Pos0*COR2[i];
       COI2[i]=Pos0*COI2[i];
      }

      Ampl=0;
      goto SIMULAC;
  }
else;

SIMULAC:
      if(ESTAB== 0)
           {
                MENOR = fabs(PartReal[1]);
           for(i = 1;i<=NUMP;i++)
                {
                if(MENOR >fabs(PartReal[i]))
                MENOR = fabs(PartReal[i]);
                 }
           if((MENOR < 2)|(MENOR > 20))
```

```
                {
                 MessageBox(NULL,
                 "Las constantes ingresadas para la acci≤n\n"
                 "de control propuesta no permiten obtener\n"
                 "un tiempo de establecimiento dentro de\n"
                 "los rangos permitidos por el sistema\n"
                 "0.2 seg < Ts (2%) < 1 seg",
                 "Error",MB_OK);
                   return;
                     }
                  else if(MENOR < 0.01)
                 {
                  Tsim = 10;
                  Psim = Tsim / 100;
                 }
                  else
                 {
                 Tsim = 5 / MENOR;
                   Psim = Tsim / 100;
                     if(Tsim > 3)
                      {
                       Psim = 0.02;
                       Tsim = 3;
                          }
                     }
          }
       else
          {
           Psim = 0.04;
           Tsim = 12;
          }

         for(i = 0;i<=NUMINT;i++)
           TMT[i] = 0;

       Cond = 1.1;

if(RESP2_R==TRUE)
  {
         for(i = 0;i<=NUMINT;i++)
        TMT[i] = 0;
       VecTiempo(ESTAB,  Cond,  Psim, Tsim,  GP  +  GQ, COR1,  COI1,  PartReal,
PartImag, TMT, NUMINT);

         MaxEsp=0.0;
       for(i=0;i<=NUMINT;i++) // maximo valor de TMT
          {
        tempTMT=fabs(TMT[i]);
        if(MaxEsp<tempTMT)
        MaxEsp=tempTMT;
          }

  }
else if(RESP2_C==TRUE)
  {
       for(i=0;i<=NUMINT;i++)
```

```
      TMT[i] = 0;
      VecTiempo(ESTAB,Cond,Psim,  Tsim, GP  + GQ  - 1,  COR2, COI2,  PartReal,
PartImag, Tmp2, NUMINT);

        MaxEsp=0.0;
      for(i=0;i<=NUMINT;i++) // maximo valor de TMT
         {
       TMT[i] = TMT[i] + Tmp2[i];
       tempTMT=fabs(TMT[i]);
       if(MaxEsp<tempTMT)
       MaxEsp=tempTMT;
       }

  }
else;

// Error
   MaxErr=0;
   for(i=0;i<= NUMINT;i++)
   {
    Tmp1[i] = Ampl - 13.2 * TMT[i];
    tempTmp1=fabs(Tmp1[i]);
    if(MaxErr<tempTmp1)
    MaxErr=tempTmp1;
   }

//  Velocidad vs Posicion
   Tmp2[0] = VEL0;
   MaxVel=fabs(Tmp2[0]);
   for(i = 1;i<=(NUMINT - 2);i++)
   {
    Tmp2[i] = (TMT[i + 1] - TMT[i - 1]) / (2 * Psim);
    tempTmp2=fabs(Tmp2[i]);
    if(MaxVel<tempTmp2)
    MaxVel=tempTmp2;
   }

SimDinEsf:

Mp=Mp*100;        // para realizar de nuevo cβlculos
Calculos=TRUE;
}

void TTransferWindow::Resultado(double NUMP1)
{
int i;
char str[25];
int sig = 6; // significant digits
int longitud;

HDC hDC= GetDC(HWindow); // establecer el contexto de visualización

TextOut(hDC,0,220,"Los Polos de Lazo Cerrado son:",30);
TextOut(hDC,0,240,"P1 = ",5);
TextOut(hDC,0,260,"P2 = ",5);
TextOut(hDC,0,280,"P3 = ",5);
```

```
if(NUMP1==4)
TextOut(hDC,0,300,"P4 = ",5);

      for(i = 1; i<=NUMP1;i++)
        {
      gcvt(PartReal[i], sig, str);
      longitud=strlen(str);
      TextOut(hDC,50,220+i*20,str,longitud);
          if(PartImag[i] > 0)
          {
            gcvt(PartImag[i], sig, str);
            longitud=strlen(str);
            TextOut(hDC,150,220+i*20,str,longitud);
             }
          else if(PartImag[i] < 0)
             {
            gcvt(PartImag[i], sig, str);
            longitud=strlen(str);
            TextOut(hDC,150,220+i*20,str,longitud);
             }
      }

      FLAGG = 0;
      for(i = 1;i<=NUMP1;i++)
        {
       if(PartReal[i] >= 0)
       FLAGG = 1;
       ESTAB = 0;
      }
      if(FLAGG== 1)
        {
       TextOut(hDC,0,320,"SISTEMA INESTABLE !",19);
       ESTAB = 1;
        }

ReleaseDC(HWindow,hDC);   // liberar el contexto de visualización
}

void TTransferWindow::CalcFase(double NUMZ,double NUMP,double ZR[],double
ZI[],double PLR[],double PLI[],double SG,double JW, double &FASE, double
&FASEC)
{
double RE,IM,ANG;
int i;
      FASE = 0;
      if(NUMZ == 0)
      goto NoCeros;
//      Calculo la fase que introducen los ceros
      for (i = 1;i<= NUMZ;i++)
        {
          RE = SG - ZR[i];
          IM = JW - ZI[i];
          Cuad(RE,IM,ANG);
            FASE = FASE + ANG;
      }
//      Calculo fase que introducen los polos y la resto de la anterior
```

```
NoCeros:
      for(i = 1;i<= NUMP;i++)
        {
          RE = SG - PLR[i];
          IM = JW - PLI[i];
          Cuad(RE,IM,ANG);
            FASE = FASE - ANG;
      }
//      Encuentro la fase del que debe introducir el compensador
        FASEC = -PI - FASE;
}

// Subrutina que encuentra el cuadrante en el que se halla el angulo
// basandose en los signos de la parte real (RE) y la parte Imaginaria (IM)

void TTransferWindow::Cuad(double RE1,double IMS,double &ANG1)
{
      if(RE1 > 0 && IMS > 0)
         ANG1 = atan(IMS / RE1);
      else if(RE1 == 0 && IMS > 0)
         ANG1 = PI / 2;
      else if(RE1 == 0 && IMS < 0)
         ANG1 = -PI / 2;
      else if(RE1 > 0 && IMS < 0)
         ANG1 = 2 * PI + atan(IMS / RE1);
        else
         ANG1 = PI + atan(IMS / RE1);
}

void TTransferWindow::Bicectriz (double SG,double JW,double FASEC, double
&CEROC,double &POLOC)
{
double DenPoloc, DenCeroc;
      DenPoloc = tan(.5 * (PI - atan(fabs(JW / SG))) - .5 * FASEC);
      DenCeroc = tan(.5 * (PI - atan(fabs(JW / SG))) + .5 * FASEC);
        POLOC = SG - (JW / DenPoloc);
      CEROC = SG - (JW / DenCeroc);
}

void TTransferWindow::GanaComp(double NUMZ, double NUMP,double ZR[],double
ZI[],double PLR[], double PLI[],double SG,double JW,double GAN, double &Kcp)
{
int i;
double MDL,RE,IM;
//Calculo el modulo del numerador de la Gtot(S)
      if(NUMZ == 0)
        goto NoCero;
            for(i = 1;i<= NUMZ;i++)
                {
              RE = SG - ZR[i];
              IM = JW - ZI[i];
              MDL = sqrt(RE*RE + IM*IM);

            if(MDL == 0)
            goto Siga;
```

```
                GAN = GAN * MDL; //Caso de simplificacion de polo con cero
              Siga:
                }
//Calculo el modulo del denominador y lo divido entre el anterior

NoCero: for(i = 1;i<=NUMP;i++)
        {
      RE = SG - PLR[i];
      IM = JW - PLI[i];
      MDL = sqrt(RE*RE + IM*IM);
      if(MDL == 0)
      goto Otro;
        GAN = GAN / MDL;
      Otro:
      }
//Encuentro el Valor de la ganancia del compensador Kcp

Kcp = 1 / GAN;
}

void  TTransferWindow::InfEsfera(double  Mp,  double  ts,double  SG,double
JW,double POLOC,double CEROC,double FASEC,double Kcp)
{
   char str[25];
   int sig = 6; // significant digits
   int longitud;

HDC hDC= GetDC(HWindow);   // establecer el contexto de visualización

TextOut(hDC,0,0,"                    RESULTADOS DEL ANALISIS TEORICO",50);
Mp = Mp*100;
TextOut(hDC,0,20,"Sobretiro porcentual.............:",34);
gcvt(Mp, sig, str);
longitud=strlen(str);
TextOut(hDC,190,20,str,longitud);
TextOut(hDC,250,20,"%",1);

TextOut(hDC,0,40,"Tiempo de establecimiento..:",28);
gcvt(ts, sig, str);
longitud=strlen(str);
TextOut(hDC,190,40,str,longitud);
TextOut(hDC,250,40,"segundos",8);

TextOut(hDC,30,100,"Polos deseados en..............:",33);
gcvt(SG, sig, str);
longitud=strlen(str);
TextOut(hDC,220,100,str,longitud);
TextOut(hDC,280,100,"+/-j",4);
gcvt(JW, sig, str);
longitud=strlen(str);
TextOut(hDC,300,100,str,longitud);

TextOut(hDC,30,120,"Fase compensada..............:",32);
gcvt((FASEC*180/PI), sig, str);
longitud=strlen(str);
TextOut(hDC,220,120,str,longitud);
```

```
TextOut(hDC,280,120,"grados",6);

TextOut(hDC,30,140,"Polo del compensador en.....:",29);
gcvt(POLOC, sig, str);
longitud=strlen(str);
TextOut(hDC,220,140,str,longitud);

TextOut(hDC,30,160,"Cero del compensador en.....:",29);
gcvt(CEROC, sig, str);
longitud=strlen(str);
TextOut(hDC,220,160,str,longitud);

TextOut(hDC,30,180,"Ganancia del compensador..:",27);
gcvt(Kcp, sig, str);
longitud=strlen(str);
TextOut(hDC,220,180,str,longitud);

ReleaseDC(HWindow,hDC);  // liberar el contexto de visualización
}

void   TTransferWindow::CeroArb(double  SG,double   JW,  double   FASEC,double
CEROC,double &POLOC)
{
double Den;
      if((SG - CEROC )> 0)
           Den = tan(atan(fabs(JW) / (SG - CEROC)) - FASEC);
      else if((SG - CEROC) == 0)
           Den = tan((PI / 2) - FASEC);
        else
           Den = tan(PI - atan(fabs(JW) / (SG - CEROC)) - FASEC);
        //Encuentro la posicion del polo del compensador en el plano "s"
      POLOC = SG - (fabs(JW) / Den);
}

void   TTransferWindow::MultPoli(double   GP,double    GQ,double   Po[],double
Qo[],double PROD[])
{
int i, j;
double AUX[8]; //ojo  GP+GQ+1=5

for(i = 0;i<=(GP + GQ);i++)
PROD[i] = 0;

   for( j = 0;j<= GP;j++)
    {
        for (i = j;i<= (j + GQ);i++)
            AUX[i] = Po[j] * Qo[i - j];

        for (i = j;i<=(j + GQ);i++)
          PROD[i] = PROD[i] + AUX[i];
    }
}

void      TTransferWindow::SumPoli(double     GRADO,double      PROD1[],double
PROD2[],double PLC[])
{
```

```
int i;
        for(i = 0;i<= GRADO;i++)
          PLC[i] = PROD1[i] + PROD2[i];
}

void      TTransferWindow::RootPoli(double     GRADO,double      POLI[],double
PtRe[],double PtIm[])
{
double A1[8], B1[8], c1[7]; // A1(GT), B1(GT), c1(GT - 1)
double GT,Q1,TEST;
int H,K, L, i;
double U1,V1;
double R1I,R2R,R1R,R2I;

      GT = GRADO;
      L = 1;
      U1 = -2;
      V1 = 1;
      for(i = 0;i<= GT;i++)
      A1[i] = POLI[GT - i];

E500:       U1 = U1 + 1;
            V1 = V1 - 1;
E700:       if(GT <= 2)
            goto E100;
            B1[0] = A1[0];
            c1[0] = A1[0];
            K = 0;
E600:       B1[1] = A1[1] + U1 * B1[0];
            c1[1] = B1[1] + U1 * c1[0];
            for(i= 2;i<=(GT - 1);i++)
        {
            B1[i] = A1[i] + U1 * B1[i - 1] + V1 * B1[i - 2];
            c1[i] = B1[i] + U1 * c1[i - 1] + V1 * c1[i - 2];
      }
      B1[GT] = A1[GT] + U1 * B1[GT - 1] + V1 * B1[GT - 2];
      TEST = c1[GT - 2]*c1[GT - 2] - c1[GT - 3] * c1[GT - 1];
      if(TEST ==0)
      goto E200;
      p1 = U1;
      Q1 = V1;
      U1 = U1 - (B1[GT - 1] * c1[GT - 2] - B1[GT] * c1[GT - 3]) / TEST;
      V1 = V1 - (B1[GT] * c1[GT - 2] - B1[GT - 1] * c1[GT - 1]) / TEST;
      if(fabs(U1 - p1) <= .0001)
      goto E300;
      else
      goto E1200;
E300:       if(fabs(V1 - Q1) <= .0001)
            goto E400;
E1200:      if(K ==30)
            goto E500;
E200:       K = K + 1;
            goto E600;
E400:       RUTINA(U1,V1,R1R,R2R,R1I,R2I);
      PtRe[L] = R1R;
      PtIm[L] = R1I;
```

```
      PtRe[L + 1] = R2R;
      PtIm[L + 1] = R2I;
      L = L + 2;
      GT = GT - 2;
        for(H = 0;H<=GT;H++)
        A1[H] = B1[H];
      goto E700;
E100:       if(GT== 1)
            {
            PtRe[L] = -A1[1] / A1[0];
            PtIm[L] = 0;
            goto E1100;
            }
      U1 = -A1[1] / A1[0];
      V1 = -A1[2] / A1[0];
      RUTINA(U1,V1,R1R,R2R,R1I,R2I);
      PtRe[L] = R1R;
      PtIm[L] = R1I;
      PtRe[L + 1] = R2R;
      PtIm[L + 1] = R2I;
      L = L + 2;
      goto E1100;
E1100:
}

void TTransferWindow::RUTINA(double U1R,double V1R,double  &R1RR,double &R2RR,
double &R1IR, double &R2IR)
{
double D1;
      D1 = U1R*U1R + 4 * V1R;
      if(D1 >= 0)
      goto E800;
      goto E900;
E800:       R1RR = (U1R + sqrt(D1)) / 2;
            R1IR = 0;
            R2RR = (U1R - sqrt(D1)) / 2;
            R2IR = 0;
      goto E1000;
E900:       R1RR = U1R / 2;
            R2RR = R1RR;
      R1IR = sqrt(fabs(D1)) / 2;
      R2IR = -R1IR;
E1000:
}

void  TTransferWindow::CoeFrac  (double NUMZ,double  NUMP,  double ZR[],double
ZI[],double XR[], double XI[], double COR[], double COI[])
{
int i,j;
double RE1C,RE2C,IM1C,IM2C,RELP,IMGP,NUR,NUI,CUOI;
double DER,DEI,CUOR;
for(i = 1;i<=NUMP;i++)
 {
        RE1C = XR[i] - ZR[1];
      IM1C = XI[i] - ZI[1];
```

```
      if(NUMZ <= 1)
      goto UnCero;

      for(j = 2;j<=NUMZ;j++)
         {
          RE2C = XR[i] - ZR[j];
          IM2C = XI[i] - ZI[j];

            RELP = RE1C * RE2C - IM1C * IM2C;
            IMGP = RE1C * IM2C + RE2C * IM1C;

          RE1C = RELP;
          IM1C = IMGP;
         }

      NUR = RE1C;
      NUI = IM1C;

        goto Cont;
UnCero: if(NUMZ ==0)
      {
      NUR = 1;
      NUI = 0;
      goto Cont;
      }
      NUR = XR[i] - ZR[1];
      NUI = XI[i];

Cont:   for(j = 1;j<=NUMP;j++)
       {
          DER = XR[i] - XR[j];
            DEI = XI[i] - XI[j];
          if(DER ==0 && DEI ==0)
          goto CORTE;

                CUOR = (NUR * DER + NUI * DEI) / (DER*DER + DEI*DEI);
                CUOI = (NUI * DER - NUR * DEI) / (DER*DER + DEI*DEI);

          NUR = CUOR;
          NUI = CUOI;
CORTE:   }

        COR[i] = CUOR;
        COI[i] = CUOI;

 }
goto Term;
Term:
}

void  TTransferWindow::VecTiempo  (int  ESTAB,double  Cond,double  Psim,double
Tsim,double GRADO,double  COR[],double COI[],double PTR[],double  PTI[],double
Tempo[],double &NUMINT)   // ojo Psim por Pim
{
int i,j,FLAGG5=0;
double te;
```

```
NUMINT = 0;

recalculo:

te = 0;
i = 0;

do{
    Tempo[i] = 0;
    j = 1;
    do{
        if(PTI[j]== 0)
       {
          Tempo[i] = Tempo[i] + (COR[j] * exp(PTR[j] * te));
          j = j + 1;
         }
      else
       {
            Tempo[i] = Tempo[i]  + exp(PTR[j] *  te) * COR[j]  * cos(PTI[j]  *
te) + exp(PTR[j + 1] * te) * COR[j + 1] * cos(PTI[j + 1] * te);
          Tempo[i] = Tempo[i] - exp(PTR[j] * te)  * COI[j] * sin(PTI[j] *  te)
- exp(PTR[j + 1] * te) * COI[j + 1] * sin(PTI[j + 1] * te);
            j = j + 2;
         }

      } while (j <= GRADO + 2);

   if( ESTAB == 1)
    {
          if(FLAGG5 ==0 && (fabs(Tempo[i]) > Cond))
             {
             Psim = te / 100;
             Tsim = te;
             te = 0;
             FLAGG5 = 1;
               goto recalculo;
             }
    }
       te = te + Psim;

      i = i + 1;
  }while (te < Tsim);

NUMINT = i;

}

//   Funcion para realizar calculos de Realimentaci≤n de Estado

void TTransferWindow::RealimEstad(void)
{
int i,j,k,PruebaM;
int M[2]={0,0};
int Flag[3]={0,0,0};

  char stringo[10],*strptro;
```

```
if(((RESP2_C==TRUE)|(RESP2_R==TRUE))==0)
 {
  MessageBox(HWindow,
  "Falta ingresar datos",
  "Advertencia",
  MB_ICONEXCLAMATION | MB_OK);
  return;
 }

p1=0;
p2=0;
p3=0;

if(Pimag==0)
 {
    for(i=0;i<2;i++)
     {
       for(j=i+1;j<3;j++)
      {
          if(Flag[i]==0)
           {
            if(Pr[i]==Pr[j])
           {
            Flag[j]=1;
            M[i]=M[i]+1;
             }
           }
      }
     }

  PruebaM=10*M[0]+M[1];
  strptro=itoa(PruebaM, stringo, 10);
  MessageBox(HWindow, strptro, "PruebaM", MB_OK);

  switch(PruebaM)
   {
    case 0:   // 3 reales diferentes
     p1=Pr[0];
     p2=Pr[1];
     p3=Pr[2];
     PolosRealesDif();
     break;

    case 10:   // 2 reales iguales y un real
     p1=Pr[0];
      for(k=1;k<3;k++)
       {
        if(Pr[k]==Pr[0])
         k=k;
        else
        p2=Pr[k];
       }
     DosPRIyUnPRD();
     break;

    case 1:
```

```
      p1=Pr[1];
      p2=Pr[0];
      DosPRIyUnPRD();
      break;

     case 20:   // 3 reales iguales
      p1=Pr[0];
      TresPRI();
      break;

     default:
     exit(1);
    }
 }

if(fabs(Pimag)>0) // un complejo conjugado y un real
 {
  p1=Pr[0];
  p2=Pimag;
  p3=Pr[2];
  UnrealyUnComp();
 }

Calculos=TRUE;
}

// Cuando los Polos son reales y diferentes

void TTransferWindow::PolosRealesDif(void)
{
double a0,a1,a2,t;
double a_es,b_es,c_es,d_es,a_de,b_de,c_de;
double Tesp,Tvel,Terr,K;
int i;

// los polos deben ser negativos
a0=-p1*p2*p3;
a1=p1*p2+p2*p3+p1*p3;
a2=-(p1+p2+p3);

K=a0/629.64;  //bo = 47.7

k1=(a0+47322)/(K*629.64);
k2=(a1+449.6)/(K*629.64);
k3=(a2-104.6)/(K*629.64);

ac[0][0]=1;
ac[0][1]=1;
ac[0][2]=1;
ac[0][3]=1;
ac[1][0]=-(p1+p2+p3);
ac[1][1]=-(p2+p3);
ac[1][2]=-(p1+p3);
ac[1][3]=-(p1+p2);
ac[2][0]=p1*p2+p2*p3+p1*p3;
ac[2][1]=p2*p3;
```

```
ac[2][2]=p1*p3;
ac[2][3]=p1*p2;
ac[3][0]=-p1*p2*p3;
ac[3][1]=0;
ac[3][2]=0;
ac[3][3]=0;

//  Para el espacio
b[3]=629.64*Ampl*K;
b[2]=Pos0*(-449.6+629.64*k2*K)+VEL0*(104.6+629.64*k3*K)+Ace0;
b[1]=Pos0*(104.6+629.64*k3*K)+VEL0;
b[0]=Pos0;

nEc=4;
SisEcuaciones();
a_es=x[0];
b_es=x[1];
c_es=x[2];
d_es=x[3];

//  Para la velocidad
ac[0][0]=1;
ac[0][1]=1;
ac[0][2]=1;
ac[1][0]=-(p2+p3);
ac[1][1]=-(p1+p3);
ac[1][2]=-(p1+p2);
ac[2][0]=p2*p3;
ac[2][1]=p1*p3;
ac[2][2]=p1*p2;

b[2]=629.64*Ampl*K-Pos0*(-47322+629.64*k1*K);
b[1]=VEL0*(104.6+629.64*k3*K)+Ace0;
b[0]=VEL0;

nEc=3;
SisEcuaciones();
a_de=x[0];
b_de=x[1];
c_de=x[2];

t=0;
Psim=0.05;
MaxEsp=0.0;    // inicializar a cero para que no se haga
MaxVel=0.0;
MaxErr=0;
NUMINT=200;      // hallar maximo valor de simulacion

  for(i=0;i<=349;i++)
  {
    TMT[i]=a_es + b_es*exp(p1*t) + c_es*exp(p2*t) + d_es*exp(p3*t);
    Tmp2[i]=a_de*exp(p1*t) + b_de*exp(p2*t) + c_de*exp(p3*t);
    Tmp1[i] = Ampl - 13.2 * TMT[i];

    t=t+Psim;
    Tesp=fabs(TMT[i]);
```

```
    if (MaxEsp<Tesp)
    MaxEsp=Tesp;        // obtener el maximo valor del espacio

    Tvel=fabs(Tmp2[i]);
    if (MaxVel<Tvel)
    MaxVel=Tvel;        // obtener el maximo valor de la velocidad

    Terr=fabs(Tmp1[i]);
    if(MaxErr<Terr)
    MaxErr=Terr;
  }
}

//  Cuando los Polos son 2 iguales y uno diferente
void TTransferWindow::DosPRIyUnPRD(void)
{
double a0,a1,a2,t;
double a_es,b_es,c_es,d_es,a_de,b_de,c_de;
double Tesp,Tvel,Terr,K;
int i;

// los polos deben ser negativos
a0=-p1*p1*p2;
a1=p1*p1+2*p1*p2;
a2=-2*p1-p2;

K=a0/629.64;

k1=(a0+47322)/(K*629.64);
k2=(a1+449.6)/(K*629.64);
k3=(a2-104.6)/(K*629.64);

ac[0][0]=1;
ac[0][1]=1;
ac[0][2]=0;
ac[0][3]=1;
ac[1][0]=-2*p1-p2;
ac[1][1]=-p1-p2;
ac[1][2]=1;
ac[1][3]=-2*p1;
ac[2][0]=p1*p1+2*p1*p2;
ac[2][1]=p1*p2;
ac[2][2]=-p2;
ac[2][3]=p1*p1;
ac[3][0]=-p1*p1*p2;
ac[3][1]=0;
ac[3][2]=0;
ac[3][3]=0;

//  Para el espacio
b[3]=629.64*Ampl*K;
b[2]=Pos0*(-449.6+629.64*k2*K)+VEL0*(104.6+629.64*k3*K)+Ace0;
b[1]=Pos0*(104.6+629.64*k3*K)+VEL0;
b[0]=Pos0;

nEc=4;
```

```
SisEcuaciones();
a_es=x[0];
b_es=x[1];
c_es=x[2];
d_es=x[3];

//  Para la velocidad
ac[0][0]=1;
ac[0][1]=0;
ac[0][2]=1;
ac[1][0]=-p1-p2;
ac[1][1]=1;
ac[1][2]=-2*p1;
ac[2][0]=p1*p2;
ac[2][1]=-p2;
ac[2][2]=p1*p1;

b[2]=629.64*Ampl*K-Pos0*(-47322+629.64*k1*K);
b[1]=VEL0*(104.6+629.64*k3*K);
b[0]=VEL0;

nEc=3;
SisEcuaciones();
a_de=x[0];
b_de=x[1];
c_de=x[2];

t=0;
Psim=0.05;
MaxEsp=0.0;    // inicializar a cero para que no se haga
MaxVel=0.0;
MaxErr=0;
NUMINT=200;      // hallar maximo valor de simulacion

  for(i=0;i<=349;i++)
  {
    TMT[i]=a_es + (b_es + c_es*t)*exp(p1*t) + d_es*exp(p2*t);
    Tmp2[i]=(a_de + b_de*t)*exp(p1*t) + c_de*exp(p2*t);
    Tmp1[i] = Ampl - 13.2 * TMT[i];

    t=t+Psim;
    Tesp=fabs(TMT[i]);
    if (MaxEsp<Tesp)
    MaxEsp=Tesp;         // obtener el maximo valor del espacio

    Tvel=fabs(Tmp2[i]);
    if (MaxVel<Tvel)
    MaxVel=Tvel;         // obtener el maximo valor de la velocidad

    Terr=fabs(Tmp1[i]);
    if(MaxErr<Terr)
    MaxErr=Terr;
  }
}

//  Cuando los Polos son iguales
```

```
void TTransferWindow::TresPRI(void)
{
double a0,a1,a2,t;
double a_es,b_es,c_es,d_es,a_de,b_de,c_de;
double Tesp,Tvel,Terr,K;
int i;

// los polos deben ser negativos

a0=-p1*p1*p1;
a1=3*p1*p1;
a2=-3*p1;

K=a0/629.64;

k1=(a0+47322)/(K*629.64);
k2=(a1+449.6)/(K*629.64);
k3=(a2-104.6)/(K*629.64);

ac[0][0]=1;
ac[0][1]=1;
ac[0][2]=0;
ac[0][3]=0;
ac[1][0]=-3*p1;
ac[1][1]=-2*p1;
ac[1][2]=1;
ac[1][3]=0;
ac[2][0]=3*p1*p1;
ac[2][1]=p1*p1;
ac[2][2]=-p1;
ac[2][3]=1;
ac[3][0]=-p1*p1*p1;
ac[3][1]=0;
ac[3][2]=0;
ac[3][3]=0;

//  Para el espacio
b[3]=629.64*Ampl*K;
b[2]=Pos0*(-449.6+629.64*k2*K)+VEL0*(104.6+629.64*k3*K)+Ace0;
b[1]=Pos0*(104.6+629.64*k3*K)+VEL0;
b[0]=Pos0;

nEc=4;
SisEcuaciones();
a_es=x[0];
b_es=x[1];
c_es=x[2];
d_es=x[3];

//  Para la velocidad
ac[0][0]=1;
ac[0][1]=0;
ac[0][2]=0;
ac[1][0]=-2*p1;
ac[1][1]=1;
ac[1][2]=0;
```

```
ac[2][0]=p1*p1;
ac[2][1]=-p1;
ac[2][2]=1;

b[2]=629.64*Ampl*K-Pos0*(-47322+629.64*k1*K);
b[1]=VEL0*(104.6+629.64*k3*K)+Ace0;
b[0]=VEL0;

nEc=3;
SisEcuaciones();
a_de=x[0];
b_de=x[1];
c_de=x[2];

t=0;
Psim=0.05;
MaxEsp=0.0;    // inicializar a cero para que no se haga
MaxVel=0.0;
MaxErr=0;
NUMINT=200;      // hallar maximo valor de simulacion

  for(i=0;i<=349;i++)
  {
    TMT[i]=a_es + (b_es + c_es*t + d_es*t*t/2)*exp(p1*t);
    Tmp2[i]=(a_de + b_de*t + c_de*t*t/2)*exp(p1*t);
    Tmp1[i] = Ampl - 13.2 * TMT[i];

    t=t+Psim;
    Tesp=fabs(TMT[i]);
    if (MaxEsp<Tesp)
    MaxEsp=Tesp;         // obtener el maximo valor del espacio

    Tvel=fabs(Tmp2[i]);
    if (MaxVel<Tvel)
    MaxVel=Tvel;         // obtener el maximo valor de la velocidad

    Terr=fabs(Tmp1[i]);
    if(MaxErr<Terr)
    MaxErr=Terr;
  }
}

//  Cuando hay un polo real y un par de polos complejos conjugados
void TTransferWindow::UnrealyUnComp(void)
{
double a0,a1,a2,t;
double a_es,b_es,c_es,d_es,a_de,b_de,c_de;
double Tesp,Tvel,Terr,K;
int i;

// los polos deben ser negativos
a0=-p1*p1*p3-p2*p2*p3;
a1=p1*p1+p2*p2+2*p1*p3;
a2=-2*p1-p3;

K=a0/629.64;
```

```
k1=(a0+47322)/(K*629.64);
k2=(a1+449.6)/(K*629.64);
k3=(a2-104.6)/(K*629.64);

ac[0][0]=1;
ac[0][1]=1;
ac[0][2]=0;
ac[0][3]=1;
ac[1][0]=-2*p1-p3;
ac[1][1]=-p1-p3;
ac[1][2]=p2;
ac[1][3]=-2*p1;
ac[2][0]=p1*p1+p2*p2+2*p1*p3;
ac[2][1]=p1*p3;
ac[2][2]=-p2*p3;
ac[2][3]=p1*p1+p2*p2;
ac[3][0]=-p1*p1*p3-p2*p2*p3;
ac[3][1]=0;
ac[3][2]=0;
ac[3][3]=0;

//  Para el espacio
b[3]=629.64*Ampl*K;
b[2]=Pos0*(-449.6+629.64*k2*K)+VEL0*(104.6+629.64*k3*K)+Ace0;
b[1]=Pos0*(104.6+629.64*k3*K)+VEL0;
b[0]=Pos0;

nEc=4;
SisEcuaciones();
a_es=x[0];
b_es=x[1];
c_es=x[2];
d_es=x[3];

//  Para la velocidad
ac[0][0]=1;
ac[0][1]=0;
ac[0][2]=1;
ac[1][0]=-p1-p3;
ac[1][1]=p2;
ac[1][2]=-2*p1;
ac[2][0]=p1*p3;
ac[2][1]=-p2*p3;
ac[2][2]=p1*p1+p2*p2;

b[2]=629.64*Ampl*K-Pos0*(-47322+629.64*k1*K);
b[1]=VEL0*(104.6+629.64*k3*K);
b[0]=VEL0;

nEc=3;
SisEcuaciones();
a_de=x[0];
b_de=x[1];
c_de=x[2];

t=0;
```

```
Psim=0.05;
MaxEsp=0.0;    // inicializar a cero para que no se haga
MaxVel=0.0;
MaxErr=0;
NUMINT=200;      // hallar maximo valor de simulacion

  for(i=0;i<=349;i++)
  {

    TMT[i]=a_es    +    (b_es*cos(p2*t)    +    c_es*sin(p2*t))*exp(p1*t)    +
d_es*exp(p3*t);
    Tmp2[i]=(a_de*cos(p2*t) + b_de*sin(p2*t))*exp(p1*t) + c_de*exp(p3*t);
    Tmp1[i] = Ampl - 13.2 * TMT[i];

    t=t+Psim;
    Tesp=fabs(TMT[i]);
    if (MaxEsp<Tesp)
    MaxEsp=Tesp;        // obtener el maximo valor del espacio

    Tvel=fabs(Tmp2[i]);
    if (MaxVel<Tvel)
    MaxVel=Tvel;        // obtener el maximo valor de la velocidad

    Terr=fabs(Tmp1[i]);
    if(MaxErr<Terr)
    MaxErr=Terr;
  }
}

//   Funcion para resolver un sistema de 4 ecuaciones con 4 incognitas
//   eliminacion gausiana con pivoteo
//   Resolucion de sistema de ecuaciones complejo de 4 x 4
void TTransferWindow::SisEcuaciones(void)
{
        int n;
      int i,j,k,p;
      double m[4][4],a[4][5];
      int nrow[4],ncopy;
      double suma;
      double maximo, mimo;
//    float mimi;
        n=nEc;

      for(i=0;i<4;i++)
      {
        for(j=0;j<4;j++)
        a[i][j]=ac[i][j];
      }

//    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;
              if (a[nrow[i]][i]>0)
              maximo = a[nrow[i]][i] ;
              else
              maximo = -a[nrow[i]][i] ;
              for(j=i+1;j<n;j++){

                 if (a[nrow[j]][i]>0)
                 mimo = a[nrow[i]][i] ;
                 else
                 mimo = -a[nrow[i]][i] ;

                    if (maximo<mimo){
                                maximo=mimo;
                                p = j;
                    }
              }

//              if (a[nrow[p]][i]>0)
//              mimi = a[nrow[i]][i] ;
//              else
//              mimi = -a[nrow[i]][i] ;
//              if (mimi==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];
              }
        }

//      if ((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 del sistema de ecuaciones\n";
//    for(i=0;i<n;i++)
//          cout << "x [" <<i <<"] = " << x[i] << "\n";
}

//   Funcion para crear los Mapas de Bits para la Animacion
//
void TTransferWindow::ConstrCeldas(void)
{

  HDC hDisplayDC;                          // contexto de visualizaci≤n
  HBRUSH hBrochaRoja, hPrevBrush2;              // brochas para la esfera
  HPEN hPlumaRoja, hPrevPen2;                    // plumas para la esfera

UbicCeldaX = (int)(0.45*anchoVentx);
UbicCeldaY = (int)(0.3125*altoVenty);
AnchoCelda = (int)(0.1*anchoVentx);
AlturaCelda = (int)(0.5625*altoVenty);

OrigenCeldaY=(int)(0.359*AlturaCelda);

AnchoMapa = AnchoCelda;
AlturaMapa = (int)(0.14489*altoVenty);

CoorEsqDerX = AnchoMapa-1;
CoorEsqDerY = AlturaMapa-1;

EsqIzqBolaX = 0;
EsqDerBolaX = CoorEsqDerX;
EsqIzqBolaY = 0;
EsqDerBolaY = CoorEsqDerY;

//  configurar el contexto de visualizaci≤n
hDisplayDC= GetDC(HWindow);   // establecer el contexto de visualizaci≤n

//  crear y almacenar el bitmap de la página oculta
hPaginaOcultaDC= CreateCompatibleDC(hDisplayDC);       // DC en memoria
hPaginaOculta= CreateCompatibleBitmap(hDisplayDC,AnchoCelda,
  AlturaCelda);      // creaci≤n del mapa de bits de la pßgina oculta
hPrevBitmap2=  (HBITMAP)SelectObject(hPaginaOcultaDC, //  seleccionar mapa  de
bits
  hPaginaOculta);

PatBlt(hPaginaOcultaDC,0,0,AnchoCelda,AlturaCelda,
  WHITENESS);                     // dejar la pßgina oculta en blanco

//  crear y almacenar el mapa de bits de una esfera
hMapaDC= CreateCompatibleDC(hDisplayDC);           // DC en memoria
hMapa= CreateCompatibleBitmap(hDisplayDC,
  AnchoMapa,AlturaMapa);                  // crear el bitmap del actor
hBrochaRoja= CreateSolidBrush(RGB(0,0,0));            // crear brocha
hPlumaRoja= CreatePen(PS_SOLID,1,RGB(0,0,0));        // crear pluma
hPrevBitmap1= (HBITMAP)SelectObject(hMapaDC,hMapa); // selec. mapa de bits
```

```
PatBlt(hMapaDC,0,0,
  AnchoMapa,AlturaMapa,WHITENESS);        // blanquear el mapa de bits

hPrevBrush2= (HBRUSH)SelectObject(hMapaDC,hBrochaRoja); // seleccionar brocha
hPrevPen2= (HPEN)SelectObject(hMapaDC,hPlumaRoja);      // seleccionar pluma
Ellipse(hMapaDC,EsqIzqBolaX,EsqIzqBolaY,EsqDerBolaX,EsqDerBolaY);  // BOLA

//  limpiar el entorno gráfico
SelectObject(hMapaDC,hPrevBrush2);          // deseleccionar brocha
DeleteObject(hBrochaRoja);                              // borrar brocha
SelectObject(hMapaDC,hPrevPen2);            // deseleccionar pluma
DeleteObject(hPlumaRoja);                               // borrar pluma


//  liberar el contexto de visualización
ReleaseDC(HWindow,hDisplayDC);   // liberar el contexto de visualización
bCeldasListas = TRUE;            // establecer un token
}

//   Bucle para secuencia de animaci≤n rßpida
//
 void TTransferWindow::Animacion(void)
{
  HDC hDC;                          // contexto de visualización
  int i,j,k;
char str[25];
int sig = 6; // significant digits
int longitud;

if (bCeldasListas == FALSE) return;  // cancelar si celdas no preparadas
hDC= GetDC(HWindow);          // establecer el contexto de visualización
SetCapture(HWindow);                                    // bloqueo del ratón
hPrevCursor= SetCursor(hHourGlass);   // cambiar al cursor de espera
DeltaY=-(int)(TMT[0]*50);

// Dibujo del electroiman

MoveTo(hDC,(UbicCeldaX+AnchoCelda),UbicCeldaY-1);
LineTo(hDC,(UbicCeldaX+AnchoCelda),(UbicCeldaY+AlturaCelda));
LineTo(hDC,UbicCeldaX-1,(UbicCeldaY+AlturaCelda));
LineTo(hDC,UbicCeldaX-1,UbicCeldaY-1);
LineTo(hDC,(UbicCeldaX+AnchoCelda),UbicCeldaY-1);


for (i= 0; i < NUMINT; i++)
{                                 // comienzo del bucle de animaci≤n

//  dibujar la vi±eta actual

PatBlt(hPaginaOcultaDC,0,0,AnchoCelda,AlturaCelda,
  WHITENESS);                     // dejar en blanco la pßgina oculta
BitBlt(hPaginaOcultaDC,OrigenCeldaX,(OrigenCeldaY+DeltaY),
  AnchoMapa,AlturaMapa, hMapaDC,0,0,SRCAND);
// copiar la bola en la pßgina oculta
BitBlt(hDC,UbicCeldaX,UbicCeldaY,AnchoCelda,AlturaCelda,
  hPaginaOcultaDC,0,0,SRCCOPY); // copiar la pßgina oculta a la ventana
```

```
      gcvt(TMT[i], sig, str);
      longitud=strlen(str);
      TextOut(hDC,0,0,"                              ",25);
      TextOut(hDC,0,0,str,longitud);

  for(k=0;k<=30000;k++)
  for(j=0;j<=10;j++);

DeltaY = -(int)(TMT[i]*50);

//choque superior
if(TMT[i] >=1)
{
PatBlt(hPaginaOcultaDC,0,0,AnchoCelda,AlturaCelda,
  WHITENESS);                    // dejar en blanco la pßgina oculta
BitBlt(hPaginaOcultaDC,OrigenCeldaX,0,
  AnchoMapa,AlturaMapa, hMapaDC,0,0,SRCAND);
  // copiar la bola en la pßgina oculta
BitBlt(hDC,UbicCeldaX,UbicCeldaY,AnchoCelda,AlturaCelda,
  hPaginaOcultaDC,0,0,SRCCOPY); // copiar la pßgina oculta a la ventana

i = NUMINT;
}

// choque inferior
if(TMT[i] <= -1)
{
PatBlt(hPaginaOcultaDC,0,0,AnchoCelda,AlturaCelda,
  WHITENESS);                    // dejar en blanco la pßgina oculta
BitBlt(hPaginaOcultaDC,OrigenCeldaX,(AlturaCelda-AlturaMapa),
  AnchoMapa,AlturaMapa, hMapaDC,0,0,SRCAND);
  // copiar la bola en la pßgina oculta
BitBlt(hDC,UbicCeldaX,UbicCeldaY,AnchoCelda,AlturaCelda,
  hPaginaOcultaDC,0,0,SRCCOPY); // copiar la pßgina oculta a la ventana

i = NUMINT;
}

} // volver al principio del bucle y dibujar la siguiente vi±eta...

SetCursor(hPrevCursor);          // restablecer el cursor por defecto
ReleaseCapture();                              // desbloquear el rat≤n

ReleaseDC(HWindow,hDC);          // liberar el contexto de visualizaci≤n
}

void TTransferWindow::LimpiarPantalla(void)
{
InvalidateRect(HWindow,NULL,TRUE);
UpdateWindow(HWindow);
}

void TTransferWindow::GraficaEjes(void)
{
  HDC hDC;                       // contexto de visualización
  HPEN hNegro, hPrev1;           // pluma para los ejes
```

```
  float idivX, idivY,inTiemX;
  int i,j,sig=6;
  char str[25];

pxmax=anchoVentx-20;   // maximo valor de puntos en el eje x

 if(NUMINT>=pxmax)
  incX=NUMINT/pxmax;       // valor de escalamiento del tiempo
 else
  incX=1;

 if(NUMINT<pxmax)
  jncX=pxmax/NUMINT;
 else
  jncX=1;

hDC= GetDC(HWindow);       // establecer el contexto de visualización

// dibujo de los ejes
hNegro=CreatePen(0,1,0x00000000);
hPrev1=(HPEN)SelectObject(hDC,hNegro);
MoveTo(hDC,10,UbicEjex);
LineTo(hDC,(anchoVentx-10),UbicEjex);
MoveTo(hDC,10,40);
LineTo(hDC,10,(altoVenty-10));
SelectObject(hDC,hPrev1);
DeleteObject(hNegro);

  for(j=jncX;j<pxmax;j=j+jncX);

  j=j-jncX;
  idivX=(float)j/10.0;
  idivY=(float)(altoVenty-50)/20.0;

  for(i=1;i<11;i++)
  {
   j=(int)(10+idivX*i);
   MoveTo(hDC,j,(UbicEjex-1));
   LineTo(hDC,j,(UbicEjex+2));
  }

  for(i=0;i<21;i++)
  {
   j=(int)(40+idivY*i);
   if(i==10)
   {
    MoveTo(hDC,8,UbicEjex);
    LineTo(hDC,12,UbicEjex);
   }
   else
   {
    MoveTo(hDC,8,j);
    LineTo(hDC,12,j);
   }
  }
```

```
ReleaseDC(HWindow,hDC);   /* liberar el contexto de visualización */

inTiemX= Psim*idivX*incX/jncX;
gcvt(inTiemX,sig,str);

strcpy(incTiemp, "(");
strcat(incTiemp, str);
strcat(incTiemp, " seg/div)");

}
//
//   Funcion para graficar el Espacio versus el Tiempo
//
void TTransferWindow::GrafEspacio(void)
{
  HDC hDC;                          // contexto de visualización
  HPEN hRojo, hPrev2;                    // pluma para la curva
  int ic, j, longi,sig=6;
  double y, escY,inEspaY;
  char str[25];

inEspaY=MaxEsp/10;
gcvt(inEspaY,sig,str);
strcpy(incEspac, "(");
strcat(incEspac, str);
strcat(incEspac, " cm/div)");

escY=(altoVenty-50)*0.5/MaxEsp; // factor de escala en el eje y
hDC= GetDC(HWindow);      // establecer el contexto de visualización

//poner algun texto
TextOut(hDC,0,20,"x=                              s",27);
TextOut(hDC,300,20,"y=                                 cm",29);
SetTextColor(hDC, RGB(255,0,0));
longi=strlen(incTiemp);
TextOut(hDC,140,20,incTiemp,longi);
longi=strlen(incEspac);
TextOut(hDC,440,20,incEspac,longi);

TextOut(hDC,0,0,"espacio vs t",12);

// dibujar la figura
hRojo=CreatePen(0,1,RGB(255,0,0));
hPrev2=(HPEN)SelectObject(hDC,hRojo);
j=1;
  y=(double)UbicEjex-TMT[0]*escY;
  MoveTo(hDC,10,(int)y);
  for(ic=incX,j=jncX;j<pxmax;ic=ic+incX,j=j+jncX)
  {
    y=(double)UbicEjex-TMT[ic]*escY;
    LineTo(hDC,(10+j),(int)y);
  }
SelectObject(hDC,hPrev2);
DeleteObject(hRojo);

ReleaseDC(HWindow,hDC);   // liberar el contexto de visualización
```

```
}
//
//   Funcion para graficar la Velocidad versus el Tiempo
//
void TTransferWindow::GrafVelocidad(void)
{
  HDC hDC;                          // contexto de visualización
  HPEN hRojo, hPrev2;               // pluma para la curva
  int ic, j, longi,sig=6;
  double y, escY,inVeloY;
  char str[25];

inVeloY=MaxVel/10;
gcvt(inVeloY,sig,str);
strcpy(incVeloc, "(");
strcat(incVeloc, str);
strcat(incVeloc, " (cm/s)/div)");


escY=(altoVenty-50)*0.5/MaxVel;
                  // factor de escala en el eje y
hDC= GetDC(HWindow);      // establecer el contexto de visualización

//poner algun texto
TextOut(hDC,0,20,"x=                              s",27);
TextOut(hDC,300,20,"y=                             cm/s",29);
SetTextColor(hDC, 0x00808080);
longi=strlen(incTiemp);
TextOut(hDC,140,20,incTiemp,longi);
longi=strlen(incVeloc);
TextOut(hDC,440,20,incVeloc,longi);

TextOut(hDC,155,0,"vel vs t",8);

// dibujar la figura
hRojo=CreatePen(0,1,0x00808080);
hPrev2=(HPEN)SelectObject(hDC,hRojo);
j=1;
  y=(double)UbicEjex-Tmp2[0]*escY;
  MoveTo(hDC,10,(int)y);
  for(ic=incX,j=jncX;j<pxmax;ic=ic+incX,j=j+jncX)
  {
    y=(double)UbicEjex-Tmp2[ic]*escY;
    LineTo(hDC,(10+j),(int)y);
  }
SelectObject(hDC,hPrev2);
DeleteObject(hRojo);

ReleaseDC(HWindow,hDC);   // liberar el contexto de visualización
}
//
//   Funcion para graficar el Error de posici≤n versus el Tiempo
//
void TTransferWindow::GrafError(void)
{
  HDC hDC;                          // contexto de visualización
```

```
  HPEN hAzul, hPrev2;              // pluma para la curva
  int ic, j, longi,sig=6;
  double y, escY,inErroY;
  char str[25];

inErroY=MaxErr/10;
gcvt(inErroY,sig,str);
strcpy(incError, "(");
strcat(incError, str);
strcat(incError, " cm/div)");

escY=(altoVenty-50)*0.5/MaxErr;
                  // factor de escala en el eje y
hDC= GetDC(HWindow);      // establecer el contexto de visualización

//poner algun texto
TextOut(hDC,0,20,"x=                              s",27);
TextOut(hDC,300,20,"y=                              mV",28);
SetTextColor(hDC, 0x00FF0000);
longi=strlen(incTiemp);
TextOut(hDC,140,20,incTiemp,longi);
longi=strlen(incError);
TextOut(hDC,440,20,incError,longi);
TextOut(hDC,255,0,"err vs t",8);

// dibujar la figura
hAzul=CreatePen(0,1,0x00FF0000);
hPrev2=(HPEN)SelectObject(hDC,hAzul);

j=1;
  y=(double)UbicEjex-Tmp1[0]*escY;
  MoveTo(hDC,10,(int)y);
  for(ic=incX,j=jncX;j<pxmax;ic=ic+incX,j=j+jncX)
  {
    y=(double)UbicEjex-Tmp1[ic]*escY;
    LineTo(hDC,(10+j),(int)y);
  }
SelectObject(hDC,hPrev2);
DeleteObject(hAzul);

ReleaseDC(HWindow,hDC);   // liberar el contexto de visualización
}
//
//   Funcion para graficar la Velocidad versus el Espacio
//
void TTransferWindow::GrafFase(void)
{
  HDC hDC;                         // contexto de visualización
  HPEN hNegro, hPrev1;             // pluma para los ejes
  HPEN hAzul, hPrev2;              // pluma para la curva
  int ic;
  double x, y, escX, escY;

escX=(anchoVentx-20)*0.5/MaxEsp; // factor de escala en el eje x
escY=(altoVenty-50)*0.5/MaxVel; // factor de escala en el eje y
```

```
hDC= GetDC(HWindow);      // establecer el contexto de visualización

//poner algun texto
TextOut(hDC,0,20,"x=                              cm",27);
TextOut(hDC,300,20,"y=                                cm/s",29);
SetTextColor(hDC, RGB(0,0,255));
TextOut(hDC,0,0,"vel vs. esp",11);

// dibujo de los ejes
hNegro=CreatePen(0,1,0x00000000);
hPrev1=(HPEN)SelectObject(hDC,hNegro);
MoveTo(hDC,10,UbicEjex);
LineTo(hDC,(anchoVentx-10),UbicEjex);
MoveTo(hDC,(anchoVentx/2),40);
LineTo(hDC,(anchoVentx/2),(altoVenty-10));
SelectObject(hDC,hPrev1);
DeleteObject(hNegro);

// dibujar la figura
hAzul=CreatePen(0,1,RGB(0,0,255));
hPrev2=(HPEN)SelectObject(hDC,hAzul);

  y=(double)UbicEjex-Tmp2[0]*escY;
  x=(double)(anchoVentx/2)+TMT[0]*escX;

  MoveTo(hDC,(int)x,(int)y);
  for(ic=1;ic<(NUMINT-3);ic++)
  {
    y=(double)UbicEjex-Tmp2[ic]*escY;
    x=(double)(anchoVentx/2)+TMT[ic]*escX;
    LineTo(hDC,(int)x,(int)y);
  }
SelectObject(hDC,hPrev2);
DeleteObject(hAzul);

ReleaseDC(HWindow,hDC);   // liberar el contexto de visualización
}
//
//    Funcion para obtener las Coordenadas del Espacio
//
void TTransferWindow::CoorEspacio(void)
{
  HDC hDisplayDC;                      // contexto de visualización
  HPEN hBlackPen, hPrevPen;          // plumas
  double xEsp,yEsp;
  char str[25];
  int sig = 6;    // numero de digitos significativos
  int longitud;

hDisplayDC= GetDC(HWindow);   // establecer el contexto de visualización
hBlackPen= CreatePen(PS_SOLID,1,RGB(0,0,0));       // crear pluma
hPrevPen= (HPEN)SelectObject(hDisplayDC,hBlackPen); // seleccionar pluma

xEsp= (coordenadaX-10)*Psim*incX/jncX;
yEsp= (UbicEjex-coordenadaY)*MaxEsp/(UbicEjex-40);
```

```
//coordenadas x and y del cursor

gcvt(xEsp, sig, str);
longitud=strlen(str);
TextOut(hDisplayDC,20,20,"                                    ",22); //borrar el  dato
anterior
TextOut(hDisplayDC,20,20,str,longitud);

gcvt(yEsp, sig, str);
longitud=strlen(str);
TextOut(hDisplayDC,320,20,"                                    ",22); //borrar el dato
anterior
TextOut(hDisplayDC,320,20,str,longitud);


SelectObject(hDisplayDC,hPrevPen);     // deseleccionar pluma
DeleteObject(hBlackPen);                // borrar pluma
ReleaseDC(HWindow,hDisplayDC);  // liberar el contexto de visualización
}
//
//     Funcion para obtener las Coordenadas de la Velocidad
//
void TTransferWindow::CoorVelocid(void)
{
  HDC hDisplayDC;                       // contexto de visualización
  HPEN hBlackPen, hPrevPen;          // plumas
  double xVel,yVel;

  char str[25];
  int sig = 6;    // numero de digitos significativos
  int longitud;

hDisplayDC= GetDC(HWindow);   // establecer el contexto de visualización
hBlackPen= CreatePen(PS_SOLID,1,RGB(0,0,0));       // crear pluma
hPrevPen= (HPEN)SelectObject(hDisplayDC,hBlackPen);   // seleccionar pluma

xVel= (coordenadaX-10)*Psim*incX/jncX;
yVel= (UbicEjex-coordenadaY)*MaxVel/(UbicEjex-40);

//coordenadas x and y del cursor

gcvt(xVel, sig, str);
longitud=strlen(str);
TextOut(hDisplayDC,20,20,"                                    ",22); //borrar el  dato
anterior
TextOut(hDisplayDC,20,20,str,longitud);

gcvt(yVel, sig, str);
longitud=strlen(str);
TextOut(hDisplayDC,320,20,"                                    ",22); //borrar el dato
anterior
TextOut(hDisplayDC,320,20,str,longitud);

SelectObject(hDisplayDC,hPrevPen);       // deseleccionar pluma
DeleteObject(hBlackPen);              // borrar pluma
ReleaseDC(HWindow,hDisplayDC);   // liberar el contexto de visualización
```

```
}
//
//      Funcion para obtener las Coordenadas del Error
//
void TTransferWindow::CoorError(void)
{
  HDC hDisplayDC;                  // contexto de visualización
  HPEN hBlackPen, hPrevPen;            // plumas
  double xErr,yErr;

  char str[25];
  int sig = 6;     // numero de digitos significativos
  int longitud;

hDisplayDC= GetDC(HWindow);  // establecer el contexto de visualización
hBlackPen= CreatePen(PS_SOLID,1,RGB(0,0,0));       //  crear pluma
hPrevPen= (HPEN)SelectObject(hDisplayDC,hBlackPen);  // seleccionar pluma

xErr= (coordenadaX-10)*Psim*incX/jncX;
yErr= (UbicEjex-coordenadaY)*MaxErr/(UbicEjex-40);

//coordenadas x and y del cursor

gcvt(xErr, sig, str);
longitud=strlen(str);
TextOut(hDisplayDC,20,20,"                                    ",22); //borrar el  dato
anterior
TextOut(hDisplayDC,20,20,str,longitud);

gcvt(yErr, sig, str);
longitud=strlen(str);
TextOut(hDisplayDC,320,20,"                                    ",22); //borrar el dato
anterior
TextOut(hDisplayDC,320,20,str,longitud);

SelectObject(hDisplayDC,hPrevPen);      // deseleccionar pluma
DeleteObject(hBlackPen);              // borrar pluma
ReleaseDC(HWindow,hDisplayDC);    // liberar el contexto de visualización
}
//
//      Funcion para obtener las Coordenadas de la Velocidad vs. el Espacio
//
void TTransferWindow::CoorFase(void)
{
  HDC hDisplayDC;                   // contexto de visualización
  HPEN hBlackPen, hPrevPen;           // plumas
  double xEsp,yVel;

  char str[25];
  int sig = 6;     // numero de digitos significativos
  int longitud;

hDisplayDC= GetDC(HWindow);    // establecer el contexto de visualización
hBlackPen= CreatePen(PS_SOLID,1,RGB(0,0,0));            // crear pluma
hPrevPen= (HPEN)SelectObject(hDisplayDC,hBlackPen);    // seleccionar pluma
```

```
xEsp= (coordenadaX-anchoVentx/2)*MaxEsp/(anchoVentx/2-10);
yVel= (UbicEjex-coordenadaY)*MaxVel/(UbicEjex-40);

//coordenadas x and y del cursor

gcvt(xEsp, sig, str);
longitud=strlen(str);
TextOut(hDisplayDC,20,20,"                                    ",22); //borrar el  dato
anterior
TextOut(hDisplayDC,20,20,str,longitud);

gcvt(yVel, sig, str);
longitud=strlen(str);
TextOut(hDisplayDC,320,20,"                                   ",22); //borrar el  dato
anterior
TextOut(hDisplayDC,320,20,str,longitud);

SelectObject(hDisplayDC,hPrevPen);      // deseleccionar pluma
DeleteObject(hBlackPen);              // borrar pluma
ReleaseDC(HWindow,hDisplayDC);    // liberar el contexto de visualización
}
```

VARIABLES.

A continuación un listado de las variables principales que se utilizan en el programa.

Kpl Ganancia de la planta.

Ksn Ganancia del sensor.

Mp máximo sobreimpulso porcentual.

ts tiempo de establecimiento.

CEROC Ubicación del cero del compensador en la red de adelanto de fase.

POLOC Ubicación del polo del compensador en la red de adelanto de fase.

CAM Coeficiente de amortiguamiento de las raíces de lazo cerrado.

Wn Frecuencia natural no amortiguada.

SG Parte real de las raíces deseadas.

JW Parte imaginaria de las raíces deseadas.

FASE Fase de la planta en las raíces deseadas.

FASEC Fase que debe tener el compensador.

Ampl Amplitud de la señal de entrada.

Kp Constante proporcional de la Acción de Control.

Ki Constante proporcional de la Acción de Control.

Kd Constante proporcional de la Acción de Control.

Pos0 Condición inicial de posición.

VEL0 Condición inicial de velocidad.

Ace0 Condición de aceleración.

ZR[i] Parte real de los ceros de la función de transferencia considerada.

ZI[i] Parte imaginaria de los ceros de la función de transferencia considerada.

PLR[i] Parte real de los polos de la función de transferencia considerada.

PLI[i] Parte imaginaria de los ceros de la función de transferencia considerada.

Po[i] Polinomio a multiplicar.

Qo[i] Polinomio a multiplicar.

GP Grado del polinomio a multiplicar.

GQ Grado del polinomio a multiplicar.

PROD1[i] Resultado de la multiplicación de polinomios.

PROD2[i] Resultado de la suma de polinomios.

PartReal[i] Parte real de las raíces del polinomio considerado.

PartImag[i] Parte imaginaria de las raíces del polinomio considerado.

NUMINT Número de instantes tomados encuenta para el vector temporal.

CRX[i] Parte real de los coeficientes resultantes de la expansión en fracciones parciales de la función de transferencia considerada.

CRI[i] Parte imaginaria de los coeficientes resultantes de la expansión en fracciones parciales de la función de transferencia considerada.

MENOR Constante de tiempo en la expresión exponencial para las compensaciones.

Psim Período de simulación.

Tsim Tiempo de simulación.

FLAGGi Banderas utilizadas en la selección de las bifurcaciones.

ESTAB Bandera de estabilidad del sistema.

p1,2,3 Polos de lazo cerrado para realimentación de estado.

k1,2,3 Elementos del vector de realimentación.

a0,1,2 Polos de lazo abierto para realimentación de estado.

ac[i][i] Coeficientes para la obtención de las fracciones parciales en un sistema de ecuaciones, para realimentación de estados.

K Ganancia de la compensación con realimentación de estado.

TMT[i] Vector de espacio para realimentación de estado.

Tmp2[i] Vector de velocidad para realimentación de estado.

Tmp1[i] Vector de error para realimentación de estado.

anchoVentx Ancho de la zona de atracción.

altoVenty Altura de la zona de atracción.

coordenadaX Variable de animación.

coordenadaY Variable de animación.

AnchoCelda Ancho de la celda de animación.

AlturaCelda Altura de la celda de animación.

AnchoMapa Ancho del mapa de bits de la animación.

AlturaMapa Altura del mapa de bits de la animación.

CONDICIONES INICIALES.

Para conseguir la respuesta del sistema cuando la esfera parte de condiciones iniciales de posición y de velocidad, se desarrollan las instrucciones que permiten conseguir esto.

Si partimos de las ecuaciones que gobiernan la dinámica del sistema tenemos:

$$m \frac{d}{dt}\dot{x} = 1.176\ x + 0.059\ i$$

$$e_0 = L \frac{d}{dt} i + R\ i + 0.059\ \dot{x}$$

Si en las expresiones anteriores se aplica la transformada de Laplace considerando condiciones iniciales tanto de posición (x) como de corriente (i) y luego las ordenamos adecuadamente es posible reducirlas de la siguiente manera:

$$X(s) = \frac{22.7}{s^2 - 452.3}\left(I(s) + \frac{s\,X(0) + \dot{X}(0)}{22.7}\right) = G_3\,(I(s) + G_4)$$

$$I(s) = \frac{2.1}{s + 104.6}\left(E_0(s) - 0.059\,sX(s) - LI(0) + 0.059\,X(0)\right)$$

$$= G_2\,(E_0(s) - H(s) + G_1)$$

Expresiones que pueden representarse por medio del siguiente diagrama de bloques:

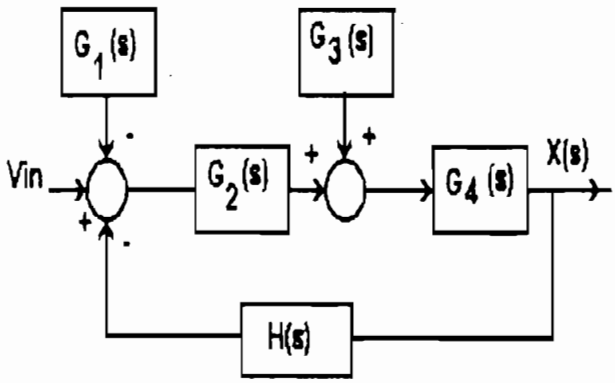


Diagrama de Bloques considerando condiciones iniciales.

Si se colocan las funciones de transferencia del compensador C(s) y del sensor de posición H1(s) se puede obtener el diagrama de bloques del sistema de control. Si se reduce el diagrama de bloques de las consideraciones de condiciones iniciales, se puede obtener el diagrama de bloques completo del sistema de control, el cual se lo indica a continuación:

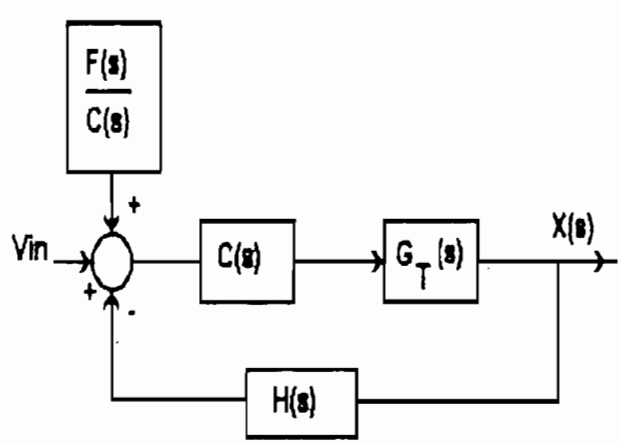


Diagrama de Bloques del Sistema de Control considerando Condiciones Iniciales.