UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO

FACULTAD DE INGENIERIA

PAQUETE DE COMPUTADORA PARA EL DISEÑO Y SIMULACION DE CONTROLADORES DIGITALES

T E S I S

Que para obtener el Título de
INGENIERO EN COMPUTACION

P r e s e n t a n

JORGE GOMEZ LOPEZ
LUIS SERGIO HERNANDEZ OLEA
OSCAR RODRIGUEZ CARMONA

Director de Tesis:
Ing. Francisco José Rodriguez Ramirez

MEXICO, D. F. 1995

A mis Padres.

Quienes me han apoyado incondicionalmente
en todos mis triunfos, así como en mis
derrotas.
Siempre les estaré agradecido por la
educación que me han dado.

Jorge Gómez López

Gracias ...

A Dios

En quien pongo siempre mis esperanzas.

A mis Padres

Por su ejemplo y porque gracias a sus esfuerzos he podido llegar ha este momento.

A mis Hermanos

Por su cariño y compresión.

A mi Novia

Por el amor que siempre me impulsó a seguir adelante.

A mis Amigos

Por estar siempre a mi lado.

A mis Maestros

Que me guiaron hasta completar mis estudios.

Luis Sergio Hernández Olea

A mis Padres

y Hermanos

por el apoyo

y cariño recibidos

al emprender todos mis proyectos

Oscar Rodriguez Carmona

INDICE

INTRODUCCION.

El estudio y análisis de sistemas de control requiere de conocimientos firmes en el área, para comprender la problemática que ésto implica y así encontrar la mejor solución a estos sistemas. Esto requiere la aplicación adecuada de métodos numéricos, así como de análisis de sistemas dinámicos.

Sin embargo, si se requiere de la obtención de resultados rápidos, aún para pruebas sencillas, el realizar este análisis a mano implica mucho tiempo en cada prueba, sin tomar en cuenta los probables errores de cálculo.

La finalidad del sistema presentado en este trabajo es la de proporcionar al usuario relacionado con el área una herramienta útil y básica para la obtención de resultados rápidos en el análisis de sistemas de control digital, en cuanto al diseño y simulación de controladores digitales.

El paquete diseñado permite realizar análisis de sistemas de control digital en malla abierta, malla cerrada con controlador y el diseño de controladores por el método de asignación de polos y ceros.

El presente trabajo esta dividido en los siguientes VI capitulos:

Capitulo I. Contiene la introducción del trabajo, así como los objetivos a cubrir.

Capitulo II. Contiene los algoritmos y métodos numéricos utilizados y desarrollados para la implementación del paquete.

Capitulo III. En este capitulo se presenta el análisis desarrollado, así como una descripción del código y el seudocódigo realizados, además se presenta la estructura general del paquete.

Capitulo IV. Aquí se proveen algunas pruebas realizadas al paquete a nivel módulo.

Capitulo V. En este capitulo se presenta el manual de usuario.

Capitulo VI. Conclusiones.

Anexo. Código fuente de los programas que conforman el sistema.

Bibliografía. Bibliografía consultada.

CAPITULO II
ALGORITMOS Y METODOS NUMERICOS.

En el presente capitulo se describen los diferentes algoritmos y métodos numéricos utilizados en el paquete.

Algunos algoritmos utilizados no fueron tomados de la bibliografía de algún autor, sino que fueron diseñados expresamente para el sistema, los cuales también se encuentran explicados en el este capitulo. Asi mismo se comenta donde y para que fueron ocupados estos algoritmos y métodos.

ELIMINACION GAUSSIANA ELEMENTAL.

El método de eliminación Gaussiana elemental fue empleado para la solución de sistemas de ecuaciones del tipo: Ax = b, donde A es una matriz cuadrada. Aqui se muestra como resolver el problema utilizando métodos aritméticos.

El presente método se ocupó dentro del paquete para la solución de sistemas de ecuaciones resultantes de la expansión en fracciones parciales dentro de la discretización de la función de transferencia de la planta G(S) y del sistema H(S).

Se empezó por considerar el caso especial donde la matriz de coeficientes es triangular superior. Considerar el sistema

$$
\begin{aligned}
b_{11}x_1 + b_{12}x_2 + \ldots + b_{1n}x_n &= c_1 \\
b_{22}x_2 + \ldots + b_{2n}x_n &= c_2 \\
\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots & \qquad (1) \\
b_{n-1,n-1}x_{n-1} + b_{n-1,n}x_n &= c_{n-1} \\
b_{nn}x_n &= c_n
\end{aligned}
$$

La última ecuación involucra solamente a x_n, y por lo tanto si $b_{nn} \neq 0$, se obtiene $x_n = c_n / b_{nn}$. La penúltima ecuación involucra solamente a x_n y a x_{n-1}, y, como x_n es conocida, se puede resolver para x_{n-1} para obtener $x_{n-1} = (c_{n-1} - b_{n-1,n} x_n) / b_{n-1,n-1}$ asumiendo $b_{n-1,n-1} \neq 0$. Continuando de esta forma, podemos resolver para las incógnitas $x_n, \ldots, x_1$ (en este orden) usando $n, n-1, \ldots, 1$ ecuaciones. Este proceso es conocido como solución por sustitución hacia atrás.

ALGORITMO DE SUSTITUCION HACIA ATRAS.

Dada una matriz **B** triangular superior $n \times n$ con $b_{ii} \neq 0$ para todas las i y un vector **c**, entonces la solución de **Bx** = **c** es c_n

$$x_n \leftarrow \frac{}{b_{nn}}$$

Hacer $k = n - 1, n - 2, \ldots, 1$:

$$x_k \leftarrow \frac{c_k - \sum_{j=k+1}^{n} b_{kj} x_j}{b_{kk}}$$

Una consecuencia inmediata del algoritmo de sustitución hacia atrás es la siguiente. Para una matriz **U** triangular superior, todos sus elementos diagonales son distintos de cero, el sistema **Ux** = **c** tiene una solución para cada **c**(n). Por el teorema I esto es equivalente a la invertibilidad de **U**. Por lo tanto que **U** es invertible si y sólo si todos sus elementos diagonales son distintos de cero.

Teorema I.

Si **A** es una matriz cuadrada, entonces se cumple lo siguiente:

(i) La única solución de **Az** = **0** es **z** = **0**.

(ii) **Ax** = **b** tiene una solución para cada **b**.

(iii) **A** es invertible.

El método de eliminación Gaussiana es un procedimiento que reduce el problema **Ax** = **b** a un problema **Ux** = **c**, el cual esta en la forma (1). La justificación para el proceso de eliminación involucra operaciones elementales de renglón y el concepto de problemas equivalentes. Se dice que los problemas **Ax** = **b** y **Ux** = **c** son **equivalentes** si cada solución del inicial es una solución de el posterior y viceversa.

Dado el problema **Ax** = **b** con $\mathbf{A} = (a_{ij})_{n,n}$ la matriz **W** $n \times (n + 1)$ esta dada por

$$\mathbf{W} = [\mathbf{A}|\mathbf{b}] \qquad (2)$$

W se conoce como la matriz **aumentada** del sistema **Ax** = **b**. Suponer que **W** está sujeta a una secuencia finita de operaciones del siguiente tipo (operaciones elementales de renglón):

(i) Intercambiar dos renglones.
(ii) Multiplicar un renglón por un escalar distinto de cero.
(iii) Sumar un múltiplo de un renglón a otro.

$$\tilde{\mathbf{W}} = [\mathbf{U}|\mathbf{c}] \qquad (3)$$

Entonces **Ax** = **b** y **Ux** = **c** son equivalentes.

El proceso involucrado en la transformación de (2) en (3) es indicativo del método de eliminación para un sistema general de ecuaciones lineales. La idea principal de la eliminación Gaussiana es usar operaciones elementales de renglón sobre **W** en (2) para producir $\tilde{\mathbf{W}}$ en (3) tal que **U** es triangular superior. Entonces la sustitución hacia atrás puede ser usada para encontrar la solución **x**.

El primer paso del proceso consiste en eliminar x_1 de cada ecuación excepto la primera. En términos de **W** esto corresponde a

usar operaciones renglón elementales para producir una nueva matriz aumentada para la cual el elemento (1,1) es distinto de cero y donde $(i,1)$ es cero para $i = 2,\ldots,n$. Si el elemento (1,1) de **W** es distinto de cero, entonces sumando $-(w_{i1}/w_{11})$ veces el renglón 1 al renglón i producirá un cero en el elemento $(i,1)$. Si el elemento (1,1) de **W** es cero, se buscará la primera columna de **W** para el elemento distinto de cero, suponer que esto ocurre en el renglón r, entonces intercambiar los renglones 1 y r y proceder como antes.

Habiendo completado el $(k-1)$ paso, **W** ha sido reducida a la siguiente forma :

$$\left[\begin{array}{ccccccc|c} w_{11} & w_{12} & \cdots & & w_{1k} & \cdots & w_{1n} & w_{1,n+1} \\ 0 & w_{22} & \cdots & & w_{2k} & \cdots & w_{2n} & w_{2,n+1} \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ & & & 0 & w_{kk} & \cdots & w_{kn} & w_{k,n+1} \\ & & & 0 & w_{k+1,k} & \cdots & w_{k+1,n} & w_{k+1,n+1} \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & \cdots & & 0 & w_{nk} & \cdots & w_{nn} & w_{n,n+1} \end{array}\right]$$

El k-ésimo paso consiste en eliminar x_k de las ecuaciones $k+1,\ldots,n$. Esto se realiza de la siguiente manera:

(a) Buscar la k-ésima columna, de los renglones k al n, para el primer elemento distinto de cero, esto ocurre en el renglón r.

(b) Si $k \neq r$, entonces intercambiar los renglones k y r.

(c) Calcular los multiplicadores

$$m_{ik} = \frac{-w_{ik}}{w_{kk}} \qquad i = k+1,\ldots,n$$

donde w_{kk} es el "nuevo" elemento (k,k).

(d) Calcular para $i = k+1,\ldots,n$

$$w_{ij}^{\text{nuevo}} = w_{ij}^{\text{anterior}} + m_{ik}\, w_{kj}^{\text{anterior}} \qquad j = k+1,\ldots,n+1$$

ALGORITMO DE ELIMINACION GAUSSIANA ELEMENTAL.

Dada una matriz A de orden $n \times n$ y un vector b de dimensión n, formar la matriz aumentada $W = [A \mid b]$.

Hacer $k = 1, 2, \ldots, n - 1$:

  Buscar la columna k, renglones k a n, para el primer elemento distinto de cero, digamos w_{rk}

  Si no existe el renglón r detenerse

  Si $r \neq k$ entonces | Hacer $j = k, \ldots, n + 1$:

    $temp \leftarrow w_{kj}$

    $w_{kj} \leftarrow w_{rj}$

    $w_{rj} \leftarrow temp$

  Hacer $i = k + 1, \ldots, n$:

    $m_{ik} \leftarrow -w_{ik}/w_{kk}$

    $w_{ik} \leftarrow 0$

    Hacer $j = k + 1, \ldots, n + 1$:

      $w_{ij} \leftarrow w_{ij} + m_{ik} w_{kj}$

Si en el paso k se encuentra que los renglones k a n de la columna k cada uno contiene cero, entonces el algoritmo requiere detener los cálculos. La razón de está terminación es que A no es invertible en este caso. Si $w_{kk} = w_{k+1,k} = \ldots = w_{nk} = 0$ en el paso k, entonces las últimas $n - k + 1$ ecuaciones

$$\begin{aligned} 0 \quad & x_k + w_{k,k+1} x_{k+1} + \ldots + w_{kn} x_n = w_{k,n+1} \\ & \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ 0 \quad & x_k + w_{n,k+1} x_{k+1} + \ldots + w_{nn} x_n = w_{n,n+1} \end{aligned} \qquad (4)$$

involucran solamente las $n - k$ incógnitas $x_{k+1}, \ldots, x_n$. Hay más ecuaciones que incógnitas en (4), y por lo tanto, por el teorema II parte (iv), este subsistema no tiene una solución para algún elemento $b(n)$. Por lo tanto el sistema original no tiene una

solución para algún elemento $b_{(m)}$. De este resultado y del teorema I se tiene que **A** no es invertible.

Teorema II.

(i) Si **x** es una solución de **Ax = b**, entonces cualquier otra solución **y** puede ser escrita como **y = x + z** con **Az = 0**, donde **0** denota el vector cuyos elementos son todos cero.

(ii) La ecuación **Ax = b** tiene como máximo una solución si y sólo si la única solución de **Az = 0** es **z = 0**.

(iv) Si **Ax = b** tiene una solución para cada elemento $b_{(m)}$, entonces $m \leq n$.

El método de eliminación Gaussiana proporciona un medio sistemático para resolver sistemas de ecuaciones lineales que es conceptualmente simple e idealmente ajustable para su implantación en computadoras. Un producto resultante del proceso de eliminación es una prueba computacional para la invertibilidad de **A**, es decir, la matriz **A** es invertible si y sólo si la matriz aumentada final de la eliminación Gaussiana aplicada a **Ax = b** es de la forma (3), donde **U** es triangular superior con elementos distintos de cero en la diagonal.

II.2 SOLUCION DE ECUACIONES DIFERENCIALES.

El método utilizado para la solución de ecuaciones diferenciales en el sistema es el de Runge-Kutta de cuarto orden, tras haberlo comparado con el método de Euler. Este método utiliza el error local de truncamiento de alto orden y se empleará para la solución de la planta, dentro de los módulos de Diseño y de Controlador propuesto.

METODO RUNGE-KUTTA DE CUARTO ORDEN.

El método Runge-Kutta de uso más común es el de cuarto orden, el cual en forma diferencial está dado por:

$$
\begin{aligned}
&w_0 = \alpha,\\
&k_1 = hf(t_i,\ w_i),\\
&k_2 = hf\left(t_i + \frac{h}{2},\ w_i + \frac{1}{2}k_1\right),\\
&k_3 = hf\left(t_i + \frac{h}{2},\ w_i + \frac{1}{2}k_2\right),\\
&k_4 = hf(t_{i+1},\ w_i + k_3),\\
&w_{i+1} = w_i + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4),
\end{aligned}
\qquad (I)
$$

para cada $i = 0,1,\ldots,N-1$. Este método tiene un error de truncamiento local de $O(h^4)$, y la solución $y(t)$ tiene cinco derivadas continuas. La razón para introducir la terminología k_1, k_2, k_3, k_4 en el método es para eliminar la necesidad de anidamientos sucesivos en la segunda variable de la función $f(t, y)$.

ALGORITMO DEL METODO RUNGE-KUTTA (CUARTO ORDEN)

Para aproximar la solución al problema de valor-inicial

$$y' = f(t, y), \qquad a \le t \le b, \qquad y(a) = \alpha,$$

seleccionar un entero positivo N.

Paso 1 Fijar $h = (b - a)/N$, $t_0 = a$, y $w_0 = \alpha$.

Paso 2 Fijar $i = 0$.

Paso 3 Fijar $k_1 = hf(t_i,\ w_i)$

$$k_2 = hf\left[t_i + \frac{h}{2}, w_i + \frac{1}{2}k_1\right]$$

$$k_3 = hf\left[t_i + \frac{h}{2}, w_i + \frac{1}{2}k_2\right]$$

$$k_4 = hf(t_i + h, w_i + k_3)$$

$$w_{i+1} = w_i + 1/6(k_1 + 2k_2 + 2k_3 + 2k_4),$$

$$t_{i+1} = a + (i+1)h.$$

Paso 4 Si $i = N - 1$, ir al paso 6.

Paso 5 Sumar 1 a i e ir al paso 3.

Paso 6 El procedimiento está completo, y w_i aproxima $y(t_i)$, para cada $i = 1,2,\ldots,N$.

El principal esfuerzo computacional en aplicar los métodos Runge-Kutta es en la evaluación de f. En los métodos de segundo orden el error de truncamiento es $O(h^2)$, pero el costo es de dos evaluaciones funcionales por paso. El método Runge-Kutta de cuarto orden requiere cuatro evaluaciones por paso, pero el error de truncamiento es de $O(h^4)$. Se han establecido las siguientes relaciones entre el número de evaluaciones por paso y el orden del error de truncamiento local:

Evaluaciones por paso	2	3	4	5	6	7	n≥8
Mejor posible error de truncamiento local	$O(h^2)$	$O(h^3)$	$O(h^4)$	$O(h^4)$	$O(h^5)$	$O(h^6)$	$O(h^{n-2})$

Consecuentemente, los métodos de menor orden con menor número de pasos son usados en preferencia de los posibles métodos de mayor orden con mayor número de pasos.

Una medida con la cual comparar a los métodos Runge-Kutta de menor orden es descrita a continuación: como el método Runge-Kutta de cuarto orden requiere cuatro evaluaciones por paso, esto dará respuestas más exactas que el método de Euler con un cuarto de tamaño de malla (donde por tamaño de malla se refiere a la diferencia entre puntos consecutivos) si éste es superior.
Similarmente si el método Runge-Kutta de cuarto orden es superior a los métodos Runge-Kutta de segundo orden, entonces éste daría más precisión con h paso de integración que un método de segundo orden con $\frac{1}{2}h$ paso de integración, porque el método de cuarto orden requiere el doble de evaluaciones por paso.

Un ejemplo de la superioridad del método Runge-Kutta de cuarto orden dado en (I) es mostrado en el siguiente ejemplo.

Para el problema:

$$y'' - y = 5 \text{ SEN } 2t \quad ; \quad 0 \leq t \leq 2,$$
$$y(0) = 0 \quad ; \quad y'(0) = 0$$

El método de Euler con $h = 0.025$ y el método Runge-Kutta de cuarto orden (I) con $h = 0.1$, serán comparados en los puntos 0.1, 0.2, 0.3, 0.4, ..., hasta 2.0. Los resultados dados en la siguiente tabla muestran una clara superioridad del método Runge-Kutta de cuarto orden sobre el método de Euler en cuanto a precisión, razón por la cual fué seleccionado.

t	Método de Euler $h = 0.025$	Método de Runge-Kutta 4o. orden $h = 0.1$	Diferencia	Error entre valor Runge-Kutta vs real	Valor real
0.0	0.000000	0.000000	0.000000	0.000000	0.00000
0.1	0.001581	0.001667	0.000106	0.000003	0.001664
0.2	0.013057	0.013260	0.000203	0.000006	0.013254
0.3	0.044128	0.044408	0.000280	0.000010	0.044398
0.4	0.103834	0.104161	0.000327	0.000012	0.104149
0.5	0.200397	0.200733	0.000336	0.000013	0.200720
0.6	0.340979	0.341282	0.000323	0.000006	0.341288
0.7	0.531505	0.531730	0.000223	0.000012	0.531718
0.8	0.776547	0.776648	0.000101	0.000010	0.776638
0.9	1.079256	1.079192	0.000064	0.000006	1.079186
1.0	1.441370	1.441106	0.000264	0.000001	1.441105
1.1	1.863283	1.862794	0.000489	0.000005	1.862799
1.2	2.344179	2.343449	0.000730	0.000011	2.343460
1.3	2.882219	2.881247	0.000972	0.000017	2.881264
1.4	3.474795	3.473592	0.001203	0.000023	3.473615
1.5	4.118818	4.117411	0.001407	0.000028	4.117439
1.6	4.811050	4.809479	0.001571	0.000031	4.809510
1.7	5.548457	5.546771	0.001686	0.000034	5.546805
1.8	6.328574	6.326835	0.001739	0.000034	6.326869
1.9	7.149876	7.148152	0.001724	0.000032	7.148184
2.0	8.012134	8.010496	0.001638	0.000027	8.010523

II.3 BINOMIO DE NEWTON.

El binomio de Newton permite reducir polinomios al compactar términos en grupos de elementos binomios, o bien puede ser utilizado para la función inversa, o sea, el desarrollo de los binomios en los polinomios que representan de forma extendida.

En este caso el binomio de Newton fue utilizado para desarrollar a los binomios en forma expandida, y así poder agrupar términos semejantes dentro de la solución de ecuaciones en la sección de expansión en fracciones parciales.

Ejemplo:

$$(a+b)^2 = a^2b^0 + 2ab + a^0b^2$$

$$(a+b)^3 = a^3b^0 + 3a^2b + 3ab^2 + a^0b^3$$

Fórmula:

$$(a+b)^n = a^nb^0 + C(n,1)a^{n-1}b^1 + C(n,2)a^{n-2}b^2 + C(n,3)a^{n-3}b^3 + \ldots + C(n,n-1)ab^{n-1} + C(n,n)a^0b^n$$

donde :

$$C(n,r) = \frac{n!}{r!(n-r)!} .$$

II.4 POLINOMIOS Y RAICES DE POLINOMIOS.

En la aplicación de diferentes métodos, tales como fracciones parciales y el cálculo del tiempo de análisis, así como para determinar la estabilidad del sistema fué necesario encontrar las raices, tanto reales como complejas, de los polinomios involucrados, para lo cual se utilizó el método de Lin-Bairstow.

Introducción.

Sea un polinomio de grado n de la forma

$$P_n(x) = a_0x^n + a_1x^{n-1} + a_2x^{n-2} + \ldots + a_{n-1}x + a_n \quad (1)$$

el cual se utiliza tipicamente en la solución de problemas matemáticos. Los polinomios pueden ser fácilmente multiplicados y divididos por otros polinomios. Ellos pueden ser diferenciados, integrados y evaluados. Permitiendo que los polinomios tomen formas distintas de (1), es posible derivar las diferentes formas de diferencias finitas y la forma ordinaria de interpolación de polinomios, la cual es usada extensivamente en la derivación de fórmulas para diferenciación e integraciones números. Aquí sólo nos conciernen los polinomios de la forma (1).

Los polinomios son utilizados extensivamente por todas las disciplinas cientificas, especialmente en relación con ecuaciones diferenciales. La ecuación característica de un sistema de n ecuaciones diferenciales lineales de primer orden con coeficientes constantes (matriz de términos característicos) se puede expresar en la forma de una ecuación polinomial. Equivalentemente la ecuación característica de una ecuación diferencial lineal de orden n es una ecuación polinomial de grado n. En este caso el problema de la matriz de términos característicos y la determinación de las raíces de una ecuación polinomial son matemáticamente un sólo problema. Sin embargo el problema de esta matriz es usualmente resuelto sin reducir la ecuación característica a la forma polinomial explícita.

Las raíces de ecuaciones polinomiales de grado 1, 2, 3 y 4 pueden ser determinadas por métodos (en forma cerrada) clásicos. Sin embargo, estos métodos están bastante involucrados en las ecuaciones de tercero y cuarto grados. Por ésta razón, y porque debemos resolver ecuaciones polinomiales de grados mayores, métodos

números de aproximaciones sucesivas son usados extensivamente para la determinación de las raíces de ecuaciones polinomiales. Antes de desarrollar tales métodos numéricos, se presentarán algunos teoremas básicos y relaciones concernientes a polinomios.

Teorema Fundamental del Algebra.

Cada ecuación algebraica (polinomial) con coeficientes complejos tiene por lo menos una raíz real o compleja.

Algoritmo de la división.

Si $P(x)$ y $F(x)$ son polinomios en x y $F(x) \neq 0$, entonces los polinomios $Q(x)$ y $R(x)$ pueden ser encontrados, tal que satisfagan la relación

$$P(x) = Q(x)F(x) + R(x) \qquad (2)$$

donde $R(x) = 0$, o bien, el grado de $R(x)$ < grado de $F(x)$.

Teorema del residuo.

El residuo obtenido al dividir $P(x)$ por $(x-\alpha)$ es el valor de $P(\alpha)$.

Prueba. Dividiendo $P(x)$ por $(x-\alpha)$ obtenemos

$$P(x) = (x-\alpha)Q(x) + R \qquad (3)$$

donde $R = 0$ o R = constante, y como el grado de R < 1 por el algoritmo de la división, evaluando (3) para $x = \alpha$, obtenemos:

$$P(\alpha) = (\alpha-\alpha)Q(\alpha) + R \qquad (4)$$

tal que

$$R = P(\alpha) \quad y \quad P(x) = (x-\alpha)Q(x) + P(\alpha).$$

Teorema del Factor.

Cada ecuación polinomial de la forma

$$P_n(x) \equiv x^n + a_1x^{n-1} + a_2x^{n-2} + \ldots + a_{n-1}x + a_n = 0 \qquad (5)$$

tiene cuando más n distintas raíces α_i $(i = 1, 2, \ldots, n)$.

Por el teorema fundamental del álgebra, existe cuando menos una raiz de la ecuación algebraica (5). Si α_1 es una raiz, por ejemplo, y $P_n(\alpha_1) = 0$, entonces por el teorema del residuo:

$$P_n(x) = (x-\alpha_1)P_{n-1}(x) + P_n(\alpha_1) = (x-\alpha_1)P_{n-1}(x). \quad (6)$$

Similarmente, ahi existe cuando menos una raiz, sea α_2, de $P_{n-1}(x) = 0$, tal que

$$P_{n-1}(x) = (x-\alpha_2)P_{n-2}(x). \quad (7)$$

Este proceso se continúa hasta que se obtiene

$$P_1(x) = (x-\alpha_n). \quad (8)$$

Ahora, por sustitución sucesiva de la relación

$$P_{n-i}(x) = (x-\alpha_{i+1})P_{n-(i+1)}(x) \quad (9)$$

en (6), se obtiene

$$P_n(x) = (x-\alpha_1)(x-\alpha_2)\ldots(x-\alpha_n) \quad (10)$$

La ecuación (10) establece que un polinomio $P_n(x)$ de grado n puede ser factorizado en n factores lineales $(x-\alpha_i)$, donde las $\alpha_i (i = 1, 2, \ldots, n)$ son raices de la correspondiente ecuación polinomial. Se deberá notar que la α_i no necesita ser distinta. Puede ser probado por el principio de inducción finita que ésta factorización es única excepto para el orden en que aparecen los factores.

EL METODO ITERATIVO DE LIN-BAIRSTOW.

El método de Lin-Bairstow es un procedimiento iterativo para calcular las raices (reales o complejas) de una ecuación polinomial de coeficientes constantes, el cual requiere solamente de la manipulación de números reales en los cálculos. El método está basado en la extracción sucesiva de factores cuadráticos $F_m(x)$ $(m = 1, 2, \ldots)$ del polinomio original de grado N y factores polinomiales sucesivos de grado $N-2m$. Cada factor cuadrático está

determinado por un procedimiento iterativo de corrección diferencial.

El teorema del factor establece que el polinomio de grado N

$$P_N(x) = x^N + a_1 x^{N-1} + a_2 x^{N-2} + a_{N-1} x + a_N \qquad (11)$$

puede ser expresado como el producto de N factores lineales, de la forma

$$P_N(x) = (x-\alpha_1)(x-\alpha_2)\ldots(x-\alpha_N) \qquad (12)$$

donde los α_j $(j = 1, 2, \ldots, N)$, no necesariamente distintos, son las raíces de la ecuación polinomial

$$P_N(x) = 0. \qquad (13)$$

Obviamente, entonces, un polinomio de grado $N = 2k$, donde k, es un entero positivo, puede ser expresado como un producto de k factores cuadráticos, como puede ser visto de la ecuación (12) por la agrupación de factores lineales pares, y considerando el producto de cada par como un factor cuadrático del polinomio. Como las raíces complejas de una ecuación polinomial de coeficientes reales ocurren en pares conjugados, α y $\bar{\alpha}$, los conjugados complejos deben ser agrupados de tal manera que los productos cuadráticos solamente tendrán coeficientes reales.

Por lo tanto podemos expresar el polinomio $P_N(x)$, donde $N = 2k$, de la forma

$$P_N(x) = \prod_{m=1}^{k} F_m(x) \qquad (14)$$

donde

$$F_m(x) = x^2 + r_m x + s_m \qquad (m = 1, 2, \ldots, k). \qquad (15)$$

Similarmente, un polinomio de grado impar $N = 2k + 1$ puede ser expresado como un producto de k factores cuadráticos y un factor lineal sencillo, como sigue:

$$P_N(x) = (x - \alpha_{2k+1}) \prod_{m=1}^{k} F_m(x) \qquad (16)$$

donde $F_m(x)$ es un factor cuadrático como el definido en (15), y

a_{2k+1} es la raiz real correspondiente al factor lineal simple del polinomio.

Las raices del polinomio de grado arbitrario *N* pueden por lo tanto ser calculadas determinando sucesivamente los factores cuadráticos de ese polinomio, y además (en el caso de un polinomio de grado impar) por la determinación final del factor lineal sencillo. El método de Lin-Bairstow es un algoritmo eficiente para calcular las raices de un polinomio por extracción sucesiva de sus factores cuadráticos.

Ejemplos:

N par (polinomio con todos los ceros complejos):

$$P_4(x) = [x-(a_1+b_1i)][x-(a_1-b_1i)][x-(a_2+b_2i)][x-(a_2-b_2i)]$$

$$= (x^2 - 2a_1x + a_1^2 + b_1^2)(x^2 - 2a_2x + a_2^2 + b_2^2).$$

N par (polinomio con ceros reales y complejos mezclados):

$$P_4(x) = [x-a_1][x-a_2][x-(a+bi)][x-(a-bi)]$$

$$= [x^2-(a_1+a_2)x + a_1a_2][x^2- 2ax + a^2 + b^2].$$

N impar (polinomio con ceros reales y complejos mezclados):

$$P_5(x) = [x-(a+bi)][x-(a-bi)](x-a_1)(x-a_2)(x-a_3)$$

$$= [x^2 - 2ax + a^2 + b^2][x^2 - (a_1+a_2)x + a_1a_2](x-a_3).$$

N impar (polinomio con todos los ceros reales):

$$P_5(x) = (x-a_1)(x-a_2)(x-a_3)(x-a_4)(x-a_5)$$

$$= [x^2-(a_1+a_2)x + a_1a_2][x^2-(a_3+a_4)x + a_3a_4](x-a_5).$$

Determinando un factor cuadrático de un polinomio.

Denotando al grado del polinomio a ser factorizado en una dada representación del método de Lin-Bairstow. Se tiene, $n = N-2k$ (k = 0, 1, ...), donde k factores cuadráticos han sido ya extraidos del polinomio original $P_N(x)$.

Si $P_n(x)$ es dividido por un factor cuadrático de prueba $F(x) = x^2 + rx + s$, donde r y s son constantes reales arbitrarias, se obtiene:

$$P_n(x) = F(x)P_{n-2}(x) + Rx + S. \qquad (17)$$

En forma expandida ésta ecuación puede ser escrita como

$$x^n + a_1x^{n-1} + a_2x^{n-2} + \ldots + a_{n-1}x + a_n$$

$$= (x^2 + rx + s)(x^{n-2} + b_1x^{n-3} + b_2x^{n-4} + \ldots + b_{n-3}x + b_{n-2}) + Rx + S. \qquad (18)$$

Nótese que cambios en r ó s causarán cambios en los coeficientes b_k de el cociente polinomial $P_{n-2}(x)$ y en los coeficientes R y S de el término residuo. Por lo tanto, considerando r y s como "variables" independientes, y expresando los b_k, R y S como funciones de estas "variables", denotamos estas funciones por $b_k(r,s)$, $R(r,s)$ y $S(r,s)$, respectivamente.

El requerimiento de que $F(x)$ sea un factor de $P_n(x)$ impone las restricciones

$$R(r,s) = 0$$
$$S(r,s) = 0 \qquad (19)$$

El problema de extraer un factor cuadrático de un polinomio puede por lo tanto ser resuelto calculando las raices de las ecuaciones (19).

Calculo de las raices de $R(r,s) = 0 = S(r,s)$.

Suponiendo que estimados iniciales r_o, s_o de las raices de sistemas de ecuaciones (19) sean conocidos. Si estos valores iniciales son incrementados respectivamente por pequeñas variaciones δ_r y δ_s, entonces las aproximaciones de primer orden de las variaciones resultantes en las funciones $R(r,s)$ y $S(r,s)$, respectivamente, son dadas por las diferenciales totales

$$\delta R = R_r\delta r + R_s\delta s$$
$$\delta S = S_r\delta r + S_s\delta s \qquad (20)$$

donde R_r, R_s, S_r, S_s denotan las derivadas parciales de R y S con respecto a r y s, en los valores actuales de r y s.

Dadas raices estimadas r_0, s_0, se puede calcular $R(r_0,s_0)$ y $S(r_0,s_0)$ dividiendo $P_n(x)$ por $x^2 + r_0x + s_0$, obteniendo los coeficientes R y S del término residuo, como

$$R = R(r_0,s_0) \approx 0$$
$$S = S(r_0,s_0) \approx 0. \qquad (21)$$

Se debe por lo tanto determinar δr y δs tales que las diferenciales totales δR y δS satisfagan las condiciones

$$R(r_0,s_0) + \delta R = 0$$
$$S(r_0,s_0) + \delta S = 0. \qquad (22)$$

Esto es, imponer las condiciones

$$\delta R = -R(r_0,s_0)$$
$$\delta S = -S(r_0,s_0). \qquad (23)$$

Las ecuaciones (20) y (23) juntas dan las siguientes relaciones:

$$R_r\delta r + R_s\delta s = -R(r_0,s_0)$$
$$S_r\delta r + S_s\delta s = -S(r_0,s_0). \qquad (24)$$

Las ecuaciones (24) son llamadas las ecuaciones de corrección diferencial. Las soluciones δr y δs de estas ecuaciones son referidas como las correcciones diferenciales.

Habiendo calculado las correcciones diferenciales δr y δs, se puede evaluar las expansiones de la serie de Taylor de primer orden:

$$R(r_0 + \delta r, s_0 + \delta s) \doteq R(r_0,s_0) + R_r\delta r + R_s\delta s$$
$$S(r_0 + \delta r, s_0 + \delta s) \doteq S(r_0,s_0) + S_r\delta r + S_s\delta s \qquad (25)$$

De (24) se tiene que los lados derechos de (25) son cero. Por lo tanto,

$$R(r_0 + \delta r, s_0 + \delta s) \doteq 0$$

$$S(r_0 + \delta r, s_0 + \delta s) \doteq 0.$$

Esto es, $r_1 = r_0 + \delta r$ y $s_1 = s_0 + \delta s$ son aproximaciones de primer orden de los ceros de funciones $R(r,s)$ y $S(r,s)$.

Usando aproximaciones solución r_1, s_1 se pueden calcular aproximaciones solución redefinidas r_2, s_2 por el mismo procedimiento definido antes usando r_0, s_0 para calcular r_1, s_1. Similarmente, r_3, s_3 pueden ser calculados de r_2, s_2. De ésta manera una secuencia convergente (r_0,s_0), (r_1,s_1), (r_2,s_2), (r_3,s_3),... de aproximaciones solución sucesivas para las ecuaciones (19) pueden ser generadas. Si, para alguna iteración k, las condiciones

$$\begin{aligned} |r_{k+1} - r_k| &\leq \varepsilon \\ |s_{k+1} - s_k| &\leq \varepsilon \end{aligned} \qquad (26)$$

son satisfechas simultáneamente para algún criterio de convergencia "ε", con un valor $\varepsilon > 0$, entonces la rutina de corrección diferencial se dice que ha convergido, y tenemos

$$\begin{aligned} R(r_{k+1}, s_{k+1}) &= 0 \\ S(r_{k+1}, s_{k+1}) &= 0. \end{aligned} \qquad (27)$$

Denotando r_{k+1} y s_{k+1} por r^* y s^*, respectivamente, se tiene que el término cuadrático

$$x^2 + r^*x + s^*$$

es un factor del polinomio $P_n(x)$, satisfaciendo

$$\begin{aligned} P_n(x) &= (x^2 + r^*x + s^*)(x^{n-2} + b_1^*x^{n-3} + b_2^*x^{n-4} \\ &\qquad + \ldots + b_{n-3}^*x + b_{n-2}^*) \\ &= F(x)P_{n-2}(x) \end{aligned} \qquad (28)$$

y $P_{n-2}(x)$ es el polinomio a ser factorizado en la siguiente etapa.

Cálculo de los coeficientes R y S de el residuo.

Como se notó antes, la división de $P_n(x)$ por $F(x) = x^2+rx+s$ da un cociente $P_{n-2}(x)$ y un residuo $Rx + S$. Aquí se trata la derivación de fórmulas recursivas para calcular R y S para usarlas en las ecuaciones de corrección diferencial.

Dividiendo $P_n(x)$ por $F(x)$, se obtiene

$$x^n + a_1x^{n-1} + a_2x^{n-2} + \ldots + a_{n-1}x + a_n$$

$$= (x^2+ rx + s)(x^{n-2} + b_1x^{n-3} + b_2x^{n-4} + \ldots + b_{n-3}x + b_{n-2}) + Rx + S. \qquad (29)$$

Sacando las multiplicaciones indicadas e igualando coeficientes de potencias similares de x, se tiene

$$\begin{aligned} a_1 &= b_1 + r \\ a_2 &= b_2 + rb_1 + s \\ a_3 &= b_3 + rb_2 + sb_1 \\ &\vdots \\ a_k &= b_k + rb_{k-1} + sb_{k-2} \\ &\vdots \\ a_{n-1} &= R + rb_{n-2} + sb_{n-3} \\ a_n &= S + sb_{n-2}. \end{aligned} \qquad (30)$$

Si ahora se introduce la fórmula de recursión

$$b_k = a_k - rb_{k-1} - sb_{k-2} \qquad (k = 3, 4, \ldots, n) \qquad (31)$$

donde

$$b_1 = a_1 - r$$
$$b_2 = a_2 - rb_1 - s$$

son usados para iniciar la recursión, se obtienen:

$$
\begin{aligned}
b_1 &= a_1 - r \\
b_2 &= a_2 - rb_1 - s \\
b_3 &= a_3 - rb_2 - sb_1 \\
&\vdots \\
b_k &= a_k - rb_{k-1} - sb_{k-2} \\
&\vdots \\
b_{n-1} &= a_{n-1} - rb_{n-2} - sb_{n-3} \\
b_n &= a_n - rb_{n-1} - sb_{n-2}.
\end{aligned} \tag{32}
$$

Comparando las últimas dos ecuaciones de (30) con las correspondientes ecuaciones (32), se tiene que

$$
\begin{aligned}
R &= b_{n-1} \\
S &= b_n + rb_{n-1}.
\end{aligned} \tag{33}
$$

Nótese que R y S son las funciones $R(r,s)$ y $S(r,s)$, requeridas en las ecuaciones de corrección diferencial (24). Los cuatro términos restantes R_r, R_s, S_r, S_s requeridos en las ecuaciones de corrección diferencial son obtenidos tomando las derivadas parciales de R y S en (33), con respecto a r y s. Por lo tanto:

$$
\begin{aligned}
R_r &= \frac{\delta b_{n-1}}{\delta r} \\
R_s &= \frac{\delta b_{n-1}}{\delta s} \\
S_r &= \frac{\delta b_n}{\delta r} + r\,\frac{\delta b_{n-1}}{\delta r} + b_{n-1} \\
S_s &= \frac{\delta b_n}{\delta s} + r\,\frac{\delta b_{n-1}}{\delta s}.
\end{aligned} \tag{34}
$$

Cálculo de R_r, R_s, S_r, S_s.

Para simplificar la notación en la derivación de las fórmulas de recursión para las derivadas parciales, se definirán:

$$\rho_k = \frac{\delta b_k}{\delta r}$$
$$q_k = \frac{\delta b_k}{\delta s}. \qquad (35)$$

Diferenciando los $b_k (k = 1, 2, \ldots, n)$ de (32) con respecto a r y s, respectivamente, se llega a:

$$\begin{aligned}
\rho_1 &= -1 & q_1 &= 0 \\
\rho_2 &= r - b_1 & q_2 &= -1 \\
\rho_3 &= -b_2 - r\rho_2 - s\rho_1 & q_3 &= -b_1 - rq_2 - sq_1 \\
&\vdots & &\vdots \\
\rho_k &= -b_{k-1} - r\rho_{k-1} - s\rho_{k-2} & q_k &= -b_{k-2} - rq_{k-1} - sq_{k-2} \\
&\vdots & &\vdots \\
\rho_{n-1} &= -b_{n-2} - r\rho_{n-2} - s\rho_{n-3} & q_{n-1} &= -b_{n-3} - rq_{n-2} - sq_{n-3} \\
\rho_n &= -b_{n-1} - r\rho_{n-1} - s\rho_{n-2} & q_n &= -b_{n-2} - rq_{n-1} - sq_{n-2}.
\end{aligned} \qquad (36)$$

Sustituyendo la notación de (35) en (34):

$$\begin{aligned}
R_r &= \rho_{n-1} \\
R_s &= q_{n-1} \\
S_r &= \rho_n + r\rho_{n-1} + b_{n-1} \\
S_s &= q_n + rq_{n-1}.
\end{aligned} \qquad (37)$$

Se tienen ahora fórmulas de recursión para calcular los seis términos requeridos en la solución de ecuaciones de corrección diferencial. Esto es, R y S pueden ser calculados por (32) y (33), y R_r, R_s, S_r, S_s pueden ser calculados por (36) y (37).

Notas.

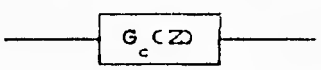
1. La técnica de corrección diferencial descrita aquí para el cálculo de ceros de $R(r,s)$ y $S(r,s)$ simultáneamente es simplemente el método de Newton para resolver dos ecuaciones no lineales con dos incógnitas.

2. Los coeficientes de los polinomios reducidos $P_{n-2}(x)$, $P_{n-4}(x)$, $P_{n-6}(x)$,... no son exactos por la acumulación de redondeo, porque los coeficientes residuo R, S en cualquier etapa de la reducción son sólo aproximadamente cero. Por ésta razón, el valor casi exacto de $x^2 + rx + s$, en la etapa m, debería ser usado como los valores iniciales para la extracción de un factor más exacto del polinomio original $P_N(x)$.

3. Para la mayoría de los casos, el método de Bairstow puede ser iniciado usando estimados iniciales $r_o^o = 0$, $s_o^o = 0$. El procedimiento puede, sin embargo, tener que "buscar" hasta que encuentre valores de r y s los cuales lo lleven a converger.

II.5 SOLUCION DE ECUACIONES EN DIFERENCIAS.

La solución de ecuaciones en diferencias se emplea para resolver la función de transferencia del controlador $G_c(Z)$, el proceso utilizado se describe a continuación:



Tenemos definida a $G_c(Z)$ como:

$$G_c(Z) = \frac{a_o + a_1 Z + a_2 Z^2 + \ldots + a_m Z^m}{a_o + a_1 Z + a_2 Z^2 + \ldots + a_n Z^n} \qquad n \geq m$$

$$G_c(Z) = \frac{U(Z)}{E(Z)}$$

Ahora multiplicando $U(Z)$ y $E(Z)$ por Z^{-n} se obtiene:

$$\frac{a_m Z^{m-n} + \ldots + a_2 Z^{2-n} + a_1 Z^{1-n} + a_0 Z^{-n}}{a_n + \ldots + a_2 Z^{2-n} + a_1 Z^{1-n} + a_0 Z^{-n}} = \frac{U(Z)}{E(Z)}$$

desarrollando:

$$E(Z)(a_m Z^{m-n} + \ldots + a_2 Z^{2-n} + a_1 Z^{1-n} + a_0 Z^{-n}) =$$

$$= U(Z)(a_n + \ldots + a_2 Z^{2-n} + a_1 Z^{1-n} + a_0 Z^{-n})$$

antitransformando:

$$a_m e((k-n+m)T) + \ldots + a_2 e((k-n+2)T) + a_1 e((k-n+1)T) + a_0 e((k-n)T) =$$

$$= a_n u(k) + \ldots + a_2 u((k-n+2)T) + a_1 u((k-n+1)T) + a_0 u((k-n)T)$$

despejando u(kT):

$$u(kT) = \left[a_m e((k-n+m)T) + \ldots + a_2 e((k-n+2)T) + a_1 e((k-n+1)T) + a_0 e((k-n)T)\right]\frac{1}{a_n}$$

$$- \left[a_{n-1} u((k-n-1)T) + \ldots + a_2 u((k-n+2)T) + a_1 u((k-n+1)T) + a_0 u((k-n)T)\right]\frac{1}{a_n}$$

ahora aplicando esto a un ejemplo:

$$G(Z) = \frac{Z + 0.5}{Z^2 + 2Z + 1} = \frac{U(Z)}{E(Z)}$$

multiplicando por Z^{-2} :

$$\frac{Z^{-1} + 0.5Z^{-2}}{1 + 2Z^{-1} + Z^{-2}} = \frac{U(Z)}{E(Z)}$$

desarrollando:

$$E(Z)(Z^{-1} + 0.5Z^{-2}) = U(Z)(1 + 2Z^{-1} + Z^{-2})$$

antitransformando:

$$e(kT-T) + 0.5e(kT-2T) = u(kT) + 2u(kT-T) + u(kT-2T)$$

despejando u(kT) queda la función:

$$u(kT) = e(kT-T) + 0.5e(kT-2T) - 2u(kT-T) - u(kT-2T)$$

La fórmula de transformación utilizada fue la de retraso, y es

la siguiente:

$$\mathcal{Z}\left\{f(kT-nT)\right\} = Z^{-n}F(Z) - Z^{-n}\sum_{j=0}^{n-1} Z^{j} f(jT).$$

II.6 TRANSFORMADA INVERSA DE LAPLACE.

La transformada inversa de Laplace se utilizó para resolver ecuaciones diferenciales, dentro de la discretización de la planta G(S) y del sistema H(S), y en la solución de la planta en malla abierta. Primero se revisan las propiedades de la transformada inversa, y después se explica el procedimiento empleado.

Si : $F(S) = L\{f(t)\}$

se dice que f(t) es la transformada inversa de Laplace de F(S) y se puede representar como :

$$f(t) = L^{-1}\{F(S)\}$$

Ahora se mencionarán sus propiedades:

1) Linealidad.

$$L^{-1}\{aF_1(S) + bF_2(S)\} = aL^{-1}\{F_1(S)\} + bL^{-1}\{F_2(S)\}$$

2) Translación.

a) $L^{-1}\{F(S-m)\} = e^{mt}L^{-1}\{F(S)\}$

b) $L^{-1}\{F(S)\} = e^{-mt}L^{-1}\{F(S-m)\}$

3) Cambio de escala.

$$L^{-1}\{f(mS)\} = \frac{1}{m} f(t/m)$$

$$L^{-1}\{F(S/m)\} = m\, f(mt)$$

La transformada de Laplace se puede usar para resolver ecuaciones diferenciales mediante el siguiente procedimiento:

Sea la ecuación diferencial:

$$\frac{d^n x}{dt^n} + a_1 \frac{d^{n-1} x}{dt^{n-1}} + a_2 \frac{d^{n-2} x}{dt^{n-2}} + \ldots + a_{n-1} \frac{dx}{dt} + a_n x = b\, f(t) \quad \ldots(1)$$

1) Aplicar la transformada de Laplace en la ecuación (1):

$$L\left\{\frac{d^n x}{dt^n} + a_1 \frac{d^{n-1} x}{dt^{n-1}} + a_2 \frac{d^{n-2} x}{dt^{n-2}} + \ldots + a_{n-1} \frac{dx}{dt} + a_n x\right\} = L(b\, f(t)) \quad \ldots(2)$$

o bien:

$$L\left\{\frac{d^n x}{dt^n}\right\} = S^n X(S) - \sum_{k=0}^{n-1} S^{n-1-k} x^k(0)$$

$$L\left\{a_1 \frac{d^{n-1} x}{dt^{n-1}}\right\} = a_1 S^{n-1} X(S) - a_1 \sum_{k=0}^{n-2} S^{n-2-k} x^k(0)$$

$$L\left\{a_2 \frac{d^{n-2} x}{dt^{n-2}}\right\} = a_2 S^{n-2} X(S) - a_2 \sum_{k=0}^{n-3} S^{n-3-k} x^k(0)$$

$$\vdots$$

$$L\left\{a_{n-1} \frac{dx}{dt}\right\} = a_{n-1} S X(S) - a_{n-1} x(0)$$

$$L\{a_n x\} = a_n X(S)$$

$$L\{b f(t)\} = b\, F(S)$$

$\ldots(3)$

Sustituyendo las ecuaciones (3) en la ecuación (2), se llega a la ecuación:

$$X(S)(S^n + a_1 S^{n-1} + a_2 S^{n-1} + a_2 S^{n-2} + \ldots + a_{n-1} S + a_n) =$$
$$= b\, F(S) + R(S) = Q(S) \quad \ldots(4)$$

2) De la ecuación (4) se despeja la función X(S):

$$X(S) = \frac{Q(S)}{S^n + a_1 S^{n-1} + a_2 S^{n-2} + \ldots + a_{n-1} S + a_n} \qquad \ldots(5)$$

3) Para obtener la solución general de la ecuación diferencial, se necesita obtener la transformada inversa de Laplace de la función X(S):

$$x(t) = L^{-1}\{X(S)\} = L^{-1}\left\{\frac{Q(S)}{S^n + a_1 S^{n-1} + a_2 S^{n-2} + \ldots + a_{n-1} S + a_n}\right\} \qquad \ldots(6)$$

X(S) es una función racional y se puede expresar como:

$$X(S) = \frac{Q(S)}{P(S)} \qquad \ldots(7)$$

en donde el numerador y el denominador son polinomios en S y el orden de P(S) es mayor que el de Q(S). Debido a que X(S) es una fracción propia, se puede descomponer como la suma algebraica de sus fracciones parciales. Por otra parte, la expansión en fracciones parciales de la función X(S) depende de la naturaleza de las raíces que satisfacen a la siguiente ecuación:

$$P(S) = 0 \qquad \ldots(8)$$

De esta manera se tiene que X(S) se puede expresar como:

$$X(S) = \frac{Q(S)}{(S+P_1)(S+P_2)\ldots(S+P_{n-1})(S-P_n)} \qquad \ldots(9)$$

o bien:

$$X(S) = \frac{Q(S)}{\prod_{i=1}^{n} (S+P_i)} \qquad \ldots(10)$$

Así una fracción propia puede ser expresada como la suma de fracciones parciales de los siguientes tipos:

a) Raíces reales diferentes.

b) Raíces reales iguales.

c) Raíces complejas.

La expansión en fracciones parciales se encuentra explicada en el apartado de expansión en fracciones parciales, y nos da como resultado una expresión de la forma:

$$\frac{K_1 S + K_2}{S^2 + 2\alpha S + \alpha^2 + \omega^2} = \frac{K_1 S + K_2}{(S + \alpha)^2 + \omega^2}$$

ahora desarrollando tenemos:

$$\frac{K_1 S + K_2}{(S + \alpha)^2 + \omega^2} = \frac{K_1 (S + K_2/K_1)}{(S + \alpha)^2 + \omega^2} = \frac{K_1 (S + \alpha + K_2/K_1 - \alpha)}{(S + \alpha)^2 + \omega^2}$$

$$= \frac{K_1 (S + \alpha + (K_2/K_1 - \alpha)\frac{1}{\omega}\,\omega}{(S + \alpha)^2 + \omega^2}$$

$$= K_1 \frac{S + \alpha}{(S + \alpha)^2 + \omega^2} + (K_2 - K_1\alpha)\frac{1}{\omega} - \cdot \frac{\omega}{(S + \alpha)^2 + \omega^2}$$

Lo cual puede ser aplicado directamente en tablas.

A continuación se muestra una tabla de las transformadas de Laplace de algunas funciones empleadas:

$f(t)$	$L(f(t))$
$\delta(t)$	1
te^{mt}	$\frac{1}{(S - m)^2}$
$e^{-\alpha t}\cos\omega t$	$\frac{S + \alpha}{(S + \alpha)^2 + \omega^2}$
$e^{-\alpha t}\operatorname{sen}\omega t$	$\frac{\omega}{(S + \alpha)^2 + \omega^2}$

II.7 EXPANSION EN FRACCIONES PARCIALES.

Las fracciones parciales fueron utilizadas para determinar la transformada zeta de G(S)/S, dentro de la discretización de la función de transferencia de la planta G(S), del sistema H(S) y para la solución del sistema en malla abierta.

A continuación se explica el procedimiento seguido para determinar la expansión en fracciones parciales.

Partiendo de que G(S) puede ser representada como :

$$G(S) = K \frac{Q(S)}{P(S)} \quad ; \qquad K = Cte.$$

donde Q(S) y P(S) son polinomios en S y el orden de P(S) es mayor que el orden de Q(S), esto es :

$$Q(S) = b_0 S^m + b_1 S^{m-1} + \ldots + b_{m-1} S + b_m$$

$$P(S) = a_0 S^n + a_1 S^{n-1} + \ldots + a_{n-1} S + a_n \qquad n > m$$

por lo que Q(S) y P(S) pueden ser representados como :

$$Q(S) = \prod_{j=1}^{m} (S + q_j)$$

$$P(S) = \prod_{i=1}^{n} (S + p_i)$$

Debido a que G(S) es una fracción propia, ésta puede ser expresada como la suma algebraica de sus fracciones parciales. De ésta forma se tiene que G(S) puede ser expresada como:

$$G(S) = \frac{Q(S)}{(S + p_1) + (S + p_2) + \ldots + (S + p_{n-1}) + (S + p_n)}$$

o bien:

$$G(S) = \frac{Q(S)}{\prod_{i=1}^{n} (S + p_i)}$$

Así, se desarrolló en fracciones parciales considerando los siguientes casos :

A) Si P(S) es un polinomio de grado n y posee n raíces reales distintas, éste puede ser escrito de la forma

$P(S) = (S + p_1) + (S + p_2) + \ldots + (S + p_{n-1}) + (S + p_n)$

por lo tanto G(S)/S esta dada por :

$$\frac{G(S)}{S} = \frac{Q(S)}{S(S + p_1) + (S + p_2) + \ldots + (S + p_{n-1}) + (S + p_n)}$$

y su expansión en fracciones parciales es :

$$\frac{G(S)}{S} = \frac{A_0}{S} + \frac{A_1}{(S + p_1)} + \frac{A_2}{(S + p_2)} + \ldots + \frac{A_{n-1}}{(S + p_{n-1})} + \frac{A_n}{(S + p_n)}$$

Donde A_0, A_1, A_2, ..., A_n son constantes por determinar. Esto se logró en el presente trabajo por igualdad de polinomios, esto es

ya que $$\frac{Q(S)}{S\,P(S)} = \frac{A_0}{S} + \frac{A_1}{(S + p_1)} + \ldots + \frac{A_n}{(S + p_n)}$$

y despejando Q(S) :

$$Q(S) = S\,P(S)\left[\frac{A_0}{S} + \frac{A_1}{(S + p_1)} + \ldots + \frac{A_n}{(S + p_n)}\right]$$

Desarrollando el lado izquierdo de esta ecuación e igualando cada coeficiente A con los coefiecientes de Q(S) se obtiene un sistema de ecuaciones que fué resuelto por el método de Eliminación Gaussiana con Substitución Hacia Atras (descrito en el presente trabajo) para encontrar el valor de las constantes A.

B) Si P(S) es un polinomio de grado n y posee n raíces reales iguales, éste puede ser escrito de la forma:

$P(S) = (S + p)^n$

por lo tanto G(S)/S está dada por :

$$\frac{G(S)}{S} = \frac{Q(S)}{S\,(S + P)^n}$$

y su expansión en fracciones parciales es :

$$\frac{G(S)}{S} = \frac{A_0}{S} + \frac{A_1}{(S + p)^n} + \frac{A_2}{(S + p)^{n-1}} + \ldots + \frac{A_n}{(S + p)}$$

Donde A_0, A_1, A_2, ..., A_n son constantes por determinar. Esto se logró en el presente trabajo por igualdad de polinomios, de la misma forma que se describe en el caso A).

C) Si P(S) es un polinomio de grado n (para n par), y posee n/2 factores cuadráticos irreducibles en el campo de los números reales, éste puede ser escrito de la forma :

$$P(S) = (S^2 + a_1 S + b_1) + (S^2 + a_2 S + b_2) + \ldots + (S^n + a_{n/2} S + b_{n/2}).$$

donde, cada factor cuadrático $(S^2 + a_i S + b_i)$ se puede expresar como $(S + \alpha_i)^2 + \beta_i^2$ en donde :

$$\alpha = a_i/2 \quad y \quad \beta_i^2 = b_i - \alpha^2 = b_i - (a_i/2)^2$$

Así, P(S) se puede expresar de la siguiente forma :

$$P(S) = [(S + \alpha_1)^2 + \beta_1^2\,]\,[(S + \alpha_2)^2 + \beta_2^2\,] \ldots [(S + \alpha_{n/2})^2 + \beta^2_{n/2}]$$

Por lo tanto, G(S)/S está dada por:

$$\frac{G(S)}{S} = \frac{Q(S)}{S\,[(S+\alpha_1)^2+\beta^2{}_1]\ \ldots\ [(S+\alpha_{n/2})^2+\beta^2{}_{n/2}]}$$

y su expansión en fracciones parciales es:

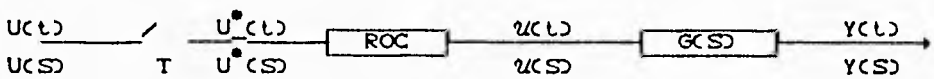
$$\frac{G(S)}{S} = \frac{A_0}{S} + \frac{A_1 S + B_1}{[(S+\alpha_1)^2+\beta^2_1]} + \ldots + \frac{A_{n/2}S + B_{n/2}}{[(S+\alpha_{n/2})^2+\beta^2{}_{n/2}]}$$

donde A_0, A_1, B_1, ... ,$A_{n/2}$, $B_{n/2}$ son constantes por determinar. Esto se logró en el presente trabajo por igualdad de polinomios, de la misma forma que se describe en el caso A).

II.8 DISCRETIZACION DE LA FUNCION DE TRANSFERENCIA DE LA PLANTA G(S) Y DEL SISTEMA H(S).

La discretización se utilizó porque para aplicar el método de diseño, es necesario contar con los equivalentes discretos de la planta G(S) y del sistema H(S). La discretización del sistema depende de si se cuenta, o no con el modelo discreto del sistema.

La discretización tanto de la función de transferencia de la Planta como del Sistema esta basada en el modelo Muestreador/Retén:



ROC : Retén de Orden Cero.

$Y(s) = G(s)\ \mathcal{L}\ (u(t))$

$$u(t) = u_0 u_{-1}(t) - u_0 u_{-1}(t-T) + u(t)u_1(t-T) - u(T)u(t-2T) + u(2T)u_1(t-2T) - u(2T)u(t-3T) + u(3t)u_1(t-3T) - u(3T)u(t-4T) + \ldots$$

$$\mathcal{L}\{u(t)\} = \frac{u(0)}{s} - \frac{u(0)}{s}e^{-sT} + \frac{u(T)}{s}e^{-sT} - \frac{u(T)}{s}e^{-2sT} + \frac{u(2T)}{s}e^{-2sT} - \frac{u(2T)}{s}e^{-3sT} + \frac{u(3T)}{s}e^{-3sT} - \frac{u(3T)}{s}e^{-4sT} + \ldots$$

$$= \frac{1}{s}\left[u(0) - u(0)e^{-sT} + u(T)e^{-sT} - u(T)e^{-2sT} + u(2T)e^{-3sT} + u(3T)e^{-3sT} + u(4T)e^{-4sT} + \ldots \right]$$

$$= \frac{1}{s}\left[u(0)(1-e^{-sT}) + u(T)(e^{-sT}-e^{-2sT}) + u(2T)(e^{-2sT}-e^{-3sT}) + u(3T)(e^{-3sT}-e^{-4sT}) + \ldots \right]$$

$$= \frac{1}{s}\left[u(0)(1-e^{-sT}) + u(T)e^{-sT}(1-e^{-sT}) + u(2T)e^{-2sT}(1-e^{-3sT}) + u(3T)e^{-3sT}(1-e^{-4sT}) + \ldots \right]$$

$$= \frac{1-e^{-sT}}{s}\left[u(0) + u(T)e^{-sT} + u(2T)e^{-2sT} + u(3T)e^{-3sT} + \ldots \right]$$

Donde $\frac{1-e^{-sT}}{s}$ es la Transformada de Laplace del retén de orden cero.

$$Y(s) = G(s)\frac{1-e^{-sT}}{s}\left[u(0) + u(T)e^{-sT} + u(2T)e^{-2sT} + u(3T)e^{-3sT} + \ldots \right]$$

$$Y(Z) = \mathcal{Z}\left\{\frac{G(S)}{S}\right\}[1-Z^{-1}]\,[\,U(0) + U(T)Z^{-1} + U(2T)Z^{-2} + U(3T)Z^{-3} + \ldots]$$

$$Y(Z) = [1-Z^{-1}]\,\mathcal{Z}\left\{\frac{G(S)}{S}\right\}[\Sigma_{K=0}\,U(KT)Z^{-K}\,]$$

como $U(Z) = [\Sigma_{K=0}\,U(KT)Z^{-K}\,]$

$$Y(Z) = [1-Z^{-1}]\,\mathcal{Z}\left\{\frac{G(S)}{S}\right\}U(Z)$$

por lo tanto :

$$\frac{Y(Z)}{U(Z)} = G(Z) = [1-Z^{-1}]\,\mathcal{Z}\left\{\frac{G(S)}{S}\right\}$$

A ésta forma de discretizar se le conoce como Respuesta Pulso Invariante.

Para determinar la transformada $\mathcal{Z}$ de $G(S)/S$ se utilizó el método de Expansión en Fracciones Parciales, que ya fue descrito, y los distintos casos que se consideraron fueron:

A) Si $P(S)$ es un polinomio de grado n y posee n raíces reales distintas.

La transformada $\mathcal{Z}$ es :

$$\mathcal{Z}\left\{\frac{K}{S}\right\} = \frac{KZ}{Z-1}$$

$$\mathcal{Z}\left\{\frac{K}{S+p}\right\} = \frac{KZ}{Z-e^{-pT}}$$

B) Si $P(S)$ es un polinomio de grado n y posee n raíces reales iguales.

La transformada $\mathcal{Z}$ correspondiente es:

$$\mathcal{Z}\left\{\frac{K}{S}\right\} = \frac{KZ}{Z-1}$$

$$\mathcal{Z}\left\{\frac{K}{(S+p)^n}\right\} = \frac{T}{n!}\frac{Bn(z)}{(Z-e^{-pT})^n}$$

Donde:

$$Bn(z) = b_1^n z^{n-1} + b_2^n z^{n-2} + \ldots + b_n^n$$

Y:

$$b_k^k = \sum_{i=1}^{k} (-1)^{k-i} i^n \binom{n+1}{k-1}, \qquad k = 1,2,3, \ldots, n.$$

C) Si P(S) es un polinomio de grado n (para n par), y posee n/2 factores cuadráticos irreducibles en el campo de los números reales.

La transformada $\mathcal{Z}$ correspondiente es:

$$\mathcal{Z}\left\{\frac{K}{S}\right\} = \frac{KZ}{Z-1}$$

$$\mathcal{Z}\left\{\frac{K_1S + K_2}{[(S+\alpha_1)^2 + \beta_2^2]}\right\} = \mathcal{Z}\left\{\frac{K_1(S + K_2/K_1)}{[(S+\alpha_1)^2 + \beta_2^2]}\right\} =$$

$$\mathcal{Z}\left\{\frac{K_1(S + \alpha + K_2/K_1 + \alpha)}{[(S+\alpha_1)^2 + \beta_2^2]}\right\} = \mathcal{Z}\left\{\frac{K_1[(S+\alpha) + ((K_2/K_1 - \alpha)/\beta)\beta]}{[(S+\alpha_1)^2 + \beta_2^2]}\right\} =$$

$$\mathcal{Z}\left\{K_1\frac{S+\alpha}{[(S+\alpha_1)^2 + \beta_2^2]} + K_1[((K_2/K_1)-\alpha)/\beta]\frac{\beta}{[(S+\alpha_1)^2 + \beta^2]}\right\}$$

Por lo tanto :

$$\mathcal{Z}\left\{\frac{K_1S + K_2}{[(S+\alpha_1)^2 + \beta^2]}\right\} = \frac{K_1(Z(Z - r\cos\phi)) + K'Zr\,\mathrm{Sen}\phi}{Z^2 - 2r\cos\phi\, Z + r^2}$$

Donde : $K' = K_1[((K_2/K_1)-\alpha))/\beta]$

$r = e^{-\alpha T}$

$\cos\phi = \cos\beta T$

$\mathrm{Sen}\phi = \mathrm{Sen}\beta T$

II.9 DISEÑO DE CONTROLADORES BASADOS EN LA ASIGNACION DE POLOS Y CEROS.

El método de diseño utilizado en la presente tesis es: Diseño de controladores basados en la asignación de polos y ceros, escrito por el profesor K. J. Astrom y el Doctor B. Wittenmark. El método empieza con un diseño para sistemas con parámetros conocidos, sustituye los parámetros del modelo conocido del sistema por estimados, los cuales son obtenidos recursivamente y recalcula los parámetros de control en cada paso. Los controladores obtenidos pueden ser usados para ajustar lazos de control cuando los parámetros o el sistema controlado son desconocidos, o lentamente variantes en el tiempo. Se asume que la fuente principal de distorsiones son cambios en el valor de referencia u.

METODO PARA ASIGNACION DE POLOS Y CEROS.

Una breve revisión del método de asignación de polos y ceros para sistemas con parámetros conocidos será dada aquí.

Notación.

Los sistemas y controladores son descritos usando una representación polinomial. La siguiente notación es usada:

$$A(q^{-1}) = a_0 + a_1 q^{-1} + \ldots + a_{n_a} q^{-n_a}, \qquad a_0 \neq 0$$

donde q^{-1} es el operador de corrimiento atrazado. Si $a_0 = 1$ el polinomio está normalizado. El grado de un polinomio $A(q^{-1})$ es escrito como grado A ó n_a.

La entrada, salida y señales de control del proceso son denotadas como $u(t)$, $y(t)$ y $u_c(t)$, respectivamente, y $v(t)$ es una distorsión. Z es una región fuera del circulo unitario. Si los ceros de un polinomio pertenecen a Z esto implica que las funciones correspondientes son suficientemente bien amortiguadas. Esta región es llamada la región de estabilidad restringida.

Formulación.

Considerando un proceso caracterizado por el operador racional

$$G(q^{-1}) = \frac{q^{-k}B(q^{-1})}{A(q^{-1})} \qquad (1)$$

es asumido que A y B son coprimos, que A está normalizada, y que el retraso en el proceso es tal que

$$k \geq 1. \qquad (2)$$

Se desea encontrar un controlador tal que el lazo cerrado sea estable y que la función de transferencia de la entrada de control u_c a la salida este dada por

$$G_m(q^{-1}) = \frac{q^{-k}B_m(q^{-1})}{A_m(q^{-1})} \qquad (3)$$

donde A_m y B_m son coprimos y A_m está normalizado. Las raíces de A_m se asumen dentro de Z.

Se asume que en la función de lazo propuesta el retraso de tiempo es al menos tan grande como el de la ecuación 1.

Procedimiento de diseño.

Un controlador general lineal puede ser descrito por

$$R(q^{-1})u(t) = T(q^{-1})u_c(t) - S(q^{-1})y(t) \qquad (4)$$

La función de transferencia de lazo cerrado desde y hasta u_c está dada por:

$$\frac{q^{-k}TB}{AR+q^{-k}BS} = \frac{q^{-k}B_m}{A_m} \qquad (5)$$

donde el lado derecho es la función de transferencia deseada de lazo cerrado G_m dada por la ecuación 3.

El problema de diseño es así equivalente al problema algebraico de encontrar polinomios R, S y T tales que la ecuación 5 se mantenga. Siguiendo de la ecuación 5, aquellos factores de B que

no sean también factores de B_m deben ser factores de R. Como los factores de B corresponden a ceros de lazo abierto ello significa que los ceros de lazo abierto que no son ceros deseados de lazo cerrado deben ser cancelados. Factorizar B como:

$$B = B^+B^- \qquad (6)$$

donde todos los ceros de B^+ están en la región de estabilidad restringida Z y todos los ceros de B^- fuera de Z. Esto significa que todos los ceros de B^+ corresponden a funciones bastante amortiguadas, y todos los ceros de B^- corresponden a funciones inestables o pobremente amortiguadas.

Una condición necesaria para resolver el problema es tal que las especificaciones son:

$$B_m = B_{m_1}B^- \qquad (7)$$

Como el grado de A_m es normalmente menor que el grado de $AR + q^{-k}BS$ es claro que hay factores en la ecuación 5 que se cancelan. Puede ser demostrado que el controlador de la ecuación 4 corresponde a una combinación de un observador y un estado realimentado. Es natural asumir que el observador es diseñado de tal forma que cambios en las señales de control no generan errores en él mismo. Esto significa que el factor que se cancela en el lado derecho de la ecuación 5 puede ser interpretado como el observador polinomial A_o. Es asumido que todos los ceros de A_o están en la región de estabilidad restringida Z.

Un diagrama de bloques del sistema de lazo cerrado es mostrado en la fig. 1. El controlador puede ser interpretado como si estuviera compuesto de una ruta de realimentación adelantada con la función de transferencia T/R y una ruta de retroalimentación con la función de transferencia $-S/R$.

El método de diseño puede ser descrito como sigue.

Datos.

Dado un modelo matemático (1) de el proceso caracterizado por

los polinomios A y B, la respuesta deseada (3) está caracterizada por los polinomios A_m y B_m y el deseado observador polinomial A_o. Se asume que los datos satisfacen las condiciones de las ecuaciones 2, 6, y 7 y todos los ceros de A_o están dentro de Z.

Paso 1.

Resolver la ecuación

$$AR_1 + q^{-k}B^-S = A_mA_o \qquad (8)$$

con respecto a R_1 y S.

Paso 2.

El controlador que da la respuesta deseada de lazo cerrado está dado por la ecuación 4 con

$$R = R_1B^+ \qquad (9)$$

y $$T = A_oB_{m_1} \qquad (10)$$

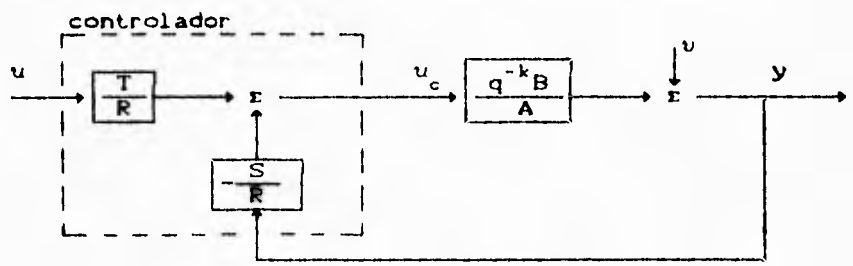


Figura 1.

La ecuación 8 puede siempre ser resuelta porque fué asumido que A y B eran coprimos. Esto implica por supuesto que A y B^- también son coprimos.

La ecuación 8 tiene infinidad de soluciones. Si R_1^o y S^o, son soluciones, entonces

$$R_1 = R_1^0 + B^- U$$

$$S = S^0 - AU$$

donde U que es un polinomio arbitrario, también es una solución. Todas las soluciones darán un sistema de lazo cerrado con la deseada función de transferencia de lazo cerrado G_m. Las diferentes soluciones darán, sin embargo, sistemas con diferentes propiedades de rechazo al ruido. La función de transferencia de la distorsión v actuando en la salida del proceso a la salida (ver fig. 1) está dada por

$$\frac{AR}{AR + q^k \; BS} = \frac{AR}{A_m A_o B^+}$$

Las soluciones particulares usadas para los controladores auto-ajustables son tales que las funciones de transferencia S/R y T/R son causales, sin retraso extra. Las siguientes son algunas soluciones:

$$\text{grado } S = \text{grado } A - 1$$
$$\text{grado } R_1 = \text{grado } A_m + \text{grado } A_o - \text{grado } A \qquad (11)$$

o

$$\text{grado } S = \text{grado } A_m + \text{grado } A_m = \text{grado } B^- - k$$
$$\text{grado } R_1 = \text{grado } B^- + k - 1 \qquad (12)$$

El caso de la ecuación 11 corresponde a compensación "integral", y el caso de la ecuación 12 corresponde a compensación "derivativa". Hay, sin embargo, muchas otras posibilidades.

Para fines del presente trabajo se optó por la compensación integral.

Interpretación del modelo.

El controlador de la ecuación 4 puede ser interpretado como un modelo. De las ecuaciones 8, 9 y 10 se tiene que:

$$\frac{T}{R_1} = \frac{A_0 B_m}{B^+ R_1} = \frac{(AR_1 + q^{-k} B^- S) B_m}{A_m B^+ R_1}$$

$$= \frac{AB_m}{B^+ A_m} + \frac{q^{-k} S B^- B_m}{B^+ R_1 A_m} = \frac{A}{B} \ast \frac{B_m}{A_m} + \frac{q^{-k} S}{R} \ast \frac{B_m}{A_m}$$

La ley de retroalimentación de la ecuación 4 puede así ser escrita como:

$$u(t) = \frac{A}{B} y_c(t + k) + \frac{S}{R} [y_c(t) - y(t)] \qquad (13)$$

donde:

$$y_c(t) = \frac{q^{-k} B_m}{A_m} u_c(t)$$

La señal y_c puede ser interpretada como la salida obtenida cuando la señal de control u_c es aplicada al modelo $q^{-k}B_m/A_m$. Cuando el controlador de la ecuación 4 es reescrito como la ecuación 13, es claro que puede ser pensado como compuesto de dos partes, un término de alimentación adelantada $A/By_c(t + k)$ y un término de retroalimentación $(S/R)(y_c(t) - y(t))$. La alimentación adelantada es una combinación de el modelo ideal y un inverso del modelo del proceso. El término de retroalimentación es obtenido alimentando el error $y_c - y$ através de un sistema dinámico caracterizado por el operador S/R. La liga entre ajuste de polos y el diseño del modelo es así establecido. Nótese, sin embargo, que el sistema $q^k A/B$ no es realizable, aunque la combinación $AB_m/(BA_m)$ si lo es.

Casos especiales.

Para llevar a cabo el diseño, es necesario tener procedimientos para descomponer un polinomio B en sus factores B^+ y B^-, y para resolver la ecuación polinomial lineal 8. La descomposición es esencialmente un problema de factorización espectral. La ecuación 8 puede ser resuelta usando eliminación Gaussiana o usando el algoritmo de Euclid. En el presente trabajo se optó por la utilización del método de eliminación Gaussiana. En los algoritmos adaptativos, estos cálculos tienen que ser repetidos

en cada paso de la iteración. Entonces es de interés ver si hay algunos casos especiales donde los cálculos de diseño puedan ser simplificados. Existen dos casos especiales donde el problema de descomposición es evitado, los cuales se describen a continuación.

Ejemplo 1: (todos los ceros del proceso son cancelados).

Asumir que todos los ceros del proceso son cancelados y que no son introducidos ceros adicionales. Además, asumir que el grado A_m = grado A y el grado A_o = grado A - 1. La ecuación 8 entonces se reduce a

$$AR_1 + q^{-k}S = A_m A_o \qquad (14)$$

y el controlador está dado entonces por la ecuación 4 con

$$R = R_1 B$$

$$B_m = B_{m_1} = K$$

$$T = A_o$$

donde K es una constante. Notar que B aparece como un factor de R, el cual es el denominador de la función de transferencia del controlador. También notar que las especificaciones son normalmente tales que la función de transferencia deseada de lazo cerrado tiene ganancia unitaria a bajas frecuencias. Esto significa que los polinomios A_m y B_m deberían ser normalizados tales que $B_m(1)/A_m(1) = 1$. Como A está normalizado, los polinomios q^{-k} y A son siempre primos. La ecuación 14 puede así siempre ser resuelta. Las soluciones correspondientes a

$$\text{grado } S = \text{grado } A - 1$$
$$\text{grado } R_1 = \text{grado } A_m + \text{grado } A_o - \text{grado } A \qquad (15)$$

ó

$$\text{grado } S = \text{grado } A_m + \text{grado } A_o - k$$
$$\text{grado } R_1 = k - 1 \qquad (16)$$

son elegidas. En el caso especial del ejemplo 1, los cálculos de diseño se reducen así a la solución de la ecuación 14. Notar que la ecuación 14 es fácil de resolver para el caso de la ecuación 16.

Los coeficientes de los polinomios R_1 y S pueden entonces ser obtenidos uno a la vez.

Ejemplo 2: (no son cancelados los ceros del proceso).

Asumir que las especificaciones son tales que los ceros deseados de lazo cerrado son iguales a los ceros del proceso, por ejemplo $B_m = KB$, donde K es una constante. Las especificaciones son normalmente tales que la ganancia de baja frecuencia del deseado lazo cerrado es unitaria. El factor constante del polinomio B_m es entonces escogido tal que $B_m(1) = A_m(1)$. Se asume que ésta normalización es hecha. La ecuación 8 entonces da

$$AR + q^{-k}BS = A_m A_o \qquad (17)$$

El procedimiento de diseño es usado otra vez para escoger los polinomios A_m y A_o. La ecuación 17 es entonces resuelta con respecto a R y S, y el controlador es entonces dado por la ecuación 4, donde $T = KA_o$.

II.10 GRAFICACION.

Para realizar la graficación, primero se detecta el tipo de monitor con el que se está trabajando, para lo cual se utiliza la función INITGRAPH de Turbo Pascal. A continuación es necesario obtener la máxima resolución del monitor, es decir las máximas coordenadas en X y Y para lo que se utilizaron las funciones GetmaxX y GetmaxY de Turbo Pascal.

Para desplegar las gráficas en la pantalla es necesario establecer una relación entre los puntos de la pantalla y la magnitud total de la gráfica, obteniendo :

$$\frac{\text{No. de ptos. de la pantalla}}{\text{Magnitud total de la gráfica}} = \frac{\text{X Pixel}}{\text{X Magnitud}}$$

Despejando X Pixel obtenemos la relación entre cada pixel y la magnitud a ser graficada :

$$\text{X Pixel} = \text{X Magnitud} * \frac{\text{No. de ptos. de la pantalla}}{\text{Magnitud total de la gráfica}}$$

Donde X Pixel es el pixel a ser pintado en la pantalla,
X Magnitud es el valor real del punto a ser desplegado.

Al número de puntos en la pantalla entre la magnitud total de la gráfica le llamaremos factor de escala con el fin de multiplicar cada magnitud real por éste y obtener el pixel correspondiente en pantalla.

De acuerdo a lo anterior definimos los siguientes factores de escala :

Escala Y = (Máxima coordenada en Y - 39) / (Pto. max. - Pto. min.)
Escala X = (Máxima coordenada en X - 89) / Numero de Ptos.

Las constantes 39 y 89 son el número de pixeles utilizados para el despliegue de los ejes y acotaciones.

La escala X se dividió entre el número de puntos pues ya fueron seleccionados los puntos a desplegarse.

Para el caso de graficación múltiple, debido a que pueden ser graficadas para distintos tiempos, es necesario establecer una relación entre los tiempos de cada gráfica y el tiempo máximo de despliegue, obteniendo el siguiente factor de escala en X:

Escala X = TGA * (MCX - 89)/TMAX/NP

donde: TGA = tiempo de la gráfica actual a desplegar
MCX = máxima coordenada en X
TMAX = tiempo de análisis máximo
NP = número de puntos.

II.II TIEMPO DE ANALISIS

El sistema tiene la capacidad de autocalcular el tiempo de análisis cuando los polos del sistema son estables, tomando la frecuencia natural del sistema dada por m. Donde m es el polo con la parte real negativa más cercana al origen (m < 0). El inverso de esta frecuencia nos da el tiempo en que la respuesta permanente alcanza el 63.2 % de su valor final. Para fines prácticos este factor es multiplicado por 8 con el fin de observar hasta un 95 % de su valor final.

$$\text{tiempo de análisis} = 8 \left(\frac{1}{m} \right)$$

Es importante señalar que el paquete permite modificar el tiempo de análisis si el usuario así lo desea, con lo cual podrá visualizar la respuesta según su conveniencia.

CAPITULO III.
DESARROLLO DEL PAQUETE.

III.1 ANALISIS DEL SISTEMA.

En el desarrollo de un sistema que debe de cumplir con más de una función especifica, debe realizarse un análisis detallado de cada una de ésas funciones, pero sin perder de vista la idea general. Con éste fin se adoptó la técnica Top Down para el análisis y diseño del paquete.

La técnica Top Down se basa en analizar un problema yendo de lo general a lo particular, teniendo primeramente una perspectiva general del problema y posteriormente bajando de nivel hasta llegar a niveles especificos del análisis.

De ésta manera, el análisis para llevar a cabo la programación del paquete, partió de la idea general de construir un sistema para el diseño y simulación de controladores digitales que realizara las siguientes tres funciones principales :

- Análisis de sistemas en malla abierta.
- Análisis de sistemas en malla cerrada con controlador.
- Diseño de controladores.

De ésta manera el análisis se dividió en tres módulos correspondientes a las tres funciones principales del paquete.

Bajando de nivel en el análisis de cada uno de los módulos, se realizó un estudio detallado para determinar las funciones matemáticas, funciones de sistemas dinámicos, y las funciones de control, así como los métodos numéricos a desarrollar en cada uno.

En todo el análisis desarrollado siempre se bajó de nivel y se profundizó tanto como fué necesario para su programación, puesto que aún en el desarrollo de la programación se presentan imprevistos que obligan a un análisis todavía más profundo.

En las figuras V.1, V.2, V.3 y V.4 se presentan los diagramas de flujo de las funciones principales, ya mencionadas, con las que cuenta el paquete.

Como parte del análisis realizado, se presenta a continuación el pseudocódico desarrollado para la elaboración del paquete.

III.2 SEUDOCODIGO.

```
PROGRAM DSC;
BEGIN
   REPEAT
      DESPLEGAR MENU PRINCIPAL
      LEER (OPCION)
      CASE OPCION OF
         1 : TIPO DE SISTEMA
         2 : MENU RESULTADOS
         3 : SALVAR DATOS
         4 : LEER DATOS
         5 : DATOS ACTUALES
         6 : SALIDA = TRUE
      END CASE;
   UNTIL SALIDA = TRUE;
END.
```

```
PROCEDURE TIPO DE SISTEMA;
BEGIN
   DESPLEGAR MENU TIPO SISTEMA
   LEER (OPCION)
   CASE OPCION OF
      1 : BEGIN
             SISTEMA = PLANTA
             LEER FUNCION DE TRANSFERENCIA (PLANTA)
             LEER TIEMPO DE ANALISIS
             LEER ENTRADA
             PROCESAR DATOS
      2 : BEGIN
             SISTEMA = CONTROLADOR
             LEER FUNCION DE TRANSFERENCIA (PLANTA)
             LEER TIEMPO DE ANALISIS
             LEER FUNCION DE TRANSFERENCIA (CONTROLADOR)
             LEER GANANCIA DE LAZO
             LEER PERIODO DE MUESTREO
             LEER (ENTRADA)
             PROCESAR DATOS
          END
      3 : BEGIN
             SISTEMA = DISEÑO
             LEER FUNCION DE TRANSFERENCIA (PLANTA)
             LEER TIEMPO DE ANALISIS
             LEER FUNCION DE TRANSFERENCIA (LAZO)
             LEER PERIODO DE MUESTREO
             LEER (ENTRADA)
             PROCESAR DATOS
          END
   END CASE;
END.
```

```
PROCEDURE PROCESAR DATOS;
BEGIN
   CASE SISTEMA OF
      PLANTA : BEGIN
                  CALCULAR RAICES
                  DESARROLLO FRACCIONES PARCIALES
                  ANTITRANSFORMADA DE LAPLACE
               END
      CONTROLADOR : BEGIN
                       WHILE TIEMPO < TIEMPO DE ANALISIS
                          BEGIN
                             RESOLVER ECUACION DIFERENCIA 1
                             FROM I = 1 TO 10 DO
                                RESOLVER RUNGE-KUTTA
                          END
                    END
      DISEÑO : BEGIN
                  CALCULA RAICES
                  DISCRETIZA (PLANTA)
                  IF SISTEMA DISCRETO = FALSE THEN
                     DISCRETIZA (SISTEMA);
                  DISEÑO CONTROLADOR
                  WHILE TIEMPO < TIEMPO DE ANALISIS
                     BEGIN
                        RESOLVER ECUACION DIFERENCIA 1
                        RESOLVER ECUACION DIFERENCIA 2
                        FOR I = 1 TO 10 DO
                           RESOLVER RUNGE-KUTTA;
                     END
               END
   END CASE;
END.
```

DIAGRAMA DE FLUJO DE DATOS

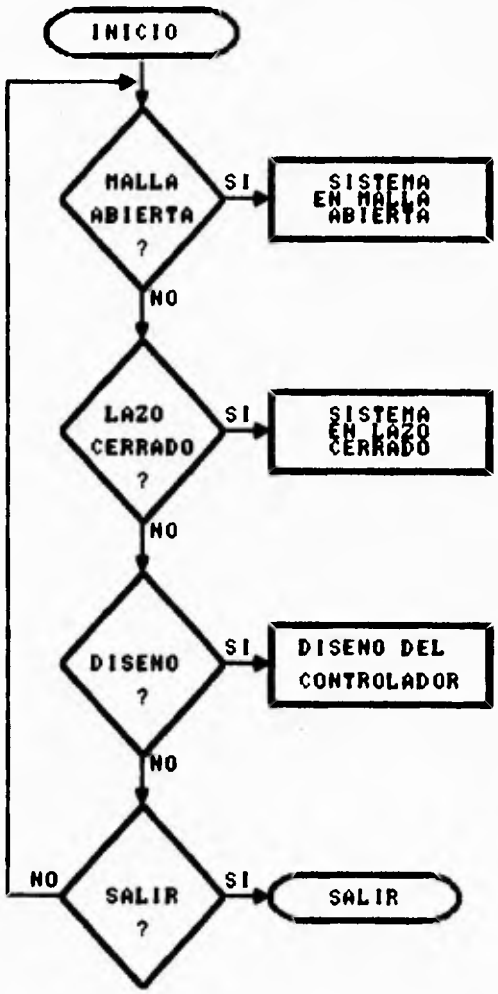


FIGURA V.1

SISTEMA EN MALLA ABIERTA

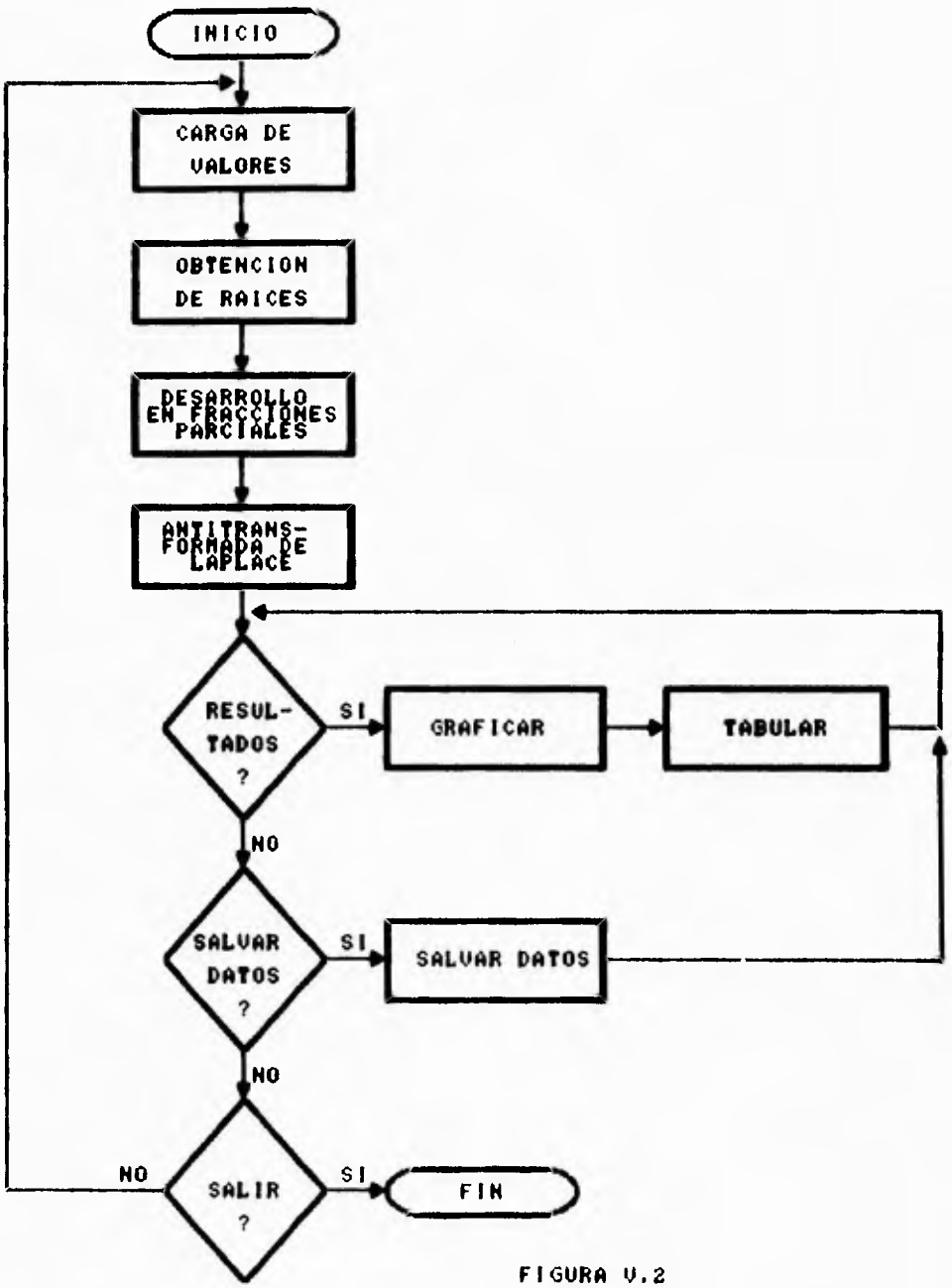


FIGURA V.2

SISTEMA EN LAZO CERRADO

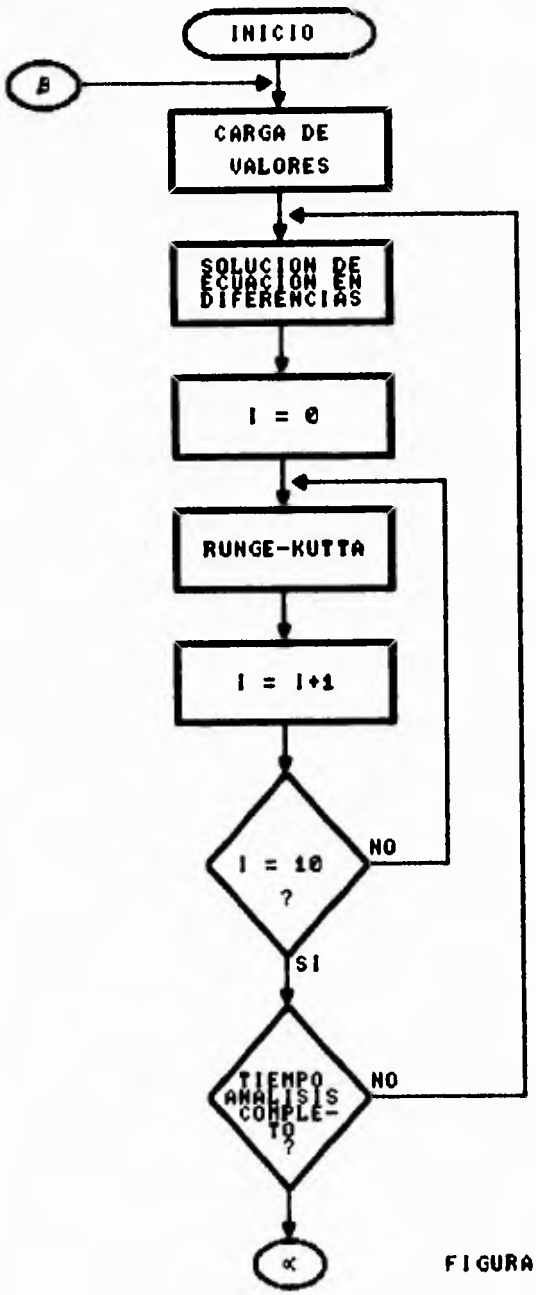
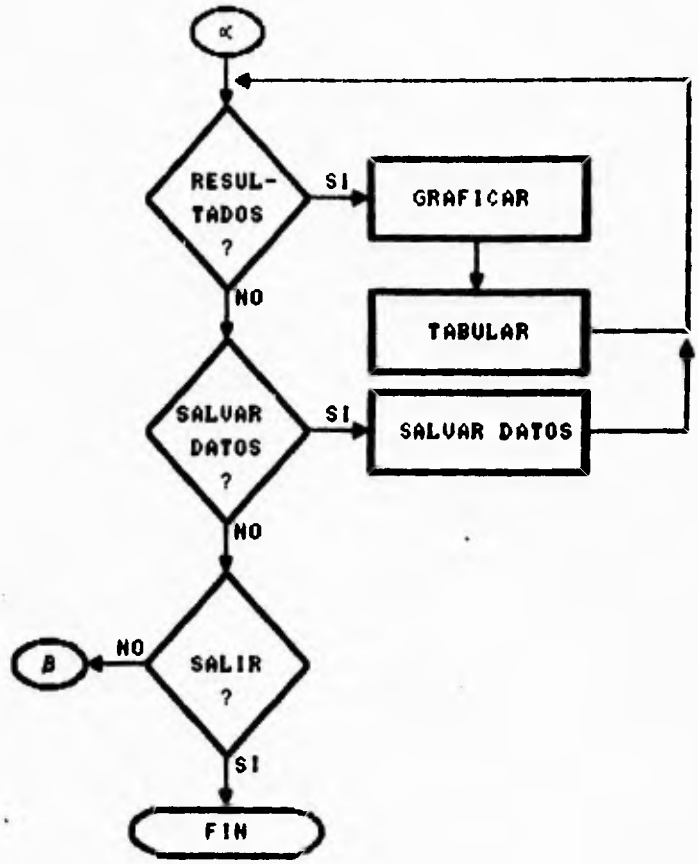


FIGURA V.3


∝
RESUL-
TADOS
?
SI
GRAFICAR
TABULAR
NO
SALVAR
DATOS
?
SI
SALVAR DATOS
NO
B
NO
SALIR
?
SI
FIN

DISENO

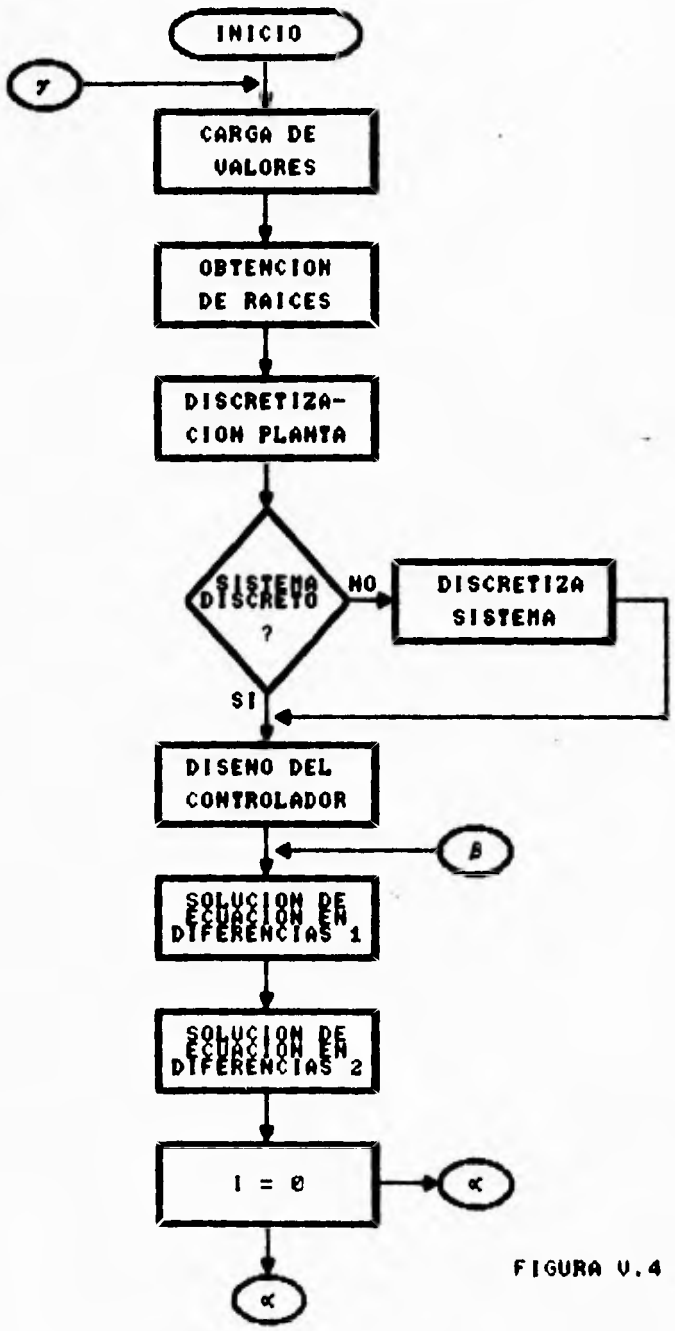
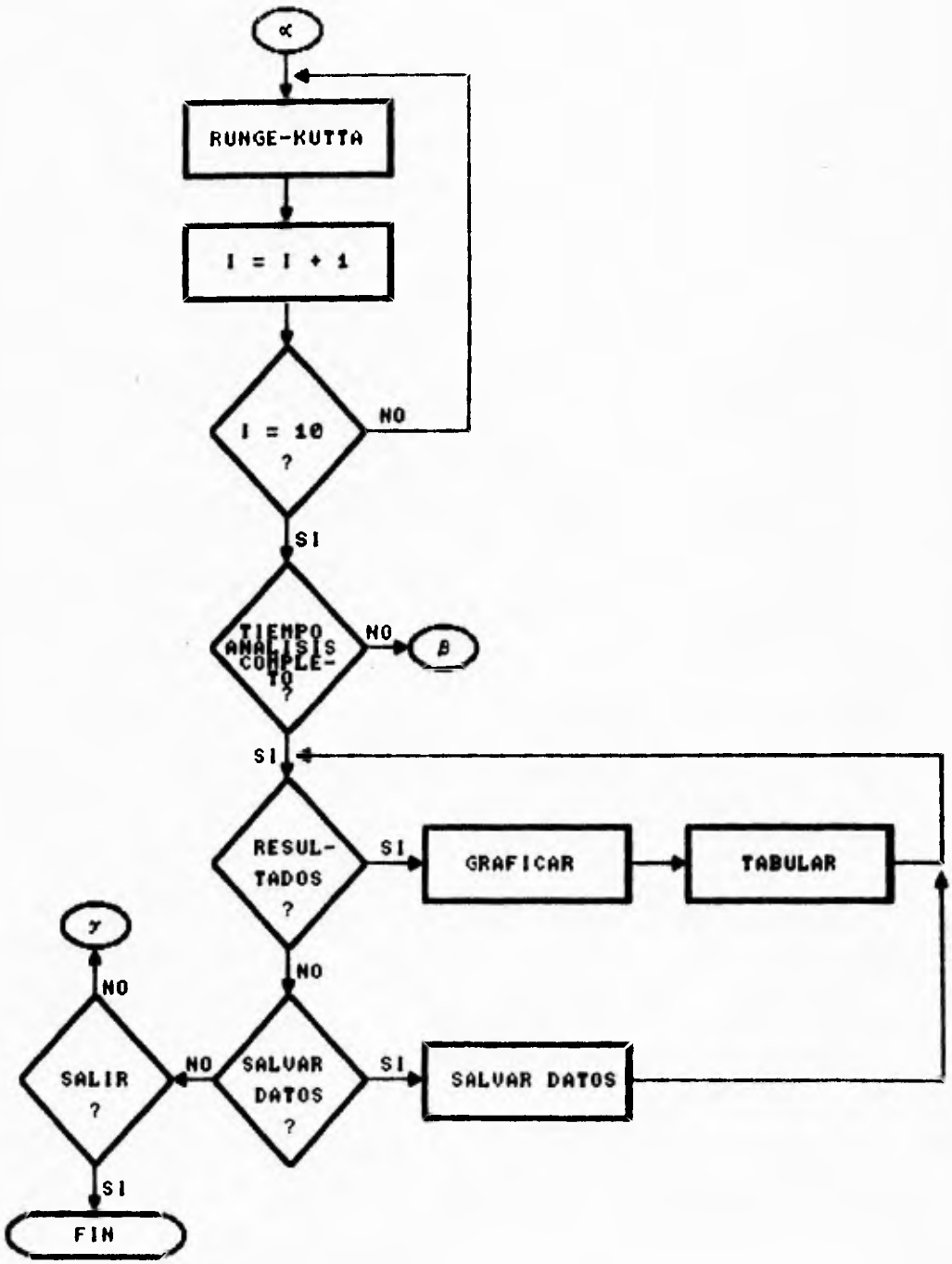


FIGURA V.4


α
RUNGE-KUTTA
I = I + 1
I = 10 ?
NO
SI
TIEMPO ANALISIS COMPLE-TO ?
NO
β
SI
RESUL-TADOS ?
SI
GRAFICAR
TABULAR
NO
γ
NO
SALIR ?
NO
SALVAR DATOS ?
SI
SALVAR DATOS
SI
FIN

Nota: En el pseudocódigo presentado se detalló hasta invocaciones a algoritmos y métodos numéricos, explicados a detalle en el capitulo de algoritmos y métodos numéricos presentado en este trabajo, por lo que no se incluyó el pseudocódigo de éstos. Tampoco fue incluido el pseudocódigo de pantallas, lectura de datos, almacenamiento de datos y despliegue de resultados, puesto que estos procedimientos imvolucran funciones especificas del lenguaje utilizado en la programación.

III.3 DESCRIPCION DEL CODIGO.

El código del paquete está hecho en Turbo Pascal 5.5 por ser éste el lenguaje sobre el que se tenia más dominio en el momento do iniciar el presente trabajo, así como por sus caracteristicas propias para el manejo de funciones matemáticas.

Para su mejor entendimiento y manejo, el código del paquete está dividido en unidades. En Turbo Pascal una unidad es un programa que puede funcionar de manera independiente o bien ser invocado desde otro programa para realizar funciones especificas. Así, una unidad puede invocar a otra unidad.

El paquete está dividido en las siguientes siete unidades: DECLARA.PAS, DISENO.PAS, ARCHIVOS.PAS, MENSAJES.PAS, LAZOPL.PAS, GRAFICAR.PAS, y DATOSAC.PAS las cuales son invocadas desde el programa principal DSC.PAS, además se hace uso de las unidades propias de Turbo Pascal 5.5: DOS, CRT y GRAPH.

A continuación se describe la función de cada una de las unidades y del programa principal:

- **DECLARA.PAS** : Esta unidad contiene la declaración de todas las variables globales empleadas en el paquete.

- **DISENO.PAS** : Esta unidad realiza el diseño del controlador así como la discretización de las funciones de transferencia de la planta y del sistema.

- **ARCHIVOS.PAS** : Esta unidad se encarga del almacenamiento y recuperación de los archivos de datos y de gráficas.

- **MENSAJES.PAS** : Esta unidad contiene todos los mensajes desplegados durante la ejecución del paquete.

- **LAZOPL.PAS** : Esta unidad lleva a cabo la antitransformada de Laplace de la función de transferencia de la planta y la evaluación de la misma en el tiempo.

- **GRAFICAR.PAS** : Esta unidad realiza la graficación de las diferentes salidas de los sistemas analizados dentro del paquete, así como el despliegue múltiple de gráficas en la pantalla.

- **DATOSAC.PAS** : Esta unidad se encarga de la presentación de los datos que se están manejando en la sesión, ya sea que hayan sido capturados o recuperados de disco.

- **PROGRAMA PRINCIPAL DSC.PAS** : Es el encargado de coordinar la ejecución de las unidades, despliegue de pantallas de menús y de captura de datos, así como pantallas de despliegue de resultados y presentación de mensajes al usuario.

En la programación del sistema se utilizaron varias estructuras de datos para el manejo de la información, tales como registros, arreglos de registros, pilas y colas doblemente ligadas.

Para el funcionamiento del sistema fue necesario el manejo de una gran cantidad de ecuaciones, suma, resta, multiplicación y división de polinomios, asi como encontrar las raices (reales y complejas) de polinomios. También fue necesario solucionar sistemas de ecuaciones y ecuaciones en diferencias asi como el manejo de matrices.

Para el manejo de polinomios se utilizaron arreglos unidimensionales, arreglos bidimensionales (matrices) y arreglos de registros.

Fue necesario el almacenamiento temporal de un volumen elevado de resultados, por ejemplo los puntos calculados para cada señal, para esto se utilizaron estructuras de datos del tipo colas doblemente ligadas a través de apuntadores. Estas estructuras de datos se crean al ser requeridas y se destruyen al dejar de ser necesarias. Esto se llevó a cabo con el fin de poder destruir las estructuras una vez que fué utilizada la información para evitar así la saturación de la memoria de la computadora y hacer más eficiente su uso.

III.4 ESTRUCTURA GENERAL DEL PAQUETE.

En el presente capitulo se presenta la forma en que está estructurado el paquete desarrollado. Esta presentación se dará, para un mejor entendimiento, en forma de navegación dentro del mismo.

El paquete está organizado en menús y submenús de captura y presentación de datos, asi como de despliegue de resultados.

La navegación dentro del paquete se inicia desde el Menú Principal el cual presenta las siguientes opciones :

- Cargar datos.
- Análisis de resultados.
- Salvar datos.
- Lectura de datos de disco.
- Consulta de datos actuales.
- Salir

Ver figura V.5.

Al elegir la opción de Cargar Datos se pasa a otra pantalla en donde se presentan las diferentes opciones para el análisis de sistemas de control con que cuenta el paquete : (Ver figura V.6)

- Análisis de la planta (G(S)) en malla abierta.
- Análisis de un sistema con controlador (G(Z)) en lazo cerrado.
- Diseño del controlador a partir de H(Z).
- Cambios en los parámetros.

Al elegir cualquiera de éstas opciones se pasa a la pantalla de Carga de Datos (Ver fig V.7) en donde se presentan los diferentes datos a capturar dependiendo del sistema a analizar :

- Datos de la planta.
- Tiempo a analizar.
- Datos del controlador.
- Ganancia de lazo.
- Periodo de muestreo.
- Datos del lazo (H(Z)).
- Tipo de entrada.

Cada uno de los datos a capturar mencionados en esta pantalla llevan a una pantalla particular en la que se realiza la captura formal de cada dato requerido para el análisis a realizar.

Una vez terminada la captura de los datos y se ha llevado a

cabo el procesamiento de los mismos, se puede elegir desde el menú principal la opción de Análisis de Resultados que nos lleva a una pantalla de menú con las siguientes opciones : (Ver figura V.8)

- Graficar.
- Tabular.
- Salvar gráficas.
- Leer gráficas.
- Controlador.

La opción de Graficar presenta una pantalla en donde se elige la señal a graficar y posteriormente se presenta la gráfica de la señal deseada.

La opción de Tabular presenta una pantalla en donde se elige la señal a tabular, pasando despues a una pantalla en donde se presenta la tabulación de los datos de la señal elegida.

Con la opción de Salvar Gráficas no se pasa a otra pantalla, sino que en ésta misma se realiza solamente la captura del nombre del archivo que contendrá los datos de las gráficas optenidas.

En la opción Leer Gráficas, en ésta misma pantalla se realiza la captura del número de gráficas a leer y posteriormente se pasa a otra pantalla en donde se elige la(s) grafica(s) a recuperar de disco.

Con la opción de Controlador se pasa a otra pantalla en donde se presentan los datos del controlador diseñado.

Regresando al Menú principal, al elegir la opción de Salvar Datos, no se pasa a otra pantalla, sino que en ésta misma se realiza la petición y captura del nombre del archivo que contendrá toda la

cabo el procesamiento de los mismos, se puede elegir desde el menú principal la opción de Análisis de Resultados que nos lleva a una pantalla de menú con las siguientes opciones : (Ver figura V.8)

- Graficar.
- Tabular.
- Salvar gráficas.
- Leer gráficas.
- Controlador.

La opción de Graficar presenta una pantalla en donde se elige la señal a graficar y posteriormente se presenta la gráfica de la señal deseada.

La opción de Tabular presenta una pantalla en donde se elige la señal a tabular, pasando despues a una pantalla en donde se presenta la tabulación de los datos de la señal elegida.

Con la opción de Salvar Gráficas no se pasa a otra pantalla, sino que en ésta misma se realiza solamente la captura del nombre del archivo que contendrá los datos de las gráficas optenidas.

En la opción Leer Gráficas, en ésta misma pantalla se realiza la captura del número de gráficas a leer y posteriormente se pasa a otra pantalla en donde se elige la(s) grafica(s) a recuperar de disco.

Con la opción de Controlador se pasa a otra pantalla en donde se presentan los datos del controlador diseñado.

Regresando al Menú principal, al elegir la opción de Salvar Datos, no se pasa a otra pantalla, sino que en ésta misma se realiza la petición y captura del nombre del archivo que contendrá toda la

información del sistema que se esté analizando. Ver figura V.5.

La opción de Consulta de datos actuales del Menú Principal nos lleva a otra pantalla en donde se despliegan los datos del sistema que se está analizando.

Finalmente, con la opción de Salir se abandona el paquete.

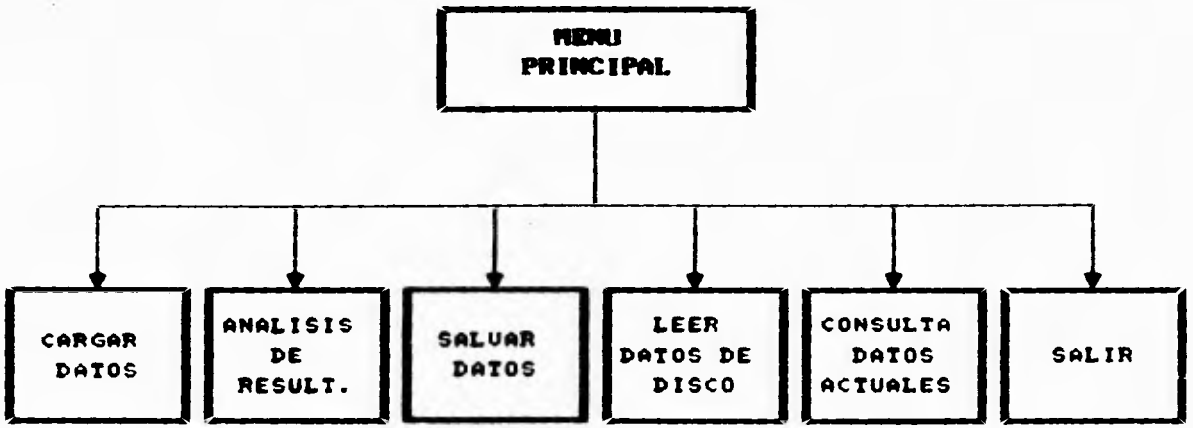

MENU
PRINCIPAL
CARGAR
DATOS
ANALISIS
DE
RESULT.
SALUAR
DATOS
LEER
DATOS DE
DISCO
CONSULTA
DATOS
ACTUALES
SALIR

FIGURA U.5

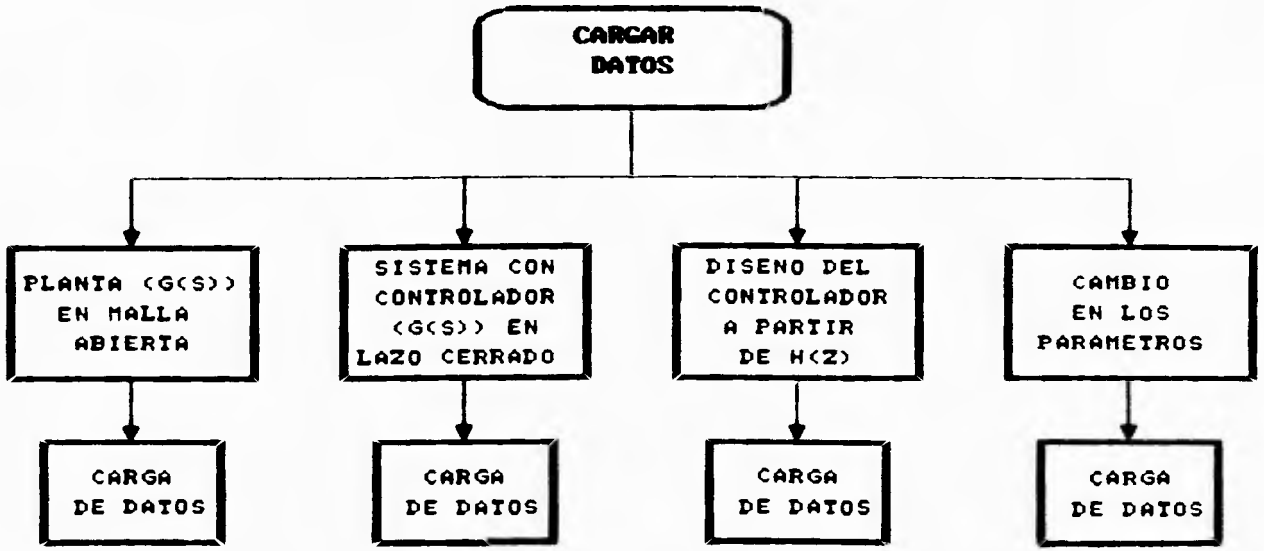

CARGAR DATOS
PLANTA (G(S)) EN MALLA ABIERTA
SISTEMA CON CONTROLADOR (G(S)) EN LAZO CERRADO
DISENO DEL CONTROLADOR A PARTIR DE H(Z)
CAMBIO EN LOS PARAMETROS
CARGA DE DATOS
CARGA DE DATOS
CARGA DE DATOS
CARGA DE DATOS

FIGURA U.6

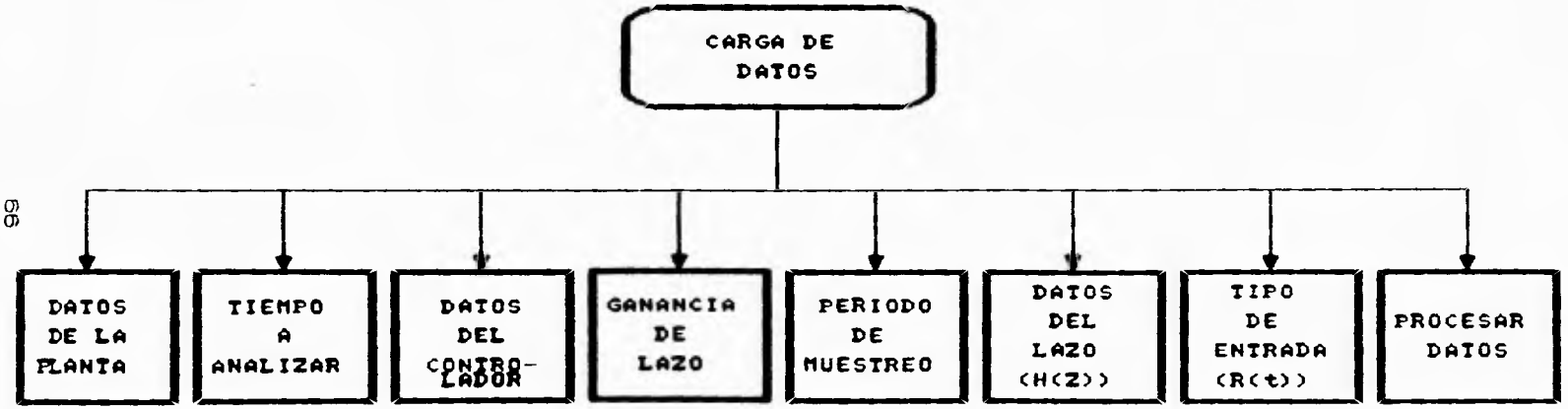

CARGA DE DATOS
DATOS DE LA PLANTA
TIEMPO A ANALIZAR
DATOS DEL CONTRO-LADOR
GANANCIA DE LAZO
PERIODO DE MUESTREO
DATOS DEL LAZO (H(Z))
TIPO DE ENTRADA (R(t))
PROCESAR DATOS

FIGURA U.7

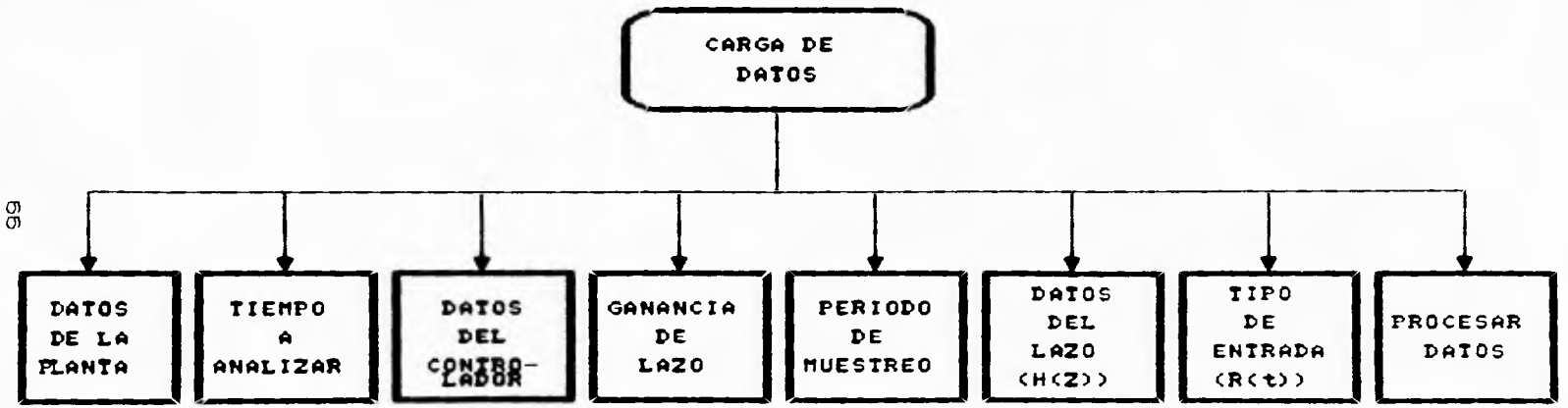

CARGA DE DATOS
DATOS DE LA PLANTA
TIEMPO A ANALIZAR
DATOS DEL CONTRO-LADOR
GANANCIA DE LAZO
PERIODO DE MUESTREO
DATOS DEL LAZO (H(Z))
TIPO DE ENTRADA (R(t))
PROCESAR DATOS

FIGURA U.7

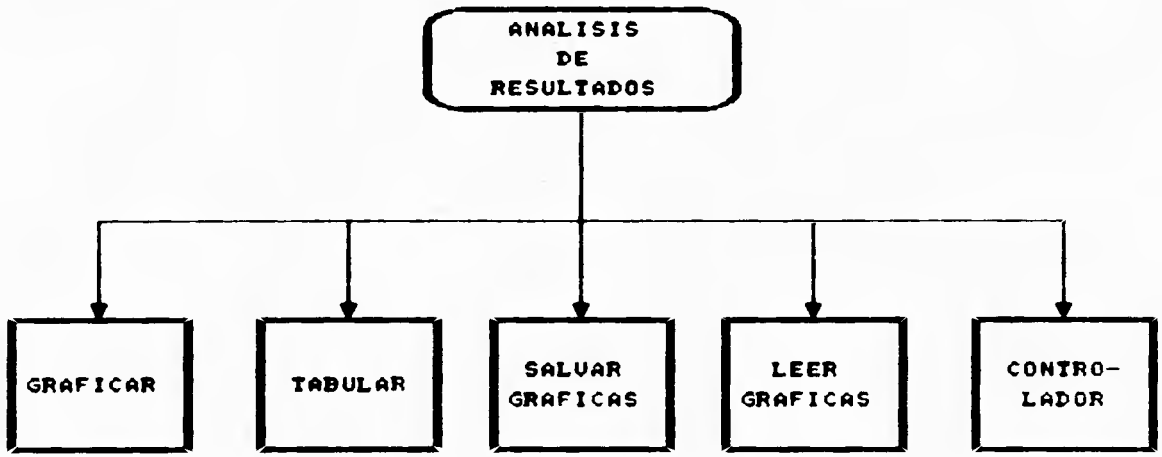

ANALISIS
DE
RESULTADOS
GRAFICAR
TABULAR
SALVAR
GRAFICAS
LEER
GRAFICAS
CONTRO-
LADOR

FIGURA V.8

CAPITULO IV
PRUEBAS A NIVEL MODULO.

El paquete está dividido en tres módulos principales:

1) Sistema en malla abierta.

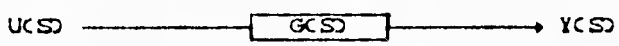
2) Sistema en malla cerrada con controlador.

3) Diseño del controlador por el método de ubicación de polos.

A continuación se describen los ejemplos aplicados a cada módulo.

1) Pruebas en malla abierta.

Todas las pruebas realizadas en éste módulo tuvieron como entrada un impulso $\delta(t)$.



a) Sistema sobreamortiguado (polos con raices reales negativas diferentes).

$$G(s) = \frac{s+4}{(s+3)(s+2)}$$

Ver figura 5.1

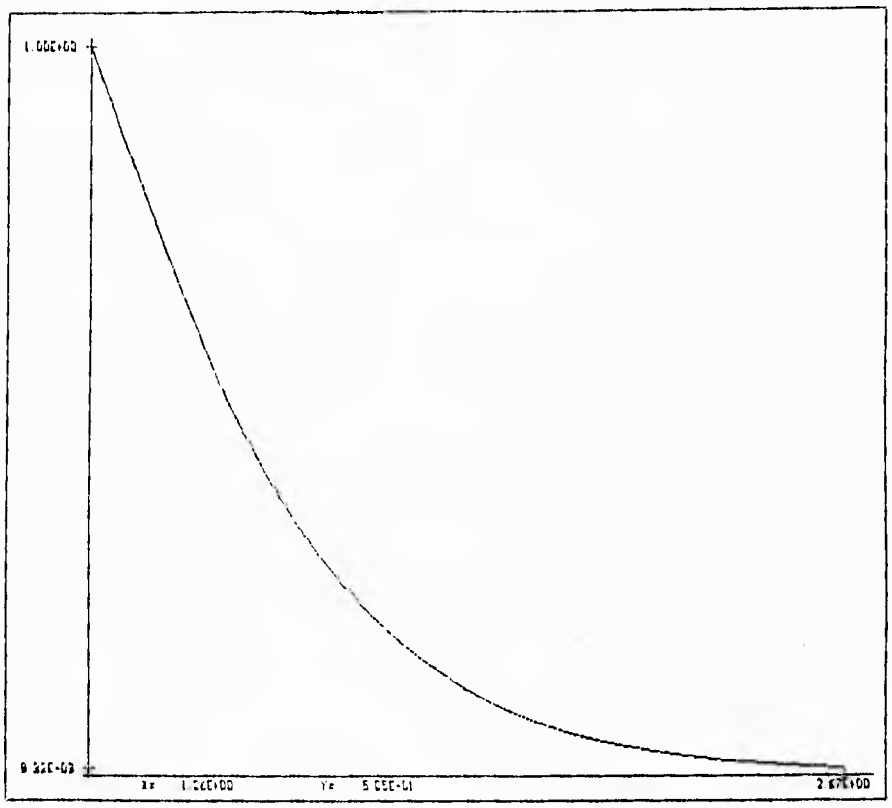


Figura 5.1

b) Sistema criticamente amortiguado (polos con raíces reales repetidas).

$$G(s) = \frac{s+5}{(s+2)(s+2)}$$

Ver figura 5.2

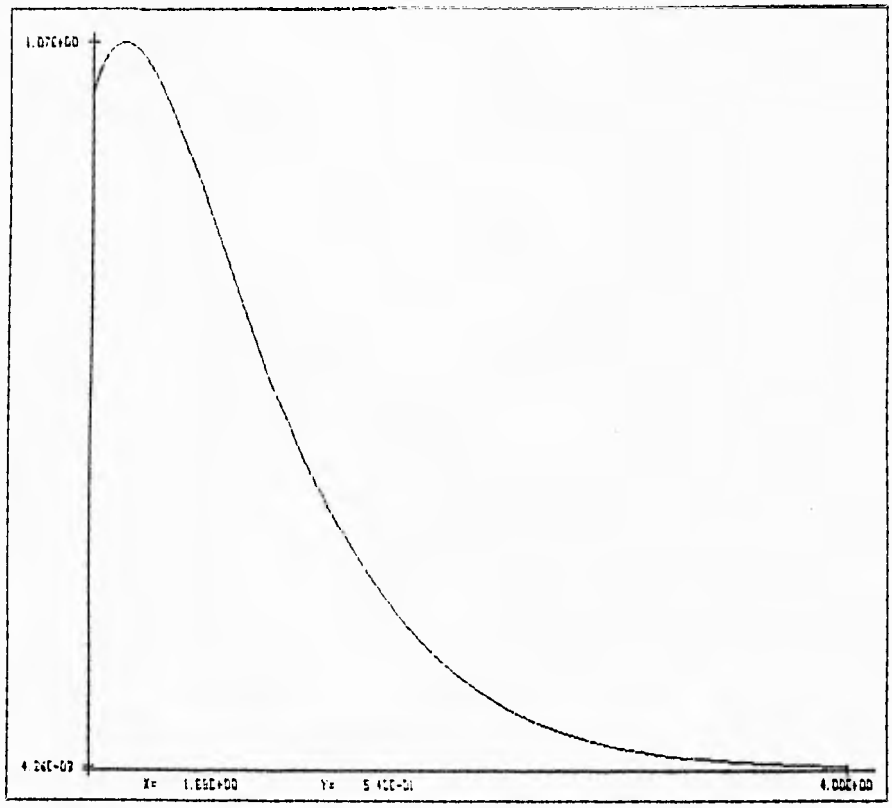


Figura 5.2

c) Sistema subamortiguado (polos con raíces complejas y parte real negativa).

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

Ver figura 5.3

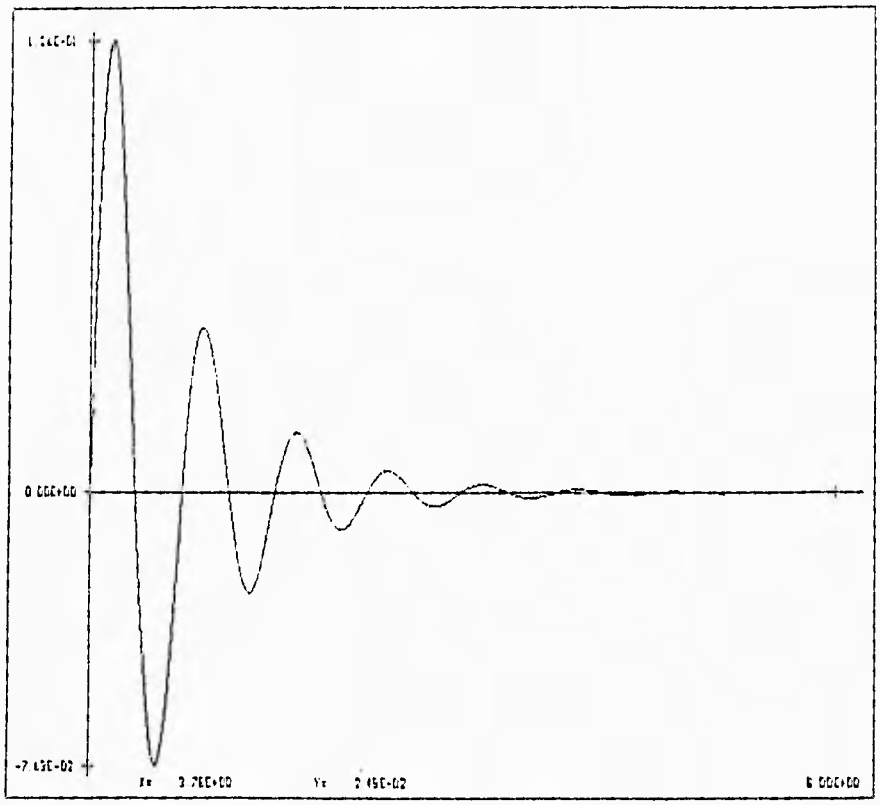


Figura 5.3

d) Sistema no amortiguado (polos con raíces complejas y parte real nula).

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

Tiempo a analizar = 10 seg.

Ver figura 5.4

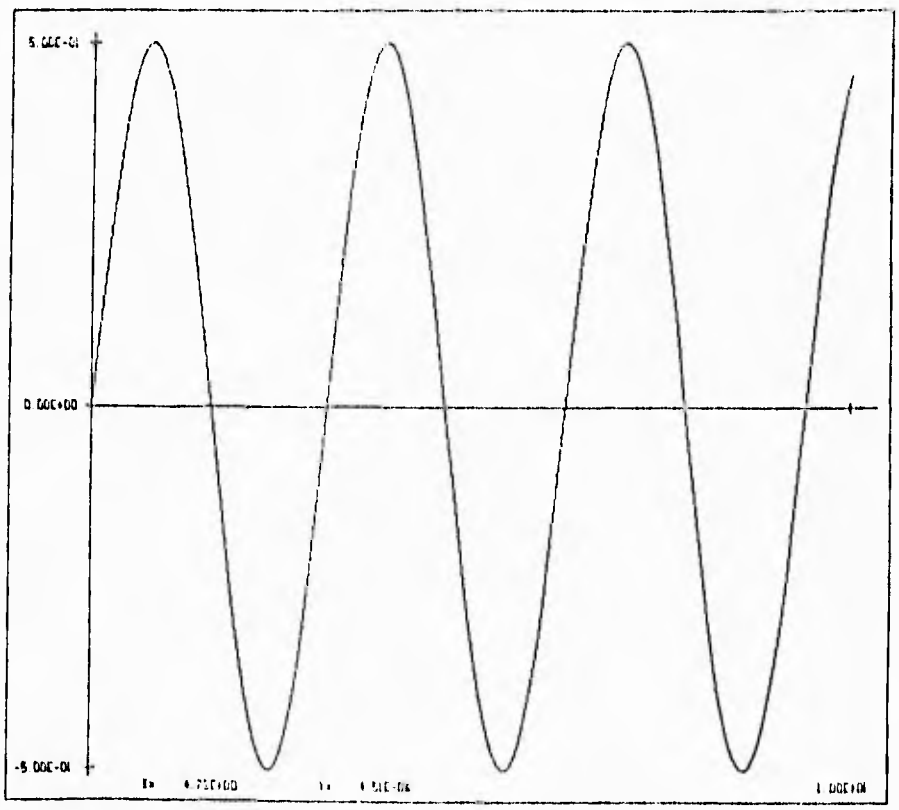


Figura 5.4

2) Pruebas en malla cerrada con controlador.

Datos de entrada al paquete :

U(t) = 1 (escalón unitario)

$$G(s) = \frac{0.1}{s(s+0.1)}$$

$$G_c(Z) = \frac{13.0682Z^2 - 12.8605Z + 0.93789}{Z^2 + 0.54918Z - 0.4043}$$

Controlador obtenido por el método directo de Ragazzini.
Ganancia de lazo = 1
Período de muestreo = 1 seg.
Tiempo a analizar = 40 seg.

Ver figuras 5.5, 5.6 y 5.7

3) Pruebas para el diseño del controlador por el método de ubicación de polos.

Datos de entrada al paquete :

U(t) = 1 (escalón unitario)

$$G(s) = \frac{0.1}{s(s+0.1)}$$

$$H(Z) = \frac{0.63212Z - 0.05014}{Z^2 - 0.7859Z + 0.3679}$$

Período de muestreo = 1 seg.
Tiempo a analizar = 80 seg.

Polinomios obtenidos del diseño del controlador:

R(Z) = Z + 0.047
S(Z) = 1.1189Z - 0.5369
T(Z) = 0.6321Z - 0.0501

Donde:

$$G_1(Z) = \frac{T(Z)}{R(Z)}, \qquad G_2(Z) = \frac{S(Z)}{R(Z)}$$

Ver figuras 5.8 y 5.9

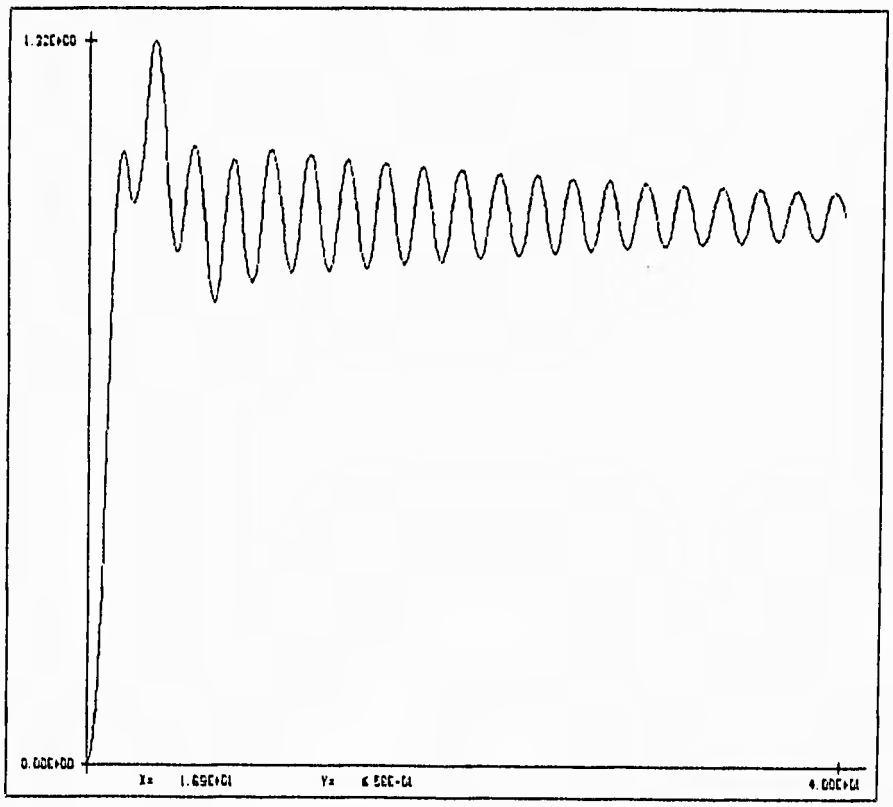


Figura 5.5

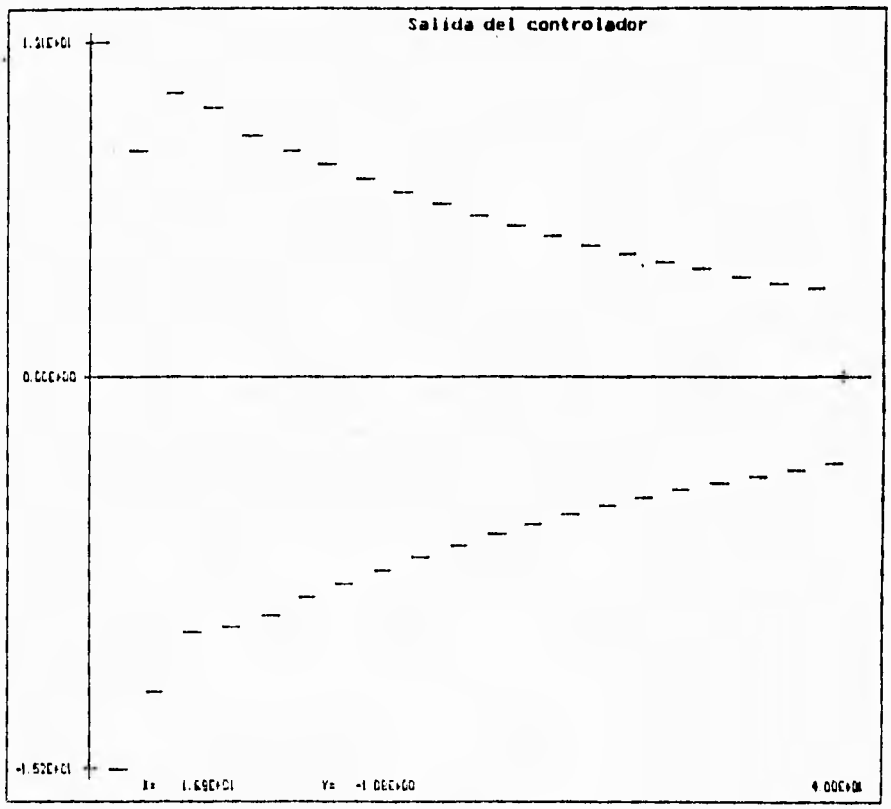

Salida del controlador
1.31E+01
0.00E+00
-1.52E+01
X= 1.69E+01
Y= -1.00E+00
4.00E+00

Figura 5.6

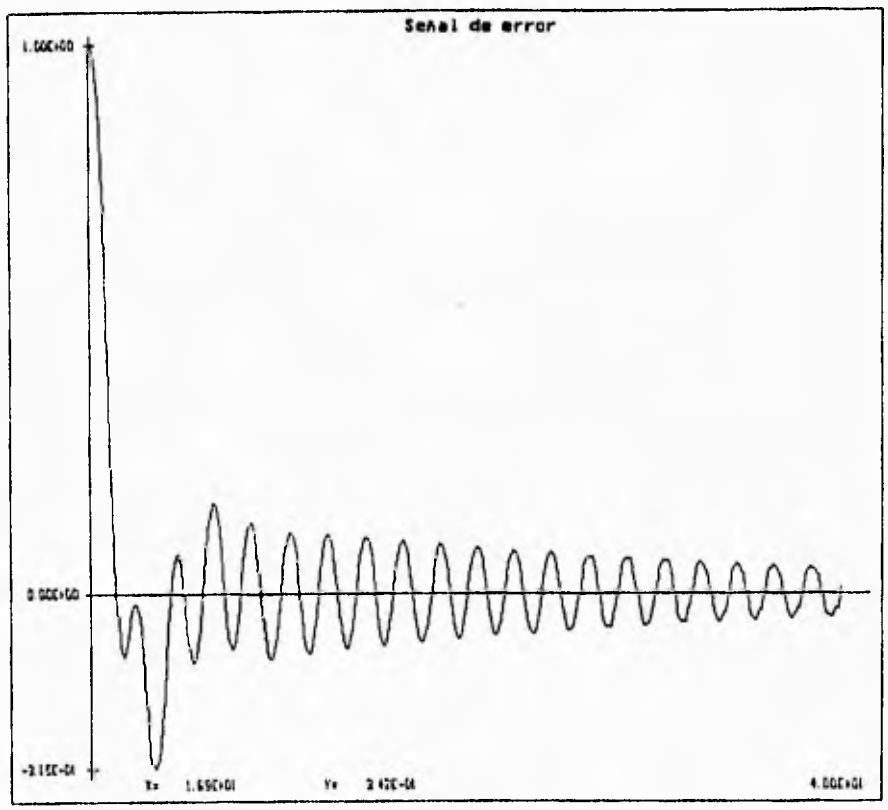

Señal de error
1.00E+00
0.00E+00
-2.15E-04
X= 1.65E+01
Y= 2.42E-04
4.00E+01

Figura 5.7

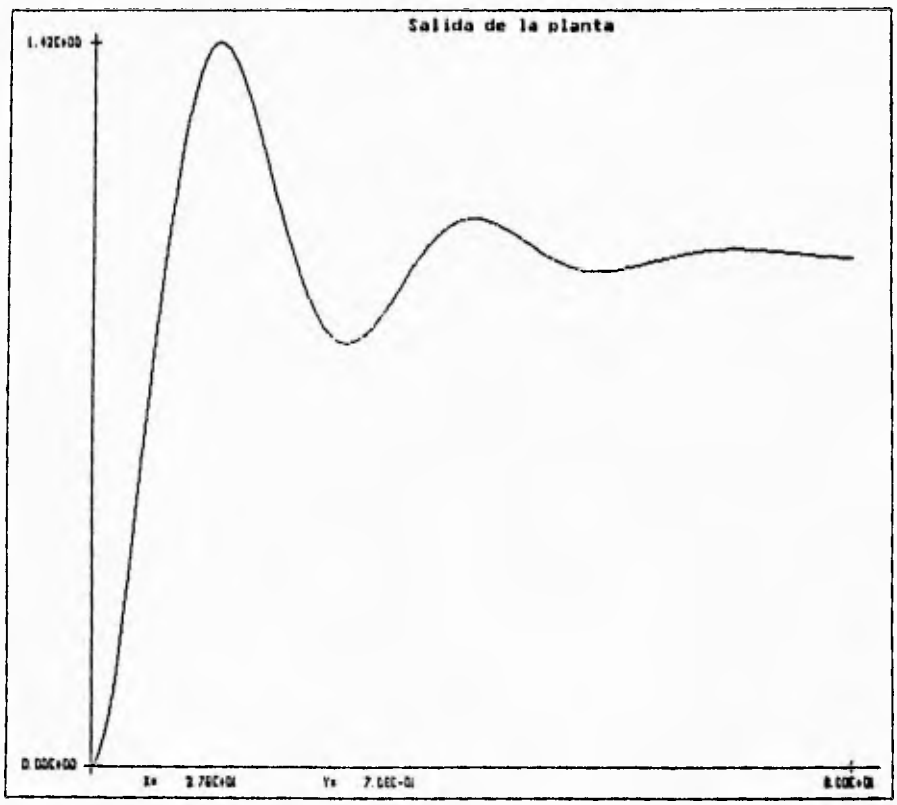


Figura 5.8

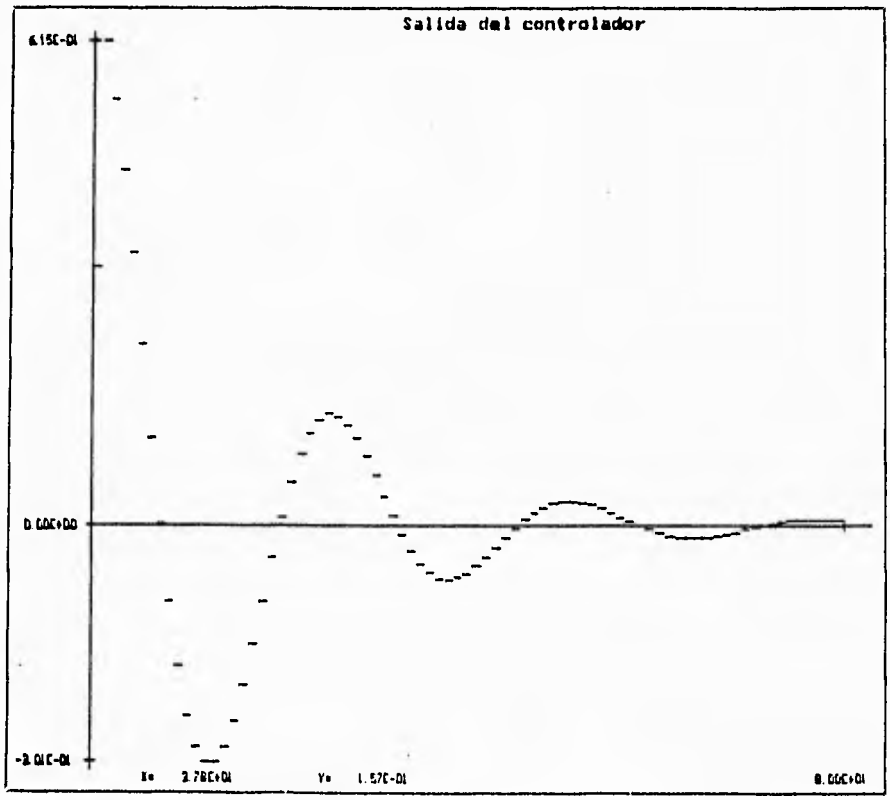

Salida del controlador
6.15E-01
0.00E+00
-8.01E-01
X= 2.78E+01
Y= 1.57E-01
8.00E+01

Figura 5.9

CAPITULO V
MANUAL DE USUARIO

El paquete está diseñado para resolver :

A) Sistemas en malla abierta con una entrada y una planta proporcionadas por el usuario.

B) Sistemas en malla cerrada con una entrada, una planta y un controlador proporcionados por el usuario.

C) Diseño de un controlador por el método de ubicación de polos a partir de una planta y la función de transferencia del sistema proporcionados por el usuario.

Descripción del funcionamiento del paquete.

La navegación y potición de datos dentro del paquete se realiza a través de menus. A continuación se realizará una descripción detallada de los mismos.

Menú Principal. A través de éste se pueden realizar las funciones de carga de datos, análisis de resultados, salvar datos, leer datos de disco, consulta de datos presentes y abandonar el paquete. La figura 6.1 muestra el menú principal.

Menú principal

1) Cargar datos
2) Análisis de resultados
3) Salvar datos
4) Leer datos de disco
5) Consulta datos actuales
6) Salir

Figura 6.1

Menú de Carga de Datos. A través de éste se pueden seleccionar los tipos de sistemas que se pueden procesar así como la opción de cambio de parámetros que permite la modificación de cualquier dato introducido previamente. La figura 6.2 muestra el menú de Carga de Datos.

Tipo de sistema

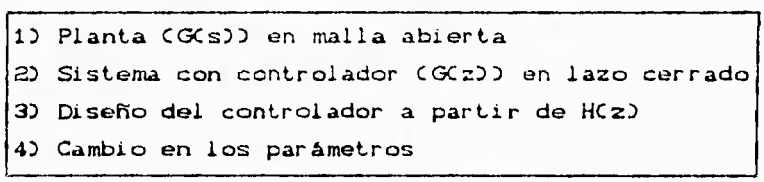
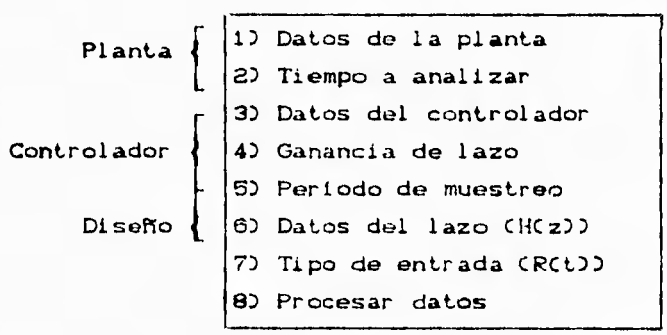


Figura 6.2

La introducción y procesamiento de los datos se realiza a través del submenú de carga de datos mostrado en la figura 6.3.

Carga de datos



Opciones seleccionadas: 1 2 3 4 5 7

Figura 6.3

A través de la opción 1 se puede introducir la función de transferencia de la planta dando los coeficientes del polinomio numerador y denominador o bien por sus respectivas raíces como se muestra en la figura 6.4. En el caso de existir polos inestables, el sistema pedirá el tiempo a analizar, de lo contrario el paquete calculará un tiempo estimado de análisis marcando la opción 2) como seleccionada.

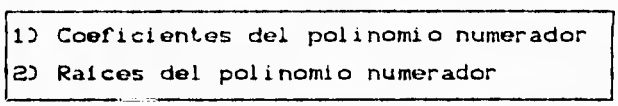


Figura 6.4

Con la opción 2) se puede introducir el tiempo a analizar en segundos como se muestra en la figura 6.5

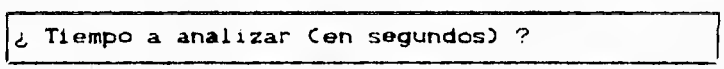


Figura 6.5

En la opción 3) se introducirá la función de transferencia del controlador de manera similar a la opción 1).

La opción 4) permite introducir la ganancia de lazo (sensor) para sistemas en malla cerrada con controlador como se ilustra en la figura 6.6.

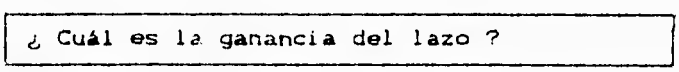


Figura 6.6

A través de la opción 5) se introduce el período de muestreo requerido en sistemas en malla cerrada ó en el diseño del controlador. Ver figura 6.7.

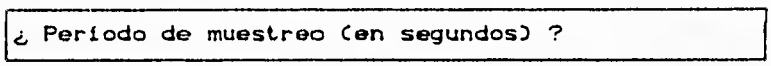


Figura 6.7

Con la opción 6) se introduce la función de transferencia del sistema H(Z) de la misma forma que en las opciones 1) y 3), necesaria para el diseño del controlador.

La opción 7) permite la selección de los diferentes tipos de entrada que pueden alimentarse al sistema, como se muestra en la figura 6.8. Debe señalarse que la entrada impulso δ(t) sólo se puede seleccionar para sistemas en malla abierta.

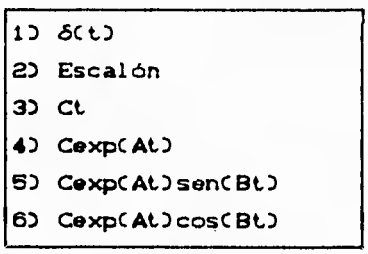


Figura 6.8

El procesamiento de los datos se realiza seleccionando la opción 8).

En la parte inferior de la pantalla se muestran las opciones que ya han sido seleccionadas, y sólo se podrá procesar la información cuando se cuente con todos los datos requeridos para cada tipo de análisis.

Menú de Análisis de Resultados. En este menú se puede realizar el análisis gráfico y tabular de los sistemas procesados. Además se pueden consultar las funciones de transferencia del controlador diseñado, los datos presentes en el análisis así como salvar y recuperar gráficas. Ver figura 6.9.

Menú de resultados

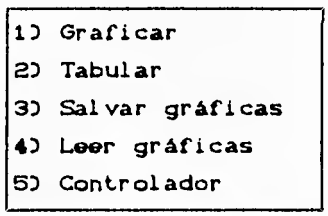


Figura 6.9

Submenú de gráficas. A través de este se puede seleccionar el graficar cualquiera de las señales que se muestran en la figura 6.10.

Las teclas de cursor (↑, ↓, ←, →), permiten desplazarse en las gráficas en modo pixel por pixel. Y para desplazamientos rápidos, debe activarse la tecla [Numlock] y desplazarse con las teclas numéricas 2, 4, 6 y 8.

Graficación

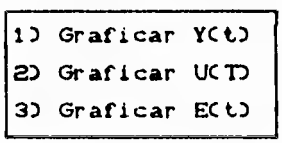


Figura 6.10

Submenú de Tabulación. Permite tabular cualquiera de las señales que se muestran en la figura 6.11. Para desplazarse en la

tabulación se pueden usar las teclas de cursor ↑, ↓, ←, →, o bien las teclas [PgUp] y [PgDn]. Ver figura 6.12.

Tabulación

1) Tabular Y(t)
2) Tabular U(T)
3) Tabular E(t)

Figura 6.11

Salida del sistema

t	Salida(t)
0.000	1.0000000
0.100	0.9934442
0.200	0.9736889
0.300	0.9406015
0.400	0.8940480
0.500	0.8338931
0.600	0.7600001
0.700	0.6722309
0.800	0.5704460
0.900	0.4545046

Figura 6.12

En la opción de Salvar Gráficas se pueden salvar gráficas ya procesadas incluyendo una breve descripción de la información. Las gráficas se grabarán en archivos con la extensión ".GRA".

La opción Leer Graficas permite recuperar hasta cinco gráficas a la vez, visualizando todos los archivos con extensión ".GRA" así como las descripciones de cada uno. Se cuenta además con la opción de cambio de directorio. Ver figura 6.13.

Carga de datos

Archivos
GRAFICA1.GRA
GRAFICA2.GRA
GRAFICA3.GRA
GRAFICA4.GRA
SALIDA.GRA
SALCONT.GRA
CONTDIS.GRA
SALPLANT.GRA
ERROR.GRA

Directorio actual

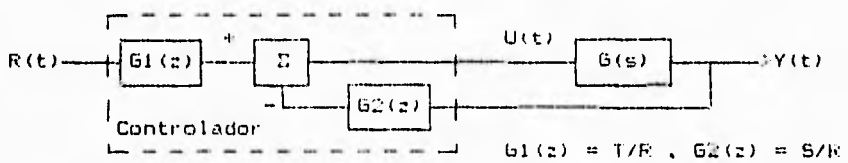
C:\TUF8055\CONTROL

Cambio de directorio

grafica salida del controlador malla cerrada

Figura 6.13

La opción Controlador permite visualizar las funciones de transferencia del controlador diseñado. Ver figura 6.14.



G1(z) = T/R , G2(z) = S/R

Grado en z	Polinomio R	Polinomio S	Polinomio T
z^ 0	0.0530	-0.7373	-0.0501
z^ 1	1.0000	1.3193	0.6321

Figura 6.14

La opción 3) Salvar datos del menú principal permite grabar los datos presentes a disco dándoles la extensión ".DSC". Se requiere una breve descripción de los datos antes de ser grabados.

La opción 4) Leer datos de disco, del menú principal introduce a una pantalla en la que se pueden visualizar y seleccionar todos los archivos con extención ".DSC". Se puede ver también la descripción de cada uno, contando con la opción de cambio de directorio. Ver figura 6.15.

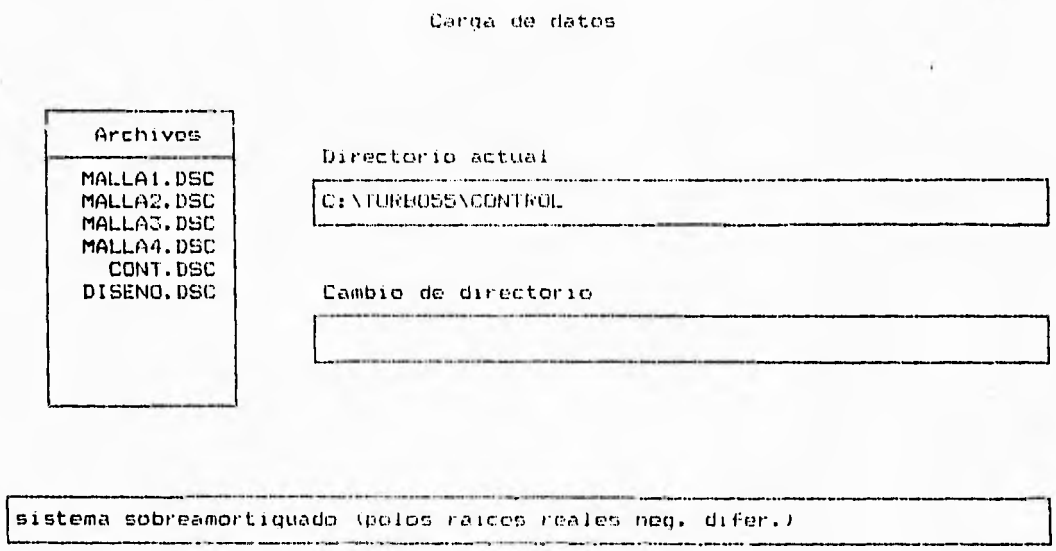


Figura 6.15

La opción 5) Consulta de datos actuales del menú principal muestra los datos que actualmente se están analizando en el paquete, pudiéndose visualizar la función de transferencia de la

planta G(s), del controlador Gc(z) o del sistema H(z). Ver figura 6.16.

Datos actuales

z⁰	Numerador	Denominador
0	0.9379	-0.4043
1	-12.8605	0.5492
2	13.0680	1.0000

1) G(s)
2) Gc(z)
3) H(z)

T= 40.000

TM= 1.000

Ganancia= 1.000

Figura 6.16

La opción 6) Salir, permite abandonar el paquete.

Con la tecla [Esc] se permite regresar a menus anteriores.

Ejemplo numérico.

1) Planta en malla abierta.

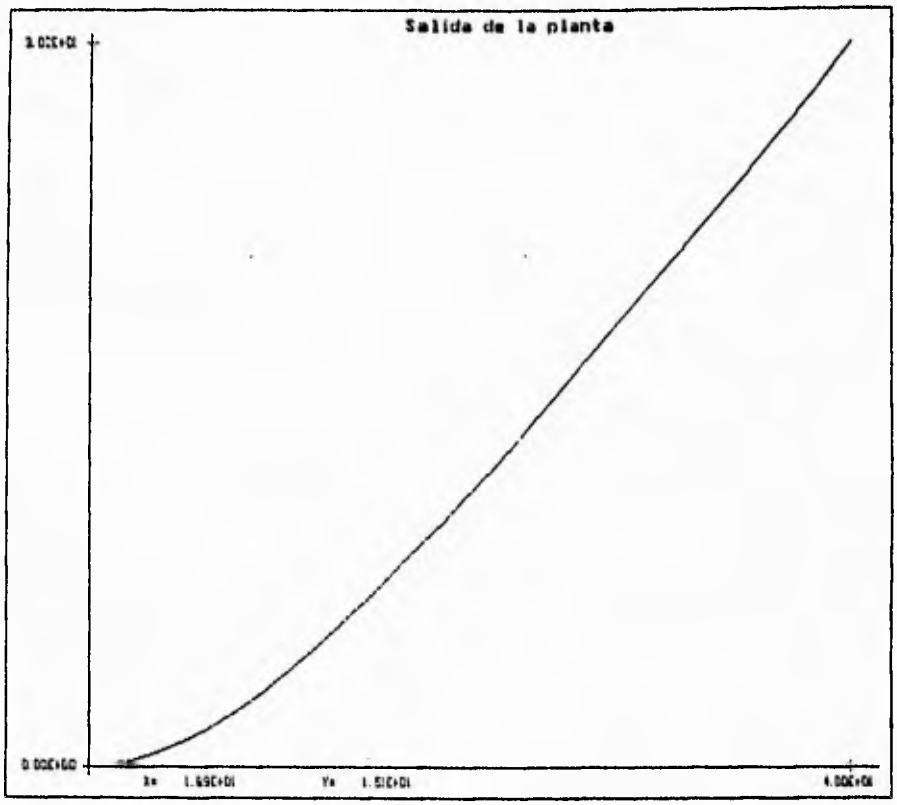
Datos :

$$G(s) = \frac{0.1}{s(s+0.1)}$$

U(t) = 1 para t ≥ 0 (escalón unitario)

Tiempo de análisis = 20 seg.

Resultados :



2) Planta en malla cerrada.

Datos :

$$G(s) = \frac{0.1}{s(s+0.1)}$$

$$G_c(Z) = 6.8112 \frac{Z - 0.9048}{Z - 0.1353}$$

Controlador obtenido por el método del equivalente discreto.

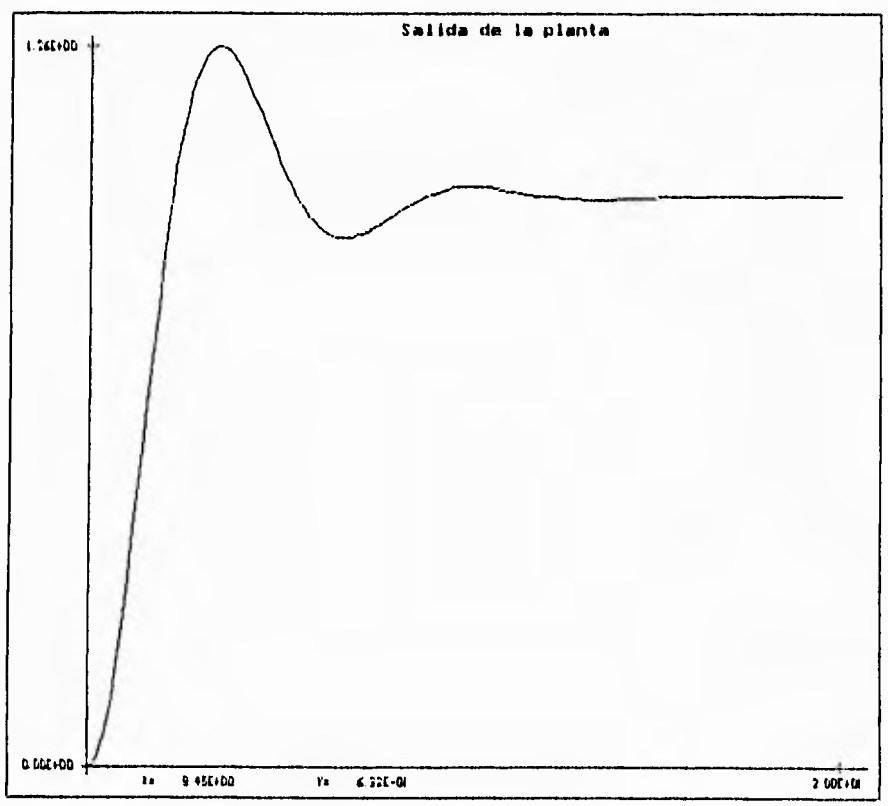
U(t) = 1 para t ≥ 0 (escalón unitario)

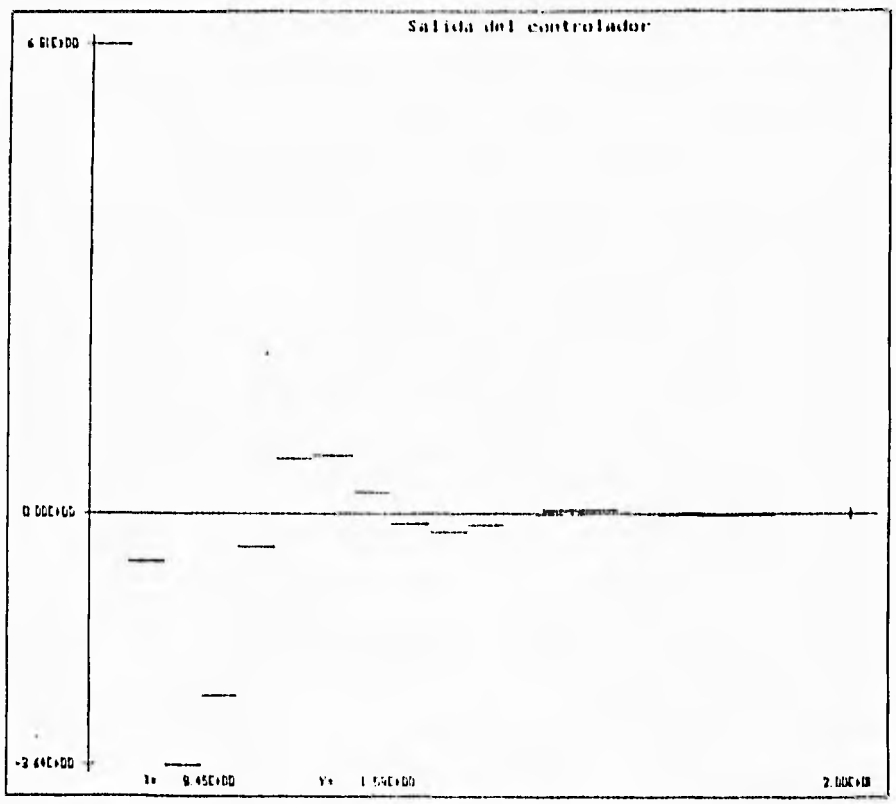
Tiempo de análisis = 20 seg.

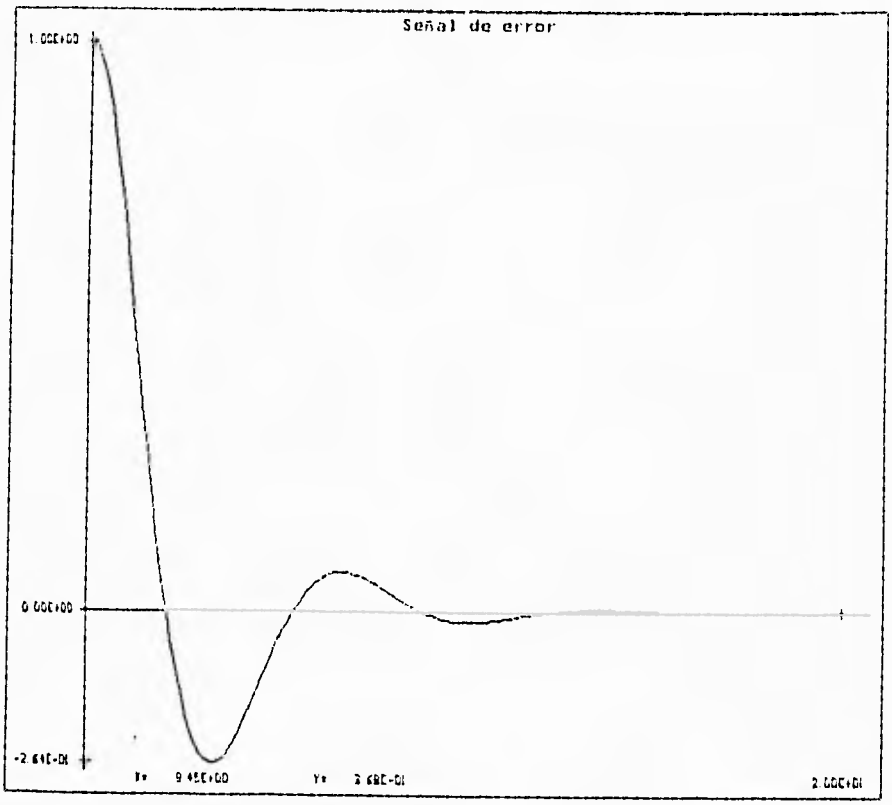
Periodo de muestreo = 1 seg.

Ganancia de lazo = 1

Resultados :




Salida del controlador
6.81E+00
0.00E+00
-2.64E+00
Xa 9.45E+00
Ya 1.95E+00
2.00E+01



3) Diseño.

Datos :

$$G(s) = \frac{0.1}{s(s+0.1)}$$

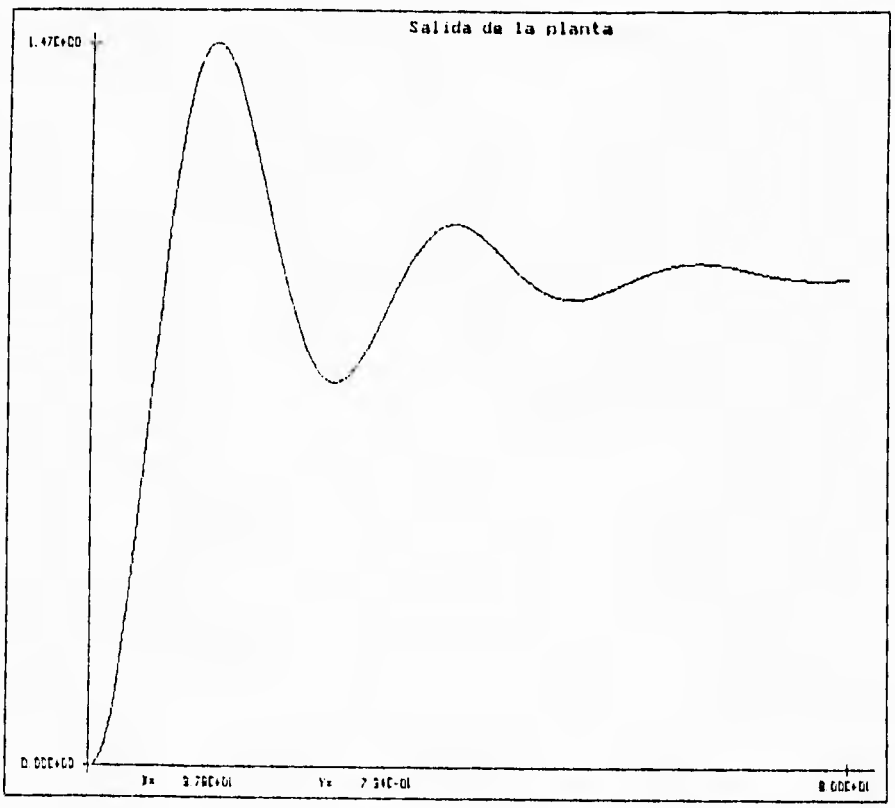
$$H(Z) = \frac{0.3295Z - 0.3187}{Z^2 - 0.8058Z + 0.454}$$

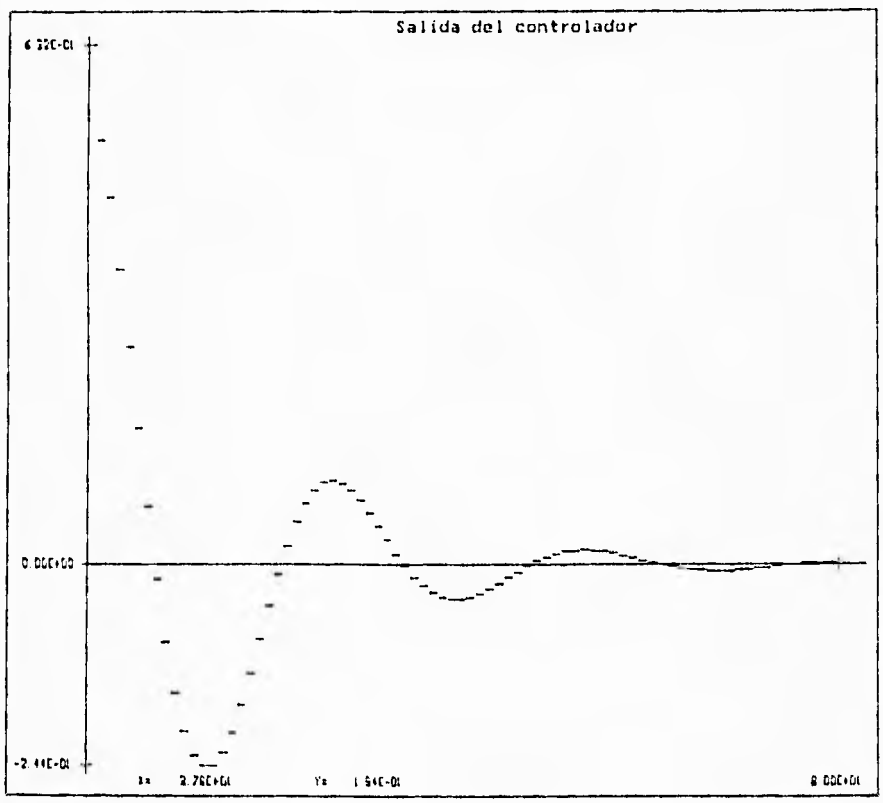
U(t) = 1 (escalón unitario)

Tiempo de análisis = 80 seg.

Período de muestreo = 1 seg.

Resultados :




Salida del controlador
6.22E-01
0.00E+00
-2.44E-01
X= 2.76E+01
Y= 1.54E-01
8.00E+01

CONCLUSIONES.

Como planteamos en la introducción del presente trabajo, es de gran utilidad el contar con la ayuda de un paquete de computadora para llevar a cabo la solución de sistemas de control digital.

Tras las pruebas realizadas al paquete, consideramos que es una herramienta útil para los usuarios del área de control, ya que pueden realizar un análisis más eficiente y rápido, comparado con el tiempo que les llevaría realizar el mismo análisis a mano debido a la cantidad de desarrollo matemático que esto implica.

El modelo construido es un prototipo bastante aceptable para su aplicación a nivel licenciatura, el cual puede ser de utilidad en un laboratorio de análisis de sistemas de control para fines docentes y de desarrollo docente ya que cuenta con todos los elementos para ser modificado y así ampliar el campo de acción del paquete acorde a los requerimientos del usuario.

El presente trabajo fue realizado en base a los conocimientos adquiridos durante nuestra formación profesional, conjugando las bases matemáticas con el análisis de sistemas para la solución de problemas de control.

Esperamos que este esfuerzo sea de utilidad para la formación de futuros ingenieros en computación, dado que es una muestra de la aplicación de la computación en la automatización de métodos utilizados en la ingeniería para su apoyo.

BIBLIOGRAFIA

- Introduction to Numerical Computation.
  J. Thomas King.
  U.S.A. 1984.
  Mc Graw Hill.

- Proc. Inst. Elec Eng.
  Part. D. Vol 127, pp. 120-130.
  The Institution of Electrical Engineers.
  May 1980.

- Numerical Analisys
  Richard L. Burden,
  J. Douglas Faires,
  Albert A. Reynols.
  U.S.A. 1978.
  Prindle, Weber & Schimidt.
  Fifth Printing 1980.

- Ejercicios de Ecuaciones Diferenciales y en Diferencias.
  Gabriel Dorantes Gómez,
  Prospero Gracia Márquez,
  Carlos de la Lanza Elton,
  Angel Victoria Rosales.
  Universidad Nacional Autónoma de México.
  Facultad de Ingeniería.
  Division de Ciencias Básicas,
  Departamento de Matemáticas Aplicadas.
  México 1984.

- Apuentes de Sistemas Dinámicos.
  Francisco José Rodríguez Ramírez.
  Universidad Nacional Autónoma de México.
  Facultad de Ingeniería.
  División de Ingeniería Mecánica y Eléctrica,
  Departamento de Ingeniería de Control.
  México 1986.

A N E X O

L I S T A D O D E P R O G R A M A S F U E N T E

```
PROGRAM PRINCIPAL_DSC;

USES DOS,CRT,GRAPH,DECLARA,DISENO,ARCHIVOS,MENSAJES,LAZOPL,GRAFICAR,DATOSAC;

{=========================================================================}
{ DESPLIEGA EN LA PANTALLA EL DIAGRAMA DE LAZO CERRADO                    }

 PROCEDURE LAZO(CH:CHAR);  { BLOQUE ACTIVO }
 BEGIN
  GOTOXY(15,4);
  WRITE('       +     E(t)        U(t)        ');        GOTOXY(15,5);
  WRITE('R(t)---| Σ |-------| Gc(z) |-------| G(s) |---->Y(t)');GOTOXY(15,6);
  WRITE('       -                                 ');    GOTOXY(15,7);
  WRITE('         |                               ');    GOTOXY(15,8);
  WRITE('         |_____| Sensor |________________');    GOTOXY(15,9);
  WRITE('                                         ');
  TEXTCOLOR(30);
  CASE CH OF
       'R': BEGIN
            GOTOXY(15,5);
            WRITE('R(t)');
           END;
       'P': BEGIN
            GOTOXY(52,5);
            WRITE('G(s)');
           END;
       'C': BEGIN
            GOTOXY(37,5);
            WRITE('Gc(z)');
           END;
       'S': BEGIN
            GOTOXY(40,8);
            WRITE('Sensor');
           END;
       '6': BEGIN
            GOTOXY(30,4);
            WRITE('E(t)');
            GOTOXY(45,4);
            WRITE('U(t)');
            GOTOXY(65,5);
            WRITE('Y(t)');
           END;
  END; {DEL CASE}
  TEXTCOLOR(15);
 END;

{=========================================================================}
{ DESPLIEGA EL DIAGRAMA DEL LAZO CON EL CONTROLADOR DISEÑADO              }

 PROCEDURE LAZOCONT(CH: CHAR);   { BLOQUE ACTIVO }
 VAR D: INTEGER;

 BEGIN
  IF CH<>'C' THEN
     D:=2
  ELSE
```

```
D:=0;
 GOTOXY(10,D+1);
 WRITE('       ┌ - - - - - - - - - - - - ┐');              GOTOXY(10,D+2);
 WRITE('                                  U(t)           ');GOTOXY(10,D+3);
 WRITE('R(t)──┤G1(z)├──┤ Σ ├──────────┤ G(s) ├──────>Y(t)');GOTOXY(10,D+4);
 WRITE('                                                 ');GOTOXY(10,D+5);
 WRITE('                  -└──┤G2(z)├──┘                 ');GOTOXY(10,D+6);
 WRITE('       Controlador                               ');GOTOXY(10,D+7);
 WRITE('       └ - - - - - - - - - - - - ┘  G1(z) = T/R , G2(z) = S/R');
 TEXTCOLOR(30);
 CASE CH OF
      'P': BEGIN
             GOTOXY(54,D+3);
             WRITE('G(s)');
           END;
      'C': BEGIN
             GOTOXY(19,3);
             WRITE('G1(z)');
             GOTOXY(36,5);
             WRITE('G2(z)');
           END;
      'M': BEGIN
             GOTOXY(54,D+3);
             WRITE('G(s)');
             GOTOXY(19,D+3);
             WRITE('G1(z)');
             GOTOXY(36,D+5);
             WRITE('G2(z)');
           END;
      'R': BEGIN
             GOTOXY(10,D+3);
             WRITE('R(t)');
           END;
 END; (DEL CASE)
 GOTOXY(1,9);
 TEXTCOLOR(15);
END;

{=============================================================================}
{ DESPLIEGA EL DIAGRAMA DE LAZO ABIERTO                                       }

 PROCEDURE PLANTA(CH:CHAR);   { BLOQUE ACTIVO }
 BEGIN
  GOTOXY(23,6);
  WRITE('                  ');           GOTOXY(23,7);
  WRITE('R(t)──────┤ G(s) ├──────>Y(t)'); GOTOXY(23,8);
  WRITE('                  ');
  TEXTCOLOR(30);
  CASE CH OF
       'R': BEGIN
              GOTOXY(23,7);
              WRITE('R(t)');
            END;
       'P': BEGIN
              GOTOXY(39,7);
              WRITE('G(s)');
```

```
          END;
      'G': BEGIN
           GOTOXY(55,7);
           WRITE('Y(t)');
           END;
      'H': BEGIN
           GOTOXY(39,7);
           WRITE('H(z)');
           END;
  END; {DEL CASE}
  TEXTCOLOR(15);
END;

{=============================================================================}
{ VALIDA QUE EL NUMERO TECLEADO SEA UN NUMERO REAL                            }

PROCEDURE VALIDA(VAR NUMERO:REAL);     { NUMERO A VALIDAR }
VAR
 POSX,POSY: BYTE;        { POSICION DEL CURSOR }
 BANDERAERROR: BOOLEAN;  { INDICADOR DE ERROR }
 NUMEROCH: STRING[10];   { NUMERO ALFANUMERICO }
 C: INTEGER;             { NUMERO DE LA POSICION DONDE SE ENCONTRO EL ERROR}

BEGIN
   I:=1;
   NUMEROCH:='';
   REPEAT
    BANDERAERROR:=FALSE;
    REPEAT
      REPEAT
        TECLA3:=READKEY;
        IF (ORD(TECLA3)=8) AND (I>1) THEN
          BEGIN
           I:=I-1;
           DELETE(NUMEROCH,I,1);
           GOTOXY(WHEREX-1,WHEREY);
           WRITE(' ');
           GOTOXY(WHEREX-1,WHEREY);
          END;
      UNTIL (((ORD(TECLA3)>44) AND (ORD(TECLA3)<58) AND (ORD(TECLA3)<>47))
             OR (ORD(TECLA3)=13) OR (ORD(TECLA3)=27));
      IF (ORD(TECLA3)>44) AND (ORD(TECLA3)<58) AND (ORD(TECLA3)<>47) THEN
          IF I>10 THEN
            BEGIN
             POSX:=WHEREX;
             POSY:=WHEREY;
             MENSAJE(0);
             MENSAJE(6);
             GOTOXY(POSX,POSY);
            END
          ELSE
           BEGIN
            WRITE(TECLA3);
            NUMEROCH:=NUMEROCH+TECLA3;
            I:=I+1;
           END;
```

```
   UNTIL ((ORD(TECLA3)=13)AND(NUMEROCH<>'') OR (ORD(TECLA3)=27)) ;
IF ORD(TECLA3)<>27 THEN
   BEGIN
    VAL(NUMEROCH,NUMERO,C);
    if C <> 0 then
       BEGIN
        POSX:=WHEREX;
        POSY:=WHEREY;
        MENSAJE(0);
        MENSAJE(7);
        WRITE(C);
        GOTOXY(POSX,POSY);
        BANDERAERROR:=TRUE;
       END;
  END;
 UNTIL ((ORD(TECLA3)=27) OR (BANDERAERROR=FALSE));
END;

{##########################################################################}
{ VALIDA QUE EL GRADO DEL POLINOMIO SEA UN NUMERO ENTERO MAYOR QUE CERO Y   }
{ MENOR QUE 16, ASI COMO QUE EL GRADO DE LOS POLOS SEA CORRECTO            }

PROCEDURE VALIDA_GRADO (VAR GRADO : INTEGER; { GRADO DEL POLINOMIO }
                         POLOS : BOOLEAN; { INDICADOR DE POLOS O CEROS }
                         ES_PLANTA : CHAR); { INDICADOR DE TIPO DE
                                              .    FUNCION DE TRANSFERENCIA }

VAR GRADO_CORRECTO,NoVALIDO : BOOLEAN; { INDICADORES DE ERROR }

BEGIN
  GRADO_CORRECTO := TRUE;
  REPEAT
    REPEAT
      GOTOXY(48,14);
      WRITE('          ');
      GOTOXY(48,14);
      VALIDA(BUFFER);
      MENSAJE(0);
      NoVALIDO:=TRUE;
      IF ((FRAC(BUFFER)<>0) OR (TRUNC(BUFFER)>15) OR (TRUNC(BUFFER)<0)) AND
          (ORD(TECLA3)<>27) THEN
         BEGIN
          MENSAJE(8);
          NoVALIDO:=FALSE;
         END;
    UNTIL NoVALIDO;
    GRADO :=TRUNC(BUFFER);
    IF POLOS AND (ORD(TECLA3)<>27) THEN
       IF ES_PLANTA = 'P' THEN
          IF GRADOOP > GRADO THEN
             BEGIN
               MENSAJE(2);
               GRADO_CORRECTO := FALSE;
             END
             ELSE GRADO_CORRECTO := TRUE
       ELSE
```

```
        IF ES_PLANTA = 'C' THEN
           IF GRADOZOC > GRADO THEN
              BEGIN
                MENSAJE(2);
                GRADO_CORRECTO := FALSE;
              END
           ELSE GRADO_CORRECTO := TRUE
        ELSE
          IF GRADPOLBMS > GRADO THEN
              BEGIN
                MENSAJE(2);
                GRADO_CORRECTO := FALSE;
              END
           ELSE GRADO_CORRECTO := TRUE;
  UNTIL GRADO_CORRECTO ;
  MENSAJE(0);
END;

{************************************************************************}
{ LEE POLINOMIOS POR RAICES                                              }

PROCEDURE LEEPOLRAIZ (VAR GRADPOLX : INTEGER; { GRADO DEL POLINOMIO }
                      VAR POLX : POLINOMIO;   { COEFICIENTES DEL POLINOMIO }
                      POLOS : BOOLEAN;        { INDICADOR DE POLOS O CEROS }
                      ES_PLANTA : CHAR);      { INDICADOR DE TIPO DE
                                                FUNCION DE TRANSFERENCIA }

VAR     POLPAR : POLIRAIZ;  { RAICES DEL POLINOMIO }
        I,J    : INTEGER;   { CONTADORES AUXILIARES }
        GANANCIA : REAL;    { GANANCIA INHERENTE }
        NoVALIDO : BOOLEAN; { INDICADOR DE ERROR }

BEGIN
   PRENDECURSOR;
   TEXTBACKGROUND(3);
   MENSAJE(0);
   GOTOXY(12,13);
   WRITE('┌──────────────────────────────────────────────┐'); GOTOXY(12,14);
   WRITE('│ ¿ Cuál es el grado del polinomio ?           │'); GOTOXY(12,15);
   WRITE('└──────────────────────────────────────────────┘');
   VALIDA_GRADO(CONRA,POLOS,ES_PLANTA);
   IF NOT POLOS THEN
      BEGIN
        GANANCIA := 0;
        GOTOXY(12,14);
        WRITE('¿ Valor de la ganancia inherente ?                    '); GOTOXY(12,14);
        WHILE (GANANCIA = 0)  AND (ORD(TECLA3) <> 27) DO
        BEGIN
          GOTOXY(48,14);
          WRITE('         ');
          GOTOXY(48,14);
          VALIDA(GANANCIA);
          MENSAJE(0);
          NoVALIDO:=TRUE;
          IF (GANANCIA = 0) AND (ORD(TECLA3) <> 27) THEN
            MENSAJE(12);
```

```
      END ;
    END;
  J := 1;
  I := 0;
  WHILE (J <= CONRA) AND (ORD(TECLA3) <> 27) DO
  BEGIN
   I := I+1;
   GOTOXY(12,14);
   WRITE('|  Raiz ',J:2,', parte real :                              |');
   GOTOXY(38,14);
   VALIDA(RAIZ[I].RE);
   GOTOXY(20,16);
   IF (J < CONRA) AND (ORD(TECLA3) <> 27) THEN
     BEGIN
      GOTOXY(12,14);
      WRITE('|  Raiz ',J:2,',parte imaginaria :                         |');
      GOTOXY(43,14);
      VALIDA(RAIZ[I].IM);
      GOTOXY(20,17);
     END
   ELSE
     RAIZ[I].IM := 0;
   WITH RAIZ[I] DO
     BEGIN
       IF IM = 0 THEN
         BEGIN
         J:=J+1;
         WITH POLPAR[I] DO
           BEGIN
            ORDENPOLRAIZ := 1;
            POLRAIZ[1] := 1;
            POLRAIZ[0] := -RE
           END
         END
       ELSE
         BEGIN
          J := J+2;
          WITH POLPAR[I] DO
            BEGIN
              ORDENPOLRAIZ := 2;
              POLRAIZ[2] := 1;
              POLRAIZ[1] := -2*RE;
              POLRAIZ[0] := SQR(RE) + SQR(IM)
            END
         END
       END
     END;
  OPTPOLCOMBXAX(GRADPOLX,POLX,POLPAR,I);
  CONRA := I;
  IF NOT POLOS THEN
    FOR I := 0 TO GRADPOLX DO POLX[I] := POLX[I] * GANANCIA;
  APAGACURSOR;
  TEXTBACKGROUND(0);
  CLRSCR;
END;
```

```
{=================================================================}
{ LEE POLINOMIOS POR COEFICIENTES                                 }

PROCEDURE LEEPOLINOMIO(VAR GRADO : INTEGER;   { GRADO DEL POLINOMIO }
                       VAR COEF  : POLINOMIO; { COEFICIENTES DEL POLINOMIO }
                       POLOS : BOOLEAN;       { INDICADOR DE POLOS O CEROS }
                       ES_PLANTA : CHAR);     { INDICADOR DE TIPO DE
                                                FUNCION DE TRANSFERENCIA }

VAR  R : INTEGER;    { CONTADOR AUXILIAR }

 BEGIN
  PRENDECURSOR;
  TEXTBACKGROUND(3);
  MENSAJE(0);
  GOTOXY(12,13);
  WRITE('                                                  '); GOTOXY(12,14);
  WRITE('¿ Cuál es el grado del polinomio ?                '); GOTOXY(12,15);
  WRITE('                                                  ');
  VALIDA_GRADO(GRADO,POLOS,ES_PLANTA);
  R := GRADO;
  WHILE (R <> 0) AND (ORD(TECLA3) <> 27) DO
  BEGIN
    GOTOXY(12,14);
    WRITE('¦Dame el coeficiente de la ',R:2,'ª potencia              ¦');
    GOTOXY(52,14);
    VALIDA(COEFIR]);
    R:=R-1;
  END;
  IF ORD(TECLA3)<>27 THEN
     BEGIN
       GOTOXY(12,14);
       WRITE('¦Dame el coeficiente del término independiente        ¦');
       GOTOXY(59,14);
       VALIDA(COEF[0]);
     END;
  APAGACURSOR;
  TEXTBACKGROUND(0);
  CLRSCR;
END;

{=================================================================}
{ ACEPTA EL TIEMPO A ANALIIAR DEL SISTEMA                         }

PROCEDURE TIEMPODEANALISIS;
BEGIN
  PRENDECURSOR;
  TEXTBACKGROUND(3);
  MENSAJE(0);
  GOTOXY(17,12);
  WRITE('                                          '); GOTOXY(17,13);
  WRITE('¿ Tiempo a analizar (en segundos) ?       '); GOTOXY(17,14);
  WRITE('                                          ');
  IF ORD(TECLA1) <> 50 THEN
     BEGIN
       MENSAJE(0);
```

```
      MENSAJE(15);
    END;
  REPEAT
    GOTOXY(54,13);
    WRITE(' ':10);
    GOTOXY(54,13);
    VALIDA(TIEMPO);
    IF (TIEMPO <= 0) AND (ORD(TECLA3) <> 27) THEN
       BEGIN
         MENSAJE(0);
         MENSAJE(24);
       END;
  UNTIL (TIEMPO > 0) OR (ORD(TECLA3) = 27);
  APAGACURSOR;
  TEXTBACKGROUND(0);
  CLRSCR;
END;

{=========================================================================}
{ CALCULA EL TIEMPO A ANALIZAR EN BASE A LA FRECUENCIA NATURAL DEL SISTEMA }

PROCEDURE BUSCATIEMPO;

VAR I,CONTADOR : INTEGER; { CONTADORES AUXILIARES }
    MENOR : REAL;         { RAIZ MENOR }

BEGIN
   IF ORD(TECLA2) = 49 THEN
      BEGIN
        CONTADOR := GRADOPP;
        IF GRADOPP >= 2 THEN
           BEGIN
             PTOINIC := 0;
             IF GRADOPP > 2 THEN
                IF (COEFSPP[0] = 0)  OR (COEFSPP[2] = 0) THEN PTOINIC := 0.1;
             M := 0;
             J := 1;
             FOR I := 0 TO GRADOPP DO POLAUXILIAR[I] := COEFSPP[GRADOPP-I];
             LLAMADD(POLAUXILIAR,GRADOPP,'NINGUNO');
           END
        ELSE
           IF -COEFSPP[0] < 0 THEN
              TIEMPO := 0/COEFSPP[0]
           ELSE
             TIEMPODEANALISIS;  (MENSAJES)
      END
   ELSE
     BEGIN
       CONTADOR := CONRA;
       IF CONRA = 1 THEN
          IF RAIZ[1].RE < 0 THEN
             TIEMPO := -0/RAIZ[1].RE
          ELSE
            TIEMPODEANALISIS; { MENSAJES }
     END;
   IF CONTADOR >= 2 THEN
```

```
      BEGIN
        I := 1;
        MENOR := RAIZ[1].RE;
        WHILE (MENOR < 0) AND (I < CONTADOR) DO
          BEGIN
            I := I+1;
            WITH RAIZ[I] DO
              IF (MENOR > RE) or (RE >= 0) THEN
                 MENOR := RE;
          END;
        IF MENOR < 0 THEN
          TIEMPO := -8/MENOR
        ELSE
          TIEMPODEANALISIS; { MENSAJES }
      END;
   IF (POS(' 2',DATOS)=0) AND (ORD(TECLA2)<>27) AND (ORD(TECLA3)<>27) THEN
      BEGIN
        DELETE(DATOS,2,2);
        INSERT(' 2',DATOS,2);
      END;
END;

{=============================================================}
{ PREGUNTA SI SE CUENTA CON EL EQUIVALENTE DISCRETO DE H(Z)    }

PROCEDURE DISCRETO;
VAR I : INTEGER;  { CONTADOR AUXILIAR }

BEGIN
   GOTOXY(1,1);
   WRITE('Función de transferencia del sistema':59);
   I := 54;
   REPEAT
     TEXTBACKGROUND(3);
     GOTOXY(12,13);
     WRITE('┌──────────────────────────────────────────┐'); GOTOXY(12,14);
     WRITE('│ ¿ Cuenta con el equivalente discreto ?  Si  No │'); GOTOXY(12,15);
     WRITE('└──────────────────────────────────────────┘');
     TEXTBACKGROUND(5);
     GOTOXY(I,14);
     CASE I OF
       54 : WRITE('Si');
       58 : WRITE('No');
     END; {DEL CASE}
     TECLA3:=READKEY;
     CASE ORD(TECLA3) OF
       75 : BEGIN
              I:=I-4;
              IF I<54 THEN I:=58;
            END;
       77 : BEGIN
              I:=I+4;
              IF I>58 THEN I:=54;
            END;
     END; {DEL CASE}
   UNTIL (ORD(TECLA3)=79) OR (ORD(TECLA3)=83) OR (ORD(TECLA3)=13) OR
```

```
        (ORD(TECLA3)=27);
  IF ORD(TECLA3) <> 27 THEN
     IF ORD(TECLA2) = 17 THEN
        IF I = 54 THEN
          BEGIN
            NO_DISCRETO := FALSE;
            LECTURA_NODISCRETA := FALSE;
          END
        ELSE
          BEGIN
            NO_DISCRETO := TRUE;
            LECTURA_NODISCRETA := TRUE;
          END
     ELSE
        IF ORD(TECLA3) = 83 THEN
           BEGIN
             NO_DISCRETO := FALSE;
             LECTURA_NODISCRETA := FALSE;
           END
        ELSE
           BEGIN
             NO_DISCRETO := TRUE;
             LECTURA_NODISCRETA := TRUE
           END;
  APAGACURSOR;
  TEXTBACKGROUND(0);
  CLRSCR;
END;

{=========================================================================}
{ NORMALIZA UNA FUNCION DE TRANSFERENCIA                                  }

PROCEDURE NORMALIZA(GRADOP,GRADOO : INTEGER; { GRADO DE POLOS Y CEROS }
                    VAR COEFSP,COEFSO : POLINOMIO); { COEFICIENTES DE POLOS
                                                      Y CEROS }
VAR I : INTEGER;  { CONTADOR AUXILIAR }

BEGIN
  IF COEFSP[GRADOP] <> 1 THEN
     BEGIN
       FOR I:=0 TO GRADOO DO COEFSO[I]:=COEFSO[I]/COEFSP[GRADOP];
       FOR I:=0 TO GRADOP DO COEFSP[I]:=COEFSP[I]/COEFSP[GRADOP];
     END;
END;

{=========================================================================}
{ LEE LA FUNCION DE ENTRADA AL SISTEMA                                    }

PROCEDURE LEEENTRADA;
BEGIN
  GOTOXY(1,1);
  WRITE('Tipos de entrada al sistema :':56);
  I:=49;
  REPEAT
    TEXTBACKGROUND(3);
    GOTOXY(14,13);
```

```
  WRITE('┌--------------------┐'); GOTOXY(14,14);
  WRITE('|1) δ(t)             |'); GOTOXY(14,15);
  WRITE('|2) Escalón          |'); GOTOXY(14,16);
  WRITE('|3) Ct               |'); GOTOXY(14,17);
  WRITE('|4) Cexp(At)         |'); GOTOXY(14,18);
  WRITE('|5) Cexp(At)sen(Bt)  |'); GOTOXY(14,19);
  WRITE('|6) Cexp(At)cos(Bt)  |'); GOTOXY(14,20);
  WRITE('└--------------------┘');
  TEXTBACKGROUND(5);
  GOTOXY(15,I-35);
  CASE I OF
       49 : WRITE('1) δ(t)           ');
       50 : WRITE('2) Escalón        ');
       51 : WRITE('3) Ct             ');
       52 : WRITE('4) Cexp(At)       ');
       53 : WRITE('5) Cexp(At)sen(Bt) ');
       54 : WRITE('6) Cexp(At)cos(Bt) ');
  END; {DEL CASE}
  TECLA2:=READKEY;
  CASE ORD(TECLA2) OF
       72 : BEGIN
              I:=I-1;
              IF I<49 THEN I:=54;
            END;
       80 : BEGIN
              I:=I+1;
              IF I>54 THEN I:=49;
            END;
  END; {DEL CASE}
UNTIL ((ORD(TECLA2)>49) AND (ORD(TECLA2)<55)) OR (ORD(TECLA2)=27) OR
      ((ORD(TECLA2)=49) AND (TIPO='P')) OR ((ORD(TECLA2)=13) AND
      ((I<>49) OR (TIPO='P')));
IF ORD(TECLA2)<>27 THEN
   BEGIN
    IF ORD(TECLA2)=13 THEN TECLA2:=CHR(I);
    TEXTBACKGROUND(3);
    PRENDECURSOR;
    FUNCION:=ORD(TECLA2)-48;
    GOTOXY(52,16);
    WRITE('┌----------┐'); GOTOXY(52,18);
    WRITE('└----------┘'); GOTOXY(52,17);
    CASE FUNCION OF
          5,6 : BEGIN
                  WRITE('|C=         |'); GOTOXY(55,17);
                  VALIDA(C); GOTOXY(52,17);
                  IF ORD(TECLA3)<>27 THEN
                     BEGIN
                      WRITE('|A=         |'); GOTOXY(55,17);
                      VALIDA(A); GOTOXY(52,17);
                      IF ORD(TECLA3)<>27 THEN
                         BEGIN
                          WRITE('|B=         |'); GOTOXY(55,17);
                          VALIDA(W);
                         END;
                     END;
                END;
```

```
            4 : BEGIN
                WRITE('|C=         |'); GOTOXY(55,17);
                VALIDA(C); GOTOXY(52,17);
                IF ORD(TECLA3)<>27 THEN
                   BEGIN
                    WRITE('|A=         |'); GOTOXY(55,17);
                    VALIDA(A);
                   END;
               END;
        1,2,3 : BEGIN
                WRITE('|C=         |'); GOTOXY(55,17);
                VALIDA(C);
               END;
     END; {DEL CASE}
    END;
  APAGACURSOR;
  TEXTBACKGROUND(0);
  CLRSCR;
END;

{=========================================================================}
{ LEE EL NUMERO DE GRAFICAS A DESPLEGAR                                   }

PROCEDURE LEENoGRAFICAS;
VAR NoVALIDO : BOOLEAN;  { INDICADOR DE ERROR }

BEGIN
  REPEAT
    TEXTBACKGROUND(3);
    MENSAJE(0);
    MENSAJE(20);
    VALIDA(BUFFER);
    MENSAJE(0);
    NoVALIDO:=TRUE;
    IF ((FRAC(BUFFER)<>0) OR (TRUNC(BUFFER)>5) OR (TRUNC(BUFFER)<=0)) AND
        (ORD(TECLA3)<>27) THEN
       BEGIN
        MENSAJE(9);
        NoVALIDO:=FALSE;
       END;
  UNTIL NoVALIDO;
  NoGRAFICAS:=TRUNC(BUFFER);
  TEXTBACKGROUND(0);
  CLRSCR;
END;


{=========================================================================}
{ DESPLIEGA  UN MENSAJE DE QUE LA INFORMACION ESTA EN PROCESO             }

PROCEDURE PROCESANDO;
BEGIN
  TEXTBACKGROUND(3);
  GOTOXY(17,12);
  WRITE('┌──────────────────────────────────────┐'); GOTOXY(17,13);
  WRITE('│                                      │'); GOTOXY(17,14);
```

```
  WRITE('└──────────────────────────────────────────┘');
  TEXTCOLOR(30);
  GOTOXY(36,13);
  WRITE('Procesando...');
  TEXTCOLOR(15);
END;


{=============================================================================}
{ SELECCIONA EL DATO A SER MANIPULADO                                         }

FUNCTION SALIDAPILA(NoSALTOS:INTEGER;{INDICE DEL DATO QUE SE QUIERE ACCESAR)
                    VAR PILA : APUNTADOR): REAL; { APUNTADOR DE REFERENCIA }
VAR SALIDA: APUNTADOR; { APUNTADOR AUXILIAR }
    I : INTEGER;       { CONTADOR AUXILIAR }

BEGIN
  SALIDA:=PILA;
  FOR I:=1 TO NoSALTOS DO SALIDA:=SALIDA^.ANTERIOR;
  SALIDAPILA:=SALIDA^.DATO
END;

{=============================================================================}
{ EVALUA LA FUNCION DE ENTRADA                                                }

FUNCTION FUNCIONT(S:REAL) : REAL; { TIEMPO PARA LA FUNCION }
VAR
  R:REAL;  { ESULTADO DE LA FUNCION DE ENTRADA }
BEGIN
  CASE FUNCION OF
     1,2 : R:=C;
       3 : R:=A@S+C;
       4 : IF (M@S)<-88 THEN R:=C
             ELSE R:=A@EXP(M@S)+C;
       5 : R:=A@SIN(M@S)+C;
       6 : R:=A@COS(M@S)+C;
  END; (DEL CASE)
  IF TIPO = 'C' THEN
     IF BLOQUE = 'C' THEN
       FUNCIONT:=R-SALIDAPILA(JR - TRUNC(S/H),ULTIMOPLANTA) @ GANANCIA
     ELSE
       FUNCIONT := ULTIMOCONTI^.DATO
  ELSE
    IF TIPO='H' THEN
       IF BLOQUE = 'S' THEN
         FUNCIONT := SALIDAPILA(JR - TRUNC(S/H),ULTIMOPLANTA)
       ELSE
         IF BLOQUE = 'P' THEN
            FUNCIONT := SALIDACONT1-SALIDACONT2
         ELSE
           FUNCIONT:=R
    ELSE
      FUNCIONT := R;
END;

{=============================================================================}
```

```
{ ALMACENA LA SEÑAL DE ERROR EN UN ARREGLO                              }

PROCEDURE SENAL_ERROR;
VAR ERROR : REAL;  { SENAL DE ERROR }

BEGIN
  BLOQUE := 'C';
  ERROR := FUNCIONT(T);
  ARREGLODATOS(ERROR,PRIMEROCONT2,ULTIMOCONT2);
  PUNTOERROR := ROUND(ESCALAPLANTA*JR);
  SALIDAERROR[PUNTOERROR] := ERROR;
END;

{=======================================================================}
{ METODO DE RUNGE-KUTTA                                                 }

PROCEDURE RUNGE ;
VAR K : ARRAY [1..4,1..15] OF REAL; { VARIABLE K DEL METODO }
    SALIDAPL : REAL;                { SALIDA ACTUAL DE LA PLANTA }

BEGIN
     JR:=JR+1;
     FOR I:=1 TO (GRADOPP-1) DO K[1,I]:=Y[I+1];
     K[1,GRADOPP]:=FUNCIONT(T);
     FOR I:=1 TO GRADOPP DO K[1,GRADOPP]:=K[1,GRADOPP]-COEFSPP[I-1]*Y[I];
     FOR I:=1 TO (GRADOPP-1) DO K[2,I]:=Y[I+1]+H*K[1,I+1]/2;
     K[2,GRADOPP]:=FUNCIONT(T+H/2);
     FOR I:=1 TO GRADOPP DO
        K[2,GRADOPP]:=K[2,GRADOPP]-COEFSPP[I-1]*(Y[I]+H*K[1,I]/2);
     FOR I:=1 TO (GRADOPP-1) DO K[3,I]:=Y[I+1]+H*K[2,I+1]/2;
     K[3,GRADOPP]:=FUNCIONT(T+H/2);
     FOR I:=1 TO GRADOPP DO
        K[3,GRADOPP]:=K[3,GRADOPPI-COEFSPP[I-1]*(Y[I]+H*K[2,I]/2);
     FOR I:=1 TO (GRADOPP-1) DO K[4,I]:=Y[I+1]+H*K[3,I+1];
     K[4,GRADOPP]:=FUNCIONT(T+H);
     FOR I:=1 TO GRADOPP DO
         BEGIN
           K[4,GRADOPP]:=K[4,GRADOPP]-COEFSPP[I-1]*(Y[I]+H*K[3,I]);
           Y[I]:=Y[I]+H*(K[1,I]+2*K[2,I]+2*K[3,I]+K[4,I])/6
         END;
     SALIDAPL:=0;
     IF FUNCION <> 1 THEN
       FOR I:=1 TO (GRADOOP+1) DO SALIDAPL:=SALIDAPL+COEFSOP[I-1]*Y[I
     ELSE
       FOR I:=2 TO (GRADOOP+2) DO SALIDAPL:=SALIDAPL+COEFSOP[I-2]*Y[I];
     ARREGLODATOS(SALIDAPL,PRIMEROPLANTA,ULTIMOPLANTA);
     T:=JR*H;
     PUNTOPLANTA:=ROUND(ESCALAPLANTA*JR);
     SALIDAPLANTA[PUNTOPLANTA]:=SALIDAPL;
   IF (ABS(SALIDAPL)>1E10) THEN
      BEGIN
        MENSAJE(0);
        REBOSAMIENTO:=TRUE;
        MENSAJE(3);
      END;
END;
```

```
{=============================================================================}
( LEE EL PERIODO DE MUESTREO DEL SISTEMA                                      )

PROCEDURE PARAMETROSCONT;
BEGIN
  PRENDECURSOR;
  TEXTBACKGROUND(3);
  MENSAJE(0);
  GOTOXY(16,12);
  WRITE('                                                  '); GOTOXY(16,13);
  WRITE('¿ Periodo de muestreo (en segundos) ?             '); GOTOXY(16,14);
  WRITE('                                                  ');
  REPEAT
    GOTOXY(55,13);
    WRITE(' ':10);
    GOTOXY(55,13);
    VALIDA(TM);
    IF (TM <= 0) AND (ORD(TECLA3) <> 27) THEN
       BEGIN
         MENSAJE(0);
         MENSAJE(24);
       END;
  UNTIL (TM > 0) OR (ORD(TECLA3) = 27);
  IF LECTURA_NODISCRETA AND (ORD(TECLA3)<>27) THEN
     NO_DISCRETO := TRUE;
  APAGACURSOR;
  TEXTBACKGROUND(0);
  CLRSCR;
END;

{=============================================================================}
( LEE LA GANANCIA DEL LAZO DE REALIMENTACION                                  )

PROCEDURE GANANCIA_LAZO;
BEGIN
  PRENDECURSOR;
  TEXTBACKGROUND(3);
  MENSAJE(0);
  GOTOXY(19,12);
  WRITE('                                          '); GOTOXY(19,13);
  WRITE('¿ Cual es la ganancia del lazo ?          '); GOTOXY(19,14);
  WRITE('                                          ');
  REPEAT
    GOTOXY(53,13);
    WRITE(' ':10);
    GOTOXY(53,13);
    VALIDA(GANANCIA);
    IF (GANANCIA <= 0) AND (ORD(TECLA3) <> 27) THEN
       BEGIN
         MENSAJE(0);
         MENSAJE(24);
       END;
  UNTIL (GANANCIA > 0) OR (ORD(TECLA3) = 27);
  APAGACURSOR;
  TEXTBACKGROUND(0);
  CLRSCR;
```

```
END;

{=========================================================================}
{ RESUELVE LA ECUACION EN DIFERENCIA PARA EL CONTROLADOR                  }

PROCEDURE EC_DIFERENCIA (GRADOZPC,GRADOZOC : INTEGER; {GRADO DE POLOS Y
                                                       CEROS DEL CONTROLADOR}
                         COEFIPC,COEFZOC   : POLINOMIO);{COEFICIENTES DE
                                                         POLOS Y CEROS DEL CONTROLADOR }
BEGIN
    S[GRADOZPC]:=0;
    FOR I:=0 TO GRADOZOC DO
        IF (KC+I-GRADOZPC) >= 0 THEN
           S[GRADOZPC]:=S[GRADOZPC]+COEFZOC[I]*FUNCIONT(TM*(KC+I-GRADOZPC));
    FOR I:=0 TO (GRADOZPC-1) DO
        IF (KC+I-GRADOZPC) >= 0 THEN
           S[GRADOZPC]:=S[GRADOZPC] - COEFZPC[I]*S[I];
    IF TIPO = 'H' THEN
       SALIDACONT1 := S[GRADOZPC]
    ELSE
      BEGIN
        ARREGLODATOS(S[GRADOZPC],PRIMERDCONT1,ULTIMDCONT1);
        PUNTOCCONT:=ROUND(ESCALACONT*KC);
        SALIDACONT[PUNTOCONT]:=S[GRADOZPC]
      END;
    FOR I:=0 TO (GRADOZPC-1) DO S[I]:=S[I+1];
    IF (ABS(S[GRADOZPC])>1E10) THEN
      BEGIN
        MENSAJE(0);
        REBOSAMIENTO:=TRUE;
        MENSAJE(4);
      END;
    KC:=KC+1;
END;

{=========================================================================}
{ RESUELVE LA ECUACION EN DIFERENCIA PARA EL CONTROLADOR 2 (PARA DISEÑO)   }

PROCEDURE EC_DIFERENCIA2;
BEGIN
    S2[GRADPOLR]:=0;
    FOR I:=0 TO GRADPOLS DO
        IF (KC+I-GRADPOLR) >= 0 THEN
            S2[GRADPOLR]:=S2[GRADPOLR]+POLS[I]*FUNCIONT(TM*(KC+I-GRADPOLR));
    FOR I:=0 TO (GRADPOLR-1) DO
        IF (KC+I-GRADPOLR) >= 0 THEN
           S2[GRADPOLR]:=S2[GRADPOLR] - POLR[I]*S2[I];
    SALIDACONT2 := S2[GRADPOLR];
    FOR I:=0 TO (GRADPOLR-1) DO S2[I]:=S2[I+1];
    IF (ABS(S2[GRADPOLR])>1E10) THEN
      BEGIN
        MENSAJE(0);
        REBOSAMIENTO:=TRUE;
        MENSAJE(4);
      END;
END;
```

```
{=============================================================================}
{ BORRA DE MEMORIA LOS ARREGLOS QUE YA NO SE UTILIZARAN                       }

PROCEDURE BORRARREGLO(VAR ULTIMO : APUNTADOR); { APUNTADOR DE REFERENCIA AL
                                                 ULTIMO ELEMENTO DEL ARREGLO}
VAR DATO : APUNTADOR; { APUNTADOR AUXILIAR }

BEGIN
  WHILE ULTIMO <> NIL DO
    BEGIN
      DATO := ULTIMO;
      ULTIMO := DATO^.ANTERIOR;
      DISPOSE(DATO);
    END;
END;

{=============================================================================}
{ INICIALIZA LAS VARIABLES PARA LA PLANTA                                     }

PROCEDURE INICIALIZA;
BEGIN
  T:=0; JR:=0;
  SALIDAPLANTA[0]:=0;
  FOR I:=1 TO GRADOPP DO Y[I]:=0;
  REBOSAMIENTO:=FALSE;
  IF TIPO = 'P' THEN
     H := TIEMPO/550
  ELSE
     H:=TM/10;
  NoPUNTOSPLANTA:=INT(TIEMPO/H);
  IF NoPUNTOSPLANTA > 550 THEN
     ESCALAPLANTA:=550/NoPUNTOSPLANTA
  ELSE
     ESCALAPLANTA:=1;
  BORRARREGLO(ULTIMOPLANTA);
  PRIMEROPLANTA:=NIL;
  IF TIPO <> 'P' THEN
     ARREGLODATOS(0,PRIMEROPLANTA,ULTIMOPLANTA);
END;

{=============================================================================}
{ INICIALIZA LAS VARIABLES PARA EL CONTROLADOR                                }

PROCEDURE INICIALIZACONT;
BEGIN
  KC:=0;
  NoPUNTOSCONT:=INT(TIEMPO/TM-1);
  IF NoPUNTOSCONT > 550 THEN
     ESCALACONT:=550/NoPUNTOSCONT
  ELSE
     ESCALACONT:=1;
  BORRARREGLO(ULTIMOCONT1);
  BORRARREGLO(ULTIMOCONT2);
  PRIMEROCONT1:=NIL;
  PRIMEROCONT2:=NIL;
END;
```

```
{=================================================================}
( MENU DE LECTURA POR RAICES O POR COEFICIENTES DE LA FUNCION DE TRANSFE-  )
( RENCIA DE LA PLANTA, EL CONTROLADOR O EL SISTEMA                         )

PROCEDURE MENUPLANTACONT;

VAR  N : INTEGER;              ( CONTADOR AUXILIAR )
     TIPOPOL : STRING;         ( IDENTIFICADOR DEL TIPO DE POLINOMIO QUE SE
                                 LEERA )
     DENOMINADOR_OK : BOOLEAN; (BANDERA DE ERROR PARA VALIDAR EL DENOMINADOR)

BEGIN
  N := 49;
  TIPOPOL := 'numerador  ';
  DENOMINADOR_OK := FALSE;
  REPEAT
    IF ORD(TECLA1) = 49 THEN
       BEGIN
         WRITE('Función de transferencia de la planta':59);
         IF TIPO='P' THEN
           PLANTA('P')
         ELSE
           IF TIPO = 'H' THEN
              LAZOCONT('P')
           ELSE LAZO('P');
       END
    ELSE
      IF ORD(TECLA1) = 51 THEN
         BEGIN
           WRITE('Función de transferencia del controlador':60);
           LAZO('C');
         END
      ELSE
        BEGIN
          WRITE('Función de transferencia del sistema':59);
          PLANTA('H');
        END;
    REPEAT
      TEXTBACKGROUND(3);
      GOTOXY(19,12);
      WRITE('┌──────────────────────────────────────┐'); GOTOXY(19,13);
      WRITE('│1) Coeficientes del polinomio ',TIPOPOL,'  │'); GOTOXY(19,14);
      WRITE('│2) Raíces del polinomio ',TIPOPOL,'        │'); GOTOXY(19,15);
      WRITE('└──────────────────────────────────────┘');
      TEXTBACKGROUND(5);
      GOTOXY(20,N-36);
      CASE N OF
          49 : WRITE('1) Coeficientes del polinomio ',TIPOPOL,'  ');
          50 : WRITE('2) Raíces del polinomio ',TIPOPOL,'        ');
      END; (DEL CASE)
      TECLA2 := READKEY;
      CASE ORD(TECLA2) OF
          72 : BEGIN
                 N := N-1;
                 IF N < 49 THEN N := 50;
               END;
```

```
      80 : BEGIN
             N := N+1;
             IF N > 50 THEN N := 49;
           END;
  END;
UNTIL ((ORD(TECLA2)>48) AND (ORD(TECLA2)<51)) OR (ORD(TECLA2)=13) OR
      (ORD(TECLA2)=27);
IF ORD(TECLA2) = 13 THEN TECLA2 := CHR(N);
TEXTBACKGROUND(0);
CLRSCR;
IF ORD(TECLA2) <> 27 THEN
  BEGIN
    CASE ORD(TECLA1) OF
      49 : BEGIN
             WRITE('Función de transferencia de la planta':59);
             IF TIPO = 'P' THEN
               PLANTA('P')
             ELSE
               IF TIPO = 'H' THEN
                 LAZOCONT('P')
               ELSE
                 LAZO('P');
             CASE ORD(TECLA2) OF
               49 : IF TIPOPOL = 'numerador  ' THEN
                      LEEPOLINOMIO(GRADOOP,COEFSOP,FALSE,'P')
                    ELSE
                     BEGIN
                      LEEPOLINOMIO(GRADOPP,COEFSPP,TRUE,'P');
                      IF ORD(TECLA3) <> 27 THEN
                        BEGIN
                          NORMALIZA(GRADOPP,GRADOOP,COEFSPP,COEFSOP);
                          BUSCATIEMPO;
                        END;
                     END;
               50 : IF TIPOPOL = 'numerador  ' THEN
                      LEEPOLRAIZ(GRADOOP,COEFSOP,FALSE,'P')
                    ELSE
                      BEGIN
                       LEEPOLRAIZ(GRADOPP,COEFSPP,TRUE,'P');
                       IF ORD(TECLA3) <> 27 THEN
                         BEGIN
                           NORMALIZA(GRADOPP,GRADOOP,COEFSPP,COEFSOP);
                           BUSCATIEMPO;
                         END;
                      END;
             END; (DEL CASE)
           END;
      51 : BEGIN
             WRITE('Función de transferencia del controlador':60);
             LAZO('C');
             CASE ORD(TECLA2) OF
                  49 : IF TIPOPOL = 'numerador  ' THEN
                         LEEPOLINOMIO(GRADOZOC,COEFZOC,FALSE,'C')
                       ELSE
                         BEGIN
                           LEEPOLINOMIO(GRADOZPC,COEFZPC,TRUE,'C');
```

```
                        NORMALIZA(GRADOZPC,GRADOZOC,COEFZPC,COEFZOC)
                      END;
                50 : IF TIPOPOL = 'numerador  ' THEN
                      LEEPOLRAIZ(GRADOZOC,COEFZOC,FALSE,'C')
                    ELSE
                      BEGIN
                        LEEPOLRAIZ(GRADOZPC,COEFZPC,TRUE,'C');
                        NORMALIZA(GRADOZPC,GRADOZOC,COEFZPC,COEFZOC)
                      END;
              END; (DEL CASE)
            END;
        54 : BEGIN
              WRITE('FUNCION DE TRANSFERENCIA DEL SISTEMA':59);
              PLANTA('H');
              CASE ORD(TECLA2) OF
                49 : IF TIPOPOL = 'numerador  ' THEN
                       LEEPOLINOMIO(GRADPOLBMS,POLBMS,FALSE,'H')
                     ELSE
                       BEGIN
                         LEEPOLINOMIO(GRADPOLAMS,POLAMS,TRUE,'H');
                         NORMALIZA(GRADPOLAMS,GRADPOLBMS,POLAMS,POLBMS);
                       END;
                50 : IF TIPOPOL = 'numerador  ' THEN
                       BEGIN
                         LEEPOLRAIZ(GRADPOLBMS,POLBMS,FALSE,'H');
                       END
                     ELSE
                       BEGIN
                         LEEPOLRAIZ(GRADPOLAMS,POLAMS,TRUE,'H');
                         NORMALIZA(GRADPOLAMS,GRADPOLBMS,POLAMS,POLBMS);
                       END;
              END; (DEL CASE)
            END;
      END; (DEL CASE)
      IF ORD(TECLA3) <> 27 THEN
         IF TIPOPOL = 'denominador' THEN
           DENOMINADOR_OK := TRUE
         ELSE
           TIPOPOL := 'denominador';
    END;
  UNTIL (ORD(TECLA2) = 27) OR DENOMINADOR_OK;
END;

{=============================================================================}
{ PROCESA EL SISTEMA EN LAZO ABIERTO                                          }

PROCEDURE PROCESA_PLANTA;

BEGIN
  IF DATOS='1 2        7' THEN
    BEGIN
      PROCESANDO;
      INICIALIZA;
      RESPUESTA(GRADODP,GRADOPP,COEFSOP,COEFSPP);
      DATOS_PROCESADOS := TRUE;
      TEXTBACKGROUND(0);
```

```
      CLRSCR;
    END
  ELSE
    BEGIN
      MENSAJE(0);
      MENSAJE(5);
    END;
END;

{=========================================================================}
{ PROCESA EL SISTEMA EN LAZO CERRADO CON CONTROLADOR                      }

PROCEDURE PROCESA_CONTROLADOR;
VAR K : INTEGER;  { CONTADOR AUXILIAR PARA EVALUAR LA PLANTA }

BEGIN
  IF DATOS='1 2 3 4 5   7' THEN
    BEGIN
      PROCESANDO;
      INICIALIZA;
      INICIALIZACONT;
      SENAL_ERROR;
      REPEAT
        BLOQUE := 'C';
        EC_DIFERENCIA(GRADOZPC,GRADOZOC,COEFZPC,COEFZOC);
        K:=0;
        WHILE (K < 10) AND (NOT REBOSAMIENTO) DO
          BEGIN
            BLOQUE:='P';
            RUNGE;
            SENAL_ERROR;
            K:=K+1;
          END;
      UNTIL (JR)=NoPUNTOSPLANTA) OR (REBOSAMIENTO);
      DATOS_PROCESADOS := TRUE;
      TEXTBACKGROUND(0);
      CLRSCR;
    END
  ELSE
    BEGIN
      MENSAJE(0);
      MENSAJE(5);
    END;
END;

{=========================================================================}
{ PROCESA EL SISTEMA CON EL CONTROLADOR DISENADO                          }

PROCEDURE PROCESA_DISENO;
VAR K : INTEGER;              { CONTADOR AUXILIAR PARA EVALUAR LA PLANTA }
    SALIDA_CONTROLADOR : REAL; { SALIDA ACTUAL DEL CONTROLADOR }

BEGIN
  IF DATOS='1 2    5 6 7' THEN
    BEGIN
      PROCESANDO;
```

```
GRADPOLB := GRADOOP;
GRADPOLA := GRADOPP;
POLB     := COE OP;
POLA     := COEF5PP;
DISCRETPLANTA(GRADPOLB,GRADPOLA,POLB,POLA);
IF SINGULAR THEN
  BEGIN
    IF NO_DISCRETO OR NOT LECTURA_NODISCRETA THEN
       BEGIN
         GRADPOLBM := GRADPOLBMS;
         GRADPOLAM := GRADPOLAMS;
         POLBM     := POLBMS;
         POLAM     := POLAMS;
       END;
    IF NO_DISCRETO THEN
      BEGIN
        DISCRETPLANTA(GRADPOLBM,GRADPOLAM,POLBM,POLAM);
        IF NOT SINGULAR THEN
          BEGIN
            MENSAJE(0);
            MENSAJE(13);
          END
        ELSE
          NO_DISCRETO := FALSE;
      END;
  END
ELSE
  BEGIN
    MENSAJE(0);
    MENSAJE(10);
  END;
IF SINGULAR THEN
  BEGIN
    IF (GRADPOLAM >= GRADPOLA) THEN
      BEGIN
        CREABMASBMENOSBM1T;
        OPTENMATRIZ;
        CREA_R1_S_R;
        IF SINGULAR THEN
          BEGIN
            INICIALIZA;
            INICIALIZACONT;
            REPEAT
              BLOQUE := 'S';
              EC_DIFERENCIA2;
              IF NOT REBOSAMIENTO THEN
                BEGIN
                  BLOQUE := 'T';
                  EC_DIFERENCIA(GRADPOLR,GRADPOLT,POLR,POLT);
                  SALIDA_CONTROLADOR := SALIDACONT1-SALIDACONT2;
                  PUNTOCONT := ROUND(ESCALACONT*(KC-1));
                  SALIDACONT[PUNTOCONT] := SALIDA_CONTROLADOR;
                  ARREGLODATOS(SALIDA_CONTROLADOR,PRIMEROCONT1,
                               ULTIMOCONT1);
                END;
              BLOQUE := 'P';
```

```
                    K:=0;
                    WHILE (K < 10) AND (NOT REBOSAMIENTO) DO
                      BEGIN
                        RUNGE;
                        K:=K+1;
                      END;
                  UNTIL (JR >= NoPUNTOSPLANTA) OR REBOSAMIENTO;
                  DATOS_PROCESADOS := TRUE;
                END
              ELSE
                BEGIN
                  MENSAJE(0);
                  MENSAJE(11);
                END;
            END
          ELSE
            BEGIN
              MENSAJE(0);
              MENSAJE(14);
            END;
          TEXTBACKGROUND(0);
          CLRSCR;
        END;
      END
    ELSE
      BEGIN
        MENSAJE(0);
        MENSAJE(5);
      END;
END;

{=============================================================================}
{ PRESENTA EL MENU DE CARGA DE DATOS HACIENDO LAS LLAMADAS NECESARIAS        }
{ A LOS PROCEDIMIENTOS DE LECTURA DE DATOS                                  }

PROCEDURE MENUCARGA;
VAR M : INTEGER; { CONTADOR AUXILIAR }

BEGIN
 M:=49;
 REPEAT
  WRITE('Carga de datos':48);
  GOTOXY(24,9);
  WRITE('┌'); GOTOXY(24,10);
  WRITE('│'); GOTOXY(24,11);
  WRITE('┌'); GOTOXY(24,12);
  WRITE('│'); GOTOXY(24,13);
  WRITE('└'); GOTOXY(24,14);
  WRITE('└'); GOTOXY(17,9);
  WRITE('Planta'); GOTOXY(12,12);
  WRITE('Controlador'); GOTOXY(17,14);
  WRITE('Diseño');
  GOTOXY(22,20);
  WRITE('Opciones seleccionadas: ',DATOS);
  REPEAT
   TEXTBACKGROUND(3);
```

```
GOTOXY(26,8);
WRITE('┌──────────────────────────┐'); GOTOXY(26,9');
WRITE('│1) Datos de la planta     │'); GOTOXY(26,10);
WRITE('│2) Tiempo a analizar      │'); GOTOXY(26,11);
WRITE('│3) Datos del controlador  │'); GOTOXY(26,12);
WRITE('│4) Ganancia de lazo       │'); GOTOXY(26,13);
WRITE('│5) Periodo de muestreo    │'); GOTOXY(26,14);
WRITE('│6) Datos del lazo (H(z))  │'); GOTOXY(26,15);
WRITE('│7) Tipo de entrada (R(t)) │'); GOTOXY(26,16);
WRITE('│8) Procesar datos         │'); GOTOXY(26,17);
WRITE('└──────────────────────────┘');
TEXTBACKGROUND(5);
GOTOXY(27,M-40);
CASE M OF
     49 : WRITE('1) Datos de la planta      ');
     50 : WRITE('2) Tiempo a analizar       ');
     51 : WRITE('3) Datos del controlador   ');
     52 : WRITE('4) Ganancia de lazo        ');
     53 : WRITE('5) Periodo de muestreo     ');
     54 : WRITE('6) Datos del lazo (H(z))   ');
     55 : WRITE('7) Tipo de entrada (R(t))  ');
     56 : WRITE('8) Procesar datos          ');
END; (DEL CASE)
 TECLA1:=READKEY;
 CASE ORD(TECLA1) OF
     72 : BEGIN
          M:=M-1;
          IF M<49 THEN M:=56;
          END;
     80 : BEGIN
          M:=M+1;
          IF M>56 THEN M:=49;
          END;
 END; (DEL CASE)
UNTIL ((ORD(TECLA1)>48) AND (ORD(TECLA1)<57)) OR (ORD(TECLA1)=13) OR
      (ORD(TECLA1)=27);
IF ORD(TECLA1)=13 THEN TECLA1:=CHR(M);
TEXTBACKGROUND(0);
CLRSCR;
CASE ORD(TECLA1) OF
    49 : BEGIN
         MENUPLANTACONT;
         IF (ORD(TECLA2) <> 27) AND (ORD(TECLA3) <> 27) THEN
           BEGIN
             DELETE(DATOS,1,1);
             INSERT('1',DATOS,1);
           END
         ELSE
           BEGIN
             DELETE(DATOS,1,1);
             INSERT(' ',DATOS,1);
           END
        END;
    50 : BEGIN
         IF TIPO='P' THEN
           PLANTA('P')
```

```
     ELSE
        IF TIPO = 'H' THEN
           PLANTA('H')
        ELSE LAZO('P');
     TIEMPODEANALISIS;
     IF (POS(' 2',DATOS)=0) AND (ORD(TECLA3)<>27) THEN
        BEGIN
         DELETE(DATOS,2,2);
         INSERT(' 2',DATOS,2);
        END;
    END;
51 : IF TIPO='C' THEN
       BEGIN
        MENUPLANTACONT;
        IF (ORD(TECLA2) <> 27) AND (ORD(TECLA3) <> 27) THEN
           BEGIN
            DELETE(DATOS,4,2);
            INSERT(' 3',DATOS,4);
           END
        ELSE
          BEGIN
            DELETE(DATOS,4,2);
            INSERT('  ',DATOS,4);
          END;
       END;
52 : IF TIPO='C' THEN
       BEGIN
        LAZO('S');
        GANANCIA_LAZO;
        IF (POS(' 4',DATOS)=0) AND (ORD(TECLA3)<>27) THEN
           BEGIN
            DELETE(DATOS,6,2);
            INSERT(' 4',DATOS,6);
           END;
       END;
53 : IF (TIPO='C') OR (TIPO = 'H') THEN
       BEGIN
        IF TIPO = 'C' THEN
           LAZO('C')
        ELSE
           LAZOCONT('M');
        PARAMETROSCONT;
        IF (POS(' 5',DATOS)=0) AND (ORD(TECLA3)<>27) THEN
           BEGIN
            DELETE(DATOS,8,2);
            INSERT(' 5',DATOS,8);
           END;
       END;
54 : IF TIPO = 'H' THEN
    BEGIN
     PLANTA('H');
     DISCRETO;
     IF ORD(TECLA3) <> 27 THEN
        MENUPLANTACONT;
     IF (ORD(TECLA2) <> 27) AND (ORD(TECLA3) <> 27) THEN
        BEGIN
```

```
                DELETE(DATOS,10,2);
                INSERT(' 6',DATOS,10);
              END
            ELSE
              BEGIN
                DELETE(DATOS,10,2);
                INSERT('  ',DATOS,10);
              END;
          END;
     55 : BEGIN
          IF TIPO='C' THEN
            LAZO('R')
          ELSE
            IF TIPO='H' THEN
              LAZOCONT('R')
            ELSE PLANTA('R');
          LEEENTRADA;
          IF (POS(' 7',DATOS)=0) AND (ORD(TECLA2)<>27) AND
             (ORD(TECLA3)<>27) THEN
             BEGIN
              DELETE(DATOS,12,2);
              INSERT(' 7',DATOS,12);
             END;
          END;
     56 : BEGIN
          CASE TIPO OF
            'P': PROCESA_PLANTA;
            'C': PROCESA_CONTROLADOR;
            'H': PROCESA_DISEND;
          END; {DEL CASE}
          END;
  END; { DEL CASE }
  GOTOXY(22,20);
  WRITE('Opciones seleccionadas: ',DATOS);
  GOTOXY(1,1);
 UNTIL ORD(TECLA1)=27;
 TEXTBACKGROUND(0);
 CLRSCR;
END;

{=============================================================================}
{ PRESENTA EL MENU DE SELECCION DEL SISTEMA A ANALIZAR                        }

PROCEDURE TIPOSISTEMA;
VAR
 CAMBIOPARAMETROS: BOOLEAN; { BANDERA PARA CAMBIO DE PARAMETROS }

BEGIN
  WRITE('Tipo de sistema :':48);
  I:=49;
  REPEAT
    TEXTBACKGROUND(3);
    GOTOXY(15,9);
    WRITE('┌──────────────────────────────────────────────┐'); GOTOXY(15,10);
    WRITE('│1) Planta (G(s)) en malla abierta             │'); GOTOXY(15,11);
    WRITE('│2) Sistema con controlador (G(z)) en lazo cerrado│'); GOTOXY(15,12);
```

```
  WRITE('|3) Diseño del controlador a partir de H(z)        |'); GOTOXY(15,13);
  WRITE('|4) Cambio en los parámetros                        |'); GOTOXY(15,14);
  WRITE('└──────────────────────────────────────────────────┘');
  TEXTBACKGROUND(5);
  GOTOXY(16,I-39);
  CASE I OF
       49 : WRITE('1) Planta (G(s)) en malla abierta                ');
       50 : WRITE('2) Sistema con controlador (G(z)) en lazo cerrado ');
       51 : WRITE('3) Diseño del controlador a partir de H(z)        ');
       52 : WRITE('4) Cambio en los parámetros                       ');
  END; (DEL CASE)
  TECLA:=READKEY;
  CASE ORD(TECLA) OF
       72 : BEGIN
            I:=I-1;
            IF I<49 THEN I:=52;
           END;
       80 : BEGIN
            I:=I+1;
            IF I>52 THEN I:=49;
           END;
  END; (DEL CASE)
UNTIL ((ORD(TECLA)>48) AND (ORD(TECLA)<53)) OR (ORD(TECLA)=13) OR
      (ORD(TECLA)=27);
IF ORD(TECLA)=13 THEN TECLA:=CHR(I);
TEXTBACKGROUND(0);
CLRSCR;
IF (ORD(TECLA)=52) AND ((DATOS='1 2        7') OR (DATOS='1 2 3 4 5   7')
    OR (DATOS='1 2    5 6 7')) THEN
    BEGIN
      CAMBIOPARAMETROS:=TRUE;
      IF TIPO='C' THEN
         TECLA:=CHR(50)
      ELSE
         IF TIPO = 'H' THEN
            TECLA := CHR(51)
         ELSE TECLA:=CHR(49);
    END
ELSE CAMBIOPARAMETROS:=FALSE;
CASE ORD(TECLA) OF
      49: BEGIN
           TIPO:='P';
           IF CAMBIOPARAMETROS=FALSE THEN
              BEGIN
                DATOS:='            ';
                DATOS_PROCESADOS := FALSE;
              END;
           MENUCARGA;
          END;
      50: BEGIN
           TIPO:='C';
           IF CAMBIOPARAMETROS=FALSE THEN
              BEGIN
                DATOS:='            ';
                DATOS_PROCESADOS := FALSE;
              END;
```

```
          MENUCARGA;
         END;
      51: BEGIN
          TIPO:='H';
          IF CAMBIOPARAMETROS=FALSE THEN
            BEGIN
              DATOS:='           ';
              DATOS_PROCESADOS := FALSE;
            END;
          SINGULAR := TRUE;
          MENUCARGA;
         END;
  END; (DEL CASE)
END;

{=====================================================================}
{ PRESENTA LOS RESULTADOS DEL SISTEMA ANALIZADO                       }

PROCEDURE RESULTADOS(P:INTEGER; { NUMERO DE PUNTOS A GRAFICAR }
                     PRIMERO: APUNTADOR; { APUNTADOR DE REFERENCIA }
                     INC:REAL); { INCREMENTO EN EL TIEMPO }
VAR
 M,I : INTEGER; { CONTADORES AUXILIARES }
 APUNTADORSUP, APUNTADORINF: APUNTADOR; { APUNTADORES PARA REFERENCIAR AL
                                          PRIMERO Y ULTIMO ELEMENTOS MOSTRA-
                                          DOS EN PANTALLA }
BEGIN
   WRITELN;
   WRITE('Salida del sistema :':49);
   TEXTBACKGROUND(3);
   GOTOXY(14,10);
   WRITE('┌──────┬──────────────────────┐'); GOTOXY(14,11);
   WRITE('│  t   │      Salida(t)       │'); GOTOXY(14,12);
   WRITE('├──────┼──────────────────────┤'); GOTOXY(14,13);
   WRITE('│      │                      │'); GOTOXY(14,14);
   WRITE('│      │                      │'); GOTOXY(14,15);
   WRITE('│      │                      │'); GOTOXY(14,16);
   WRITE('│      │                      │'); GOTOXY(14,17);
   WRITE('│      │                      │'); GOTOXY(14,18);
   WRITE('│      │                      │'); GOTOXY(14,19);
   WRITE('│      │                      │'); GOTOXY(14,20);
   WRITE('│      │                      │'); GOTOXY(14,21);
   WRITE('│      │                      │'); GOTOXY(14,22);
   WRITE('│      │                      │'); GOTOXY(14,23);
   WRITE('└──────┴──────────────────────┘');
   WINDOW(15,13,64,22);
   GOTOXY(1,1);
   APUNTADORSUP:=PRIMERO;
   APUNTADORINF:=PRIMERO;
   I:=0;
   IF P>=9 THEN
      FOR I:=0 TO 8 DO
         BEGIN
            WRITELN(I*INC:9:3,'  ':4,'|','  ':2,APUNTADORSUP^.DATO:20:7);
            APUNTADORSUP:=APUNTADORSUP^.SIGUIENTE;
         END
```

```
  ELSE
    FOR I:=0 TO P-1 DO
        BEGIN
          WRITELN(I@INC:9:3,'  ':4,'|',' ':2,APUNTADORSUP^.DATO:20:7);
          APUNTADORSUP:=APUNTADORSUP^.SIGUIENTE;
        END;
   I:=I+1;
   WRITE(I@INC:9:3,'  ':4,'|',' ':2,APUNTADORSUP^.DATO:20:7);
   I:=0;
   REPEAT
    REPEAT TECLA2:=READKEY
    UNTIL (ORD(TECLA2)=0) OR (ORD(TECLA2)=80) OR (ORD(TECLA2)=73) OR
          (ORD(TECLA2)=81) OR (ORD(TECLA2)=27);
    IF ORD(TECLA2)<>27 THEN TECLA2:=READKEY;
    CASE ORD(TECLA2) OF
      72,73 : BEGIN
                IF ORD(TECLA2)=72 THEN
                   M:=9
                ELSE M:=0;
                WHILE (I>0) AND (M<>I0) DO
                BEGIN
                  M:=M+1;
                  GOTOXY(1,I0);
                  DELLINE;
                  GOTOXY(1,1);
                  INSLINE;
                  I:=I-1;
                  APUNTADORINF:=APUNTADORINF^.ANTERIOR;
                  APUNTADORSUP:=APUNTADORSUP^.ANTERIOR;
                  WRITE(I@INC:9:3,'  ':4,'|',' ':2,APUNTADORINF^.DATO:20:7);
                END
              END;
      80,81 : BEGIN
                IF ORD(TECLA2)=80 THEN
                   M:=9
                ELSE M:=0;
                WHILE (I<(P-9)) AND (M<>I0) DO
                BEGIN
                  M:=M+1;
                  GOTOXY(1,1);
                  DELLINE;
                  GOTOXY(1,I0);
                  INSLINE;
                  I:=I+10;
                  APUNTADORINF:=APUNTADORINF^.SIGUIENTE;
                  APUNTADORSUP:=APUNTADORSUP^.SIGUIENTE;
                  WRITE(I@INC:9:3,'  ':4,'|',' ':2,APUNTADORSUP^.DATO:20:7);
                  I:=I-9;
                END
              END;
    END; {DELCASE}
  UNTIL ORD(TECLA2)=27 ;
  WINDOW(1,1,80,25);
  TEXTBACKGROUND(0);
  CLRSCR;
END;
```

```
{=====================================================================}
{ MUESTRA LOS POLINOMIOS DEL CONTROLADOR DISEÑADO                      }

PROCEDURE MUESTRACONTROLADOR;

BEGIN
  LAZOCONT('C');
  TEXTBACKGROUND(5);
  WRITE ('Grado en z':12);
  WRITE ('Polinomio R':24);
  WRITE ('Polinomio S':20);
  WRITE ('Polinomio T ':21);
  TEXTBACKGROUND(0);
  WRITELN;
  FOR I := 0 TO GRADPOLR DO
  BEGIN
     WRITE('    z^',I:2,' ':15,POLR[I]:13:4,' ':7);
     IF I <= GRADPOLS THEN
        WRITE (POLS[I]:13:4,' ':7)
     ELSE
       WRITE(' ':20);
     IF I <= GRADPOLT THEN
        WRITE (POLT[I]:13:4);
     WRITELN;
   END;
   REPEAT TECLA3:=READKEY UNTIL ORD(TECLA3)=27;
   CLRSCR;
END;

{=====================================================================}
{ MENU DE SELECCION DE LA SEÑAL A TABULAR                              }

PROCEDURE MENUTABULACION;
BEGIN
 REPEAT
   WRITE('Tabulación':45);
   IF TIPO='C' THEN
      LAID('6')
   ELSE
     IF TIPO='H' THEN
       LAZOCONT('M')
     ELSE
       PLANTA('6');
   I:=49;
   REPEAT
     TEXTBACKGROUND(3);
     GOTOXY(32,13);
     WRITE('┌──────────────────┐'); GOTOXY(32,14);
     WRITE('│1) Tabular Y(t)   │'); GOTOXY(32,15);
     WRITE('│2) Tabular U(T)   │'); GOTOXY(32,16);
     WRITE('│3) Tabular E(t)   │'); GOTOXY(32,17);
     WRITE('└──────────────────┘');
     TEXTBACKGROUND(5);
     GOTOXY(33,I-35);
     CASE I OF
         49 : WRITE('1) Tabular Y(t)');
```

```
        50 : WRITE('2) Tabular U(T)');
        51 : WRITE('3) Tabular E(t)');
    END; {DEL CASE}
    TECLA1:=READKEY;
    CASE ORD(TECLA1) OF
        72 : BEGIN
              I:=I-1;
              IF I<49 THEN I:=51;
             END;
        80 : BEGIN
              I:=I+1;
              IF I>51 THEN I:=49;
             END;
    END; {DEL CASE}
  UNTIL ((ORD(TECLA1)>48) AND (ORD(TECLA1)<52)) OR (ORD(TECLA1)=13) OR
        (ORD(TECLA1)=27);
  IF ORD(TECLA1)=13 THEN TECLA1:=CHR(I);
  TEXTBACKGROUND(0);
  CLRSCR;
  CASE ORD(TECLA1) OF
      49: RESULTADOS(JR,PRIMEROPLANTA,H);
      50: IF (TIPO='C') OR (TIPO='H') THEN
            RESULTADOS(KC-1,PRIMEROCONT1,TM);
      51: IF TIPO='C' THEN
            RESULTADOS(JR,PRIMEROCONT2,H);
  END; {DEL CASE}
 UNTIL ORD(TECLA1)=27;
END;

{=============================================================================}
{ MENU DE SELECCION DE LA SEÑAL A GRAFICAR                                    }

PROCEDURE MENUGRAFICAR;
BEGIN
 REPEAT
   WRITE('Graficación':45);
   IF TIPO='C' THEN
      LAZO('G')
   ELSE
     IF TIPO='H' THEN
       LAZOCONT('H')
     ELSE
       PLANTA('G');
   I:=49;
   REPEAT
     TEXTBACKGROUND(3);
     GOTOXY(32,13);
     WRITE('┌──────────────────┐'); GOTOXY(32,14);
     WRITE('│1) Graficar Y(t)  │'); GOTOXY(32,15);
     WRITE('│2) Graficar U(T)  │'); GOTOXY(32,16);
     WRITE('│3) Graficar E(t)  │'); GOTOXY(32,17);
     WRITE('└──────────────────┘');
     TEXTBACKGROUND(5);
     GOTOXY(33,I-35);
     CASE I OF
         49 : WRITE('1) Graficar Y(t)');
```

```
        50 : WRITE('2) Graficar U(T)');
        51 : WRITE('3) Graficar E(t)');
    END; (DEL CASE)
    TECLA1:=READKEY;
    CASE ORD(TECLA1) OF
        72 : BEGIN
              I:=I-1;
              IF I<49 THEN I:=51;
             END;
        80 : BEGIN
              I:=I+1;
              IF I>51 THEN I:=49;
             END;
    END; (DEL CASE)
  UNTIL ((ORD(TECLA1)>48) AND (ORD(TECLA1)<52)) OR (ORD(TECLA1)=13) OR
        (ORD(TECLA1)=27);
  IF ORD(TECLA1)=13 THEN TECLA1:=CHR(I);
  TEXTBACKGROUND(0);
  CLRSCR;
  CASE ORD(TECLA1) OF
      49: BEGIN
           CHECA_MONITOR(PUNTOPLANTA,SALIDAPLANTA);
           GRAFICAS(PUNTOPLANTA,SALIDAPLANTA,'P');
           MUEVE_CURSOR;
           ULTIMAGRAF:= 'P'
          END;
      50: IF (TIPO='C') OR (TIPO='H') THEN
           BEGIN
             CHECA_MONITOR(PUNTOCONT,SALIDACONT);
             GRAFICAS(PUNTOCONT,SALIDACONT,'C');
             MUEVE_CURSOR;
             ULTIMAGRAF:='C'
           END;
      51 : IF TIPO = 'C' THEN
             BEGIN
               CHECA_MONITOR(PUNTOERROR,SALIDAERROR);
               GRAFICAS(PUNTOERROR,SALIDAERROR,'E');
               MUEVE_CURSOR;
               ULTIMAGRAF := 'E';
             END;
  END; (DEL CASE)
 UNTIL ORD(TECLA1)=27;
END;

{=========================================================================}
{ MENU DE MANIPULACION DE RESULTADOS                                      }

PROCEDURE MENU_RESULTADOS;
BEGIN
 GRAFICA_DISCO := FALSE;
 REPEAT
  WRITE('Menú de resultados':49);
  I:=49;
  REPEAT
   TEXTBACKGROUND(3);
   GOTOXY(30,10);
```

```
 WRITE('┌──────────────────┐'); GOTOXY(30,11);
 WRITE('│1) Graficar       │'); GOTOXY(30,12);
 WRITE('│2) Tabular        │'); GOTOXY(30,13);
 WRITE('│3) Salvar gráficas│'); GOTOXY(30,14);
 WRITE('│4) Leer gráficas  │'); GOTOXY(30,15);
 WRITE('│5) Controlador    │'); GOTOXY(30,16);
 WRITE('└──────────────────┘');
 TEXTBACKGROUND(5);
 GOTOXY(31,I-38);
 CASE I OF
      49 : WRITE('1) Graficar       ');
      50 : WRITE('2) Tabular        ');
      51 : WRITE('3) Salvar gráficas ');
      52 : WRITE('4) Leer gráficas  ');
      53 : WRITE('5) Controlador    ');
 END; (DEL CASE)
 TECLA:=READKEY;
 CASE ORD(TECLA) OF
      72 : BEGIN
             I:=I-1;
             IF I<49 THEN I:=53;
           END;
      80 : BEGIN
             I:=I+1;
             IF I>53 THEN I:=49;
           END;
 END; (DEL CASE)
UNTIL ((ORD(TECLA)>48) AND (ORD(TECLA)<54)) OR (ORD(TECLA)=13) OR
      (ORD(TECLA)=27);
IF ORD(TECLA)=13 THEN TECLA:=CHR(I);
IF (ORD(TECLA)<>51) AND (ORD(TECLA)<>52) THEN
  BEGIN
    TEXTBACKGROUND(0);
    CLRSCR;
  END;
CASE ORD(TECLA) OF
     49 : IF GRAFICA_DISCO THEN
            BEGIN
              CHECA_MONITOR(PUNTOPLANTA,SALIDAPLANTA);
              FOR I := I TO NoGRAFICAS DO
                BEGIN
                  ASSIGN(ARCHGRAF,NOMBRE_GRA[I]);
                  RESET(ARCHGRAF);
                  READ(ARCHGRAF,REGGRAF);
                  CLOSE(ARCHGRAF);
                  WITH REGGRAF DO
                     GRAFICAS(NoPUNTOS,SALIDA,TIPOGRA);
                END;
              MUEVE_CURSOR;
              ULTIMAGRAF := ' '
            END
          ELSE
            IF DATOS_PROCESADOS THEN
              MENUGRAFICAR
            ELSE
              BEGIN
```

```
                MENSAJE(0);
                MENSAJE(23);
                GOTOXY(1,1)
              END;
       50 : IF DATOS_PROCESADOS THEN
              MENUTABULACION
            ELSE
              BEGIN
                MENSAJE(0);
                MENSAJE(23);
                GOTOXY(1,1)
              END;
       51 : IF DATOS_PROCESADOS THEN
              IF ULTIMAGRAF = ' ' THEN
                 BEGIN
                   MENSAJE(0);
                   MENSAJE(22);
                   TEXTBACKGROUND(0);
                   GOTOXY(1,1);
                 END
               ELSE
                 ALMACENA('.GRA')
            ELSE
              BEGIN
                MENSAJE(0);
                MENSAJE(23);
                TEXTBACKGROUND(0);
                GOTOXY(1,1)
              END;
       52 : BEGIN
              LEENoGRAFICAS;
              IF ORD(TECLA3) <> 27 THEN
                 RECUPERADATOS(NoGRAFICAS,'.GRA');
            END;
       53 : IF (DATOS='1 2    5 6 7') THEN
               IF DATOS_PROCESADOS THEN
                  MUESTRACONTROLADOR
               ELSE
                 BEGIN
                   MENSAJE(0);
                   MENSAJE(23);
                   GOTOXY(1,1)
                 END;
  END;
 UNTIL ORD(TECLA)=27;
 GRAFICA_DISCO := FALSE;
END;

{=============================================================================}
{ MENU PRINCIPAL                                                              }

PROCEDURE MENU;
VAR SALIDA : BOOLEAN; { BANDERA PARA LA SALIDA DEL PAQUETE }

BEGIN
  SALIDA := FALSE;
```

```
TEXTBACKGROUND(0);
TEXTCOLOR(15);
CLRSCR;
REPEAT
 GOTOXY(1,1);
 WRITELN;
 WRITE('Menú principal':47);
 I:=49;
 REPEAT
  TEXTBACKGROUND(3);
  GOTOXY(26,10);
  WRITE('                          '); GOTOXY(26,11);
  WRITE('| 1) Cargar datos         |'); GOTOXY(26,12);
  WRITE('| 2) Análisis de resultados|'); GOTOXY(26,13);
  WRITE('| 3) Salvar datos          |'); GOTOXY(26,14);
  WRITE('| 4) Leer datos de disco   |'); GOTOXY(26,15);
  WRITE('| 5) Consulta datos actuales|'); GOTOXY(26,16);
  WRITE('| 6) Salir                 |'); GOTOXY(26,17);
  WRITE('|__________________________|');
  TEXTBACKGROUND(5);
  GOTOXY(27,I-38);
  CASE I OF
       49 : WRITE('1) Cargar datos           ');
       50 : WRITE('2) Análisis de resultados ');
       51 : WRITE('3) Salvar datos           ');
       52 : WRITE('4) Leer datos de disco    ');
       53 : WRITE('5) Consulta datos actuales');
       54 : WRITE('6) Salir                  ');
  END; {DEL CASE}
  TECLA:=READKEY;
  CASE ORD(TECLA) OF
       72 : BEGIN
            I:=I-1;
            IF I<49 THEN I:=54;
           END;
       80 : BEGIN
            I:=I+1;
            IF I>54 THEN I:=49;
           END;
  END; {DEL CASE}
 UNTIL ((ORD(TECLA)>48) AND (ORD(TECLA)<55)) OR (ORD(TECLA)=13) ;
 IF ORD(TECLA)=13 THEN TECLA:=CHR(I);
 IF ORD(TECLA) <> 51 THEN
    BEGIN
     TEXTBACKGROUND(0);
     CLRSCR;
    END;
  CASE ORD(TECLA) OF
       49 : TIPOSISTEMA;
       50 : MENU_RESULTADOS;
       51 : IF (DATOS='1 2       7') OR (DATOS='1 2 3 4 5   7') OR
               (DATOS='1 2    5 6 7') THEN
              ALMACENA('.DSC')
            ELSE
              TEXTBACKGROUND(0);
       52 : RECUPERADATOS(1,'.DSC');
```

```
        53 : IF (DATOS='1 2       7') OR (DATOS='1 2 3 4 5   7') OR
               (DATOS='1 2     5 6 7') THEN
               DATOS_ACTUALES;
        54 : SALIDA:=TRUE;
         ELSE;
    END; (DEL CASE)
   UNTIL SALIDA;
END;

{=============================================================================}
{ PROGRAMA PRINCIPAL                                                          }

BEGIN
 PRIMEROPLANTA := NIL;
 ULTIMOPLANTA := NIL;
 PRIMEROCONTI := NIL;
 ULTIMOCONTI := NIL;
 PRIMEROCONT2 := NIL;
 ULTIMOCONT2 := NIL;
 PRIMEROARCH := NIL;
 ULTIMOARCH := NIL;
 ULTIMAGRAF := ' ';
 DATOS_PROCESADOS := FALSE;
 APAGACURSOR;
 MENU;
 PRENDECURSOR;
 BORRARREGLO(ULTIMOPLANTA);
 BORRARREGLO(ULTIMOCONT1);
 BORRARREGLO(ULTIMOCONT2);
 RESTORECRTMODE;
 TEXTBACKGROUND(0);
 CLRSCR;
END.
```

```
UNIT DISENO;

{ UNIDAD PARA EL DISENO DEL CONTROLADOR ATRAVES DEL METODO DE UBICACION DE  }
{ POLOS Y CEROS, ASI COMO PARA LA DISCRETIZACION DE FUNCIONES DE ---        }
{ TRANSFERENCIA.                                                           }

INTERFACE


USES  CRT,DECLARA;

PROCEDURE ORDENARFRACCPARCIALESREALES(RAICES : RAREP;CONTADOR : INTEGER);

PROCEDURE ORDENARFRACCPARCIALESIMAGS(RAICES : RAREP;CONTADOR : INTEGER);

PROCEDURE VALDECTES(GRADPOLBX : INTEGER; POLBX : POLINOMIO);

FUNCTION POTENCIA (XX : REAL; ENEE : ENTERONONEGATIVO) : REAL;

FUNCTION FAC (EENE:ENTERONONEGATIVO) : ENTEROPOSITIVO;

PROCEDURE MULTIPLICA (POL1,POL2 : POLINOMIO; GRADO1,GRADO2 : INTEGER;
                      VAR POLRESULTANTE : POLINOMIO ;
                      VAR GRADOPOLRESULTANTE : INTEGER);
PROCEDURE OPTPOLCOMBXAX(VAR GRADPOLX : INTEGER; VAR POLX : POLINOMIO;
                        POLR : POLIRAIZ; CONT : INTEGER);
PROCEDURE LLAMADO(POLBB : POLINOMIO; GRADPOLBB:INTEGER; AMBIENTE : STRING);
PROCEDURE DISCRETPLANTA(VAR GRADPOLBX,GRADPOLAX : INTEGER;
                      VAR POLBX,POLAX : POLINOMIO);
PROCEDURE CREABMASBMENOSBMIT;
PROCEDURE OPTENMATRIZ;
PROCEDURE CREA_R1_S_R;

IMPLEMENTATION


{=============================================================================}
{ MULTIPLICA DOS POLINOMIOS.                                                  }

PROCEDURE MULTIPLICA (POL1,POL2 : POLINOMIO;  { POLINOMIOS A MULTIPLICAR }
                      GRADO1,GRADO2 : INTEGER; { GRADO DE LOS POLINOMIOS }
                      VAR POLRESULTANTE : POLINOMIO; { POLINOMIO RESULTANTE }
                      VAR GRADOPOLRESULTANTE : INTEGER);{ GRADO DEL POLINO-
                                                          MIO RESULTANTE }
VAR I,J : INTEGER;        { CONTADORES AUXILIARES }

BEGIN
   GRADOPOLRESULTANTE := GRADO1+GRADO2;
   FOR I := 0 TO GRADOPOLRESULTANTE DO POLRESULTANTE[I] := 0;
   FOR I := 0 TO GRADO1 DO
   FOR J := 0 TO GRADO2 DO
      POLRESULTANTE[I+J] := POLRESULTANTE[I+J]+POL1[I]*POL2[J];
END;
```

```
{=========================================================================}
{ OBTIENE LOS COEFICIENTES DEL POLINOMIO APARTIR DE SUS RAICES.           }

PROCEDURE OPTPOLCOMBXAX(VAR GRADPOLX : INTEGER;  { GRADO DEL POLINOMIO
                                                   RESULTANTE }
                        VAR POLX : POLINOMIO;    { POLINOMIO RESULTANTE }
                        POLR : POLIRAIZ;         { RAICES DEL POLINOMIO }
                        CONT : INTEGER);         { NUMERO DE RAICES }

VAR I : INTEGER;        { CONTADOR AUXILIAR }

BEGIN
     IF CONT = 0 THEN
        BEGIN
          POLX[0] := 1;
          GRADPOLX := 0
        END;
     IF CONT = 1 THEN
        BEGIN
          POLX := POLR[1].POLRAIZ;
          GRADPOLX := POLR[1].ORDENPOLRAIZ;
        END
     ELSE
       BEGIN
          IF CONT >= 2 THEN
          BEGIN
               MULTIPLICA(POLR[1].POLRAIZ,POLR[2].POLRAIZ,
                    POLR[1].ORDENPOLRAIZ,POLR[2].ORDENPOLRAIZ,POLX,GRADPOLX);
               IF CONT > 2 THEN
               BEGIN
                    FOR I := 3 TO CONT DO
                    MULTIPLICA(POLR[I].POLRAIZ,POLX,POLR[I].ORDENPOLRAIZ,
                              GRADPOLX,POLX,GRADPOLX);
               END;
          END;
       END;
END;


{=========================================================================}
{ ORDENA LAS RAICES DEL POLINOMIO B DE ACUERDO A SU UBICACION EN EL CIRCULO }
{ UNITARIO.                                                               }

PROCEDURE ORDENAPOLBMASYMENOS (W,Z,    { COEFICIENTES DEL POLINOMIO }
                               REEAL,  { PARTE REAL DE LA RAIZ }
                               IMAGINARIA : REAL); { PARTE IMAGINARIA }

VAR  DISTANCIA : REAL;    { DISTANCIA AL ORIGEN }

BEGIN
     IF IMAGINARIA = 0 THEN
     BEGIN
          IF (ABS(REEAL) <= 1 ) AND (REEAL <> 0) THEN
          BEGIN
               CONTADORPOLBMAS := CONTADORPOLBMAS+1;
               POLRAIIBMAS[CONTADORPOLBMAS].ORDENPOLRAIZ := 1;
```

```
            POLRAIZBMAS[CONTADORPOLBMAS].POLRAIZ[0] := -REEAL;
            POLRAIZBMAS[CONTADORPOLBMAS].POLRAIZ[1] := 1;
        END
        ELSE
        BEGIN
            CONTADORPOLBMENOS := CONTADORPOLBMENOS+1;
            POLRAIZBMENOS[CONTADORPOLBMENOS].ORDENPOLRAIZ := 1;
            POLRAIZBMENOS[CONTADORPOLBMENOS].POLRAIZ[0] := -REEAL;
            POLRAIZBMENDS[CONTADORPOLBMENOS].POLRAIZ[1] := 1;
        END;
    END
    ELSE
    BEGIN

        { CALCULO DE LA DISTANCIA DE LA RAIZ AL ORIGEN }
        DISTANCIA := SQRT(SQR(REEAL)+SQR(IMAGINARIA));

        { CLASIFICACION DE RAICES ESTABLES E INESTABLES }
        IF (DISTANCIA <= 1) AND (DISTANCIA <> 0) THEN
        BEGIN
            CONTADORPOLBMAS := CONTADORPOLBMAS+1;
            POLRAIZBMAS[CONTADORPOLBMAS].ORDENPOLRAIZ := 2;
            POLRAIZBMAS[CONTADORPOLBMAS].POLRAIZ[0] := Z;
            POLRAIZBMAS[CONTADORPOLBMAS].POLRAIZ[1] := W;
            POLRAIZBMAS[CONTADORPOLBMAS].POLRAIZ[2] := 1 ;
        END
        ELSE
        BEGIN
            CONTADORPOLBMENOS := CONTADORPOLBMENOS+1;
            POLRAIZBMENOS[CONTADORPOLBMENOS].ORDENPOLRAIZ := 2;
            POLRAIZBMENDS[CONTADORPOLBMENOS].POLRAIZ[0] := Z;
            POLRAIZBMENOS[CONTADDRPOLBMENDS].POLRAIZ[1] := W;
            POLRAIZBMENOS[CONTADORPOLBMENOS].POLRAIZ[2] := 1
        END;
    END;
END;


{=============================================================================}
{ ORGANIZA LAS RAICES DEL POLINOMIO Y SUS REPETICIONES..                       }

PROCEDURE ORGANIZA (B,C,    { COEFICIENTES DEL POLINOMIO }
                    RREAL,  { PARTE REAL DE LA RAIZ }
                    IMAGINARIA : REAL); { PARTE IMAGINARIA DE LA RAIZ }

VAR REP : BOOLEAN;          { INDICA REPETICION }
    I,J : INTEGER;          { CONTADORES AUXILIARES }

BEGIN
  IF IMAGINARIA = 0 THEN
    BEGIN
      TOTALREALES := TOTALREALES+1;
      REP := FALSE;
      IF REALESREPETIDAS >= 1 THEN
         BEGIN
           I := 0;
```

```
          REPEAT
           I := I+1;
           IF RAICESREALES[I].POLRAIZ[0] = -RREAL THEN
            BEGIN
             RAICESREALES[I].REPETICIONES := RAICESREALES[I].REPETICIONES+1;
             REP := TRUE;
            END;
          UNTIL (I=REALESREPETIDAS) OR (REP=TRUE);
         END;
IF REP = FALSE THEN
          BEGIN
            REALESREPETIDAS := REALESREPETIDAS+1;
            WITH RAICESREALES[REALESREPETIDAS] DO
            BEGIN
              POLRAIZ[1] := 1;
              POLRAIZ[0] := -RREAL;
              ORDEN := 1;
              REPETICIONES := 1;
            END;
          END;
     END
   ELSE
     BEGIN
       TOTALIMAGS := TOTALIMAGS+1;
       REP := FALSE;
       IF IMAGSREPETIDAS >= 1 THEN
         BEGIN
           I := 0;
           REPEAT
             I := I+1;
             IF (RAICESIMAG[I].POLRAIZ[1] = B) AND
                (RAICESIMAG[I].POLRAIZ[0]=C) THEN
               BEGIN
                 RAICESIMAG[I].REPETICIONES := RAICESIMAG[I].REPETICIONES+1;
                 REP := TRUE;
               END;
           UNTIL (I=IMAGSREPETIDAS) OR (REP=TRUE);
     END;
     IF REP = FALSE THEN
       BEGIN
         IMAGSREPETIDAS := IMAGSREPETIDAS+1;
         WITH RAICESIMAG[IMAGSREPETIDAS] DO
         BEGIN
           POLRAIZ[2] := 1;
           POLRAIZ[1] := B;
           POLRAIZ[0] := C;
           ORDEN := 2;
           REPETICIONES := 1;
         END;
       END;
     END;
END;
```

```
{=========================================================================}
{ REDONDEA UN NUMERO REAL A 3 CIFRAS DECIMALES.                           }

PROCEDURE REDONDEAR(VAR NUMERO : REAL);  { NUMERO REAL A REDONDEAR }
VAR FRACCREDOND : REAL;        { FRACCION DECIMAL REDONDEADA }
BEGIN
    FRACCREDOND := ROUND(1000*(FRAC(NUMERO)))*0.001 ;
    IF ABS(FRACCREDOND) <= 0.009 THEN NUMERO := TRUNC(NUMERO)
    ELSE
    IF ABS(FRACCREDOND) >= 0.999 THEN NUMERO := ROUND(NUMERO)
    ELSE
    NUMERO := INT(NUMERO)+ FRACCREDOND ;
END;


{=========================================================================}
{ CALCULA LAS RAICES DE UN POLINOMIO CUADRATICO.                          }

PROCEDURE CAL_RAIZPAR (VAR X,Y : REAL;  { COEFICIENTES DEL POLINOMIO }
                       VAR RAIZ : RA;   { RAICES DEL POLINOMIO }
                       AMBIENTE : STRING); { INDICA PLANTA O SISTEMA }

VAR RAD,RADTRM : REAL;  { AUXILIARES PARA EL CALCULO DE LAS RAICES }

BEGIN
     RADTRM := (X*X)-(4*Y);
     RAD := SQRT(ABS(RADTRM));
     WITH RAIZ[J] DO
     BEGIN
          IF RADTRM = 0 THEN
             BEGIN
                  RE := -X/2;
                  RAIZ[J+1].RE := RE;
                  IM := 0 ;
                  RAIZ[J+1].IM := IM
              END ;
          IF RADTRM < 0 THEN
             BEGIN
                  RE := -X/2;
                  RAIZ[J+1].RE := RE ;
                  IM := RAD/2;
                  RAIZ[J+1].IM := -IM;
             END
             ELSE
                 BEGIN
                      RE := (-X+RAD)/2;
                      RAIZ[J+1].RE := (-X-RAD)/2;
                      IM := 0;
                      RAIZ[J+1].IM := IM;
                 END;
     END;
     REDONDEAR(RAIZ[J].RE); REDONDEAR(RAIZ[J].IM);
```

```
     REDONDEAR(RAIZ[J+1].RE); REDONDEAR(RAIZ[J+1].IM);
     IF AMBIENTE = 'SISTEMA' THEN           ( SE DISCRETIZA EL SISTEMA )
     BEGIN
          IF RAIZ[J].IM = 0 THEN
          BEGIN
               ORDENAPOLBMASYMENOS(X,Y,RAIZ[J].RE,RAIZ[J].IM);
               ORDENAPOLBMASYMENOS(X,Y,RAIZ[J+1].RE,RAIZ[J+1].IM);
          END
          ELSE ORDENAPOLBMASYMENOS(X,Y,RAIZ[J].RE,RAIZ[J].IM);
     END;
     IF AMBIENTE = 'PLANTA' THEN          ( SE DISCRETIZA LA PLANTA )
     BEGIN
          IF RAIZ[J].IM = 0 THEN
          BEGIN
               ORGANIZA(X,Y,RAIZ[J].RE,RAIZ[J].IM);
               ORGANIZA(X,Y,RAIZ[J+1].RE,RAIZ[J+1].IM);
          END
          ELSE ORGANIZA(X,Y,RAIZ[J].RE,RAIZ[J].IM);
     END;
END;


{*************************************************************************}


PROCEDURE CALC_GRADO(POLBB:POLINOMIO;   ( POLINOMIO PARA CALCULO DE RAICES )
                     GRADPOLBB:INTEGER; ( GRADO DEL POLINOMIO )
                     AMBIENTE:STRING);  ( INDICA PLANTA O SISTEMA )
FORWARD;  ( ESTE PROCEDIMIENTO SE DECLARA MAS ADELANTE )


{*************************************************************************}
( METODO DE LIN-BAIRSTOW PARA EL CALCULO DE RAICES REALES Y COMPLEJAS DE    )
( POLINOMIOS MEDIANTE LA OBTENCION DE TERMINOS CUADRATICOS EN FORMA ---     )
( RECURSIVA.                                                               )

PROCEDURE DIVIDE_POL(POLBB:POLINOMIO;    ( POLINOMIO PARA CALCULO DE RAICES )
                     GRADPOLBB:INTEGER;  ( GRADO DEL POLINOMIO )
                     AMBIENTE : STRING); ( INDICA PLANTA O SISTEMA )

VAR MEMM : INTEGER;
    Dr,Ds : REAL;

BEGIN
     B[1] := POLBB[1]-ra;
     B[2] := POLBB[2]-(ra*B[1])-sa;
     FOR KA := 3 TO n DO B[KA] := POLBB[KA]-(ra*B[KA-1])-(sa*B[KA-2]) ;
     P[1] := -1;
     P[2] := ra-B[1] ;
     FOR KA := 3 TO n DO  P[KA] := -B[KA-1]-(ra*P[KA-1])-(sa*P[KA-2]);
     DENOM := SQR(P[n-1])-(P[n-2]*(P[n]+B[n-1]));
     Dr := ((-B[n-1]*P[n-1]) + (B[n]*P[n-2])) / DENOM;
     Ds := ((-P[n-1]*B[n]) + ((P[n]+B[n-1])*B[n-1])) / DENOM;
```

```
    ra := ra+Dr;
    sa := sa+Ds;
    IF (ABS(Dr) > E) AND (ABS(Ds) > E) THEN
       DIVIDE_POL(POLBB,GRADPOLBB,AMBIENTE)
    ELSE
      BEGIN
        CAL_RAIZPAR(ra,sa,RAIZ,AMBIENTE);
        a := a+1 ;
        J := J+2 ;
        NEWn := GRADPOLBB-2*a;
        FOR KA:=1 TO NEWn DO POLBB[KA] := B[KA];
        CALC_GRADO(POLBB,GRADPOLBB,AMBIENTE);
      END;
END;


{=============================================================}
{ CALCULA PUNTOS INICIALES DE ITERACIONES PARA EL METODO DE LIN-BAIRSTOW.  }

PROCEDURE CALC_GRADO(POLBB: POLINOMIO;   { POLINOMIO PARA CALCULO DE RAICES }
                     GRADPOLBB: INTEGER; { GRADO DEL POLINOMIO }
                     AMBIENTE: STRING);  { INDICA PLANTA O SISTEMA }

BEGIN
     YALLAMADO := FALSE;
     n := GRADPOLBB;
     ra := PTOINIC; sa := PTOINIC;
     IF n > 2 THEN DIVIDE_POL(POLBB,GRADPOLBB,AMBIENTE);
     if NOT(YALLAMADO) then
     IF n = 2 THEN    { SE CALCULA EL ULTIMO PAR DE RAICES }
     BEGIN
         CAL_RAIZPAR (POLBB[1],POLBB[2],RAIZ,AMBIENTE);
         YALLAMADO := TRUE;
     END
     ELSE
       BEGIN
         RAIZ[J].RE := -POLBB[1];
         REDONDEAR(RAIZ[J].RE);
         RAIZ[J].IM := 0.0;
         IF AMBIENTE = 'PLANTA' THEN
            { PARA DISCRETIZACION DE LA PLANTA }
            ORGANIZA(0,0,RAIZ[J].RE,RAIZ[J].IM);
         IF AMBIENTE = 'SISTEMA' THEN
            { SE DISCRETIZA EL SISTEMA }
           ORDENAPOLBMASYMENOS(0,0,RAIZ[J].RE,RAIZ[J].IM);
         YALLAMADO := true;
       END;
END;
```

```
{=========================================================================}

PROCEDURE LLAMADO(POLBB: POLINOMIO;    { POLINOMIO PARA CALCULO DE RAICES }
                  GRADPOLBB: INTEGER;  { GRADO DEL POLINOMIO }
                  AMBIENTE: STRING);   { INDICA PLANTA O SISTEMA }

BEGIN
   CALC_GRADO(POLBB,GRADPOLBB,AMBIENTE);
END;


{=========================================================================}
{ OBTIENE LOS POLINOMIOS B+ Y B- A PARTIR DE LAS RAICES DEL POLINOMIO POLB. }

PROCEDURE CREARPOLBMASYMENOS(VAR POLBMM: POLINOMIO; { POLINOMIO POLB }
                             POLRAIZB: POLIRAIZ;    { RAICES DE POLB }
                             CONTADOR: INTEGER;     { No. DE RAICES }
                             VAR GRADPOLBMM: INTEGER); { GRADO DE POLB }

VAR  CUENTA : INTEGER;   { CONTADOR AUXILIAR }

BEGIN
  IF CONTADORPOLBMAS = 0 THEN
     BEGIN
      POLBMENOS := POLB;
      GRADPOLBMENOS := GRADPOLB;
      GRADPOLBMAS := 0;
      POLBMASI0] := 1;
     END
  ELSE
    BEGIN
      IF CONTADORPOLBMENOS = 0 THEN
         BEGIN
           POLBMAS := POLB;
           GRADPOLBMAS := GRADPOLB;
           GRADPOLBMENOS := 0;
           POLBMENOSI0] := 1;
         END;
    END;
    IF CONTADOR = 1 THEN
      BEGIN
        POLBMM := POLRAIZB[1].POLRAIZ;
        GRADPOLBMM := POLRAIZB[1].ORDENPOLRAIZ;
      END
    ELSE
    BEGIN
      IF CONTADOR >= 2 THEN
        BEGIN
          MULTIPLICA (POLRAIZB[1].POLRAIZ,POLRAIZB[2].POLRAIZ,
                      POLRAIZB[1].ORDENPOLRAIZ,
          POLRAIZB[2].ORDENPOLRAIZ,POLBMM,GRADPOLBMM);
           IF  CONTADOR > 2 THEN
            BEGIN
```

```
            FOR CUENTA := 3 TO CONTADOR DO
            BEGIN
              FOR J := 0 TO GRADPOLBMM DO POLAUXILIAR[J] := POLBMM[J] ;
              MULTIPLICA (POLRAIZB[CUENTA],POLRAIZ,POLAUXILIAR,
                          POLRAIZB[CUENTA].ORDENPOLRAIZ,GRADPOLBMM,
                          POLBMM,GRADPOLBMM);
            END;
          END;
      END;
  END;
END;


{=========================================================================}
( LLENA LA MATRIZ DEL SISTEMA DE ECUACIONES A RESOLVER.                   )

PROCEDURE MULTIVAR (VAR MAT: MATRIX;        { MATRIZ A LLENAR }
                    VAR COLMATRIZ: INTEGER; { No. DE COLUMNAS DE LA MATRIZ }
                    POL2: POLINOMIO;        { POLINOMIO A INTRODUCIR A LA
                                              MATRIZ }
                    GRADPOL1,GRADPOL2,      { AUXILIARES PARA LLENAR LA
                                              MATRIZ }
                    GRADMAYOR: INTEGER);    { REFERENCIA DE GRADO MAYOR DE
                                              LOS POLINOMIOS A INTRODUCIR }

VAR RENMATRIZ : INTEGER;  { RENGLON DE LA MATRIZ }

BEGIN
     FOR I := GRADPOL1 DOWNTO 0 DO
     BEGIN
          COLMATRIZ := COLMATRIZ+1;
          FOR J := GRADPOL2 DOWNTO 0 DO
          BEGIN
               RENMATRIZ := GRADMAYOR - (I+J);
               MAT[RENMATRIZ,COLMATRIZ] := POL2[J];
          END;
     END;
END;


{=========================================================================}
( DIVIDE DOS POLINOMIOS.                                                  )

PROCEDURE DIVIDE(VAR DIVIDENDO,                { POLINOMIO DIVIDENDO }
                     COCIENTE: POLINOMIO;      { POLINOMIO COCIENTE }
                     DIVISOR : POLINOMIO;      { POLINOMIO DIVISOR }
                     GRADO1,                   { GRADO DEL DIVIDENDO }
                     GRADO2 : INTEGER;         { GRADO DEL DIVISOR }
                     VAR GRADPOLRES : INTEGER); { GRADO DEL COCIENTE }

VAR MIN,I,J,K,L : INTEGER;   { CONTADORES AUXILIARES }
```

```
BEGIN
     GRADPOLRES := GRADO1-GRADO2;
     FOR I := 0 TO GRADPOLRES DO COCIENTE[I] := 0;
     MIN := 1;
     IF GRADO2 = 0 THEN MIN := 0;
     FOR I := GRADO1 DOWNTO MIN DO
     BEGIN
          IF (DIVIDENDO[I] <> 0) AND (I>=GRADO2) THEN
          BEGIN
                K := I-GRADO2;
              COCIENTE[K] := DIVIDENDO[I]/DIVISOR[GRADO2];
              FOR J := GRADO2 DOWNTO 0 DO
              BEGIN
                    L := J+K;
                 DIVIDENDO[L] := DIVIDENDO[L]+COCIENTE[K]*(-DIVISOR[J]);
              END;
          END;
     END;
END;


{=====================================================================}
{ METODO DE ELIMINACION GAUSIANA SIMPLE CON SUSTITUCION ATRAZADA PARA LA -- }
{ SOLUCION DE SISTEMAS DE ECUACIONES.                                    }

PROCEDURE ELIMINA;

VAR KK,L,JJ : INTEGER;    { CONTADORES AUXILIARES }
    MULT,TEMP : REAL;     { ACUMULADORES AUXILIARES }

BEGIN
  NN := REN-1;
  KA := -1;
  REPEAT
    KA := KA+1;
    J := KA-1;
    REPEAT
      J := J+1;
      L := J;
      IF MATRIZ[J,KA] <> 0 THEN
        BEGIN
          IF L <> KA THEN
            BEGIN
              FOR JJ := KA TO COL DO
                BEGIN
                  TEMP := MATRIZ[KA,JJ];
                  MATRIZ[KA,JJ] := MATRIZ[L,JJ];
                  MATRIZ[L,JJ] := TEMP;
                END;
            END;
          KK := KA+1;
          FOR I:= KK TO REN DO
```

```
            BEGIN
              MULT := -MATRIZ[I,KA]/MATRIZ[KA,KA];
              MATRIZ[I,KA] := 0;
              FOR JJ:=KK TO COL DO
                MATRIZ[I,JJ] := MATRIZ[I,JJ]+MULT*MATRIZ[KA,JJ];
            END;
        END
      ELSE
        IF J=REN THEN SINGULAR := FALSE;
    UNTIL (MATRIZ[KA,KA] <> 0) OR (SINGULAR = FALSE) ;
  UNTIL(KA >= NN) OR (SINGULAR = FALSE);
  IF MATRIZ[REN,REN] = 0 THEN SINGULAR := FALSE;
END;


{=============================================================}
{ ENCUENTRA LAS SOLUCIONES DEL SISTEMA DE ECUACIONES.         }

PROCEDURE SUBSTITUYE;

VAR II : INTEGER;          { CONTADOR AUXILIAR }
    SUM : REAL;            { ACUMULADOR AUXILIAR }

BEGIN
     SOL[REN] := MATRIZ[REN,COL]/MATRIZ[REN,REN];
     FOR I := 0 TO NN DO
     BEGIN
          II := REN-I-1;
          SUM := MATRIZ[II,COL];
          FOR J := COL-I-1 TO REN DO SUM := SUM-MATRIZ[II,J]*SOL[J];
          SOL[II] := SUM/MATRIZ[II,II];
     END;
END;


{=============================================================}
{ CREA EL POLINOMIO T A PARTIR DE LOS POLINOMIOS B+ Y B-.     }

PROCEDURE CREABMASBMENOSBMIT;
BEGIN
     CONTADORPOLBMAS := 0;
     CONTADORPOLBMENOS := 0;
     FOR I := 0 TO GRADPOLB DO POLAUXILIAR[I] := POLB[GRADPOLB-I];
     PTOINIC := 0;
     IF (GRADPOLB > 2) AND ((POLB[0]=0) OR (POLB[2]=0)) THEN PTOINIC := 0.1;
     M := 0 ; J := 1 ;

     { SE ENCUENTRAN LAS RAICES DEL POLINOMIO B }
     CALC_GRADO(POLAUXILIAR,GRADPOLB,'SISTEMA');
     CREARPOLBMASYMENOS(POLBMENOS,POLRAIZBMENOS,CONTADORPOLBMENOS,
                        GRADPOLBMENOS);
     CREARPOLBMASYMENOS(POLBMAS,POLRAIZBMAS,CONTADORPOLBMAS,GRADPOLBMAS);
     POLAUXILIAR := POLBM;
```

```
    { POLT = POLBM / POLBMENOS }
    DIVIDE(POLAUXILIAR,POLT,POLBMENOS,GRADPOLBM,GRADPOLBMENOS,GRADPOLT);
END;


{=============================================================================}
{ CALCULA EL ORDEN DE LA MATRIZ A RESOLVER PARA EL DISENO DEL CONTROLADOR Y }
{ LLENA LA MATRIZ A RESOLVER.                                               }

PROCEDURE OPTENMATRIZ;
BEGIN
  GRADPOLS := GRADPOLA-1;
  GRADPOLR1 := GRADPOLAM - GRADPOLA;
  COLMAT := -1;
  GRADPOL_AR1 := GRADPOLA + GRADPOLR1;
  GRADPOL_BS := GRADPOLBMENOS + GRADPOLS;
  ORDMATRIZ := GRADPOLAM + 1;
  FOR I:=0 TO ORDMATRIZ-1 DO
  FOR J:= 0 TO ORDMATRIZ DO MATRIZ[I,J] := 0;
  FOR I := 0 TO GRADPOLAM DO MATRIZ[I,ORDMATRIZ] := POLAM[GRADPOLAM-I];

  { LLENADO DE LA MATRIZ }
  IF GRADPOL_AR1 >= GRADPOL_BS THEN
    BEGIN
      GRADMAYORPOL := GRADPOL_AR1;
      MULTIVAR(MATRIZ,COLMAT,POLA,GRADPOLR1,GRADPOLA,GRADMAYORPOL);
      MULTIVAR(MATRIZ,COLMAT,POLBMENOS,GRADPOLS,GRADPOLBMENOS,GRADMAYORPOL);
    END
  ELSE
    BEGIN
      GRADMAYORPOL := GRADPOL_BS;
      MULTIVAR(MATRIZ,COLMAT,POLBMENOS,GRADPOLS,GRADPOLBMENOS,GRADPOLAM);
      MULTIVAR(MATRIZ,COLMAT,POLA,GRADPOLR1,GRADPOLA,GRADPOLAM);
    END;
END;


{=============================================================================}
{ OBTIENE LOS POLINOMIOS R1, S Y R APARTIR DE LA SOLUCION DE LA MATRIZ      }

PROCEDURE CREA_R1_S_R;
BEGIN
  SINGULAR := TRUE;
  REN := ORDMATRIZ-1;
  COL := ORDMATRIZ;

  { SOLUCION DEL SISTEMA DE ECUACIONES }
  ELIMINA;
  IF SINGULAR THEN
    BEGIN

      { LLENA EL ARREGLO DE SOLUCIONES }
      SUBSTITUYE;
```

```
    IF GRADMAYORPOL=GRADPOL_AR1 THEN
      BEGIN
        FOR I := 0 TO GRADPOLR1 DO POLR1[GRADPOLR1-I] := SOL[I];
        J := GRADPOLS+1;
        FOR I := GRADPOLR1+1 TO ORDMATRIZ-1 DO
          BEGIN
            J := J-1;
            POLS[J] := SOL[I];
          END;
      END
    ELSE
      BEGIN
        FOR I := 0 TO GRADPOLS DO POLS[GRADPOLS-I] := SOL[I];
        J := GRADPOLR1+1;
        FOR I := GRADPOLS+1 TO ORDMATRIZ-1 DO
          BEGIN
            J := J-1;
            POLR1[J] := SOL[I];
          END;
      END;

      { POLR = POLR1 * POLBMAS }
      MULTIPLICA(POLR1,POLBMAS,GRADPOLR1,GRADPOLBMAS,POLR,GRADPOLR);
  END;
END;


{=====================================================================}
{ ENCUENTRA EL FACTORIAL DE UN NUMERO                                 }

FUNCTION FAC (EENE:ENTERONONEGATIVO) { NUMERO AL QUE SE LE CALCULARA EL - }
           : ENTEROPOSITIVO;       { FACTORIAL }

VAR X,I : INTEGER;       { AUXILIARES PARA EL CALCULO DEL FACTORIAL }

BEGIN
    X := 1;
    FOR I := 1 TO EENE DO X := X*I;
    FAC := X;
END;


{=====================================================================}
{ CALCULA LA COMBINACION DE (n,r)                                     }

FUNCTION COMBINACIONES (ENNE: ENTEROPOSITIVO; { NUMERO DE GRUPOS }
                        EERE: ENTERONONEGATIVO) { NUMERO DE ELEMENTOS }
                      : ENTEROPOSITIVO;

VAR COMBI : REAL;      { AUXILIAR PARA LAS COMBINACIONES }

BEGIN
    COMBI := FAC(ENNE) / (FAC(EERE) * FAC(ENNE-EERE));
```

```
    COMBINACIONES := ROUND(COMEI);
END;


{=========================================================================}
{ ELEVA UN NUMERO A UNA POTENCIA N                                        }

FUNCTION POTENCIA (XX : REAL;              { NUMERO A ELEVAR }
                   ENEE : ENTERONONEGATIVO)   { POTENCIA }
                  : REAL;

VAR I : ENTERONONEGATIVO;       { CONTADORES AUXILIARES }
   RESPUESTA : REAL;            { ACUMULADOR AUXILIAR }

BEGIN
    RESPUESTA := 1;
    FOR I := 1 TO ENEE DO RESPUESTA := RESPUESTA*XX;
    POTENCIA := RESPUESTA;
END;


{=========================================================================}
{ ELEVA UN BINOMIO A UN GRADO N MEDIANTE EL BINOMIO DE NEWTON.            }

PROCEDURE ELEVARPOL(A,B : REAL;           { COEFICIENTES }
                    VAR POL : POLINOMIO;  { POLINOMIO RESULTANTE }
                    POT : INTEGER);       { POTENCIA }

VAR I,J : INTEGER;        { CONTADORES AUXILIARES }

BEGIN
    J := -1;
    FOR I := POT DOWNTO 0 DO
    BEGIN
        J := J+1;
        POL[I] := POTENCIA(A,I)*POTENCIA(B,J)*COMBINACIONES(POT,J);
    END;
END;


{=========================================================================}
{ CREA FRACCIONES PARCIALES REALES                                        }

PROCEDURE ORDENARFRACCPARCIALESREALES(RAICES : RAREP; { RAICES REALES }
                                      CONTADOR : INTEGER); { No. DE RAICES }

BEGIN
    FOR I := 1 TO CONTADOR DO
    BEGIN
         WITH RAICES[I] DO
         BEGIN
             FOR J := REPETICIONES DOWNTO 1 DO
             BEGIN
```

```
                    CONTADORPARCIALES := CONTADORPARCIALES+1;

     { CREA CADA FRACC. PARCIAL DE GRADO SEGUN LAS REPETICIONES }
                    WITH PARCIAL[CONTADORPARCIALES] DO
                    BEGIN
                         ORDENNUM := 0;
                         NUMERADOR[0] := 0;
                         ORDENDENOM := J;
                         DENOMINADOR[1] := POLRAIZ[I];
                         DENOMINADOR[0] := POLRAIZ[0];
                    END;
               END;
          END;
     END;
END;


{=========================================================================}
{ CREA FRACCIONES PARCIALES IMAGINARIAS.                                  }

PROCEDURE ORDENARFRACCPARCIALESIMAGS(RAICES : RAREP; { RAICES IMAGINARIAS }
                                     CONTADOR : INTEGER); { No. DE RAICES }

BEGIN
  FOR I := 1 TO CONTADOR DO
    BEGIN
      WITH RAICES[I] DO
        BEGIN
          CONTADORPARCIALES := CONTADORPARCIALES+1;

          { CREA CADA PARCIAL IMAGINARIA CON DENOMINADOR DE ORDEN 2 }
          WITH PARCIAL[CONTADORPARCIALES] DO
            BEGIN
              ORDENNUM := 1;
              NUMERADOR[1] := 1;
              NUMERADOR[0] := 1;
              ORDENDENOM := 2;
              DENOMINADOR[2] := POLRAIZ[2];
              DENOMINADOR[1] := POLRAIZ[1];
              DENOMINADOR[0] := POLRAIZ[0];
            END;
        END;
    END;
END;


{=========================================================================}
{ ENCUENTRA LAS CONSTANTES DE LAS FRACCIONES PARCIALES Y DESARROLLA LOS --- }
{ BINOMIOS DE LOS DENOMINADORES                                           }

PROCEDURE VALDECTES(GRADPOLBX : INTEGER; { POLOS DE LA FUNCION }
                    POLBX : POLINOMIO);  { GRADO DE LOS POLOS }
```

```
VAR DESARROLLO,COMUN : POLINOMIO;  { POLINOMIOS AUXILIARES }
    GRADCOMUN,L : INTEGER;         { CONTADORES AUXILIARES }

BEGIN
  COLMAT := -1;
  ORDMATRIZ := 0;
  FOR I := 1 TO CONTADORPARCIALES DO
      ORDMATRIZ := ORDMATRIZ +(PARCIAL[I].ORDENNUM + 1);
  FOR I:=0 TO ORDMATRIZ-1 DO FOR J:= 0 TO ORDMATRIZ DO MATRIZ[I,J] := 0;
  J := ORDMATRIZ;
  FOR I := 0 TO GRADPOLBX DO
    BEGIN
      J := J-1;
      MATRIZ[J,ORDMATRIZ] := POLBX[I];
    END;
  GRADMAYORPOL := ORDMATRIZ-1;

  { FORMACION DE UN SISTEMA DE ECUACIONES RESOLVIENDO LAS FRACCS. PARCIALES
    POR IGUALDAD DE POLINOMIOS }
  FOR KA := 1 TO CONTADORPARCIALES DO
    BEGIN
      WITH PARCIAL[KA] DO
      BEGIN
        IF KA <= TOTALREALES THEN
          BEGIN

            { DESARROLLO DEL DENOMINADOR DE LA PARCIAL }
            IF ORDENDENOM > 1 THEN
               ELEVARPOL(DENOMINADOR[1],DENOMINADOR[0],DESARROLLO,ORDENDENOM)
            ELSE DESARROLLO := DENOMINADOR
          END
        ELSE DESARROLLO := DENOMINADOR;
        FOR L := 0 TO GRADCOMDENOM DO POLAUXILIAR[L] := COMDENOM[L];

        { LLENADO DE LA MATRIZ COMO UN SISTEMA DE ECUACIONES }
        DIVIDE(POLAUXILIAR,COMUN,DESARROLLO,GRADCOMDENOM,ORDENDENOM,
              GRADCOMUN);
        MULTIVAR(MATRIZ,COLMAT,COMUN,ORDENNUM,GRADCOMUN,GRADMAYORPOL);
      END;
    END;
  SINGULAR := TRUE;
  REN := ORDMATRIZ-1;
  COL := ORDMATRIZ;

  { SOLUCION DEL SISTEMA DE ECUACIONES }
  ELIMINA;
  IF SINGULAR THEN
    BEGIN
      { LLENADO DEL ARREGLO DE SOLUCIONES Y ASIGNACION DE CTES }
      SUBSTITUYE;
      IF TOTALREALES >= 1 THEN
        FOR I := 0 TO TOTALREALES-1 DO
            PARCIAL[I+1].NUMERADOR[0] := SOL[I];
```

```
      IF TOTALIMAGS >= 1 THEN
        BEGIN
          IF TOTALREALES = 0 THEN I := 0 ELSE I := I+1;
          FOR J := TOTALREALES+1 TO ORDMATRIZ-1 DO
            BEGIN
              PARCIAL[J].NUMERADOR[1] := SOL[I];
              I := I+1;
              PARCIAL[J].NUMERADOR[0] := SOL[I];
              I := I+1;
            END;
        END;
   END;
END;

{=============================================================}
{ REALIZA LA TRANSFORMADA INTA DE LAS FRACCIONES PARCIALES POR EL METODO   }
{ DE MUESTREADOR-RETEN.                                                    }

PROCEDURE PASOAZ;
VAR RES,FACTOR,ALFA,BETA,R,COSEN,SEN,KP : REAL;  { AUXILIARES PARA LA
                                                   TRANSFORMADA Z DE PARCIALES  }
   L : INTEGER;    { CONTADOR AUXILIAR }

BEGIN

  { TRANSFORMADA Z DE PARCIALES REALES }
  IF TOTALREALES >= 1 THEN
  FOR I:=1 TO TOTALREALES DO
    BEGIN
      WITH PARCIAL[I] DO
      BEGIN

        { TRANSFORMADA Z DE PARCIAL REAL DE ORDEN 1 }
        IF ORDENDENOM=1 THEN
          BEGIN
            ORDENNUM := 1;
            NUMERADOR[1] := NUMERADOR[0];
            NUMERADOR[0] := 0;
            IF DENOMINADOR[0]=0 THEN DENOMINADOR[0] := -1
            ELSE DENOMINADOR[0] := -EXP(-DENOMINADOR[0]*TM)
          END;

          { TRANSFORMADA Z DE PARCIAL REAL DE ORDEN 2 }
          IF ORDENDENOM=2 THEN
          BEGIN
            ORDENNUM := 1;
            NUMERADOR[1] := NUMERADOR[0]*TM;
            NUMERADOR[0] := 0;
            IF DENOMINADOR[0]=0 THEN DENOMINADOR[0] := -1
            ELSE DENOMINADOR[0] := -EXP(-DENOMINADOR[0]*TM);
            ELEVARPOL(DENOMINADOR[1],DENOMINADOR[0],DENOMINADOR,ORDENDENOM);
          END;
```

```
        ( TRANSFORMADA Z DE PARCIAL REAL DE ORDEN MAYOR A 2 )
        IF ORDENDENOM>2 THEN
         BEGIN
          ORDENNUM := ORDENDENOM-1;     {mult por z}
          DENOMINADOR[0] := -EXP(-DENOMINADOR[0]*TM);
          ELEVARPOL(DENOMINADOR[1],DENOMINADOR[0],DENOMINADOR,ORDENDENOM);
          FACTOR := NUMERADOR[0]*(POTENCIA(TM,ORDENDENOM)/FAC(ORDENDENOM));
          FOR KA:=1 TO ORDENNUM DO
            BEGIN
              NUMERADOR[0] := 0;        {por que se mult por z}
              RES := 0;                 {todo el numerador}
              FOR L:=1 TO KA DO
                RES:= RES + ( POTENCIA(-1,KA-L)*POTENCIA(L,ORDENNUM)*
                      COMBINACIONES(ORDENNUM+1,KA-L) );
              NUMERADOR[ORDENNUM-KA+1] := RES*FACTOR;
            END;
         END;
      END;
    END;

    ( TRANSFORMADA Z DE PARCIALES IMAGINARIAS )
    IF TOTALIMAGS >=1 THEN
    FOR J := TOTALREALES+1 TO CONTADORPARCIALES DO
    BEGIN
      WITH PARCIAL[J] DO
        BEGIN
          ALFA := DENOMINADOR[1]/2;
          BETA := DENOMINADOR[0]-SQR(ALFA);
          COSEN := COS(BETA*TM);
          SEN := SIN(BETA*TM);
          R := EXP(-ALFA*TM);
          KP := (NUMERADOR[0] - (NUMERADOR[1] * ALFA)) / BETA;
          ORDENNUM := 2;
          NUMERADOR[2] := NUMERADOR[1];
          NUMERADOR[1] := R * ( (KP*SEN)-(NUMERADOR[1]*COSEN) );
          NUMERADOR[0] := 0;
          DENOMINADOR[2] := 1;
          DENOMINADOR[1] := -2*R*COSEN;
          DENOMINADOR[0] := SQR(R);
        END;
    END;
  END;


{=====================================================================}
( OBTIENE UN COMUN DENOMINADOR PARA LAS FRACCIONES PARCIALES REALES    )

PROCEDURE OPTENCOMUNRE;

VAR  CUENTA : INTEGER;   ( CONTADOR AUXILIAR )

BEGIN
```

```
    IF (REALESREPETIDAS = 1) AND (IMAGSREPETIDAS = 0) THEN
    WITH PARCIAL[I] DO
    BEGIN
         COMDENOM := DENOMINADOR;
         GRADCOMDENOM := ORDENDENOM;
         COMNUM := NUMERADOR;
         GRADCOMNUM := ORDENNUM;
    END
    ELSE
    IF REALESREPETIDAS = 1  THEN
    WITH PARCIAL[I] DO
    BEGIN
         COMDENOM := DENOMINADOR;
         GRADCOMDENOM := ORDENDENOM;
    END
    ELSE
     IF REALESREPETIDAS >= 2 THEN
     BEGIN
          I := PARCIAL[1].ORDENDENOM+1;
          WITH PARCIAL[I] DO
          MULTIPLICA (DENOMINADOR,PARCIAL[1].DENOMINADOR,ORDENDENOM,
                      PARCIAL[1].ORDENDENOM,COMDENOM,GRADCOMDENOM);
         IF  REALESREPETIDAS > 2 THEN
         BEGIN
              FOR CUENTA := 3 TO REALESREPETIDAS DO
              BEGIN
                   I := I + PARCIAL[I].ORDENDENOM;
                  WITH PARCIAL[I] DO
                  BEGIN
                      POLAUXILIAR := COMDENOM;
                      MULTIPLICA (DENOMINADOR,POLAUXILIAR,
                      ORDENDENOM,GRADCOMDENOM,COMDENOM,GRADCOMDENOM);
                  END;
              END;
         END;
     END;
END;


{=========================================================================}
{ OBTIENE UN COMUN DENOMINADOR PARA LAS FRACCIONES PARCIALES IMAGINARIAS   }

PROCEDURE OPTENCOMUNIM;

VAR  CUENTA : INTEGER;     { CONTADOR AUXILIAR }

BEGIN
     IF (IMAGSREPETIDAS=1) AND (REALESREPETIDAS=0) THEN
     WITH PARCIAL[I] DO
     BEGIN
          COMDENOM := DENOMINADOR;
          GRADCOMDENOM := ORDENDENOM;
          COMNUM := NUMERADOR;
          GRADCOMNUM := ORDENNUM;
```

```
    END
    ELSE
    IF ((IMAGSREPETIDAS >= 1) AND (REALESREPETIDAS >= 1) THEN
    BEGIN
        WITH PARCIAL[TOTALREALES+1] DO
        MULTIPLICA (COMDENOM,DENOMINADOR,GRADCOMDENOM,ORDENDENOM,
                COMDENOM,GRADCOMDENOM);
       IF  IMAGSREPETIDAS ) 1 THEN
       BEGIN
           FOR CUENTA := TOTALREALES+2 TO CONTADORPARCIALES DO
           BEGIN
               WITH PARCIAL[CUENTA] DO
               BEGIN
                   POLAUXILIAR := COMDENOM;
                   MULTIPLICA (DENOMINADOR,POLAUXILIAR,
                   ORDENDENOM,GRADCOMDENOM,COMDENOM,GRADCOMDENOM);
               END;
           END;
       END;
    END;
END;


{=========================================================================}
( OBTIENE LA FUNCION DE TRANSFERENCIA DISCRETA SUMANDO TODAS LAS FRACCIONES )
( PARCIALES COMO QUEBRADOS CON EL COMUN DENOMINADOR OBTENIDO.              )

PROCEDURE OPTENPLANTADISCR(VAR GRADPOLAX,      ( GRADO DE LOS  POLOS )
                       GRADPOLBX: INTEGER; ( GRADO  DE LOS CEROS )
                       VAR POLAX,          ( COEFICIENTES DE LOS POLOS )
                       POLBX : POLINOMIO); ( COEFICIENTES DE LOS CEROS )

VAR AUXILIAR : POLINOMIO;    ( POLINOMIO AUXILIAR  )
   ANTERIOR : INTEGER;      ( ACUMULADOR AUXILIAR )

BEGIN
    FOR I:=0 TO GRADOPOL DO COMNUM[I] := 0;
    GRADCOMNUM := 0;
    IF CONTADORPARCIALES > 1 THEN
    BEGIN

        ( SUMA DE FRACCIONES PARCIALES COMO QUEBRADOS )
        FOR I := 1 TO CONTADORPARCIALES DO
        WITH PARCIAL[I] DO
        BEGIN
            AUXILIAR := COMDENOM;
            DIVIDE(AUXILIAR,DENOMINADOR,DENOMINADOR,GRADCOMDENOM,
                  ORDENDENOM,ORDENDENOM);
            MULTIPLICA(DENOMINADOR,NUMERADOR,ORDENDENOM,ORDENNUM,
                     NUMERADOR,ORDENNUM);
            IF ORDENNUM > GRADCOMNUM THEN GRADCOMNUM := ORDENNUM;
            ANTERIOR := GRADCOMNUM;
            FOR KA := ANTERIOR DOWNTO 0 DO
```

```
            BEGIN
                COMNUM[KA] := COMNUM[KA]+NUMERADOR[KA];
                IF COMNUM[GRADCOMNUM] < 1E-9 THEN
                   GRADCOMNUM := GRADCOMNUM-1;
             END;
        END;
    END;

    { SE DIVIDE TODA LA FUNCION ENTRE (Z-1) PARA ELIMINAR TERMINOS EN EL }
    { NUMERADOR }
    WITH PARCIAL[I] DO
    BEGIN
        NUMERADOR[1] := 1;
        NUMERADOR[0] := -1;
        ORDENNUM := 1;
        DENOMINADOR[1] := 1;
        DENOMINADOR[0] := 0;
        ORDENDENOM := 1;
        DIVIDE(COMDENOM,POLAX,NUMERADOR,GRADCOMDENOM,ORDENNUM,GRADPOLAX);
        IF (COMNUM[0] = 0) AND (GRADCOMNUM > 0) THEN
           BEGIN
             FOR I := 0 TO GRADCOMNUM-1 DO POLBX[I] := COMNUM[I+1];
                 GRADPOLBX := GRADCOMNUM - 1;
           END
        ELSE
          BEGIN
            GRADPOLBX := GRADCOMNUM;
            POLBX := COMNUM;

            { SE MULTIPLICA POR Z EL DENOMINADOR DE LA FUNCION OBTENICA }
            FOR I := GRADPOLAX DOWNTO 0 DO POLAX[I+1] := POLAX[I];
            GRADPOLAX := GRADPOLAX + 1;
          END;
    END;
END;

{=========================================================================}
{ INICILIZA VARIABLES PARA LA DISCRETIZACION Y LLAMA A LOS PROCEDIMIENTOS  }
{ NECESARIOS PARA LA DISCRETIZACION.                                      }

PROCEDURE DISCRETPLANTA(VAR GRADPOLBX,       { GRADO DE LOS CEROS }
                        GRADPOLAX : INTEGER; { GRADO DE LOS POLOS }
                        VAR POLBX,           { COEFICIENTES DE LOS CEROS }
                        POLAX : POLINOMID);  { COEFICIENTES DE LOS POLOS }

BEGIN

    { INICIALIZACION DE VARIABLES }
    SINGULAR := TRUE;
    REALESREPETIDAS := 0;
    IMAGSREPETIDAS := 0;
    TOTALREALES := 0;
    TOTALIMAGS := 0;
```

```
    CONTADORPARCIALES := 0;

    { CALCULO DE LAS RAICES DE LOS POLOS }
    GRADCOMDENOM := GRADPOLAI+1;
    FOR I := GRADCOMDENOM DOWNTO 1 DO COMDENOM[I] := POLAX[I-1];
    COMDENOM[0] := 0;
      PTOINIC := 0;
      IF GRADCOMDENOM > 2 THEN
         IF (COMDENOM[0]=0) OR (COMDENOM[2]=0) THEN PTOINIC := 0.1;
      M := 0; j := 1;
      FOR I := 0 TO GRADCOMDENOM DO
         POLAUXILIAR[I] := COMDENOM[GRADCOMDENOM-I];
      CALC_GRADO(POLAUXILIAR,GRADCOMDENOM,'PLANTA');

    { CREACION DE LAS FRACCIONES PARCIALES REALES }
    IF REALESREPETIDAS >=I THEN
       ORDENARFRACCPARCIALESREALES(RAICESREALES,REALESREPETIDAS);

    { CREACION DE LAS FRACCIONES PARCIALES IMAGINARIAS }
    IF IMAGSREPETIDAS >=I THEN
       ORDENARFRACCPARCIALESIMAGS(RAICESIMAG,IMAGSREPETIDAS);

    { CALCULO DE LAS CTES DA LAS FRACCS. PARCIALES }
    VALDECTES(GRADPOLBX,POLBX);
    IF SINGULAR THEN
    BEGIN
        { DISCRETIZACION DE LAS PARCIALES }
        PASOAZ;
        { OBTENCION DE UN COMUN DENOMINADOR DE LAS FRACCS. PARCIALES }
        OPTENCOMUNRE;
        OPTENCOMUNIM;

        { UNION DE LAS FRACCS. PARCIALES DISCRETIZADAS }
        OPTENPLANTADISCR(GRADPOLAI,GRADPOLBX,POLAI,POLBX);
    END;  { DE IF SINGULAR}
END;

END.
```

```
UNIT DATOSAC;

( UNIDAD QUE MUESTRA LOS DATOS ACTUALES.                                  )
( ======================================                                  )

INTERFACE

USES CRT,DECLARA;

PROCEDURE DATOS_ACTUALES;

IMPLEMENTATION

{=========================================================================)
( PRESENTA EL FORMATO PARA EL DESPLIEGUE DE LOS COEFICIENTES DE LAS  ---  )
( FUNCIONES DE TRANSFERENCIA.                                             )

PROCEDURE PANTALLA_DATOS;

BEGIN
  GOTOXY(1,4);
  WRITE('                                                  '); GOTOXY(1,5);
  WRITE('  | s⁰ |   Numerador   |  Denominador  |          '); GOTOXY(1,6);
  WRITE('                                                  '); GOTOXY(1,7);
  WRITE('  |    |               |               |          '); GOTOXY(1,8);
  WRITE('  |    |               |               |          '); GOTOXY(1,9);
  WRITE('  |    |               |               |          '); GOTOXY(1,10);
  WRITE('  |    |               |               |          '); GOTOXY(1,11);
  WRITE('  |    |               |               |          '); GOTOXY(1,12);
  WRITE('  |    |               |               |          '); GOTOXY(1,13);
  WRITE('  |    |               |               |          '); GOTOXY(1,14);
  WRITE('  |    |               |               |          '); GOTOXY(1,15);
  WRITE('  |    |               |               |          '); GOTOXY(1,16);
  WRITE('  |    |               |               |          '); GOTOXY(1,17);
  WRITE('  |    |               |               |          '); GOTOXY(1,18);
  WRITE('  |    |               |               |          '); GOTOXY(1,19);
  WRITE('  |    |               |               |          '); GOTOXY(1,20);
  WRITE('  |    |               |               |          '); GOTOXY(1,21);
  WRITE('  |    |               |               |          '); GOTOXY(1,22);
  WRITE('  |    |               |               |          '); GOTOXY(1,23);
  WRITE('                                                  ');
END;


{=========================================================================)
( DESPLIEGA EL MENU DE SELECCION DE LA FUNCION DE TRANSFERENCIA A VISUALIZAR)
( ASI COMO LOS PARAMETROS ACTUALES DEL SISTEMA.                           )

PROCEDURE DATOS_ACTUALES;

VAR M,I : INTEGER;          ( CONTADORES AUXILIARES )

BEGIN
  WRITE('Datos actuales':47);
  M := 49;
```

```
TEXTBACKGROUND(3);
PANTALLA_DATOS;
GOTOXY(42,14);

{ DESPLIEGA EL TIEMPO ANALIZADO }
WRITE('┌──────────────┐'); GOTOXY(42,15);
WRITE('│ T= ',TIEMPO:10:3,' │'); GOTOXY(42,16);
WRITE('└──────────────┘');

{ EL CASO DE NO TRATARSE DE UN SISTEMA EN MALLA ABIERTA DESPLIEGA EL
  TIEMPO DE MUESTREO }
IF TIPO <> 'P' THEN
   BEGIN
     GOTOXY(62,14);
     WRITE('┌──────────────┐'); GOTOXY(62,15);
     WRITE('│ TM= ',TM:10:3,' │'); GOTOXY(62,16);
     WRITE('└──────────────┘');
   END;

{ EN CASO DE SER UN SISTEMA CON CONTROLADOR DESPLIEGA LA GANANCIA }
IF TIPO = 'C' THEN
   BEGIN
     GOTOXY(42,19);
     WRITE('┌────────────────────┐'); GOTOXY(42,20);
     WRITE('│ Ganancia= ',GANANCIA:10:3,' │'); GOTOXY(42,21);
     WRITE('└────────────────────┘');
   END;
REPEAT
  REPEAT
    TEXTBACKGROUND(3);
    GOTOXY(42,6);
    WRITE('┌──────────┐'); GOTOXY(42,7);
    WRITE('│ 1) G(s)  │'); GOTOXY(42,8);
    WRITE('│ 2) Gc(z) │'); GOTOXY(42,9);

    { SE DEPLIEGA H(s) O H(z) SI EL SISTEMA FUE INTRODUCIDO EN FORMA
      DISCRETA O NO }
    IF LECTURA_NODISCRETA THEN
       WRITE('│ 3) H(s)  │')
    ELSE
      WRITE('│ 3) H(z)  │');
    GOTOXY(42,10);
    WRITE('└──────────┘');
    TEXTBACKGROUND(5);
    GOTOXY(43,M-42);
    CASE M OF
      49 : WRITE(' 1) G(s)  ');
      50 : WRITE(' 2) Gc(z) ');
      51 : IF LECTURA_NODISCRETA THEN
             WRITE(' 3) H(s)  ')
           ELSE
             WRITE(' 3) H(z)  ');
    END; {DEL CASE}
    TECLA:= READKEY;
    CASE ORD(TECLA) OF
```

```
TEXTBACKGROUND(3);
PANTALLA_DATOS;
GOTOXY(42,14);

{ DESPLIEGA EL TIEMPO ANALIZADO }
WRITE('┌──────────────┐'); GOTOXY(42,15);
WRITE('│ T= ',TIEMPO:10:3,' │'); GOTOXY(42,16);
WRITE('└──────────────┘');

{ EL CASO DE NO TRATARSE DE UN SISTEMA EN MALLA ABIERTA DESPLIEGA EL
  TIEMPO DE MUESTREO }
IF TIPO <> 'P' THEN
   BEGIN
     GOTOXY(62,14);
     WRITE('┌──────────────┐'); GOTOXY(62,15);
     WRITE('│ TM= ',TM:10:3,' │'); GOTOXY(62,16);
     WRITE('└──────────────┘');
   END;

{ EN CASO DE SER UN SISTEMA CON CONTROLADOR DESPLIEGA LA GANANCIA }
IF TIPO = 'C' THEN
   BEGIN
     GOTOXY(42,19);
     WRITE('┌──────────────────┐'); GOTOXY(42,20);
     WRITE('│ Ganancia= ',GANANCIA:10:3,' │'); GOTOXY(42,21);
     WRITE('└──────────────────┘');
   END;
REPEAT
  REPEAT
    TEXTBACKGROUND(3);
    GOTOXY(42,6);
    WRITE('┌──────────┐'); GOTOXY(42,7);
    WRITE('│ 1) G(s)  │'); GOTOXY(42,8);
    WRITE('│ 2) Gc(z) │'); GOTOXY(42,9);

    { SE DEPLIEGA H(s) O H(z) SI EL SISTEMA FUE INTRODUCIDO EN FORMA
      DISCRETA O NO }
    IF LECTURA_NODISCRETA THEN
       WRITE('│ 3) H(s)  │')
    ELSE
      WRITE('│ 3) H(z)  │');
    GOTOXY(42,10);
    WRITE('└──────────┘');
    TEXTBACKGROUND(5);
    GOTOXY(43,M-42);
    CASE M OF
      49 : WRITE(' 1) G(s)  ');
      50 : WRITE(' 2) Gc(z) ');
      51 : IF LECTURA_NODISCRETA THEN
             WRITE(' 3) H(s)  ')
           ELSE
             WRITE(' 3) H(z)  ');
    END; (DEL CASE)
    TECLA1 := READKEY;
    CASE ORD(TECLA1) OF
```

```
   72 : BEGIN
          M := M-1;
          IF M < 49 THEN
            M := 51;
        END;
   80 : BEGIN
          M := M+1;
          IF M > 51 THEN
            M := 49;
        END;
  END; {DEL CASE}
UNTIL ((ORD(TECLA1)>48) AND (ORD(TECLA1)<52)) OR (ORD(TECLA1)=13) OR
      (ORD(TECLA1)=27);
IF ORD(TECLA1) = 13 THEN
   TECLA1 := CHR(M);
TEXTBACKGROUND(3);
CASE ORD(TECLA1) OF
  49 : BEGIN

       { DESPLIEGA LOS COEFICIENTES DE LOS CEROS DE LA PLANTA EN S }
       PANTALLA_DATOS;
       FOR I := 0 TO GRADOP DO
         BEGIN
           GOTOXY(8,7+I);
           WRITE(COEFSOP[I]:13:4);
         END;

       { DESPLIEGA LOS COEFICIENTES DE LOS POLOS DE LA PLANTA EN S }
       FOR I := 0 TO GRADOPP DO
         BEGIN
           GOTOXY(3,7+I);
           WRITE(I:2);
           GOTOXY(24,7+I);
           WRITE(COEFSPP[I]:13:4);
         END;
     END;
  50 : IF TIPO = 'C' THEN       { SI SE TRATA DE UN SISTEMA CON UN }
       BEGIN                    { CONTROLADOR PROPUESTO }
         PANTALLA_DATOS;
         GOTOXY(3,5);
         WRITE('z');

       { DESPLIEGA LOS COEFICIENTES DE LOS CEROS DEL CONTROLADOR }
       FOR I := 0 TO GRADOZOC DO
         BEGIN
           GOTOXY(8,7+I);
           WRITE(COEFZOC[I]:13:4);
         END;

       { DESPLIEGA LOS COEFICIENTES DE LOS POLOS DEL CONTROLADOR }
       FOR I := 0 TO GRADOZPC DO
         BEGIN
           GOTOXY(3,7+I);
           WRITE(I:2);
           GOTOXY(24,7+I);
```

```
            WRITE(COEFZPC[I]:13:4);
          END;
      END;
  51 : IF TIPO = 'H' THEN     { SI SE TRATA DE UN SISTEMA PROPUESTO }
      BEGIN                   { ( DISEND DE UN CONTROLADOR ) }
        PANTALLA_DATOS;
        GOTOXY(3,5);
        IF LECTURA_NODISCRETA THEN
           WRITE('s')
        ELSE
           WRITE('z');

        { DESPLIEGA LOS COEFICIENTES DE LOS CEROS DEL SISTEMA
          COMO FUERON INTRODUCIDOS ORIGINALMENTE (EN S O EN Z) }
        FOR I := 0 TO GRADPOLBMS DO
          BEGIN
            GOTOXY(8,7+I);
            WRITE(POLBMS[I]:13:4);
          END;

        { DESPLIEGA LOS COEFICIENTES DE LOS POLOS DEL SISTEMA
          COMO FUERON INTRODUCIDOS ORIGINALMENTE (EN S O EN Z) }
        FOR I := 0 TO GRADPOLAMS DO
          BEGIN
            GOTOXY(3,7+I);
            WRITE(I:2);
            GOTOXY(24,7+I);
            WRITE(POLAMS[I]:13:4);
          END;
      END;
  END;{DEL CASE}
UNTIL ORD(TECLA1) = 27;
TEXTBACKGROUND(0);
CLRSCR;
END;

END.
```

```
UNIT LAZOPL;

{ UNIDAD QUE RESUELVE EL SISTEMA EN LAZO ABIERTO.                          }
{ ==============================================                          }

INTERFACE

USES CRT,DECLARA,DISENO,MENSAJES;

FUNCTION CALCULARE(PARCIAL : FRACCPARCIALES; T : REAL) : REAL;

PROCEDURE RESPUESTAIGRADCEROSX,GRADPOLOSX : INTEGER;
                  CEROSX,POLOSX : POLINOMIO);

PROCEDURE ARREGLODATOS(X:REAL; VAR PRIMERO,ULTIMO: APUNTADOR);


IMPLEMENTATION

{=========================================================================}
{ CALCULA LA FUNCION EXPONENCIAL ELIMINANDO NUMEROS MUY PEQUEROS           }

FUNCTION EXPI(S:REAL) : REAL;      { EXPONENTE DE LA FUNCION }
 BEGIN
 IF S<-88 THEN EXPI:=0
         ELSE EXPI:=EXP(S);
END;

{=========================================================================}
{ ALMACENA LOS PUNTOS EVALUADOS DE LA PLANTA EN UNA COLA.                  }

PROCEDURE ARREGLODATOSIX:REAL;                        { DATO A ALMACENAR }
                     VAR PRIMERO,ULTIMO: APUNTADOR);  { APUNTADORES DE -
                                                        REFERENCIA }
VAR DATONUEVO: APUNTADOR;   { APUNTADOR AUXILIAR }

BEGIN
  NEW(DATONUEVO);
  WITH DATONUEVO^ DO
    BEGIN
      DATO:=X;
      ANTERIOR:=ULTIMO;
      ULTIMO:=DATONUEVO;
      IF PRIMERO = NIL THEN
         PRIMERO:=DATONUEVO
      ELSE
         ANTERIOR^.SIGUIENTE:=ULTIMO;
    END;
END;

{=========================================================================}
{ EVALUA LA ANTITRANSFORMADA DE LAPLACE PARA LAS PARCIALES REALES           }

FUNCTION CALCULARE(PARCIAL : FRACCPARCIALES; { FRACCION PARCIAL REAL }
                 T : REAL) : REAL;           { TIEMPO ACTUAL PARA CALCULO
                                               DE LA FUNCION }
```

```
BEGIN
  WITH PARCIAL DO
    IF DENOMINADOR[0] = 0 THEN
      CALCULARE := NUMERADOR[0]*POTENCIA(T,ORDENDENOM-1)/FAC(ORDENDENOM-1)
    ELSE
      CALCULARE := NUMERADOR[0]*POTENCIA(T,ORDENDENOM-1)*
                   EXPI(-DENOMINADOR[0]*T)/FAC(ORDENDENOM-1);
END;

{=========================================================================}
{ EVALUA LA ANTITRANSFORMADA DE LAPLACE PARA LAS PARCIALES IMAGINARIAS     }

FUNCTION CALCULAIM(PARCIAL : FRACCPARCIALES;  { FRACCION PARCIAL IMAGINARIA }
                   T : REAL) : REAL;          { TIEMPO ACTUAL PARA CALCULO
                                                DE LA FUNCION }
VAR ALFA,OMEGA : REAL;

BEGIN
  WITH PARCIAL DO
  BEGIN
    ALFA := DENOMINADOR[1]/2;
    OMEGA := SQRT(DENOMINADOR[0]-SQR(ALFA));
    CALCULAIM := NUMERADOR[1]*EXPI(-ALFA*T)*COS(OMEGA*T) +
                 ((NUMERADOR[0]-NUMERADOR[1]*ALFA)/OMEGA)*
                  EXPI(-ALFA*T)*SIN(OMEGA*T);
  END;
END;

{=========================================================================}
{ MULTIPLICA LA PLANTA G(S) POR LA ENTRADA R(S), OBTIENE LAS FRACCIONES --- }
{ PARCIALES, ANTITRANSFORMA Y EVALUA LOS PUNTOS DE LA SALIDA DEL SISTEMA.  }

PROCEDURE RESPUESTA(GRADCEROSI,  { GRADO DE LOS CEROS DE LA PLANTA }
                    GRADPOLOSI : INTEGER; { GRADO DE LOS POLOS DE LA PLANTA }
                    CEROSI,      { COEFICIENTES DE LOS CEROS DE LA PLANTA}
                    POLOSI : POLINOMIO); {COEFICIENTES DE POLOS DE LA PLANTA}

VAR I,           { CONTADOR AUXILIAR }
    GRADNUMENT,  { GRADO DEL NUMERADOR DE LA F.T. DE LA ENTRADA }
    GRADDENENT : INTEGER;  { GRADO DEL DENOMINADOR DE LA F.T. DE LA ENTRADA }
    NUMENT,          { COEFICIENTES DEL NUMERADOR DE LA F.T. DE LA ENTRADA }
    DENENT : POLINOMIO; { COEFS. DEL DENOMINADOR DE LA F.T. DE LA ENTRADA }
    INC,               { INCREMENTO EN EL TIEMPO }
    ECUACION : REAL;   { RESULTADO DE LA FUNCION }

BEGIN
     SINGULAR := TRUE;
     REALESREPETIDAS := 0;
     IMAGSREPETIDAS := 0;
     TOTALREALES := 0;
     TOTALIMAGS := 0;
     CONTADORPARCIALES := 0;
     CASE FUNCION OF
       2 : BEGIN
             GRADNUMENT := 0;
             GRADDENENT := 1;
```

```
        NUMENT[0] := C;
        DENENT[0] := 0;
        DENENT[1] := 1;
      END;
3 : BEGIN
        GRADNUMENT := 0;
        GRADDENENT := 2;
        NUMENT[0] := C;
        DENENT[0] := 0;
        DENENT[1] := 0;
        DENENT[2] := 1;
    END;
4 : BEGIN
        GRADNUMENT := 0;
        GRADDENENT := 1;
        NUMENT[0] := C;
        DENENT[0] := -A;
        DENENT[1] := 1;
    END;
5 : BEGIN
        GRADNUMENT := 0;
        GRADDENENT := 2;
        NUMENT[0] := B@C;
        DENENT[0] := SQR(A)+SQR(B);
        DENENT[1] := -2@A;
        DENENT[2] := 1;
    END;
6 : BEGIN
        GRADNUMENT := 1;
        GRADDENENT := 2;
        NUMENT[0] := -C@A;
        NUMENT[1] := C;
        DENENT[0] := SQR(A)+SQR(B);
        DENENT[1] := -2@A;
        DENENT[2] := 1;
    END;
END; (DEL CASE);
IF FUNCION = 1 THEN
   FOR I := 0 TO GRADCEROSX DO
     CEROSX[I] := CEROSX[I]@C
ELSE
  BEGIN
    MULTIPLICA(CEROSX,NUMENT,GRADCEROSX,GRADNUMENT,CEROSX,GRADCEROSX);
    MULTIPLICA(POLOSX,DENENT,GRADPOLOSX,GRADDENENT,POLOSX,GRADPOLOSX);
  END;

{ CALCULO DE LAS RAICES DE LOS POLOS }
GRADCOMDENOM := GRADPOLOSX;
COMDENOM := POLOSX;
PTOINIC := 0;
IF GRADPOLOSX > 2 THEN
   IF (POLOSX[0]=0) OR (POLOSX[2]=0) THEN PTOINIC := 0.1;
N := 0; J := 1;
FOR I := 0 TO GRADPOLOSX DO POLAUXILIAR[I] := POLOSX[GRADPOLOSX-I];
LLAMADO(POLAUXILIAR,GRADPOLOSX,'PLANTA');
```

```
    { CREA FRACCIONES PARCIALES REALES }
    IF REALESREPETIDAS >=1 THEN
    ORDENARFRACCPARCIALESREALES(RAICESREALES,REALESREPETIDAS);

    { CREA FRACCIONES PARCIALES IMAGINARIAS }
    IF IMAGSREPETIDAS >=1 THEN
    ORDENARFRACCPARCIALESIMAGS(RAICESIMAG,IMAGSREPETIDAS);

    { CALCULO DE LAS CTES DE LAS FRACCS. PARCIALES }
    VALDECTES(GRADCEROSX,CEROSX);

    { ANTITRASFORMADA DE LAPLACE DE LAS FRACCS. PARCIALES }
    IF SINGULAR THEN
    BEGIN
      INC := 0;
      WHILE (JR <= NoPUNTOSPLANTA) AND (NOT REBOSAMIENTO) DO
      BEGIN
        ECUACION := 0;
        { CALCULO DE LA FUNCION }
        FOR J := 1 TO TOTALREALES DO
            ECUACION := ECUACION + CALCULARE(PARCIAL[J],INC);
        FOR J := TOTALREALES+1 TO CONTADORPARCIALES DO
            ECUACION := ECUACION + CALCULAIM(PARCIAL[J],INC);
        ARREGLODATOS(ECUACION,PRIMEROPLANTA,ULTIMOPLANTA);
        PUNTOPLANTA:=ROUND(ESCALAPLANTA*JR);
        SALIDAPLANTA[PUNTOPLANTA]:=ECUACION;
        IF (ABS(ECUACION) > 1E10) THEN
          BEGIN
            MENSAJE(0);
            MENSAJE(3);
            REBOSAMIENTO := TRUE
          END;
        INC := INC + H;
        JR := JR + 1;
      END;
      JR:=JR-1;
    END;  { DE IF SINGULAR}
END;

END.
```

```
UNIT MENSAJES;

{ UNIDAD QUE CONTIENE TODOS LOS MENSAJES DEL SISTEMA.                    }
{ ==================================================                    }

INTERFACE

USES CRT,DECLARA,DOS;

PROCEDURE MENSAJE(TIPOERROR:INTEGER);
PROCEDURE PRENDECURSOR;
PROCEDURE APAGACURSOR;

IMPLEMENTATION

{=========================================================================}

 PROCEDURE APAGACURSOR;
 VAR
   INTERRUPCION : REGISTERS;
 BEGIN
   INTERRUPCION.AX:=$0100;
   INTERRUPCION.CX:=$2000;
   INTR($10,INTERRUPCION)
 END;

{=========================================================================}

 PROCEDURE PRENDECURSOR;
 VAR
   INTERRUPCION : REGISTERS;
 BEGIN
   INTERRUPCION.AX:=$0100;
   INTERRUPCION.CX:=$0607;
   INTR($10,INTERRUPCION)
 END;

{=========================================================================}

 PROCEDURE BEEP;
 BEGIN
   SOUND(500);
   DELAY(200);
   NOSOUND;
 END;

{=========================================================================}
{ DESPLIEGA MENSAJES DE ERROR Y DE PETICION DE DATOS                      }

PROCEDURE MENSAJE(TIPOERROR:INTEGER); { TIPO DE ERROR A DESPLEGAR }
VAR ESCAPE: CHAR;  { TECLA PRESIONADA }
BEGIN
 IF TIPOERROR<>0 THEN
    BEGIN
     BEEP;
```

```
   GOTOXY(3,24);
  END;
CASE TIPOERROR OF
 0 : BEGIN
      GOTOXY(2,23);
      WRITE('┌──────────────────────────────────────────────────────────────────────────┐'); GOTOXY(2,24);
      WRITE('│                                                                          │'); GOTOXY(2,25);
      WRITE('└──────────────────────────────────────────────────────────────────────────┘');
     END;
 2 : WRITE('El grado del denominador tiene que ser mayor o igual que el del numerador');
 3 : BEGIN
      WRITE('Salida de la planta excesivamente grande, solo se calcularon ',JR,' valores');
      ESCAPE:=READKEY
     END;
 4 : BEGIN
      WRITE('Salida del controlador excesivamente grande, solo se calcularon ',KC,' valores');
      ESCAPE:=READKEY
     END;
 5 : WRITE('Faltan datos');
 6 : WRITE('Los datos se tienen que expresar en 10 digitos como máximo');
 7 : WRITE('Error en la posición ');
 8 : WRITE('El grado tiene que ser un entero positivo menor que 15');
 9 : BEGIN
      WRITE('El número de gráficas tiene que ser un entero positivo menor que 6');
      ESCAPE:=READKEY
     END;
10 : BEGIN
      WRITE('La planta no es discretizable');
      ESCAPE:=READKEY
     END;
11 : BEGIN
      WRITE('Controlador no realizable con la H(z) propuesta');
      ESCAPE:=READKEY
     END;
12 : WRITE('El valor de la ganancia inherente debe ser diferente de cero');
13 : BEGIN
      WRITE('La función de transferencia del sistema no es discretizable');
      ESCAPE:=READKEY
     END;
14 : BEGIN
      WRITE('El grado de los polos de H(z) debe ser mayor o igual que el de los de G(s)');
      ESCAPE:=READKEY
     END;
15 : WRITE('Planta con polos inestables, especifique el tiempo a analizar');
16 : BEGIN
      WRITE('Directorio inexistente');
      ESCAPE:=READKEY
     END;
17 : WRITE('Dar nombre del archivo sin extensión : ');
18 : WRITE('Archivo existente, sobreescribir (s/n) ? ');
19 : WRITE('Descripción : ');
20 : WRITE('Cuantas gráficas se leerán (max 5) ? ');
21 : WRITE('Gráfica seleccionada No. ');
22 : WRITE('Antes de salvar una gráfica es necesario graficarla');
```

```
  23 : WRITE('Los datos no han sido procesados');
  24 : WRITE('El valor de este parámetro tiene que ser mayor que cero');
 END; (CELCASE)
END;

(=============================================================================)

END.
```

```
UNIT ARCHIVOS;

{ UNIDAD PARA LA MANIPULACION DE ARCHIVOS EN DISCO                    }
{ ===========================================                        }

INTERFACE

USES CRT, DECLARA, DOS, MENSAJES;

PROCEDURE RECUPERADATOS(NUMEROARCH : INTEGER; EXTENSION : STRING);
PROCEDURE ALMACENA(EXTENSION : STRING);


IMPLEMENTATION

{====================================================================}
{ VALIDA UNA CADENA ALFANUMERICA                                     }

PROCEDURE VALIDALFA(VAR CADENA:STRING); { DESCRIPCION DEL ARCHIVO }
VAR
 I,C: INTEGER; { CONTADORES AUXILIARES }

BEGIN
   I:=1;
   CADENA:='';
   REPEAT
      REPEAT
        TECLA3:=READKEY;
        IF (ORD(TECLA3)=8) AND (I>1) THEN
          BEGIN
           I:=I-1;
           DELETE(CADENA,I,1);
           GOTOXY(WHEREX-1,WHEREY);
           WRITE(' ');
           GOTOXY(WHEREX-1,WHEREY);
          END;
      UNTIL  (((ORD(TECLA3) > 47) AND (ORD(TECLA3) < 59))   OR
              ((ORD(TECLA3) > 64) AND (ORD(TECLA3) < 91))   OR
              ((ORD(TECLA3) > 96) AND (ORD(TECLA3) < 123))  OR
               (ORD(TECLA3) = 13) OR  (ORD(TECLA3) = 27)    OR
               (ORD(TECLA3) = 92));
      IF (ORD(TECLA3) <> 13) AND (ORD(TECLA3) <> 27) THEN
         IF I < 36 THEN
            BEGIN
              WRITE(TECLA3);
              CADENA:=CADENA+TECLA3;
              I:=I+1;
            END;
   UNTIL ((ORD(TECLA3)=13) AND (CADENA<>'') OR (ORD(TECLA3)=27)) ;
END;

{====================================================================}
{ LLENA EL REGISTRO DEL ARCHIVO DEPENDIENDO DEL SISTEMA ANALIZADO    }
```

```
PROCEDURE LLENA_REGISTRO;
VAR DESCRIPCIONTEMP : STRING[65];  { DESCRIPCION DEL ARCHIVO }

BEGIN
  MENSAJE(0);
  MENSAJE(19);
  READLN(DESCRIPCIONTEMP);
  WITH REGISTRO DO
       BEGIN
         NUMERADORPL := COEFSOP;
         DENOMINADORPL := COEFSPP;
         GRADONUMPL := GRADOOP;
         GRADODENPL := GRADOPP;
         TIEMPOA := TIEMPO;
         ENTRADA.TIPOENT := FUNCION;
         ENTRADA.A := A;
         ENTRADA.B := B;
         ENTRADA.C := C;
         DESCRIPCION := DESCRIPCIONTEMP;
         PM := TM;
         DATOSARCH := DATOS;
       END;
  IF DATOS = '1 2 3 4 5   7' THEN
    WITH REGISTRO DO
      BEGIN
        NUMERADOR := COEFIOC;
        DENOMINADOR := COEFIPC;
        GRADONUM := GRADOIOC;
        GRADODEN := GRADOIPC;
        GANANCIARCH := GANANCIA;
      END
  ELSE
    WITH REGISTRO DO
    BEGIN
      NUMERADOR := POLBMS;
      DENOMINADOR := POLAMS;
      GRADONUM := GRADPOLBMS;
      GRADODEN := GRADPOLAMS;
      MODISCRETO := LECTURA_MODISCRETA;
    END;
END;

{=============================================================================}
{ RECUPERA LOS DATOS DEL ARCHIVO EN DISCO DEPENDIENDO DEL SISTEMA ANALIZADO }

PROCEDURE SACA_REGISTRO ( NOMBREARCH : STRING ); { ARHIVO LEIDO }

BEGIN
  ASSIGN(ARCHIVO,NOMBREARCH);
  RESET(ARCHIVO);
  READ(ARCHIVO,REGISTRO);
  CLOSE(ARCHIVO);
  WITH REGISTRO DO
       BEGIN
         COEFSOP := NUMERADORPL;
```

```
          COEFSPP := DENOMINADORPL;
          GRADOOP := GRADONUMPL;
          GRADOPP := GRADODENPL;
          TIEMPO := TIEMPOA;
          FUNCION := ENTRADA.TIPOENT;
          A := ENTRADA.A;
          B := ENTRADA.B;
          C := ENTRADA.C;
          TM := PM;
          DATOS := DATOSARCH;
        END;
    IF DATOS = '1 2 3 4 5   7' THEN
       WITH REGISTRO DO
        BEGIN
          TIPO := 'C';
          COEFZOC := NUMERADOR;
          COEFZPC := DENOMINADOR;
          GRADOZOC := GRADONUM;
          GRADOZPC := GRADODEN;
          GANANCIA := GANANCIARCH;
        END
    ELSE
      IF DATOS = '1 2     5 6 7' THEN
         WITH REGISTRO DO
         BEGIN
           TIPO := 'H';
          POLBMS := NUMERADOR;
          POLAMS := DENOMINADOR;
          GRADPOLBMS := GRADONUM;
          GRADPOLAMS  := GRADODEN;
          NO_DISCRETO := NODISCRETO;
          LECTURA_NODISCRETA := NODISCRETO;
         END
      ELSE
        TIPO := 'P';
    DATOS_PROCESADOS := FALSE;
END;


{=========================================================================}
{ ALMACENA LOS DATOS DEL SISTEMA ANALIZADO O SUS GRAFICAS                 }

PROCEDURE ALMACENA(EXTENSION : STRING); { EXTENSION DEL ARCHIVO }
VAR NOMARCH : STRING;     { NOMBRE DEL ARCHIVO }
    DIRINFO : SEARCHREC;  { VARIABLE PARA ATRIBUTOS DE ARCHIVO }
    DESCRIPCIONTEMP : STRING[65]; { DESCRIPCION DEL ARCHIVO }

BEGIN
  PRENDECURSOR;
  MENSAJE(0);
  MENSAJE(17);
  VALIDALFA(NOMARCH);
  IF ORD(TECLA3) <> 27 THEN
    BEGIN
      NOMARCH := NOMARCH + EXTENSION;
```

```
FINDFIRST(NOMARCH,ARCHIVE,DIRINFO);
CASE DOSERROR OF
    0 : BEGIN
        MENSAJE(0);
        MENSAJE(18);
        REPEAT
          TECLA1 := READKEY;
        UNTIL (ORD(TECLA1) = 83) OR (ORD(TECLA1) = 115) OR
              (ORD(TECLA1) = 78) OR (ORD(TECLA1) = 110) OR
              (ORD(TECLA1) = 27);
        IF (ORD(TECLA1) = 83) OR (ORD(TECLA1) = 115) THEN
          IF EXTENSION = '.DSC' THEN
            BEGIN
              ASSIGN(ARCHIVO,NOMARCH);
              RESET(ARCHIVO);
              LLENA_REGISTRO;
              WRITE(ARCHIVO,REGISTRO);
              CLOSE(ARCHIVO);
            END
          ELSE
            BEGIN
              ASSIGN(ARCHGRAF,NOMARCH);
              RESET(ARCHGRAF);
              WITH REGGRAF DO
                BEGIN
                  IF ULTIMAGRAF = 'P' THEN
                     BEGIN
                       SALIDA := SALIDAPLANTA;
                       NoPUNTOS := PUNTOPLANTA;
                     END
                  ELSE
                    IF ULTIMAGRAF = 'C' THEN
                       BEGIN
                         SALIDA := SALIDACONT;
                         NoPUNTOS := PUNTOCONT;
                       END
                    ELSE
                      BEGIN
                        SALIDA := SALIDAERROR;
                        NoPUNIOS := PUNTOERROR
                      END;
                  MAYOR := MAX;
                  MENOR := MIN;
                  TIEMPOGRA := TIEMPO;
                  TIPOGRA := ULTIMAGRAF
                END;
              MENSAJE(0);
              MENSAJE(19);
              READLN(DESCRIPCIONTEMP);
              REGGRAF.DESCRIPCION := DESCRIPCIONTEMP;
              WRITE(ARCHGRAF,REGGRAF);
              CLOSE(ARCHGRAF);
            END;
          TEXTBACKGROUND(0);
          CLRSCR;
```

```
            END;
         3 : BEGIN
               MENSAJE(0);
               MENSAJE(16);
             END;
        10 : BEGIN
               IF EXTENSION = '.DSC' THEN
                  BEGIN
                    ASSIGN(ARCHIVO,NOMARCH);
                    REWRITE(ARCHIVO);
                    LLENA_REGISTRO;
                    WRITE(ARCHIVO,REGISTRO);
                    CLOSE(ARCHIVO);
                  END
               ELSE
                  BEGIN
                    ASSIGN(ARCHGRAF,NOMARCH);
                    REWRITE(ARCHGRAF);
                    WITH REGGRAF DO
                      BEGIN
                        IF ULTIMAGRAF = 'P' THEN
                            BEGIN
                              SALIDA := SALIDAPLANTA;
                              NoPUNTOS := PUNTOPLANTA;
                            END
                         ELSE
                           IF ULTIMAGRAF = 'C' THEN
                              BEGIN
                                SALIDA := SALIDACONT;
                                NoPUNTOS := PUNTOCONT;
                              END
                           ELSE
                             BEGIN
                               SALIDA := SALIDAERROR;
                               NoPUNTOS := PUNTOERROR
                             END;
                        MAYOR := MAX;
                        MENOR := MIN;
                        TIEMPOGRA := TIEMPO;
                        TIPOGRA := ULTIMAGRAF
                      END;
                    MENSAJE(0);
                    MENSAJE(19);
                    READLN(DESCRIPCIONTEMP);
                    REGGRAF.DESCRIPCION := DESCRIPCIONTEMP;
                    WRITE(ARCHGRAF,REGGRAF);
                    CLOSE(ARCHGRAF);
                  END;
               TEXTBACKGROUND(0);
               CLRSCR;
             END;
      END; {DEL CASE}
      ULTIMAGRAF := ' ';
    END
  ELSE
```

```
    BEGIN
      TEXTBACKGROUND(0);
      CLRSCR;
    END;
  APAGACURSOR;
END;

{=========================================================================}
{ LLENA UN ARREGLO CON LOS DATOS DE LAS GRAFICAS A MANIPULAR              }

PROCEDURE ARREGLOARCH(X,  { NOMBRE DEL ARCHIVO }
                      D: STRING; { DESCRIPCION DEL ARCHIVO }
                      MAY,MEN,   { MAXIMO Y MINIMO DE LA GRAFICA }
                      T6: REAL;  { TIEMPO DE LA GRAFICA }
                      VAR PRIMERO,ULTIMO: APUNTADOR1); { APUNTADORES DE -
                                                 REFERENCIA DE LOS DATOS }

VAR DATONUEVO: APUNTADOR1; { APUNTADOR AUXILIAR }

BEGIN
  NEW(DATONUEVO);
  WITH DATONUEVO^ DO
    BEGIN
      NOMBREARCH:=X;
      DESCRIPCION := D;
      MAYOR := MAY;
      MENOR := MEN;
      TIEMPOGRA := T6;
      ANTERIOR1:=ULTIMO;
      ULTIMO:=DATONUEVO;
      IF PRIMERO = NIL THEN
         PRIMERO:=DATONUEVO
      ELSE
         ANTERIOR1^.SIGUIENTE1:=ULTIMO;
    END;
END;

{=========================================================================}
{ RECUPERA LOS NOMBRES DE LOS ARCHIVOS DE DATOS EN DISCO                  }

PROCEDURE LEEDATOS;

VAR DIRINFO : SEARCHREC; { ATRIBUTOS DE ARCHIVO }

BEGIN
  PRIMEROARCH := NIL;
  NUMARCH := 0;
  FINDFIRST('*.DSC',ARCHIVE,DIRINFO);
  WHILE DOSERROR = 0 DO
  BEGIN
    NUMARCH := NUMARCH + 1;
    ASSIGN(ARCHIVO,DIRINFO.NAME);
    RESET(ARCHIVO);
    READ(ARCHIVO,REGISTRO);
    CLOSE(ARCHIVO);
```

```
    ARREGLOARCH(DIRINFO.NAME,REGISTRO.DESCRIPCION,0,0,0,PRIMEROARCH,
                ULTIMOARCH);
    FINDNEXT(DIRINFO);
  END;
END;

{=========================================================================}
{ RECUPERA LOS NOMBRES DE LOS ARCHIVOS DE GRAFICAS EN DISCO               }

PROCEDURE LEEGRAFICAS;

VAR DIRINFO : SEARCHREC; { ATRIBUTOS DE ARCHIVO }

BEGIN
  PRIMEROARCH := NIL;
  NUMARCH := 0;
  FINDFIRST('*.GRA',ARCHIVE,DIRINFO);
  WHILE DOSERROR = 0 DO
  BEGIN
    NUMARCH := NUMARCH + 1;
    ASSIGN(ARCHGRAF,DIRINFO.NAME);
    RESET(ARCHGRAF);
    READ(ARCHGRAF,REGGRAF);
    CLOSE(ARCHGRAF);
    WITH REGGRAF DO
      ARREGLOARCH(DIRINFO.NAME,DESCRIPCION,MAYOR,MENOR,TIEMPOGRA,PRIMEROARCH,
                  ULTIMOARCH);
    FINDNEXT(DIRINFO);
  END;
END;

{=========================================================================}
{ BORRA EL ARREGLO DE LOS NOMBRES DE LOS ARCHIVOS CUANDO YA NO SON ---    }
{ REQUERIDOS                                                              }

PROCEDURE BORRARCH;
VAR DATO : APUNTADOR1; { APUNTADOR AUXILIAR }

BEGIN
  WHILE ULTIMOARCH <> NIL DO
    BEGIN
      DATO := ULTIMOARCH;
      ULTIMOARCH := DATO^.ANTERIOR1;
      DISPOSE(DATO);
    END;
END;

{=========================================================================}
{ PRESENTA LOS NOMBRES DE LOS ARCHIVOS Y SU DESCRIPCION EN UNA PANTALLA --- }
{ INICIAL.                                                                }
PROCEDURE DATOSINICIALES(VAR APUNTADORSUP,APUNTADORINF, { APUNTADOR SUPERIOR
                                                          E INFERIOR DE LOS NOMBRES DE ARCHIVOS PRESENTA-
                                                          DOS EN PANTALLA }
                         APUNTADORINT: APUNTADOR1); { APUNTADOR AL -
                                                      NOMBRE DEL ARCHIVO ILUMINADO EN PANIALLA }
```

```
VAR I : INTEGER;  { CONTADOR AUXILIAR }

BEGIN
   APUNTADORSUP:=PRIMEROARCH;
   APUNTADORINF:=PRIMEROARCH;
   apuntadorint := PRIMEROARCH;
   IF NUMARCH <> 0 THEN
      BEGIN
        WINDOW(1,1,80,25);
        MENSAJE(0);
        GOTOXY(3,24);
        WRITE(APUNTADORSUP^.DESCRIPCION);
      END;
   WINDOW(6,9,18,18);
   CLRSCR;
   GOTOXY(1,1);
   I:=1;
   IF NUMARCH > 0 THEN
      BEGIN
        TEXTBACKGROUND(5);
        WRITELN(APUNTADORSUP^.NOMBREARCH:12);
        APUNTADORSUP:=APUNTADORSUP^.SIGUIENTE1;
        TEXTBACKGROUND(3);
      END;
   IF NUMARCH>=10 THEN
      FOR I:=2 TO 9 DO
          BEGIN
             WRITELN(APUNTADORSUP^.NOMBREARCH:12);
             APUNTADORSUP:=APUNTADORSUP^.SIGUIENTE1;
          END
   ELSE
     FOR I:=2 TO NUMARCH-1 DO
         BEGIN
           WRITELN(APUNTADORSUP^.NOMBREARCH:12);
           APUNTADORSUP:=APUNTADORSUP^.SIGUIENTE1;
         END;
   IF NUMARCH > 1 THEN
      WRITE(APUNTADORSUP^.NOMBREARCH:12);
END;


{=============================================================================}
{ MENU DE RECUPERACION DE ARCHIVOS DE DISCO.                                  }

PROCEDURE RECUPERADATOS(NUMEROARCH : INTEGER; { NUMERO DE ARCHIVOS EXISTEN-
                                                TES EN EL DIRECTORIO ACTUAL }
                        EXTENSION : STRING);  { EXTENSION DEL ARCHIVO }
VAR
 M,I,r,   { CONTADORES AUXILIARES }
 SELECCIONADOS : INTEGER; { NUMERO DE ARCHIVOS SELECCIONADOS }
 MINIMO,MAXIMO : REAL; { PUNTOS MINIMOS Y MAXIMOS EN GENERAL DE LAS GRAFICAS}
 APUNTADORSUP, APUNTADORINF,apuntadorint : APUNTADOR1; { APUNTADORES DE
                                          ARCHIVOS EN PANTALLA }
 DIRECTORIO : STRING;  { DIRECTORIO ACTUAL }
```

```
BEGIN
  WRITELN;
  WRITE('Carga de datos':47);
  GOTOXY(26,8);
  WRITE('Directorio actual');
  GOTOXY(26,14);
  WRITE('Cambio de directorio');
  TEXTBACKGROUND(3);
  GOTOXY(25,9);
  WRITE('┌──────────────────────────────────────────────┐'); GOTOXY(25,10);
  WRITE('│                                              │'); GOTOXY(25,11);
  WRITE('└──────────────────────────────────────────────┘'); GOTOXY(25,15);
  WRITE('┌──────────────────────────────────────────────┐'); GOTOXY(25,16);
  WRITE('│                                              │'); GOTOXY(25,17);
  WRITE('└──────────────────────────────────────────────┘'); GOTOXY(5,6);
  WRITE('┌────────────┐'); GOTOXY(5,7);
  WRITE('│  Archivos  │'); GOTOXY(5,8);
  WRITE('└────────────┐'); GOTOXY(5,9);
  WRITE('             │'); GOTOXY(5,10);
  WRITE('             │'); GOTOXY(5,11);
  WRITE('             │'); GOTOXY(5,12);
  WRITE('             │'); GOTOXY(5,13);
  WRITE('             │'); GOTOXY(5,14);
  WRITE('             │'); GOTOXY(5,15);
  WRITE('             │'); GOTOXY(5,16);
  WRITE('             │'); GOTOXY(5,17);
  WRITE('             │'); GOTOXY(5,18);
  WRITE('             │'); GOTOXY(5,19);
  WRITE('             └');
 { MENSAJE(0);
  WINDOW(6,9,18,18);}
  IF EXTENSION = '.DSC' THEN
     LEEDATOS
  ELSE
     LEEGRAFICAS;
  DATOSINICIALES(APUNTADORSUP,APUNTADORINF,APUNTADORINT);
  SELECCIONADOS := 0;
  I := 1;
  R := 1;
  REPEAT
   WINDOW(26,10,78,10);
   CLRSCR;
   GETDIR(0,DIREC
   WRITE(DIRECTORIO);
   WINDOW(6,9,18,18);
   REPEAT
      TECLA2:=READKEY
   UNTIL (ORD(TECLA2)=0) OR (ORD(TECLA2)=67) OR (ORD(TECLA2)=99)
         OR (ORD(TECLA2)=27) OR (ORD(TECLA2) = 13);
   IF ORD(TECLA2)=0 THEN
      TECLA2:=READKEY;
   CASE ORD(TECLA2) OF
        72,73 : IF PRIMEROARCH <> NIL THEN
                BEGIN
                 IF ORD(TECLA2)=72 THEN
```

```
          if apuntadorinf = apuntadorint then
            begin
              r := 1;
              M := 9;
            end
          else
            begin
              r := r-1;
              m := 10;
              gotoxy(1,r+1);
              WRITE(APUNTADORint^.NOMBREARCH:12);
              apuntadorint := apuntadorint^.anterior1;
              textbackground(5);
              gotoxy(1,r);
              WRITE(APUNTADORint^.NOMBREARCH:12);
              textbackground(3);
            end
        ELSE
          M:=0;
        WHILE (I>1) AND (M<>10) DO
          BEGIN
           M:=M+1;
           GOTOXY(1,10);
           DELLINE;
           GOTOXY(1,1);
           INSLINE;
           I:=I-1;
           APUNTADORINF:=APUNTADORINF^.ANTERIOR1;
           APUNTADORSUP:=APUNTADORSUP^.ANTERIOR1;
           WRITE(APUNTADORINF^.NOMBREARCH:12);
           if r < 10 then
            begin
              gotoxy(1,r+1);
              WRITE(APUNTADORint^.NOMBREARCH:12);
            end;
          apuntadorint := apuntadorint^.anterior1;
          textbackground(5);
          gotoxy(1,r);
          WRITE(APUNTADORint^.NOMBREARCH:12);
          textbackground(3);
          END;
          WINDOW(1,1,80,25);
          MENSAJE(0);
          GOTOXY(3,24);
          WRITE(APUNTADORint^.DESCRIPCION);
        END;
80,81 : IF PRIMERDARCH <> NIL THEN
        BEGIN
         IF ORD(TECLA2)=80 THEN
            if apuntadorsup = apuntadorint then
               begin
                  m := 9;
                { r := 10;}
               end
            else
```

```
                IF NUMARCH <> 1 THEN
                begin
                  M:=10;
                  r := r+1;
                  gotoxy(1,r-1);
                  WRITE(APUNTADORint^.NOMBREARCH:12);
                  apuntadorint := apuntadorint^.siguiente1;
                  textbackground(5);
                  gotoxy(1,r);
                  WRITE(APUNTADORint^.NOMBREARCH:12);
                  textbackground(3);
                end
              ELSE
                 M:=0;
              WHILE (I<(NUMARCH-9)) AND (M<>10) DO
               BEGIN
                M:=M+1;
                GOTOXY(1,1);
                DELLINE;
                GOTOXY(1,10);
                INSLINE;
                I:=I+1;
                APUNTADORINF:=APUNTADORINF^.SIGUIENTE1;
                APUNTADORSUP:=APUNTADORSUP^.SIGUIENTE1;
                WRITE(APUNTADORSUP^.NOMBREARCH:12);
                if (r-1) > 0 then
                  begin
                    gotoxy(1,r-1);
                    WRITE(APUNTADORint^.NOMBREARCH:12);
                  end;
                apuntadorint := apuntadorint^.siguiente1;
                textbackground(5);
                gotoxy(1,r);
                WRITE(APUNTADORint^.NOMBREARCH:12);
                textbackground(3);
               END;
             WINDOW(1,1,80,25);
             MENSAJE(0);
             GOTOXY(3,24);
             WRITE(APUNTADORint^.DESCRIPCION);
            END;
99,67 : BEGIN
          WINDOW(1,1,80,25);
          MENSAJE(0);
          PRENDECURSOR;
          WINDOW(26,16,78,18);
          READLN(DIRECTORIO);
          APAGACURSOR;
          {$I-}
          CHDIR(DIRECTORIO);
          IF IORESULT = 0 THEN
             BEGIN
               BORRARCH;
               IF EXTENSION = '.DSC' THEN
                  LEEDATOS
```

```
                    ELSE
                      LEEGRAFICAS;
                  END
                ELSE
                  BEGIN
                    WINDOW(1,1,80,25);
                    MENSAJE(0);
                    MENSAJE(16);
                  END;
                {WINDOW(6,9,18,18);
                CLRSCR;}
                DATOSINICIALES(APUNTADORSUP,APUNTADORINF,APUNTADORINT);
                I := 1;
                R := 1;
                {$I+}
              END;
        13 : BEGIN
              SELECCIONADOS := SELECCIONADOS + 1;
              IF (EXTENSION = '.GRA') AND (APUNTADORINT <> NIL) THEN
                 BEGIN
                   IF SELECCIONADOS=1 THEN
                      BEGIN
                        MIN:=APUNTADORINT^.MENOR;
                        MAX:=APUNTADORINT^.MAYOR;
                        TIEMPO:=APUNTADORINT^.TIEMPOGRA
                      END
                   ELSE
                      BEGIN
                        IF APUNTADORINT^.MENOR < MIN THEN
                           MIN := APUNTADORINT^.MENOR;
                        IF APUNTADORINT^.MAYOR > MAX THEN
                           MAX := APUNTADORINT^.MAYOR;
                        IF APUNTADORINT^.TIEMPOGRA > TIEMPO THEN
                           TIEMPO := APUNTADORINT^.TIEMPOGRA;
                      END;
                   NOMBRE_GRAISELECCIONADOS] := APUNTADORINT^.NOMBREARCH;
                   WINDOW(1,1,80,25);
                   MENSAJE(0);
                   MENSAJE(21);
                   WRITE(SELECCIONADOS:1,'   ',APUNTADORINT^.NOMBREARCH);
                 END;
              END;
     END; {DELCASE}
  UNTIL (ORD(TECLA2) = 27) OR (SELECCIONADOS = NUMEROARCH);
  IF (ORD(TECLA2) = 13) AND (EXTENSION = '.DSC') THEN
     IF APUNTADORINT <> NIL THEN
        SACA_REGISTRO(APUNTADORINT^.NOMBREARCH);
  IF (ORD(TECLA2) <> 27) AND (EXTENSION = '.GRA') THEN
     GRAFICA_DISCO := TRUE;
  BORRARCH;
  WINDOW(1,1,80,25);
  TEXTBACKGROUND(0);
  CLRSCR;
END;
{=========================================================================}
END.
```

```
UNIT GRAFICAR;

{ UNIDAD DE GRAFICACION DE RESULTADOS                                    }
{ ===================================                                    }

INTERFACE

USES CRT,GRAPH,DECLARA,MENSAJES;

PROCEDURE CHECA_MONITOR(P : INTEGER;VAR SALIDA : ARREGLO);
PROCEDURE GRAFICAS(P:INTEGER; VAR SALIDA: ARREGLO; CH:CHAR);
PROCEDURE MUEVE_CURSOR;

IMPLEMENTATION

{=========================================================================}
{ CHECA EL TIPO DE MONITOR Y CALCULA EL PUNTO MAXIMO Y MINIMO PARA EL ---  }
{ DESPLIEGUE DE LAS GRAFICAS.                                             }

PROCEDURE CHECA_MONITOR(P : INTEGER;  { NUMERO DE PUNTOS A GRAFICAR }
                        VAR SALIDA : ARREGLO); {ARREGLO DE PUNTOS A GRAFICAR}
VAR
 MANEJADOR,MODOGRAF,CODIGOERROR : INTEGER; { VARIABLES PARA VALIDACION DE -
                                             MONITOR }
 BEGIN
  MANEJADOR:=DETECT; {DETECTA LA TARGETA DE GRAFICOS AUTOMATICAMENTE}
  INITGRAPH(MANEJADOR,MODOGRAF,'');
  IF GRAPHRESULT<>GROK THEN
     BEGIN
       WRITELN('Error gráfico : ',GraphErrorMsg(codigoerror));
       HALT(1);
     END;
   IF NOT GRAFICA_DISCO THEN
     BEGIN
       MIN:=SALIDA[0];
       MAX:=SALIDA[0];
       FOR I:=1 TO P DO
         BEGIN
           IF SALIDA[I]>MAX THEN MAX:=SALIDA[I];
           IF SALIDA[I]<MIN THEN MIN:=SALIDA[I];
         END;
       IF MAX = MIN THEN
          IF MAX > 0 THEN
             MIN:=0
          ELSE
             MAX:=0;
     END;
END;

{=========================================================================}
{ DESPLIEGA EN PANTALLA LA GRAFICA SELECCIONADA.                          }

PROCEDURE GRAFICAS(P:INTEGER;  { NUMERO DE PUNTOS A GRAFICAR }
                   VAR SALIDA: ARREGLO; { ARREGLO DE PUNTOS A GRAFICAR }
                   CH:CHAR);   { TIPO DE GRAFICA }
```

```
VAR MAXX,MAXY,   ( MAXIMAS COORDENADAS DEL MONITOR )
    I : INTEGER; ( CONTADOR AUXILIAR )
    ESCALAX,ESCALAY : REAL; ( ESCALAS DE LOS EJES )

BEGIN
  MAXX:=GETMAXX;
  MAXY:=GETMAXY;
  ( CALCULO DE LA ESCALA DEL EJE Y, EN BASE A LOS PUNTOS UTILIZABLES DE LA
    PANTALLA )
  ESCALAY:=(MAXY-39)/(MAX-MIN);
  SETUSERCHARSIZE(1,1,1,1);
  CASE CH OF
  'E','P': BEGIN
            IF GRAFICA_DISCO THEN
               ( CALCULO DE LA ESCALA DEL EJE X, EN BASE A LOS PUNTOS
                 UTILIIABLES DE LA PANTALLA Y AL TIEMPO PARTICULAR DE
                 CADA GRAFICA )
              ESCALAX:=TRUNC(REGGRAF.TIEMPOGRA@(MAXX-89)/TIEMPO)/P
            ELSE
              BEGIN
               IF CH = 'P' THEN
                  OUTTEXTXY(290,6,'Salida de la planta')
               ELSE
                  OUTTEXTXY(290,6,'Señal de error');
               ( CALCULO DE LA ESCALA DEL EJE X, EN BASE A LOS PUNTOS
                UTILIZABLES DE LA PANTALLA )
               ESCALAX := (MAXX-89)/P;
              END;
            FOR I:=0 TO P-1 DO
            LINE(60+ROUND(ESCALAX@I),MAXY-19-ROUND(ESCALAY@(SALIDA[I]-MIN)),
                 60+ROUND(ESCALAX@(I+1)),MAXY-19-ROUND(ESCALAY@
                 (SALIDA[I+1]-MIN)));
           END;
      'C': BEGIN
             IF GRAFICA_DISCO THEN
               ESCALAX:=TRUNC(REGGRAF.TIEMPOGRA@(MAXX-89)/TIEMPO)/(P+1)
             ELSE
               BEGIN
                OUTTEXTXY(290,6,'Salida del controlador');
                ESCALAX := (MAXX-89)/(P+1);
               END;
             FOR I:=0 TO (MAXX-90) DO
             PUTPIXEL(60+I,MAXY-19-trunc(ESCALAY@
                    !SALIDA[trunc(I/ESCALAX)]-MIN)),WHITE);
           END;
  END; (DEL CASE)
END;

{************************************************************************}
( POSICIONA EL CURSOR EN CUALQUIER PUNTO DE LA PANTALLA INDICANDO SUS ---  )
( COORDENADAS.                                                            )

PROCEDURE MUEVE_CURSOR;
VAR MAXX,MAXY,  ( MAXIMAS COORDENADAS DE LA PANTALLA )
    X,Y,        ( COORDENADAS ACTULES DEL PIXEL )
```

```
    EJE_Y : INTEGER; { COORDENADA DEL EJE Y }
    ESCALAY : REAL;  { FACTOR DE ESCALA Y }
    NUMALFAX,NUMALFAY : STRING; { COORDENADAS X, Y EN ALFANUMERICOS (PARA
                                  DESPLIEGUE) }

BEGIN
  MAXX:=GETMAXX;
  MAXY:=GETMAXY;
  ESCALAY:=(MAXY-39)/(MAX-MIN);
  RECTANGLE(0,0,MAXX,MAXY);
  SETTEXTSTYLE(2,0,1);
  SETUSERCHARSIZE(7,8,7,8);
  STR(MAX:9,NUMALFAY);
  OUTTEXTXY(7,15, NUMALFAY);
  STR(MIN:9,NUMALFAY);
  OUTTEXTXY(7,MAXY-24, NUMALFAY);
  IF (ROUND(ESCALAY*(0-MIN)) > 7) AND
     ((MAXY-24-ROUND(ESCALAY*(0-MIN))) > 22) THEN
     BEGIN
       STR(0.00:9,NUMALFAY);
       OUTTEXTXY(7,MAXY-24-ROUND(ESCALAY*(0-MIN)), NUMALFAY);
     END;
  STR(TIEMPO:9,NUMALFAX);
  OUTTEXTXY(585,MAXY-14,NUMALFAX);
  EJE_Y:= MAXY-19-ROUND(ESCALAY*(0-MIN));
  LINE(60,15,60,MAXY-14);
  LINE(56,EJE_Y,MAXX-9,EJE_Y);
  LINE(56,20,64,20);
  LINE(56,MAXY-19,64,MAXY-19);
  LINE(MAXX-29,EJE_Y-3,MAXX-29,EJE_Y+3);
  Y := ROUND(MAXY/2);
  X := ROUND(MAXX/2);
  IF Y = EJE_Y THEN
     PUTPIXEL(X,Y,BLACK)
  ELSE
     PUTPIXEL(X,Y,WHITE);
  OUTTEXTXY(100,MAXY-14,'X=');
  OUTTEXTXY(230,MAXY-14,'Y=');
  STR((X-60)*TIEMPO/(MAXX-89):9,NUMALFAX);
  OUTTEXTXY(123,MAXY-14,NUMALFAX);
  STR((MAXY-19-Y)/ESCALAY+MIN:9,NUMALFAY);
  OUTTEXTXY(255,MAXY-14,NUMALFAY);
  SETUSERCHARSIZE(7,8,7,8);
  REPEAT
     REPEAT
        TECLA2 := READKEY
     UNTIL (ORD(TECLA2) = 0)  OR (ORD(TECLA2) = 27) OR (ORD(TECLA2)=50) OR
           (ORD(TECLA2) = 52) OR (ORD(TECLA2) = 54) OR (ORD(TECLA2)=56) ;
     IF ORD(TECLA2) = 0 THEN
        TECLA2:=READKEY;
     IF (Y = EJE_Y) OR (X =60) THEN
        PUTPIXEL(X,Y,WHITE)
     ELSE
        PUTPIXEL(X,Y,BLACK);
     CASE ORD(TECLA2) OF
```

```
72 : IF Y > 20 THEN
       BEGIN
         Y := Y-1;
         SETCOLOR(BLACK);
         OUTTEXTXY(255,MAXY-14,NUMALFAY);
         SETCOLOR(WHITE);
         STR((MAXY-19-Y)/ESCALAY+MIN:9,NUMALFAY);
         OUTTEXTXY(255,MAXY-14,NUMALFAY);
       END;
56 : BEGIN
      IF Y > 30 THEN
        Y := Y-10
      ELSE
        Y:=20;
      SETCOLOR(BLACK);
      OUTTEXTXY(255,MAXY-14,NUMALFAY);
      SETCOLOR(WHITE);
      STR((MAXY-19-Y)/ESCALAY+MIN:9,NUMALFAY);
      OUTTEXTXY(255,MAXY-14,NUMALFAY);
    END;
80 : IF Y < (MAXY-19) THEN
       BEGIN
         Y := Y+1;
         SETCOLOR(BLACK);
         OUTTEXTXY(255,MAXY-14,NUMALFAY);
         SETCOLOR(WHITE);
         STR((MAXY-19-Y)/ESCALAY+MIN:9,NUMALFAY);
         OUTTEXTXY(255,MAXY-14,NUMALFAY);
       END;
50 : BEGIN
      IF Y < (MAXY-29) THEN
        Y := Y+10
      ELSE
        Y := MAXY-19;
      SETCOLOR(BLACK);
      OUTTEXTXY(255,MAXY-14,NUMALFAY);
      SETCOLOR(WHITE);
      STR((MAXY-19-Y)/ESCALAY+MIN:9,NUMALFAY);
      OUTTEXTXY(255,MAXY-14,NUMALFAY);
    END;
75 : IF X > 60 THEN
       BEGIN
         X := X-1;
         SETCOLOR(BLACK);
         OUTTEXTXY(125,MAXY-14,NUMALFAX);
         SETCOLOR(WHITE);
         STR((X-60)*TIEMPO/(MAXX-89):9,NUMALFAX);
         OUTTEXTXY(125,MAXY-14,NUMALFAX);
       END;
52 :BEGIN
     IF X > 70 THEN
       X := X-10
     ELSE
       X := 60;
     SETCOLOR(BLACK);
```

```
            OUTTEXTXY(125,MAXY-14,NUMALFAX);
            SETCOLOR(WHITE);
            STR((X-60)*TIEMPO/(MAXX-89):9,NUMALFAX);
            OUTTEXTXY(125,MAXY-14,NUMALFAX);
          END;
       77 : IF X < (MAXX-29) THEN
              BEGIN
                X := X+1;
                SETCOLOR(BLACK);
                OUTTEXTXY(125,MAXY-14,NUMALFAX);
                SETCOLOR(WHITE);
                STR((X-60)*TIEMPO/(MAXX-89):9,NUMALFAX);
                OUTTEXTXY(125,MAXY-14,NUMALFAX);
              END;
       54 : BEGIN
              IF X < (MAXX-39) THEN
                X := X+10
              ELSE
                X := MAXX-29;
              SETCOLOR(BLACK);
              OUTTEXTXY(125,MAXY-14,NUMALFAX);
              SETCOLOR(WHITE);
              STR((X-60)*TIEMPO/(MAXX-89):9,NUMALFAX);
              OUTTEXTXY(125,MAXY-14,NUMALFAX);
            END;
   END;
   IF (Y = EJE_Y) OR (X = 60) THEN
      PUTPIXEL(X,Y,BLACK)
   ELSE
      PUTPIXEL(X,Y,WHITE);
 UNTIL ORD(TECLA2) = 27;
 CLOSEGRAPH;
 RESTORECRTMODE;
 APAGACURSOR;
 TEXTBACKGROUND(0);
 CLRSCR;
END;

END.
```

```
UNIT DECLARA;

{ UNIDAD QUE CONTIENE LAS VARIABLES GLOBALES DEL PAQUETE                    }
{ ==========================================                                }

INTERFACE

{=============================================================================}

 CONST GRADOPOL = 17;   { GRADO MAXIMO DE LOS POLINOMIOS }

      COLUMNAS = 18;    { NUMERO MAXIMO DE COLUMNAS DE LA MATRIZ A RESOLVER}

      E = 0.00001;      { GRADO DE ERROR }

{=============================================================================}


 TYPE

 ARREGLO = ARRAY[0..600] OF REAL;         { ARREGLO DE PUNTOS A GRAFICAR }
 POLINOMIO = ARRAY[0..GRADOPOL] OF REAL;  { ARREGLO DE COEFICIENTES DE
                                            POLINOMIOS }
 APUNTADOR = ^COMPONENTE;                   { APUNTADOR DE LOS PUNTOS
                                              CALCULADOS EN EL SISTEMA }
 COMPONENTE = RECORD                        { REGISTRO DEL PUNTO CALCULADO}
              DATO: REAL;                     { PUNTO CALCULADO }
              ANTERIOR,SIGUIENTE: APUNTADOR  { LIGAS DEL APUNTADOR }
             END;

 APUNTADOR1 = ^COMPONENTE1;                 { APUNTADOR PARA EL REGISTRO DE
                                              NOMBRES DE LAS GRAFICAS }
 COMPONENTE1 = RECORD                       { REGISTRO DE DATOS DE LAS
                                              GRAFICAS }
              NOMBREARCH : STRING[12];          { NOMBRE DEL ARCHIVO }
              MAYOR,MENOR,TIEMPOGRA : REAL;     { PUNTO MAXIMO, MINIMO Y
                                                  TIEMPO DE GRAFICACION }
              DESCRIPCION : STRING[65];         { DESCRIPCION DEL ARCHIVO }
              ANTERIOR1,SIGUIENTE1: APUNTADOR1; { LIGA AL REGISTRO ANTERIOR
                                                  Y SIGUIENTE }
             END;

 ENTRADA1 = RECORD                          { REGISTRO DE DATOS DE LA ENTRADA)
                                            { PARA SU ALMACENAMIENTO EN DISCO)
             TIPOENT : INTEGER;             { TIPO DE ENTRADA }
             A,B,C   : REAL;                { CTES. DE LA ENTRADA }
           END;

 ARCHDATOS = RECORD                         { REGISTRO DEL ARCHIVO DE DATOS }
              NUMERADORPL,DENOMINADORPL,        {COEFICIENTES DE LAS FUN-)
              NUMERADOR,DENOMINADOR : POLINOMIO; {CIONES DE TRANSFERENCIA }
              GRADONUMPL,GRADODENPL,     { GRADOS DE LOS POLINOMIOS DE LAS)
              GRADONUM,GRADODEN ,        { FUNCIONES DE TRANSFERENCIA }
              NUMPUNTOS : INTEGER;       { NUMERO DE PUNTOS }
              GANANCIARCH,               { GANANCIA DEL SISTEMA }
```

```
            TIEMPOA,                ( TIEMPO A ANALIZAR )
            PM : REAL;              ( PERIODO DE MUESTREO )
            ENTRADA : ENTRADA1;     ( REGISTRO DE LA ENTRADA )
            NODISCRETO : BOOLEAN;   ( INDICA DISCRETIZACION )
            DATOSARCH : STRING[13]; ( OPCIONES SELECCIONADAS )
            DESCRIPCION : STRING[65]; ( DESCRIPCION DEL ARCHIVO )
          END;

REGISTROGRAF = RECORD                 ( REGISTRO DE DATOS DE LAS
                                        GRAFICAS )
               SALIDA : ARREGLO;      ( PUNTOS A GRAFICAR )
               NoPUNTOS : INTEGER;    ( No. DE PUNTOS A GRAFICAR )
               MAYOR,MENOR,           ( PUNTO MAXIMO Y MINIMO )
               TIEMPOGRA : REAL;      ( TIEMPO A GRAFICAR )
               TIPOGRA : CHAR;        ( TIPO DE GRAFICA )
               DESCRIPCION : STRING[65]; ( DESCRIPCION DEL ARCHIVO )
             END;

RAIZRI = RECORD               ( REGISTRO DE RAICES )
           RE : REAL;         ( PARTE REAL )
           IM : REAL          ( PARTE IMAGINARIA )
         END;

POLRA = RECORD                  ( REGISTRO DE POLINOMIOS )
          POLRAIZ : POLINOMIO;  ( COEFICIENTES DEL POLINOMIO )
          ORDENPOLRAIZ : INTEGER; ( GRADO DEL POLINOMIO )
        END;

FRACCPARCIALES = RECORD           ( REGISTRO DE FRACCIONES PARCIALES )
                   NUMERADOR : POLINOMIO;   ( POLINOMIO NUMERADOR )
                   DENOMINADOR : POLINOMIO; ( POLINOMIO DENOMINADOR )
                   ORDENNUM,                ( GRADO DEL NUMERADOR )
                   ORDENDENOM : INTEGER;    ( GRADO DEL DENOMINADOR )
                 END;

RAICESREPETIDAS = RECORD          ( REGISTRO DE RAICES REPETIDAS )
                    POLRAIZ : POLINOMIO;    ( COEFICIENTES DEL POLINOMIO )
                    ORDEN,                  ( GRADO DEL POLINOMIO )
                    REPETICIONES : INTEGER; ( NUMERO DE REFETICIONES )
                  END;

ENTERONONEGATIVO = 0.. MAXINT;  ( MAXIMO ENTERO POSITIVO )
ENTEROPOSITIVO = 1..MAXINT;     ( MAXIMO ENTERO POSITIVO DESDE EL UNO )
MATRIX = ARRAY[0..GRADOPOL,0..COLUMNAS] OF REAL; ( MATRIZ A RESOLVER )
POLIRAIZ = ARRAY[1..GRADOPOL] OF POLRA;          ( ARREGLO DE POLINOMIOS )
RA = ARRAY[1..GRADOPOL] OF RAIZRI;               ( ARREGLO DE RAICES )
RAREP = ARRAY[1..GRADOPOL] OF RAICESREFETIDAS;   ( ARREGLO DE RAICES
                                                   REPETIDAS )
FRACC = ARRAY[1..GRADOFOL] OF FRACCPARCIALES;    ( ARREGLO DE FRACCIONES
                                                   PARCIALES )

{=========================================================================}

VAR
```

```
( K : ARRAY[1..4,1..15] OF REAL;)
Y : ARRAY[1..15] OF REAL;  ( REGISTRO TEMPORAL DE SALIDAS DE RUNGE-KUTTA )

SALIDAPLANTA,          ( SALIDA DE LA PLANTA )
SALIDACONT,            ( SALIDA DEL CONTROLADOR )
SALIDAERROR : ARREGLO; ( SALIDA DE LA SENAL DE ERROR )

COEFSPP,               ( POLOS DE LA PLANTA EN S )
COEFSOP,               ( CEROS DE LA PLANTA EN S )
POLAMS,                ( POLOS DEL SISTEMA EN S)
POLBMS,                ( CEROS DEL SISTEMA EN S )
COEFZPC,               ( POLOS DEL CONTROLADOR EN Z )
COEFZOC,               ( CEROS DEL CONTROLADOR EN Z )
POLA,                  ( POLOS DE LA PLANTA EN Z )
POLAM,                 ( POLOS DEL SISTEMA EN Z )
POLB,                  ( CEROS DE LA PLANTA EN Z )
POLBM,                 ( CEROS DEL SISTEMA EN Z )
POLT,POLR1,POLR,POLS,  ( POLINOMIOS RESULTANTES DEL DISEND )
POLBMAS,               ( POLINOMIO DE CEROS ESTABLES )
POLBMENOS,             ( POLINOMIO DE CEROS INESTABLES )
POLAUXILIAR,           ( POLINOMIO AUXILIAR )
SOL,                   ( SOLUCIONES DEL SISTEMA DE ECUACIONES )
COMDENOM,              ( COMUN DENOMINADOR DE FRACCIONES PARCIALES EN Z )
COMNUM,                ( COMUN NUMERADOR DE FRACCIONES PARCIALES EN Z )
S,S2 : POLINOMIO;      ( REGISTROS TEMPORALES DE SALIDA DE LOS -
                         CONTROLADORES )
GRADOPP,               ( GRADO DE LOS POLOS DE LA PLANTA EN S )
GRADOOP,               ( GRADO DE LOS POLOS DE LA PLANTA EN S )
GRADPOLAMS,            ( GRADO DE LOS POLOS DEL SISTEMA EN S )
GRADPOLBMS,            ( GRADO DE LOS CEROS DEL SISTEMA EN S )
I,J,K,N,NN,M,          ( CONTADORES AUXILIARES )
JR,                    ( CONTADOR DE RUNGE-KUTTA )
FUNCION,               ( TIPO DE ENTRADA AL SISTEMA )
NUMARCH,               ( NUMERO DE ARCHIVOS )
GRADOZPC,              ( GRADO DE LOS POLOS DEL CONTROLADOR EN Z )
GRADOZOC,              ( GRADO DE LOS CEROS DEL CONTROLADOR EN Z )
GRADPOLBMAS,           ( GRADO DEL POLINOMIO DE RAICES ESTABLES )
GRADPOLBMENOS,         ( GRADO DEL POLINOMIO DE RAICES INESTABLES )
GRADPOLB,              ( GRADO DEL POLINOMIO DE CEROS DE LA PLANTA EN Z )
GRADPOLR1,             ( GRADO DEL POLINOMIO R1 )
GRADPOLS,              ( GRADO DEL POLINOMIO S )
GRADPOLR,              ( GRADO DEL POLINOMIO R )
GRADPOLT,              ( GRADO DEL POLINOMIO T )
GRADPOLA,              ( GRADO DEL POLINOMIO DE POLOS DE LA PLANTA EN Z )
GRADPOLAM,             ( GRADO DE LOS POLOS DEL SISTEMA EN Z )
GRADPOLBM,             ( GRADO DE LOS CEROS DEL SISTEMA EN Z )
GRADPOL_AR1,           ( GRADO RESULTANTE DE MULTIPLICAR POLA * POLR1 )
GRADPOL_BS,            ( GRADO RESULTANTE DE MULTIPLICAR POLB * POLS )
GRADCOMDENOM,          ( GRADO DEL COMUN DENOMINADOR Y NUMERADOR DE LAS - )
GRADCOMNUM,            ( FRACCIONES PARCIALES EN Z )
GRADMAYORPOL,          ( GRADO MAYOR DE UN POLINOMIO )
GRADMENORPOL,          ( GRADO MENOR DE UN POLINOMIO )
KC,                    ( VARIABLE K DE RUNGE-KUTTA )
PUNTOPLANTA,           ( INDICE DEL ARREGLO DE LOS PUNTOS A GRAFICAR -
                         DE LA PLANTA )
```

```
PUNTOCONT,              { INDICE DEL ARREGLO DE LOS PUNTOS A GRAFICAR -
                          DEL CONTROLADOR }
PUNTOERROR,             { INDICE DEL ARREGLO DE LOS PUNTOS A GRAFICAR -
                          DE LA SENAL DE ERROR }
PTOSPLANTA,             { NUMERO DE PUNTOS GENERADOS POR LA PLANTA }
COLMAT,                 { NUMERO DE COLUMNAS DE LA MATRIZ }
ORDMATRIZ,              { ORDEN DE LA MATRIZ }
CONTADORPOLOMAS,        { CONTADOR DE RAICES ESTABLES }
CONTADORPOLOMENOS,      { CONTADOR DE RAICES INESTABLES }
REN,                    { RENGLON DE LA MATRIZ }
COL,                    { COLUMNA DE LA MATRIZ }
TOTALREALES,            { TOTAL DE RAICES REALES }
TOTALIMAGS,             { TOTAL DE RAICES IMAGINARIAS }
REALESREPETIDAS,        { TOTAL DE RAICES REALES REPETIDAS }
IMAGSREPETIDAS,         { TOTAL DE RAICES IMAGINARIAS REPETIDAS}
CONTADORPARCIALES,      { CONTADOR DE FRACCIONES PARCIALES }
CONRA,                  { CONTADOR DE RAICES }
NoGRAFICAS : INTEGER;   { NUMERO DE GRAFICAS }

T,                      { TIEMPO ACTUAL }
H,                      { INCREMENTOS DE TIEMPO DE RUNGE-KUTTA }
A,B,C,                  { PARAMETROS DE LA ENTRADA AL SISTEMA }
TIEMPO,                 { TIEMPO DE ANALISIS }
TM,                     { TIEMPO DE MUESTREO }
GANANCIA,               { GANANCIA DEL SISTEMA }
BUFFER,                 { BUFFER PARA LA VALIDACION DE NUMEROS REALES }
ESCALAPLANTA,           { ESCALA TOMADA PARA LA GRAFICACION EN PANTALLA -
                          DE LA PLANTA }
ESCALACONT,             { ESCALA TOMADA PARA LA GRAFICACION EN PANTALLA -
                          DEL CONTROLADOR }
NoPUNTOSPLANTA,         { NUMERO DE PUNTOS QUE CALCULARA RUNGE-KUTTA }
NoPUNTOSCONT,           { NUMERO DE PUNTOS QUE SE CALCULARAN PARA EL -
                          CONTROLADOR }
MAX,MIN,                { PUNTO MAXIMO Y MINIMO DEL ARREGLO A GRAFICAR }
SALIDACONT1,SALIDACONT2, { SALIDAS ACTUALES DE LOS CONTROLADORES }
ra,sa,DENOM,            { ACUMULADORES AUXILIARES PARA EL METODO DE -
                          LIN-BAIRSTOW }
PTOINIC : REAL;         { PTO. DE INICIO PARA EL METODO LIN-BAIRSTOW }

TECLA,TECLA1,TECLA2,TECLA3,    { ULTIMA TECLA PRESIONADA }
TIPO,                          { TIPO DE SISTEMA }
BLOQUE,                        { BLOQUE ACTIVO DEL SISTEMA (PLANTA, CON-
                                 TROLADOR, G1, G2) }
ULTIMAGRAF : CHAR;             { INDICADOR DEL TIPO DE LA ULTIMA GRAFICA
                                 DESPLEGADA EN PANTALLA}

REBOSAMIENTO,                  { INDICA LIMITE EXCEDIDO }
GRAFICA_DISCO,                 { INDICA LECTURA DE GRAFICA DE DISCO }
DATOS_PROCESADOS,              { INDICA EL PROCESAMIENTO DE DATOS }
YALLAMADO,                     { INDICA CALCULO DE RAIZ REALIZADO }
SINGULAR,                      { INDICA SI LA MATRIZ ES SINGULAR }
NO_DISCRETO,                   { INDICA DISCRETIZACION }
LECTURA_NODISCRETA : BOOLEAN;  { INDICA LECTURA DISCRETA }

DATOS: STRING[1];              { OPCIONES SELECCIONADAS }
```

```
PRIMEROPLANTA, ULTIMOPLANTA, PRIMEROCONT1, ULTIMOCONT1,
PRIMEROCONT2, ULTIMOCONT2 : APUNTADOR; ( APUNTADORES PARA LA PLANTA Y LOS
                                         CONTROLADORES )

ARCHIVO : FILE OF ARCHDATOS;           ( ARCHIVO DE DATOS )

REGISTRO : ARCHDATOS;                  ( REGISTRO DEL ARCHIVO DE DATOS )

ARCHGRAF : FILE OF REGISTROGRAF;       ( ARCHIVO DE GRAFICAS )

REGGRAF : REGISTROGRAF;                ( REGISTRO DEL ARCHIVO DE GRAFICAS )

PRIMEROARCH,ULTIMOARCH : APUNTADOR1;   ( APUNTADORES PARA LOS NOMBRES DE
                                         ARCHIVOS )

NOMBRE_GRA : ARRAY[1..5] OF STRING[12]; ( NOMBRES DE GRAFICAS A DESPLEGAR
                                          A LA VEZ )

P,Q : ARRAY[1..GRADOPOL] OF REAL;      ( POLINOMIOS AUXILIARES PARA EL
                                         METODO DE LIN-BAIRSTOW )

MATRIZ : MATRIX;          ( MATRIZ A RESOLVER COMO UN SISTEMA DE ECUACIONES)

POLRAIZMAS,               ( POLINOMIO DE RAICES ESTABLES )
POLRAIZMENOS : POLIRAIZ;  ( POLINOMIO DE RAICES INESTABLES )

RAICESREALES,             ( RAICES REALES REPETIDAS )
RAICESIMAS : RAREP;       ( RAICES IMAGINARIAS REPETIDAS )

PARCIAL : FRACC;          ( FRACCIONES PARCIALES )

RAIZ : RA;                ( RAICES DE UN POLINOMIO (REALES E IMAGINARIAS) )

IMPLEMENTATION

END.
```