UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO

FACULTAD DE INGENIERIA

DESARROLLO DE UN PROGRAMA PARA ANALISIS SIMULACION Y DISEÑO DE SISTEMAS DINAMICOS EN EL DOMINIO DEL TIEMPO

T E S I S

QUE PARA OBTENER EL TITULO DE

INGENIERO EN COMPUTACION

P R E S E N T A :

LETICIA LANIZAR URIBE

DIRECTOR DE TESIS :

ING. FRANCISCO RODRIGUEZ RAMIREZ

MEXICO, D. F. 1992

UNAM – Dirección General de Bibliotecas Tesis Digitales Restricciones de uso

I N D I C E

INTRODUCCION

Actualmente existen muy diversos paquetes de control que brindan apoyo en el analisis y diseño de sistemas dinamicos. Entre los mas utilizados podemos citar al paquete cc, que ofrece gran capacidad para el manejo de sistemas. Este paquete utiliza 228 comandos y requiere de un coprocesador matemático.

Lo que pretendemos en un futuro es integrar un paquete de control tan poderoso como cc pero que no requiera de un coprocesador matemático y que no maneje el gran numero de comandos que utiliza cc. Se pretende que sea un paquete facil de utilizar, por lo que presentara una interface amigable con el usuario, en español y que sea el resultado del desarrollo de alumnos de la Fi de la UNAM.

En el presente trabajo muestro un sistema que forma parte de este proyecto. Se trata de un programa para el calculo de la respuesta en el tiempo de sistemas dinamicos ante diversas excitaciones.

Organización del material expuesto

En el capitulo I expongo la teoria de sistemas dinámicos sobre la que se basa el programa que presento. Dentro de los capitulos II y III expongo el desarrollo sistemático que se siguió en la elaboracion del programa de computadora; en el capitulo II la fase de definicion de requerimientos y en el capitulo III la fase de diseño. En el capitulo IV expongo algunos casos de prueba. Finalmente en el apéndice A anexo una copia del código fuente del programa.

Caracteristicas del programa

El programa principal.exe contiene la version ejecutable del

programa para cálculo de la respuesta en el tiempo de sistemas dinámicos. Para su ejecución se requiere de una PC compatible con IBM bajo el sistema operativo MS-DOS, con 256 Kilobytes de memoria RAM y una unidad de discos como mínimo.

El programa está escrito en lenguaje Turbo Pascal V5.5. El texto fuente desarrollado está conformado por 4265 líneas dentro de las unidades principal, resp1, resp2 y resp3. Para su implementación se utilizaron las librerías contenidas en la unidad MENUS, la cual fue desarrollada previamente con el propósito de presentar una interface amigable con el usuario.

CAPITULO I BASE TEORICA

I. TIPOS DE RESPUESTA

En el análisis y diseño de sistemas dinámicos en el dominio del tiempo, es importante el estudio de los diferentes tipos de respuesta que el sistema presenta cuando es sometido a diversas señales de entrada. Este estudio permitirá el desarrollo de sistemas más satisfactorios, ya que proporciona una base para comparar diseños alternos.

Como primer punto en este capítulo se desarrollará el tema de tipos de respuesta para sistemas dinámicos.

Considérese la siguiente representación esquemática de un sistema:[1]

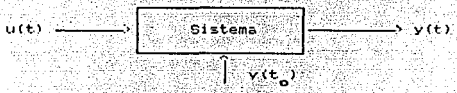


Fig. 1

En donde

u(t): Es la entrada del sistema
$y(t_o)$: Es el estado inicial
y(t): Es la salida del sistema

En general, la salida producida por el sistema en el intervalo (t_o,t), depende de la entrada del sistema u(t), las características propias del sistema y de su estado inicial $y(t_o)$. A partir de estos elementos se pueden obtener las respuestas:

(1) Rodríguez, Francisco. Apuntes de sistemas dinámicos p.170

-Libre
-Forzada
-Total
-Permanente
-Transitoria

I.1 Respuesta libre: Es la respuesta producida por el sistema cuando la entrada aplicada es nula y las condiciones iniciales son distintas de cero. Por lo que podemos decir que la respuesta libre es aquella que depende únicamente de los parámetros propios del sistema y de su estado inicial.

Tomando como base la representación del sistema en la figura 1, se tendría:

$y(t_o) \neq 0$

Fig. 2

I.2 Respuesta forzada: Es la respuesta producida por el sistema cuando la entrada aplicada es distinta de cero y su estado inicial es cero. Esto es, la respuesta forzada es aquella que depende exclusivamente de la entrada o excitación externa y de los parámetros propios del sistema. Esquemáticamente:

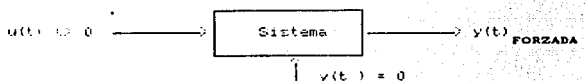


Fig. 3

I.3 Respuesta total: Es aquella que depende de la entrada aplicada, de las características propias del sistema y de su estado inicial. Esto es, la respuesta total es aquella producida cuando la entrada y las condiciones iniciales son distintas de cero. En forma esquemática, podemos representar a la respuesta total como:

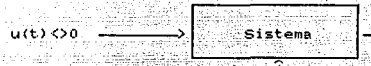


Fig. 4

La respuesta total puede ser obtenida a partir de la siguiente expresión:

$$y(t)_{TOTAL} = y(t)_{LIBRE} + y(t)_{FORZADA}$$

I.4 Respuesta permanente: Es la forma en que la salida del sistema se comporta para tiempos grandes. Es la respuesta que el sistema produce cuando se ha estabilizado.

La respuesta permanente se puede obtener a través de la siguiente expresión:

$$y(t)_{PERMANENTE} = \lim_{t \to \infty} y(t)_{TOTAL}$$

I.5 Respuesta transitoria: Cuando se aplica una señal de entrada a un sistema estable ideal, se espera que la salida siga fielmente a la señal de entrada. En forma esquemática, la respuesta escalón de un sistema ideal sería:

(2) Ibidem p. 180

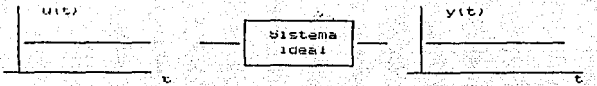


Fig. 5

Dentro de los sistemas reales esto no es posible, ya que generalmente se presenta un comportamiento transitorio antes de que el sistema alcance un estado estable.

La respuesta transitoria es la forma en que el sistema responde antes de alcanzar el estado estable, o bien, es la respuesta del sistema desde el estado inicial hasta el estado final. La respuesta transitoria se puede obtener a partir de la siguiente expresión:

$$y(t)_{\text{TRANSITORIA}} = y(t)_{\text{TOTAL}} - y(t)_{\text{PERMANENTE}}$$

II. SEÑALES DE PRUEBA TIPICAS

En la práctica, la señal real de entrada a un sistema dinámico se desconoce. La entrada instantánea no puede expresarse analíticamente, ya que ésta es de naturaleza aleatoria, a excepción de algunos casos especiales. Al analizar y diseñar sistemas, es deseable tener una base de comparación del funcionamiento de diversos sistemas. Muchos criterios de diseño se basan en el estudio de la respuesta de los sistemas a señales de entrada particulares, comparando después las respuestas del sistema con las señales de entrada. Estos criterios son convenientes si se toma en consideración el hecho de que existe gran posibilidad de que si la respuesta de un sistema a una entrada estándar de prueba es satisfactoria, también lo será la respuesta del sistema bajo condiciones de operación normal.

Muchos sistemas operan con señales de entrada muy semejantes a las señales de prueba estándar.[3]

Las señales de prueba utilizadas más comunmente son las funciones escalón, rampa, impulso, sinusoidal, etc. Estas funciones de prueba dan la facilidad de realizar análisis matemáticos y experimentales del sistema con sencillez, ya que son señales de entrada muy simples. La señal o señales de prueba a utilizar deberán de determinarse por la forma de la señal de entrada a la que el sistema estará expuesto más frecuentemente. Si por ejemplo, el sistema estará sometido a señales de entrada que varian gradualmente con el tiempo, una buena elección para la señal de prueba a utilizar, podría ser la función rampa. De igual manera, si el sistema recibe frecuentemente perturbaciones bruscas, la función impulso unitario sería buena elección. Y para sistemas que reciben comunmente entradas bruscas, la entrada escalón sería conveniente como señal de prueba.

Utilizar señales de prueba permite al diseñador comparar el comportamiento de diversos sistemas sobre una misma base.

II.1 Respuesta impulso

La función impulso unitario es una función matemática ideal, por lo que en la realidad sólo se pueden obtener aproximaciones. Se basa en una función rectangular

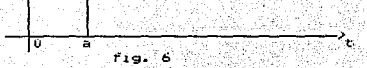


fig. 6

A medida que a tiende a cero, la función r(t) tiende a la función impulso unitario, por lo que se tiene:

(3) Dorf, Richard Sistemas automáticos de control, p. 96

$$\delta(t) = \lim_{a \to 0} f(t)$$

Algunas de las propiedades más importantes que presenta la función impulso unitario son:

$$L[\,\delta(t)\,] = 1$$

$$L[\,k\delta(t)\,] = k$$

$$\delta(t-t_o) = 0 \text{ para } t \neq t_o$$

$$\int_{t_1}^{t_2} \delta(t-t_o)\,dt = 1 \text{ para } t_1 < t < t_2$$

Gráficamente la función impulso unitario se representa:

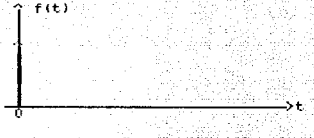


fig. 7

La señal de prueba de respuesta al impulso, físicamente se puede aproximar sometiendo al sistema a un pulso estrecho y de gran amplitud.

La respuesta impulso es la respuesta de un sistema cuando se somete a una entrada impulso y las condiciones iniciales del sistema son nulas.[5] Gráficamente:

(4) Zill, Dennis Ecuaciones diferenciales con aplicaciones p.285
(5) Ogata, Ingeniería de control moderna p.285

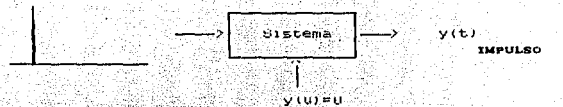


fig. 8

Como L{ $\delta(t)$ } = 1, la transformada de Laplace de la salida del sistema es igual a la función de transferencia.

La respuesta impulso es de gran utilidad e importancia, ya que contiene la misma información sobre la dinámica del sistema que la función de transferencia.

La respuesta transitoria de un sistema depende de sus condiciones iniciales. Por conveniencia y con el fin de poder comparar la respuesta transistoria de distintos sistemas, se acostumbra a utilizar condiciones iniciales nulas al aplicar señales de prueba.

11.2 Respuesta escalón:

La función escalón unitario $u_{-1}(t)$ se define como:(6)

$$u_{-1}(t) = \begin{cases} 0, & t < 0 \\ 1, & t \geq 0 \end{cases}$$

Algunas de las propiedades más importantes de la función escalón unitario son:

a) Multiplicación por una constante.
Sea f(t) una función definida como:

$$f(t) = k\, u_{-1}(t)$$

(6)Zill, Dennis, Op.cit. p.804

Para k positiva:

$$f(t) = \begin{cases} 0, & t < 0 \\ k, & t \geq 0 \end{cases}$$

Gráficamente:

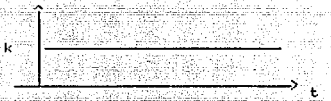


fig. 9

Para k negativa

$$f(t) = \begin{cases} 0, & t < 0 \\ k, & t \geq 0 \end{cases}$$

Gráficamente:

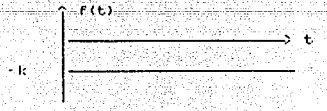


fig. 10

b) Derivación. La derivada de la función escalón unitario es la función impulso unitario, esto es:

$$\frac{d}{dt}\left[u_{-1}(t) \right] = \delta(t)$$

c) Transformada de Laplace:

$$L\left[u_{-1}(t) \right] = 1/s$$

La respuesta escalón de un sistema se obtiene cuando la entrada al sistema es la función escalón y el estado inicial es nulo.

Gráficamente se tiene:

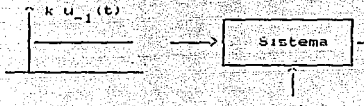


fig. 11

Dentro del análisis y diseño de sistemas dinámicos, la respuesta escalón es de gran importancia, ya que en muchas ocasiones las características deseadas de los sistemas se especifican en términos de la respuesta transitoria a una entrada escalón, ya que la función escalón es fácil de generar.

11.2.1 Respuesta escalón para sistemas de primer orden:

Sea el sistema de primer orden con la siguiente función de transferencia:

$$G(S) = \frac{1}{Ts + 1} = \frac{y(s)}{x(s)} \quad -- (1)$$

Dado que la transformada de Laplace de la función escalón unitario es 1/s, la salida del sistema se obtendría de: (7)

$$y(s) = \frac{1}{Ts + 1} \frac{1}{s} = \frac{1}{s} - \frac{T}{Ts + 1} \quad --(2)$$

$$y(t) = L^{-1}(y(s)) = 1 - e^{-t/T} \quad t \geq 0 \quad -- (3)$$

De esta ecuación se observa que la salida del sistema inicialmente toma el valor de cero y finalmente tiende a la unidad.

Una de las características importantes de la respuesta escalón para sistemas de primer orden es que para t=T, la salida del sistema y(t) alcanza el 63.2% de su valor final. Como ya es conocido, al valor de T se le da el nombre de constante de tiempo del sistema. Mientras menor sea el valor de T, más rápida será la respuesta del sistema.

II.2.2 Respuesta escalón para sistemas de segundo orden.

Sea la función de transferencia para un sistema de segundo orden: (8)

$$G(S) = \frac{\omega_n^2}{s^2 + 2\zeta\omega_n + \omega_n^2} = \frac{y(s)}{x(s)} \quad -- (4)$$

(7) Ogata op. cit. p. 288
(8) Ibidem p. 245

Donde los parámetros ω_n y ζ se definen:

ω_n: Frecuencia natural no amortiguada del sistema

ζ: Factor de amortiguamiento relativo del sistema

Para una entrada escalón unitario, se tiene:

$$y(S) = \frac{\omega_n^2}{(s^2 + 2\zeta\omega_n + \omega_n^2)\, s} \qquad -- (5)$$

El comportamiento dinámico de los sistemas de segundo orden se puede describir en función de ζ y ω_n. Para $0 < \zeta < 1$, los polos de lazo cerrado son complejos conjugados y se localizan en el semiplano complejo izquierdo. Para este caso se dice que el sistema es subamortiguado. Para $\zeta = 1$ los polos de lazo cerrado son reales iguales y se dice que el sistema tiene amortiguamiento crítico. Para $\zeta > 1$ el sistema es sobreamortiguado. Los sistemas con amortiguamiento crítico y sobreamortiguados no presentan oscilaciones en su respuesta transitoria. Cuando $\zeta = 0$ se presenta el caso no amortiguado. Los polos se localizan sobre el eje imaginario y la respuesta transitoria no se extingue.

Para la respuesta escalón de un sistema de segundo orden se considerarán cuantro casos:

(1) Caso subamortiguado ($0 < \zeta < 1$)

La ecuación (5) se puede descomponer como:[9]

[9] Idem p. 247

$$y(S) = \frac{1}{s} \cdot \frac{s + 2\zeta\omega_n}{s^2 + 2\zeta\omega_n + \omega_n^2} \quad \text{-- (6)}$$

$$= \frac{1}{s} \cdot \frac{s + \zeta\omega_n}{(s + \zeta\omega_n)^2 + \omega_d^2} - \frac{\zeta\omega_n}{(s + \zeta\omega_n)^2 + \omega_d^2} \quad \text{--(7)}$$

Donde:

$$\omega_d = \omega_n \sqrt{1-\zeta^2} \quad (10)$$

$$y(t) = 1 - e^{-\zeta\omega_n t} \cos \omega_d t + \frac{\zeta}{\sqrt{1-\zeta^2}} \text{ sen } \omega_d t) \quad \text{--(8)}$$

(2) Caso no amortiguado ($\zeta = 0$)

De la ecuación (8), haciendo $\zeta = 0$ se tiene

$$y(t) = 1 - \cos \omega_n t \qquad (t \geq 0) \qquad \text{--(9)}$$

(3) Caso de amortiguamiento crítico (ζ=1)

$$y(s) = \frac{\omega_n^2}{(s + \omega_n)^2}$$

$$y(t) = L^{-1}(y(s)) = 1 - e^{-\omega_n t} (1 + \omega_n t) \quad t \geq 0 \quad \text{--(9)}$$

(4) Caso sobreamortiguado ($\zeta > 1$)

Para este caso se tienen polos reales distintos (11)

$$y(s) = \frac{\omega_n^2}{(s + \zeta\omega_n + \omega_n \sqrt{-1 + \zeta^2})(s + \zeta\omega_n - \omega_n \sqrt{-1 + \zeta^2}}$$

(10) A ω_d se le denomina frecuencia natural amortiguada

(11) Ogata op. cit. p. 248

$$y(t) = L^{-1}\{ y(s) \} = 1 + \frac{e^{-(\zeta + \sqrt{-1 + \zeta^2}\omega_n t}}{2\sqrt{-1 + \zeta^2}(\zeta + \sqrt{-1 + \zeta^2})} \quad \text{---(10)}$$

$$t \geq 0$$

II.3 Respuesta rampa. La función rampa se define

$$f(t) = \begin{cases} 0, & t < 0 \\ t, & t \geq 0 \end{cases}$$

Su representación gráfica es:

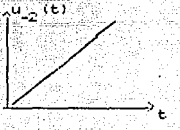


fig. 12

Entre las propiedades más importantes de la función rampa se tienen:

a) Multiplicación por una constante

Sea f(t) una función definida por

$$f(t) = k\, u_{-2}(t)$$

Donde k es una constante. Para k positiva:

$$f(t) = \begin{cases} 0, & t < 0 \\ kt, & t \geq 0 \end{cases}$$

Para k negativa:

$$f(t) = \begin{cases} 0, & t < 0 \\ -kt, & t \geq 0 \end{cases}$$

b) Derivación. La derivada de la función rampa es la función escalón.

$$\frac{d}{dt}[\, u_{-2}(t)] = u_{-1}(t)$$

c) Integración. La integral de la función rampa es la función parábola

$$\int_0^t u_{-2}(t)dt = u_{-3}(t)$$

Donde

$$u_{-3}(t) = \begin{cases} 0, & t < 0 \\ \frac{t^2}{2}, & t \geq 0 \end{cases}$$

d) Transformación (Laplace)

$$L(\, u_{-2}(t)\,) = \frac{1}{s^2}$$

La respuesta rampa de un sistema se obtiene al excitar a un sistema con la función rampa cuando el estado inicial es nulo. Gráficamente:

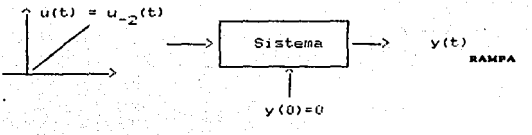


fig. 13

II.3.1 Respuesta rampa de un sistema de primer orden.

Sea la siguiente función de transferencia de un sistema de primer orden:

$$G(S) = \frac{1}{Ts + 1}$$

Como la transformada de Laplace de la función rampa es $1/s^2$ se tiene:

$$y(s) = \frac{1}{s^2} - \frac{T}{s} + \frac{T}{s + \frac{1}{T}}$$

La transformada inversa de Laplace para esta última ecuación es:

$$L^{-1}(y(s)) = y(t) = t - T + Te^{-t/T} \quad \text{para } t \geq 0 \quad -- \quad (11)$$

II.3.2 Respuesta rampa de un sistema de segundo orden.

Considerando nuevamente la función de transferencia de un sistema de 2o. orden

$$G(s) = \frac{\omega n^2}{s^2 + 2\zeta\omega n + \omega n^2}$$

Dado que la transformada de Laplace de la función rampa es $1/s^2$, se tiene:

$$y(s) = \frac{\omega n^2}{s^2 + 2\zeta\omega n + \omega n^2} \frac{1}{s^2}$$

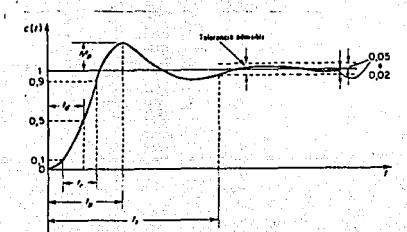
El tipo de respuesta del sistema dependerá de las raíces de la ecuación característica.

III. PARAMETROS DE DISEÑO

En el análisis y diseño de sistemas dinámicos de segundo orden, frecuentemente se especifican las características de funcionamiento del sistema a partir de la respuesta transitoria que el sistema presenta cuando se somete a una entrada escalón unitario. A estas caracteristicas se les denomina parámetros de diseño. Estos son:

-Tiempo de retardo, t_d
-Tiempo de levantamiento, t_r
-Tiempo de asentamiento, t_s
-Tiempo de sobrepaso, t_p
-Sobrepaso, M_p

En la siguiente gráfica se muestran los parámetros antes mencionados:



Los parámetros de diseño son medidas estándar que describen la forma de la curva de la respuesta transitoria de un sistema de segundo orden a una entrada escalón. A partir de ellos podemos conocer entre otros puntos la rapidez con la que el sistema responde y el valor máximo que la curva de respuesta alcanza.

III.1 Tiempo de retardo, t_d: Es el tiempo en el que la respuesta

transitoria escalón alcanza por primera vez el cincuenta por ciento de su valor final.

III.2 Tiempo de levantamiento, t_l: Es el tiempo requerido para que la respuesta escalón alcance desde el 10% al 90% de su valor final.

Generalmente es difícil expresar en forma analítica el tiempo de levantamiento para un sistema. Para obtenerlo en la mayoría de los casos se recurre a la solución de ecuaciones a través de métodos numéricos o se determina su valor gráficamente. Por conveniencia, el tiempo de levantamiento que muchas veces se toma es el tiempo que transcurre desde que la respuesta escalón toma el valor de 0 hasta que alcanza por primera vez el 100% de su valor final.

III.3 Tiempo de asentamiento, t_s: Es el tiempo mínimo necesario para que la respuesta escalón del sistema alcance y se mantenga dentro del rango definido por el valor final $\pm$ el 5%.

III.4 Tiempo de sobrepaso, t_p: Es el tiempo necesario para que la respuesta escalón alcance su valor máximo.

III.5 Sobrepaso, M_p: Es la máxima desviación que alcanza la respuesta escalón con respecto a su valor final. Generalmente se expresa como un porcentaje que queda definido como:

$$\text{Máximo sobreimpulso porcentual} = \frac{c(t_p) - c(\infty)}{c(\infty)} \times 100\% \quad (1)$$

Las especificaciones antes mencionadas no necesariamente corresponden a un caso determinado.

(12) Ogata, Katsuhiko, Ingeniería de control moderna, p. 252

Se presentarán a continuación expresiones generales para calcular los parámetros de diseño.

a) Caso subamortiguado ($0 < \zeta < 1$)

En la ecuación (8) se obtuvo que la respuesta escalón para un sistema de segundo orden con $0 < \zeta < 1$, es:

$$y(t) = 1 - e^{-\zeta\omega_n t}\left(\cos \omega_d t + \frac{\zeta}{\sqrt{1-\zeta^2}} \operatorname{sen} \omega_d t\right)$$

En este caso es posible obtener expresiones generales para determinar los parámetros de diseño.

Para el tiempo de levantamiento se procederá por conveniencia a obtener la expresión analítica del tiempo que transcurre para que la respuesta escalón alcance por primera vez el 100% de su valor final. El valor final para la respuesta escalón unitario, es igual a uno, por lo que retomando la ecuación (8) e igualándola a uno, se tiene:

$$1 = 1 - e^{-\zeta\omega_n t_r}\left(\cos \omega_d t_r + \frac{\zeta}{\sqrt{1-\zeta^2}} \operatorname{sen} \omega_d t_r\right)$$

$$\cos \omega_d t_r + \frac{\zeta}{\sqrt{1-\zeta^2}} \operatorname{sen} \omega_d t_r = 0$$

$$\frac{\operatorname{sen} \omega_d t_r}{\cos \omega_d t_r} = -\frac{\sqrt{1-\zeta^2}}{\zeta}$$

$$\tan \omega_d t_r = -\frac{\sqrt{1-\zeta^2}}{\zeta}$$

$$\boxed{t_r = \frac{1}{\omega_d} \tan^{-1}\left[-\frac{\sqrt{1-\zeta^2}}{\zeta}\right]}$$

Para obtener el tiempo de asentamiento, t_s, se emplean las ecuaciones de las curvas envolventes de la respuesta transitoria a

una entrada escalón unitario. Las expresiones analíticas de estas envolventes son:

$$1 \pm (e^{-\zeta\omega_n t} / \sqrt{1-\zeta^2})$$

La respuesta transitoria una entrada escalón siempre se mantiene dentro de estas curvas, las cuales tienen una constante de tiempo $T = 1/\zeta\omega_n$. Si se considera el criterio de que el tiempo de establecimiento es el tiempo mínimo a partir de que la respuesta tranzitoria a una entrada escalón presenta variaciones menores al 5% de su valor final, entonces el tiempo de asentamiento será aproximadamente tres veces el valor de la constante de tiempo. Esto es:

$$t_s = 3T = \frac{3}{\zeta\omega_n}$$

Una alternativa para obtener el tiempo de retardo t_d, es resolver la ecuación:

$$0.5 - e^{-\zeta\omega_n t_d}(\cos \omega_n t_d + \frac{\zeta}{\sqrt{1-\zeta^2}} \operatorname{sen} \omega_n t_d) = 0$$

utilizando métodos numéricos.

Para el tiempo de sobrepaso se procederá a derivar la ecuación (8) con respecto al tiempo y se igualará a cero:

$$\frac{dy}{dt} = \frac{\omega_n}{\sqrt{1-\zeta^2}} e^{-\omega_n t} \operatorname{sen} \omega_d t$$

Igualando a cero se tiene:

$$\frac{dy}{dt} = \frac{\omega_n}{\sqrt{1-\zeta^2}} e^{-\omega_n t_p} \operatorname{sen} \omega_d t_p = 0$$

De donde:

$$\text{sen } \omega_d t_p = 0$$

$$\omega_d t_p = N\pi \; ; \qquad N = 0, 1, 2, \ldots$$

Como el tiempo de sobreimpulso corresponde al primer pico de la respuesta escalón, $\omega_d t_p = \pi$. De donde se concluye:

$$t_p = \frac{\pi}{\omega_d}$$

El máximo sobreimpulso M_p se produce en $t = t_p$. De la ecuación (8) se tiene:

$$y(t_p) - 1 = M_p = -e^{-\zeta\omega_n(\pi/\omega_d)}\left(\cos\pi + \frac{\zeta}{\sqrt{1-\zeta^2}}\,\text{sen}\,\pi\right)$$

$$M_p = e^{-(\zeta/\sqrt{1-\zeta^2})\pi}$$

De lo anterior se puede concluir que para el caso de sistemas de segundo orden con entrada escalón, para obtener una respuesta rápida, ω_n debe ser grande y para hacer pequeño el tiempo de establecimiento y al mismo tiempo limitar el máximo sobreimpulso, ζ no debe ser muy pequeña.

b) Casos críticamente amortiguado ($\zeta = 1$) y sobreamortiguado ($\zeta > 1$)

En ambos casos, los únicos parámetros que se pueden calcular son el tiempo de retardo y el tiempo de levantamiento, ya que la respuesta escalón unitario en ambos casos no presenta oscilaciones. No existe sobrepaso y no se presentan variaciones entre el 95% y 105% del valor final.

Para obtener los valores del tiempo de levantamiento y tiempo de retardo, se puede recurrir a la solución gráfica o a la solución numérica de ecuaciones. Sólo en algunos casos se podrán determinar los valores de estos parametros en forma analítica.

c) Caso no amortiguado ($\zeta = 0$)

La respuesta a escalón tiene la forma dada en la ecuación (9):

$$y(t) = 1 - \cos \omega_n t$$

En este caso no es posible evaluar los parámetros de diseño, ya que no se puede determinar un valor final al que la respuesta escalón tienda. De la definición de los parámetros de diseño se puede observar como todos se obtienen tomando como base el valor final que adquiere la respuesta escalón.

IV. METODO DE DESARROLLO EN FRACCIONES PARCIALES

La forma general de una función de transferencia es la siguiente:

$$G(s) = \frac{Y(s)}{X(s)}$$

Donde Y(s) y X(s) son polinomios en S tales que el grado del polinomio Y(s) es menor o igual al grado del polinomio de X(S).

Un método que permite obtener la transformada inversa de Laplace de la función G(s) es el de desarrollo en fracciones parciales. A través de este método es posible descomponer la función G(s) en la suma algebraica de funciones de s simples, para las cuales se conocen las transformadas inversas de Laplace. Esto es, G(s) se descompone en:

$$G(s) = G_1(s) + G_2(s) + \ldots + G_n(s)$$

Como se conocen las transformadas inversas de $G_1(s)$, $G_2(s)$,

..., $G_n(s)$

$$L^{-1}(G(s)) = L^{-1}(G_1(s)) + L^{-1}(G_2(s)) + \ldots + L^{-1}(G_n(s))$$
$$= g_1(t) + g_2(t) + \ldots + g_n(t)$$

Para poder aplicar el método de dazarrollo en fracciones parciales, es necesario conocer previamente las raíces del polinomio X(s), de tal manera que a G(s) la podamos expresar como:

$$G(s) = \frac{y(s)}{(s+p_1)(s+p_2)\ldots(s+p_n)}$$

En donde p_1, p_2, ... , p_n pueden ser reales y diferentes, reales múltiples o complejas. La naturaleza de las raíces definen procedimientos distintos para el desarrollo en fracciones parciales de G(s). A continuación se tratará cada uno de estos casos:

a) Raíces reales diferentes

Para este caso, la expansión en fracciones parciales de G(s) es de la forma:

$$G(s) = \frac{A_1}{(s+p_1)} + \frac{A_2}{(s+p_2)} + \ldots + \frac{A_n}{(s+p_n)}$$

Donde A_1, A_2, ... A_n son constantes. Su valor se puede obtener a partir de la siguiente expresión:

$$A_k = G(s)(s+p_k)\,|_{s=-p_k}$$

b) Raíces reales múltiples

El polinomio X(s) en este caso presenta n raíces reales iguales. G(s) puede expresarse como

$$G(s) = \frac{Y(s)}{(s+p)^n}$$

La expansión en fracciones parciales tomará la forma:

$$G(s) = \frac{A_1}{(s+p)} + \frac{A_2}{(s+p)^2} + \ldots + \frac{A_n}{(s+p)^n}$$

Donde el valor de las constantes está dado por:

$$A_i = \frac{1}{(n-I)!} \frac{d^{i-1}}{ds^{i-1}} G(s)(s+p)^n \Big|_{s=-p}$$

3) Raíces complejas

El polinomio X(s) de grado n puede ser escrito como:

$$X(s) = (s^2 + a_1 s + b_1)(s^2 + a_2 s + b_2)\ldots(s^n + a_{n/2} s + b_{n/2})$$

Donde las raíces de cada término cuadrático son complejas conjugadas.

La expansión en fracciones parciales de G(s) tiene la forma:

$$G(s) = \frac{A_1 s + B_1}{(s^2 + a_1 s + b_1)} + \frac{A_2 s + B_2}{(s^2 + a_2 s + b_2)} + \ldots + \frac{A_{n/2} s + B_{n/2}}{(s^2 + a_{n/2} s + b_{n/2})}$$

$$= \frac{A_1 s + B_1}{(s + \alpha_1)^2 + \beta_1^2} + \frac{A_2 s + B_2}{(s + \alpha_2)^2 + \beta_2^2} + \ldots + \frac{A_{n/2} s + B_{n/2}}{(s + \alpha_{n/2})^2 + \beta^2_{n/2}}$$

Donde $\alpha_i = a_i/2$ $\quad \beta_i = \sqrt{b_i - \alpha_i^2}$

La expresión general para determinar los valores de A_i y de

Di es:

$$(A_i s + B_i)\Big|_{s = -\alpha_i + j\beta_i} = a(s)\left[(s+\alpha_i)^2+\beta_i^2 \right] \Big|_{s=-\alpha_i+j\beta}$$

Para i = 1, 2, ... , n/2

CAPITULO II DEFINICION DE REQUERIMIENTOS

I. DEFINICION DEL PROBLEMA

I.1 OBJETIVO

Elaborar un programa de computadora que sirva de apoyo en el analisis, simulación y diseño de sistemas dinamicos en el dominio del tiempo, utilizando algun lenguaje de programación orientado a objetos. El programa debera abarcar sistemas lineales continuos y discretos y debera ser capaz de calcular en forma analitica y numerica los diferentes tipos de respuesta del sistema a señales de entrada dadas por el usuario, a partir de la funcion de transferencia del sistema. Los datos se almacenaran en archivos para que posteriormente puedan ser graficados.

I.2 JUSTIFICACION

La idea de crear el programa para calculo de la respuesta en el tiempo de sistemas dinamicos, se basa en el proyecto inicial de crear un paquete de control que cuente con una interface amigable con el usuario, de tal manera de que el manejo del paquete sea simple y de que no requiera que el usuario final tenga que manejar un gran numero de comandos, como es el caso de algunos de los paquetes de control existentes.

La extension de este proyecto requirio dividir en varios subsistemas al programa total. Uno de estos se enfocaria al desarrollo de la interface amigable con el usuario, otro, a la graficacion de los datos que resulten al ejecutar el programa. Otros subsistemas desarrollarian soluciones a problemas especificos de control. El programa que mostrare en la siguiente documentación formara parte de este proyecto. Se ha pensado crear un conjunto de librerias generales disponibles para futuros desarrollos.

I.3 ANALISIS

El problema se puede formular de la siguiemte manera:

Dada la función de transferencia de un sistema

$$G(s) = \frac{Y(s)}{X(s)}$$ Para el caso continuo, o

$$G(z) = \frac{Y(z)}{X(z)}$$ Para el caso discreto

y la señal de entrada x(t), junto con las condiciones iniciales del sistema, se desean obtener los diferentes tipos de respuesta que el sistema presenta. Esto es, respuesta forzada, libre, total, transitoria y permanente.

X(t) puede ser una señal de prueba típica, tal como impulso, escalón o función rampa, o bien, puede ser algún otro tipo de señal introducida por el usuario.

II. METODO

Básicamente se encontraron dos alternativas para resolver este problema. La primera consiste en realizar el desarrollo en fracciones parciales de G(s) o G(z) según sea el caso, para así poder obtener fácilmente la respuesta en el tiempo y(t).

La segunda alternativa consiste en obtener la respuesta y(t) utilizando exclusivamente métodos numéricos. Se eligió finalmente la primera opción, ya que a través del desarrollo en fracciones parciales es posible obtener tanto la respuesta en forma numérica como su expresión analítica, mientras que con la segunda opción sólo es posible calcular la respuesta en el tiempo en forma numérica.

Respecto al lenguaje de programación a utilizar se decidió trabajar con algún lenguaje que permitiera el diseño orientado a objetos. Se optó por esta desición debido a que este tipo de diseño proporciona herramientas que permiten atacar las complejidades de programación que pueden surgir en diversas aplicaciones. A continuación se presentará una breve exposición de las características principales de la programación orientada a

objetos, así como de sus principales ventajas frente a la programación tradicional.

11.1 Ventajas

La programación orientada a objetos permite diseños más eficientes, mas estructurados, modulares y fáciles de mantener. Permite disminuir el tiempo de desarrollo de una aplicación. "Evita el tener que inventar todo desde el principio"(13). Esto es, un programador que utilize el diseño orientado a objetos no tiene que enfrentarse con la tarea de implementar programas que den solución a problemas que alguien ya les ha dado solución con programación orientada a objetos, sino reutilizar estos programas. Esta es una de las principales ventajas que presenta este tipo de diseño, la reutilización de código, por medio de la cual es posible utilizar el código compilado de un programa y sin necesidad de realizar modificaciones sobre el programa fuente ni de recompilarlo, poder utilizarlo adaptándolo en otro programa.

En una de sus conferencias, Luis Joyanes Aguilar señaló: "El diseño orientado a objetos implica modificar nuestro punto de vista sobre la programación. Implica mirar hacia el fin y no en los medios para alcanzarlo. Tal vez, el elemento clave en la revolución industrial del software sea la creación de un mercado de piezas estándar de software de modo que se puedan ensamblar entre sí, para diseñar soluciones de complejidad superior".(14)

El diseño orientado a objetos es un paradigma de programación que se puede utilizar para hacer mas eficiente el desarrollo de programas. Básicamente consiste en modelar las características y conducta de los elementos de un entorno en estructuras conocidas como objetos. Las características del objeto se modelan a través de un conjunto de datos, mientraz que la conducta a través de un conjunto de métodos. Los métodos y datos estarán unidos bajo un mismo nombre.

(13) Joyanes Aguilar,Luis, en: conferencia sobre programación orientado a objetos vs. programación tradicional 1991, Postgrado de ingeniería, México, D. F.

(14) Idem

Las características basicas que un lenguaje debe presentar para ser considerado como orientado a objetos son:

-Encapsulamiento: Término formal que describe el acto de agrupar datos realcionados entre sí y las operaciones que actúan sobre estos datos en estructuras conocidas como objetos. Esta propiedad reduce la interdependencia entre los componentes de software.

-Polimorfismo: Es la propiedad por la cual dos o más objetos responden al mismo llamado en forma totalmente diferente.

Con los lenguajes orientados a los procedimientos, cuando se tienen diferentes tipos de datos que llaman a un procedimiento, se debe tener una estructura de desición tal como el CASE que identifique el procedimiento a llamar, dependiendo del tipo del dato. Con programación orientada a objetos ésto no es necesario ya que con una misma llamada se tiene distinta respuesta, dependiendo del objeto receptor.

-Herencia: Propiedad por la cual se puede crear a partir de un objeto, una jerarquía de objetos, de tal manera que cada descendiente hereda todas las características y conducta de su ancestro y puede además añadir caradcterísticas y conducta propias.(15)

La complejidad de algunos problemas se puede manejar de manera más efectiva con programación orientada a objetos desde el momento en el que la creación de objetos permite clasificar la información.

Hasta el momento se han descrito algunas de las principales ventajas de la programación orientada a objetos, sin embargo, no se pueden pasar por alto algunas desventajas que este método de diseño presenta y que a continuación se señalarán:

(15) Geoffrey A Pascoe, Elements of object oriented programming en: Revista byte (agosto, 1986) p.p. 139-144

-El tiempo de ejecucion de un programa puede ser mayor al tiempo que requeriria el mismo programa escrito con algun lenguaje orientado al procedimiento.

-El programador necesita conocer gran cantidad de libreria.

Para mayor información sobre el tema se recomienda consultar la referencia (15).

III. APLICACION DEL METODO

III.1 DEFINICION DE REQUERIMIENTOS

Como siguiente punto se procederá a describir un conjunto de caracteristicas que el programa para calculo de la respuesta en el tiempo de sistemas dinamicos deberá cumplir.

III.1.1 Especificacion de requerimientos

III.1.1.1 Activacion: El programa podra activarse desde el menú principal. La primera opcion a elegir es trabajar con un sistema continuo o con uno discreto. Una vez que se elija alguna de estas dos opciones se recibirá como entrada los datos referentes a los polinomios del numerador y del denominador de la funcion de transferencia, esto es, se considerará que tanto el numerador como el denominador están formados por productos de polinomios, de tal forma que

$$G(s) = \frac{Y1(s)\ Y2(s)\ \ldots\ Yn(s)}{X1(s)\ X2(s)\ \ldots\ Xm(s)}$$

Para cada polinomio, los datos de entrada que el programa requerirá del usuario son: su grado y sus coeficientes. Una vez introducida la funcion de transferencia se pedirá al usuario que introduzca la entrada al sistema, para lo cual se desplegara un menu con las siguientes opciones:

1. Respuesta impulso
2. Respuesta escalon
3. Respuesta rampa
4. Otro tipo de respuesta

En un desarrollo posterior se pretende que la opcion 4 despliegue un editor que permita al usuario teclear directamente la entrada al sistema, para lo cual se creara un pequeño interprete. Por el momento la entrada se introducira a traves de un menú que presente las siguientes opciones:

1. A(t**n)
2. A exp(Bt)
3. A sen kt
4. A cos kt
5. A exp(Bt) * sen kt
6. A exp(Bt) * cos kt

para el caso continuo, o:

1. B exp(ak)
2. B (a**k)
3. B cos wkt
4. B sen wkt
5. B (r**k) sen wkt
6. B (r**k) cos wkt

para el caso discreto.

La entrada al sistema podra estar formada por un solo termino del menú anterior o por la suma de varios de ellos. Para esto, una vez que el usuario halla introducido el valor de las variables asociadas con cada termino, se desplegara el siguiente menú:

1. +
2. Termina entrada

La opción 1 activara nuevamente al segundo menú. Una vez que se termine de dar la entrada se requerirá recibir las condiciones iniciales del sistema.

III.1.1.2 Almacenamiento de datos de entrada.

Los datos de entrada al programa podrán obtenerse de dos fuentes: directamente del teclado o a traves de un archivo de texto que previamente contenga los datos de entrada requeridos por el programa.

III.1.1.3 Almacenamiento de datos de salida.

Para el intercambio de información con otros subsistemas, se pide como requisito que la información numerica quede almacenada en archivos de texto con el siguiente formato:

```
Encabezado          ( Que puede contener una breve descripcion
                      de la información del archivo)
INICIO:
Información tabulada
```

III.1.1.4 Validación de datos.

Para los datos de entrada de la función de transferencia, el programa deberá validar que no se den los siguientes casos:

1. Que el grado del polinomio del numerador sea mayor al grado del polinomio del denominador.

2. Que G(S) <> 0/0

3. Que el numero de polinomios que forman al numerador y al denominador sean mayores que cero

4. Que el tamaño de la función de transferencia no rebase las posibilidades del programa.

Si no se cumplen las condiciones antes citadas, el programa enviara los siguientes mensajes de error respectivamente:

1. La funcion de transferencia no corresponde a un sistema fisico realizable.

2. G(s)= 0/0. Existe una indeterminacion.

3. El numero de polinomios debe ser mayor a cero

4. El sistema es demasiado grande

I. ARQUITECTURA

I.1 Definicion de los conjuntos de informacion:

Informacion de entrada. Según se indique, el programa podra tomar la informacion desde el teclado o desde algún archivo de texto. Los datos de entrada que se requiren son:

El tipo de sistema (continuo o discreto)
Para la funcion de transferencia:
Número de polinomios en el numerador
Numero de polinomios en el denominador
Para cada polinomio:
Grado
Coeficientes
Señal de entrada al sistema
Condiciones iniciales
El valor del tiempo hasta el cual se calculara la respuesta numérica.

Información de salida:

Expresion analitica para cada uno de los siguientes tipos de respuesta: libre, forzada, total, permanente y transitoia.

Cinco archivos de texto con los siguientes nombres:

resp_lib.rtc
resp_for.rtc
resp_tot.rtc
resp_per.rtc
resp_tra.rtc

para el caso continuo. Para el caso discreto se generarán cinco archivos con los mismos nombres pero con extension rtd, en donde la extension rtc indica que se trata de la respuesta en el tiempo continua, mientras que la extension rtd indica que se trata de la respuesta en el tiempo discreta.

Cada uno de estos archivos almacenará la respuesta numérica libre, forzada, total, permanente y transitoria del sistema, respectivamente.

I.2 Identificacion de programas a desarrollar.
Básicamente se identifican tres programas : Un programa para el cálculo de la respuesta en el tiempo continuo, otro para la respuesta en el tiempo discreto y el programa principal.

II Selección del lenguaje de programación

Se seleccionó trabajar con Turbo Pascal 5.5 porque éste presenta las siguientes ventajas:

-Da el poder y la eficiencia de la programación orientada a objetos.

-Ofrece alta velocidad.

-Es didáctico: Se pretende que parte de los usuarios del programa sean estudiates.

-Es un lenguaje híbrido: Tiene las propiedades de la programación tradicional y además le suma las propiedades de la programación orientada a objetos.

-Es un lenguaje poderoso.

-Es muy flexible.

-Es un lenguaje estructurado

-Fácil de mantener.

III Abstracción de datos

En primer termino, el elemento que se quiere modelar para la obtencion de los diferentes tipos de respuesta de un sistema es la funcion de transferencia. Este problema puede abarcar dos casos, el continuo y el discreto.

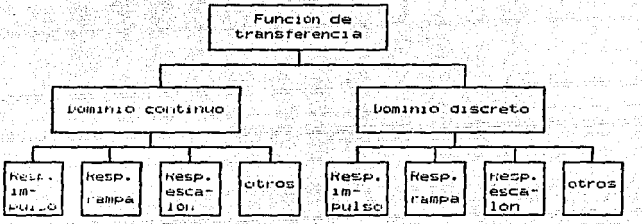


Fig. 1

Los diferentes tipos de respuesta podrán obtenerse de la funcion de transferencia de la siguiente manera:

Para el caso continuo:

$$Y_{IMPULSO}(S) = G(S) \qquad ---(1)$$

$$Y_{ESCALON}(S) = G(S)\ \frac{1}{S} \qquad ---(2)$$

$$Y_{RAMPA}(S) = G(S)\ \frac{1}{S^2} \qquad ---(3)$$

$$Y_{OTROS}(S) = G(S)\ L\ (X(t)) \qquad ---(4)$$

Para el caso discreto:

$$Y_{IMPULSO}(Z) = G(Z) \qquad ---(5)$$

$$Y_{ESCALON}(Z) = G(Z)\ \frac{Z}{Z-1} \qquad ---(6)$$

$$Y_{RAMPA}(z) = G(z) \frac{z}{(z-1)^2} \quad \text{---(7)}$$

$$Y_{OTROS}(z) = G(z) Z\{x(kt)\} \quad \text{---(8)}$$

A partir de la figura 1 se pueden modelar los conjuntos de objetos a utilizar:

Para el caso continuo:

La representación interna de la función de transferencia se llevará a cabo a través de un objeto al que se denominara transfer.

La representacion de la respuesta impulso esta implícitamente dada en el objeto transfer, por la igualdad mostrada en (1).

Los objetos descendientes de transfer serán:

-Sal_esc : En donde se representará la respuesta escalón del sistema.

-Sal_ram: En donde se representara la respuesta rampa

-Sal_otros: En este objeto se representará la respuesta del sistema a cualquier entrada introducida por el usuario.

Para el caso discreto:

El objeto sal_discreta representará a la función de transferencia y los siguientes objetos serán sus descendientes:

-Disc_escalón: Para representar la respuesta escalón.

-Disc_rampa: Para la representacion de la respuesta rampa.

-Disc_otros: Para representar cualquier otro tipo de respuesta.

A continuación se describirá para cada objeto el conjunto de métodos y datos a utilizar.

Para el caso continuo:

III.1 Objeto transfer

Este objeto estará formado por los siguientes datos:

-Num_poln: En donde se representará el número de polinomios en el numerador; variable de tipo entera.

-Num_pold: Representa el número de polinomios en el denominador; variable de tipo entera.

-Grado_n, grado_d: En estas variables se almacenarán los grados de los polinomios del numerador y denomiador respectivamente; arreglos de números enteros.

-Coef_n, coef_d: En estas variables se almacenarán los coeficientes de los polinomios del numerador y del denominador a partir del coeficiente de mayor orden; arreglos de números enteros.

-Cte: Variable de tipo real cuyo valor resulta al normalizar los polinomios del denominador de la función de transferencia.

Los métodos que forman parte del objeto transfer son:

1. Módulo inicia_arc: Este modulo prepara los archivos a utilizar durante el desarrollo del programa.

2. Módulo cierra: Cierra los archivos utilizados.

3. Módulo captura: En este módulo se realizará la captura de la función de transferencia.

Parametros del modulo:

indice: variable de tipo entero cuyo valor final será igual al numero total de coeficientes del denominador que el usuario introduzca

indice2: Variable entera cuyo valor será igual al numero de coeficientes introducidos para el numerador.

Pseudocodigo:

```
COMIENZA
   Lee el número de polinomios en el numerador
   Lee el número de polinomios en el denominador
   PARA i = 1 HASTA numero de polinomios en el numerador HAZ
   COMIENZA
      LEE grado del polinomio
      LEE coeficientes del polinomio
   TERMINA
   PARA i = 1 HASTA número de polinomios en el denominador HAZ
   COMIENZA
      LEE grado del polinomio
      LEE coeficientes del polinomio
   TERMINA
TERMINA
```

4. Modulo desp: Despliega simbólicamente un producto de polinomios.

Parametros del módulo:

Grad: Arreglo de enteros que contiene los grados de los polinomios que forman al producto.

Coef: Arreglo de numeros reales que contiene los coeficientes de los polinomios.

Num_pol: Numero de polinomios que forman al producto; variable entera.

5. Módulo imp_func. Despliega la funcion de transferencia

Pseudocodigo:

```
COMIENZA
   ESCRIBE ('G(S)=')
   DESPLIEGA los polinomios del numerador
   ESCRIBE('/')
   DESPLIEGA los polinomios del denominador
TERMINA
```

6. Módulo multiplica: Multiplica dos polinomios.

Parametros del módulo:

V1: Arreglo de numeros reales que contiene los coeficientes del polinomio 1.

V2: Arreglo que contiene los coeficientes del polinomio 2.

Pseudocodigo:

```
COMIENZA
   INICIALIZA resultado
   PARA i = 0 HASTA grado del polinomio 1 HAZ
      PARA j = 0 HASTA grado del polinomio 2 HAZ
          resultado[i+j] = resultado[i+j] + polinomio1[i]*
                                polinomio2[j]
      TERMINA (Para)
    TERMINA (para:
TERMINA
```

7. Módulo evalua: Dado un polinomio f(x) y un valor de x, el módulo encuentra el valor de y= f(x).

Parametros del modulo:

F: Resultado de la evaluación de f(x); numero real.

X: Variable independiente; número real.

Func: Polinomio f(x); arreglo de números reales.

Grado: grado de f(x); número entero.

Pseudocodigo:

```
COMIENZA
    CALCULA f(x)
    ASIGNA f(x) a F
TERMINA
```

8. Modulo res_ec: Resuelve un sistema de ecuaciones por el metodo de Gauss-Jordan.

Parametros del modulo:

Mat: Matriz de números reales que contiene un sistema de ecuaciones.

N: Numero de ecuaciones del sistema; variable entera.

Pseudocodigo:

```
COMIENZA
   ALMACENA la matriz en la variable mat
   PARA i = 1 HASTA numero de renglones de mat HAZ
      pivote = mat(i,i)
      PARA j = 1 HASTA número de columnas de mat HAZ
         mat(i,j) = mat(i,j)/pivote
      TERMINA
```

```
    PARA k = 1 HASTA n HAZ
      SI k <> i ENTONCES cero = mat(k,i)
      PARA j = 1 HASTA número de columnas de mat HAZ
        mat(k,j) = mat(k,j) - cero * mat(i,j)
      TERMINA
    TERMINA
  TERMINA
```

9. Módulo reacomoda: Sirve de apoyo al modulo res_ec. Cuando la diagonal principal de la matriz que contiene un sistema de ecuaciones contiene un valor igual a cero, realiza un reacomodo de los renglones de la matriz, de tal manera de que se elimine el cero de la diagonal principal.

Parametro del modulo:

Mat: Matriz de números reales que contiene un sistema de ecuaciones.

Pseudocodigo

```
  COMIENZA
    MIENTRAS existan ceros en la diagonal principal mat HAZ
      REACOMODA los renglones de la matriz
    TERMINA
  TERMINA
```

10. Módulo intercambia: Dados los valores de i y j, este modulo intercambia el renglon i por el j en la matriz especificada.

Parámetros del módulo:

I, J: Apuntan a los renglones que se van a intrercambiar; numeros enteros.

N: Numero de renglones de la matriz; variable entera.

Mat: Matriz de números reales que contiene un sistema de ecuaciones.

Pseudocódigo:

```
COMIENZA
   ASIGNA a una variable temporal el renglon j de la matriz
   ASIGNA al renglon j de la matriz los valores del renglon i
   ASIGNA al renglón i los valores de la variable temporal
TERMINA
```

11. Módulo dim_mat: Encuentra el grado del polinomio del numerador de la función de transferencia.

Parámetro del módulo:

N: Número de renglones que posteriormente tendra la matriz MAT; variable entera.

Pseudocódigo:

```
COMIENZA
   N=0
   PARA cada polinomio del denominador de G(S) HAZ
      n= n+grado del polinomio del denominador de G(S)
   TERMINA
TERMINA
```

12. Módulo mapea: Regresa un apuntador en donde se encuentran los coeficientes del polinomio n-ésimo.

Parametros del modulo:

K: apuntador al arreglo coef_o; número entero.

Indice: Número asignado al polinomio cuyos coeficientes se quieren localizar; variable entera.

Pseudocódigo:

```
COMIENZA
    PARA i = 1 HASTA numero de polinomios en el denomiador HAZ
        k = grado del polinomio + k + 1
    TERMINA
    k = k + 1
TERMINA
```

13. Modulo raices_rep: Cuando un polo de la función de transferencia se repite mas de una vez, se llama a este procedimiento para determinar el número de veces que se repite el polo.

Parametros del modulo:

Repetidas: Numero de veces que se repite el polo apuntado por indice; variable entera.

Indice: Apuntador al arreglo raices; número entero.

Pseudocódigo:

```
COMIENZA
    repetidas = 0;
    registro = raices[indice]
    MIENTRAS raices[indice] = registro HAZ
        repetidas = repetidas + 1
TERMINA
```

14. Modulo normaliza: Normaliza los polinomios del denominador de la funcion de transferencia.

Pseudocodigo:

```
COMIENZA
    PARA cada polinomio del denominador HAZ
        coef = coeficiente del termino de mayor orden
```

```
        PARA cada coeficiente del polinomio PA.
            divide coeficiente entre coef
        TERMINA
        cte = cte*coef
    TERMINA
TERMINA
```

15. Modulo res_ec_grad2: Aplica la fórmula general para la solución de ecuaciones de segundo grado.

Parametros del modulo:

A: Coeficiente de X^2; número real.

B: Coeficiente de X; número real.

C: Coeficiente de X^0; número real.

Pseudocodigo:

```
COMIENZA
    CALCULA x = (-b ± √(b*b-4*a*c)) / (2*a)
    SI x es compleja ENTONCES
    COMIENZA
      raices[ap_raiz].R = parte real de x
      raices[ap_raiz].i = parte imaginaria de x
      ap_raiz = ap_raiz + 1
    TERMINA
    SI x no es compleja ENTONCES
    COMIENZA
       raices[ap_raiz].r = primer valor de x
       raices[ap_raiz+1] = segundo valor de x
       ap_raiz = ap_raiz + 2
    TERMINA
TERMINA
```

16. Modulo lin: Aplica el método numérico de LIN. Este método

se elaboro para obtener las posibles raices de un polinomio, haciendo uso de la ventaja de factorizar dicho polinomio en otro de forma cuadratica. Es decir:

$$P(x) = a_0x^n + a_1x^{n-1} + a_2x^{n-2} + \ldots + a_{n-1}x + a_n = 0$$

$$= (x^2+px+q)\,(b_0x^{n-2} + b_1x^{n-3} + \ldots + b_{n-2}) + Rx + s$$

Una vez que ha calculado los valores de P y de q llama al modulo resp_ec_grad2(1,p,q)

Parametro del modulo:

Func: Arreglo de números reales que contiene los coeficientes del polinomio.

Pseudocodigo:

```
COMIENZA
   HAZ
      PARA k = 0 HASTA n-2 HAZ
         CALCULA b_k = a_k - p*b_{k-1} - q*b_{k-2}
      R = a_{n-1} - p*b_{n-2} - q*b_{n-3}
      s = a_n - q*b_{n-2}
      deltaq = R/b_{n-2}
      deltap = s/b_{n-2}
      p = p+deltap
      q = q+deltaq
   HASTA (deltap<tolerancia) y (deltaq<tolerancia)
TERMINA
```

17. Módulo biseccion: Aplica el método de biseccion para obtener las raices de un polinomio.

Parametros del modulo:

grado: grado del polinomio; numero entero.

11: Apuntador al arreglo coef_d, que contiene los coeficientes del polinomio; número entero.

Pseudocódigo:

```
COMIENZA
   REPITE
      ε = valor suficientemente pequeño
      ENCUENTRA a y b tales que f(a)*f(b)<0
      x1= (a+b)/2
      CALCULA f(x1)
      Si no(f(x1) = 0 ± ε ) ENTONCES
      COMIENZA
         SI el signo de f(x1) = signo de f(a) entonces
            a = x1
         SI NO b = x1
      TERMINA
   HASTA que f(x1) = a ± ε
TERMINA
```

18. Módulo divide: Realiza la división del numerador por el denominador de la función de transferencia.

Parametros del modulo:

Numerador: Residuo de la division; arreglo de números reales.

Grado: Grado del residuo; número entero.

Pseudocódigo:

```
COMIENZA
   MULTIPLICA polinomios del numerador de X(S)
   ASIGNA resultado a num
   MULTIPLICA polinomios del denominador de X(s)
   ASIGNA resultado a den
   DIVIDE num/den
   ASIGNA residuo a num
```

```
    ASIGNA a t el resultado de la division
  TERMINA
```

19. Modulo encuentra_raiz: Activa los métodos res_ec_grad2, lin y biseccion para encontrar todos los polos de la funcion de transferencia y luego los clasifica en cuatro clases:

Clase 1: Si se trata de una raiz real no repetida
Clase 2: Para raices reales multiples
Clase 3: Para raices complejas que no se repiten
Clase 4: Para raices complejas multiples

Pseudocodigo:

```
COMIENZA
  PARA cada polinomio en el numerador de X(s) HAZ
  COMIENZA
    SI el grado del polinomio = 1 ENTONCES
    COMIENZA
        ALMACENA raiz en raices[ap_raiz]
        ap_raiz = ap_raiz + 1
    TERMINA
    SI el grado del polinomio = 2 ENTONCES
       LLAMA a res_ec_grad2
    SI el grado del polinomio>2 ENTONCES
    COMIENZA
       LLAMA a biseccion
       ALMACENA las raices del polinomio en el arreglo raices
    TERMINA
  TERMINA
  PARA i = 1 HASTA numero de raices HAZ
  COMIENZA
    SI la raiz es real y no se repite ENTONCES
        raices[i].tipo = 1
    SI la raiz es real multiple ENTONCES
       raices[i].tipo = 2
    SI la raiz es compleja y no se repite ENTONCES
       raices[i].tipo = 3
```

```
        SI la raiz es compleja múltiple ENTONCES
            raices[i].tipo = 4
    TERMINA
TERMINA
```

20. Módulo encuentra_ec: Una vez que se ha llamado al módulo encuentra_raiz y se conocen los polos de la función de transferencia, así como su tipo, se tendrán fracciones de la forma: $A/(s-p_1)$, $B/(s-p_2)^n$, $(C+Ds)/(s^2+bs+c)$ y $(E+Fs)/(s^2+bs+c)^n$. El módulo encuentra_ec forma un sistema de ecuaciones que permite encontrar el valor de las constantes A, B, C, D, E y F.

Parámetros del módulo:

Conjunto: Variable auxiliar del tipo: set of 1..30.

I: Apuntador a un elemento del arreglo raices; variable entera.

Num_ra: Número de polos en la función de transferencia; variable entera.

Mat: Matriz de números reales en donde se almacenará el sistema de ecuaciones.

Col: Número de columna de la matriz mat a la que el método encuentra_ec le asignará valores; variable entera.

Pseudocódigo:

```
COMIENZA
    SI raices[i].tipo = 1 ENTONCES
       Divide el denominador de X(s) entre (s-raiz[i].r)
    SI raices[i].tipo = 2 ENTONCES
    COMIENZA
       Asigna a n el número de veces que se ha repetido la raíz
       Divide el denominador de X(s) entre (s-raiz[i].r)^n
    TERMINA
```

```
Si raices[i].tipo = 3 ENTONCES
COMIENZA
   b = -2*raices[i].r
   c = raices[i].r*raices[i].r+raices[i].raices[i].i
   Divide al denominador de x(s) entre s²+bs+c
TERMINA
Asigna a la columna col el resultado de la division
TERMINA
```

21. Modulo trans_inv: Imprime la expresion analitica de la respuesta en el tiempo.

Una vez que se conoce el desarrollo en fracciones parciales, se aplican las siguientes transformadas inversas:

$$L^{-1}\left(\frac{A}{s-B}\right) \longrightarrow Ae^{Bt}$$

$$L^{-1}\left(\frac{A}{(s-B)^n}\right) \longrightarrow \frac{A}{(n-1)!}t^{(n-1)}e^{Bt}$$

$$L^{-1}\left(\frac{A+Bs}{(s+C)^2+d}\right) \longrightarrow \frac{(A-BC)}{\sqrt{d}}e^{-ct}\operatorname{sen}\sqrt{d}\,t + Be^{-ct}\cos\sqrt{d}\,t$$

Parametros del módulo:

Mat: Matriz de números reales que contiene en su última columna los valores de A, B, C, D, E y F descritos en el metodo anterior.

Opción: Valor entero que indica el tipo de respuesta que se desea obtener y puede tomar cualquiera de los siguientes valores:

1: Para la respuesta libre
2: " " forzada
3: " " total
4: " " transitoria
5: " " permanente

Hnade: Variable booleana que tiene el valor de true cuando el grado del numerador de la funcion de transferencia es igual al del denominador.

Pseudocodigo:

```
COMIENZA
   j=1
   n= Numero de renglones de la matriz MAT
   PARA i= 1 HASTA número de raices HAZ
   COMIENZA
      Si raices[i].tipo = 1 ENTONCES
      COMIENZA
         imprime(mat[j,n+1]/cte,'exp(',raices[i].r,")t')
         j:= j+1
      TERMINA
      Si raices[i].tipo = 2 ENTONCES
      COMIENZA
         cont=0
         raices_rep(rep,1)
         MIENTRAS rep>0 HAZ
         COMIENZA
         coef= MAT[j,n+1]/(cte*fact)
         SI coef<>0 ENTONCES
         COMIENZA
            imprime('exp(',raices[i].r,')t')
            si cont>1 ENTONCES imprime('**',cont)
            i= i+1
            rep=rep-1
            cont=cont+1
            fact=fact*cont
            j=j+1
       TERMINA
       Si coef = 0 entonces
       COMIENZA
          raices_rep(rep,1);
          i=i+rep-1
```

```
            j=j+rep
        TERMINA
        SI raices[i].tipo = 3 ENTONCES
        COMIENZA
            coef = mat[j,n+1]/(cte*raices[i].i)
            IMPRIME(coef,'exp(',raices[i].r,'t) ')
            IMPRIME('sen',raices[i].i,'t')
            j=j+1;
            coef=mat[j,n+1]/cte
            IMPRIME('+'coef,'exp(',raices[i].r,'t) ')
            IMPRIME('cos',raices[i].i,'t')
        TERMINA
        SI raices[i].tipo = 4 ENTONCES
        COMIENZA
            comp_rep = 0
            coef2 = mat[j+1,n+1]/cte
            b= -2*raices[i].r
            c = raices[i].r*raices[i].r+raices[i].i*raices[i].i
            c2 = b/2
            c3 = -c2
            d = c-c2*c2
            ra = √ abs(d)
            coef = (mat[j,n+1]-mat[j+1,n+1]*c2)/(ra*cte)
            con1= coef/(2*ra*ra)
            con2= coef/(2*ra)
            con3 = coef2/(2*ra)
            IMPRIME('exp(',c3,'t)(',con1,'sen',ra,'t')
            IMPRIME('+',con2,'tcos',ra,'t+',con3,'tsen',ra,'t)')
        TERMINA
    TERMINA
TERMINA
```

22. Módulo frac_par: Aplica el método de desarrollo en fracciones parciales a una función, para lo cual activa al módulo encuentra_ec y trans_inv.

Parámetros del módulo:

Inestable: Variable booleana que toma el valor de true cuando algun de los metodos numericos aplicados es inestable.

Vec: Arreglo de numeros reales en donde se representa la transformada de Laplace de la entrada al sistema.

Opcion y Anade: Toman los mismos valores que en el método anterior.

23. Módulo valor_num: Calcula el valor numérico de la respuesta en el tiempo.

Se utilizan los mismos parametros que en trans_inv.

Pseudocódigo:

```
COMIENZA
   Evalua numericamente trans_inv
   Almacena resultados en archivos
TERMINA
```

24. Módulo fact: Calcula el factorial de un numero.

Parametro del modulo:

N: Numero entero al cual se determinará su factorial.

Pseudocódigo:

```
COMIENZA
   factorial=1
   MIENTRAS n>1 HAZ
   COMIENZA
      factorial=factorial*n
      n=n-1
   TERMINA
TERMINA
```

3.3.2 Objeto sal_escalon

Este objeto añade el siguiente modulo:

Modulo forma_ec_esc: Multiplica a la funcion de transferencia por 1/s, formando asi la ecuacion (2).

Pseudocodigo:

COMIENZA
añade el polinomio s al denominador
incrementa el numero de polinomios en el denominador
TERMINA

3.3.3 Objeto sal_rampa

Añade el siguiente modulo:

Modulo forma_ec_ram: Multiplica a la funcion de transferencia por $1/s^2$, para formar asi la ecuacion (3).

Pseudocodigo:

COMIENZA
Multiplica al denominador por s^2
incrementa el numero de polinomios en el denominador
TERMINA

3.4 Objeto sal_otros

Añade el siguiente dato y los siguientes módulos:

cond_inic: Arreglo de tipo real en donde se almacenaran las condiciones iniciales del sistema.

1. Modulo despliega: Despliega un menú de posibles entradas al sistema.

Parametro del modulo:

Respuesta: Opcion seleccionada por el usuario.

Pseudocodigo:

```
COMIENZA
   Despliega menu de posibles entradas al sistema
   Lee opcion
   respuesta=opcion
TERMINA
```

2. Modulo desp_func: Obtiene la transformada de Laplace de la opcion elegida por el usuario en el módulo despliega.

Parametros del modulo:

Respuesta: Es el mismo parametro que en el método anterior.

Vl: Denominador de la transformada de Laplace de la entrada; arreglo de numeros reales.

Num: Numerador de la transformada de Laplace de la entrada; arreglo de numeros reales.

Pseudocodigo:

```
COMIENZA
   Lee los valores de los argumentos de la entrada al sistema
   Obtem la transformada de Laplace del termino seleccionado por
   el usuario
   num = numerador de la transformada de Laplace
   Vl = denominador de la transformada de Laplace
TERMINA
```

3. Modulo lee: Calcula la transformada de Laplace de la señal de entrada introducida por el usuario y la multiplica por la funcion de transferencia para obtener Y(s).

Parametros del modulo:

Indice: Apuntador al arreglo coef_d; numero entero.

Indice2: Apuntador al arreglo coef_n; numero entero.

Vector1: Polinomio auxiliar que permitirá calcular la respuesta libre y la respuesta forzada del sistema; arreglo de numeros reales.

Arr: Condiciones iniciales del sistema; arreglo de numeros reales.

Grad2: Grado del vector arr; numero entero.

Pseudocódigo:

```
COMIENZA
    Llama al procedimiento despliega
    Lee la entrada al sistema
    Encuentra la transformada de Laplace de la entrada
TERMINA
```

4. Modulo polinomio_c1: Forma un polinomio auxiliar con el vector de condiciones iniciales que ayudará a calcular la respuesta libre del sistema.

Puesto que:

$$L\left\{\frac{d^n}{dt^n} y(t)\right\} = s^n Y(s) - \sum_{k=1}^{n} s^{n-k} f^{(k-1)}(0)$$

En el calculo de la respuesta forzada, el término de la sumatoria no tiene efecto, ya que las condiciones iniciales se consideran nulas. Para los demas tipos de respuesta es necesario calcular el polinomio que resulta al evaluar la sumatoria. Para cada derivada presente en el modelo matematico del sistema se

tendrá que encontrar este polinomio. La suma de todos los polinomios se almacenara en un vector auxiliar denominado Vl.

Parametros del modulo:

Grado: Grado del polinomio auxiliar; numero entero.

Vl: Coeficientes del polinomio; arreglo de números reales.

Pseudocodigo:

```
COMIENZA
    Lee las condiciones iniciales del sistema
    Encuentra un polinomio auxiliar para encontrar la respuesta
    libre del sistema
    Asigna el polinomio auxiliar a Vl
TERMINA
```

Caso discreto:

III.5 Objeto sal_discreta

Inicialmente se desarrollo el programa para encontrar los diferentes tipos de respuesta en el dominio del tiempo continuo. Se encontro que para extender la solución al caso discreto se deberia aplicar básicamente el mismo método, solo que trans_inv y valor_num deberian de ser procedimientos adaptados para el caso discreto, de tal manera de que en vez de aplicar la transformada inversa de Laplace a cada término se aplicara la transformada z inversa. Para este desarrollo se creó un objeto al que se denominó sal_discreta, el cual heredaria todas las caracteristicas del objeto transfer pero desarrollaria sus propios métodos trans_inv y valor_num. A continuación se describen los módulos que se añadieron al objeto.

1. Modulo abre: Abre los archivos a utilizar.

2. Modulo cierra: Cierra los archivos utilizados.

3. Módulo desp: Despliega un polinomio en términos de la variable z. Utiliza el mismo pseudocódigo que el que se describió para el objeto transfer, pero ahora utiliza z en vez de s.

Parametros del modulo:

Utiliza los mismos parámetros que se describieron en el caso continuo.

4. Modulo imp_func: Despliega la función de transferencia del sistema. Utiliza el mismo pseudocódigo que el descrito en el objeto transfer, pero cambia la 's' por la 'z'.

5. Módulo trans_inv: Calcula y despliega la transformada z inversa de y(z), para lo cual aplica las siguientes fórmulas:

$$z^{-1}\left\{\frac{z}{z-a}\right\} \longrightarrow a^k$$

$$z^{-1}\left\{\frac{z}{(z-a)^n}\right\} \longrightarrow \frac{1}{cte}\prod_{i=0}^{n-2}(k-i)\,a^k$$

$$\text{Donde } cte = a^{n-1}\prod_{i=1}^{n-2}(n-i)$$

$$z^{-1}\left\{\frac{z\,r\,\text{sen}\,\theta}{z^2-2r\cos\theta+r^2}\right\} \longrightarrow r^k \,\text{sen}\,\theta k$$

$$z^{-1}\left\{\frac{z(z-r\cos\theta)}{z^2-2r\cos\theta+r^2}\right\} \longrightarrow r^k\cos\theta k$$

Parametros del módulo:

Utiliza los mismos parámetros que se utilizaron en el caso continuo.

Pseudocódigo:

```
COMIENZA
   PARA i = 1 HASTA número de raices
      SI raices[i].tipo=1 ENTONCES
      COMIENZA
         SI raices[i].r<>0 ENTONCES
            Imprime(mat[j,n+1],'(',raices[i].r,')',**k)
         SI NO imprime(mat[j,n+1],'δ(k) ');
         j=j+1
      TERMINA
      SI raices[i].tipo = 2 ENTONCES
      COMIENZA
         cont=1;
         Llama a raices_rem(rep,i)
         MIENTRAS rep>0 HAZ
            imprime(mat[j,n+1]);
            despliega_func(cont)
            imprime('(',raices[i].r,')**k')
            cont=cont+1
            rep=rep-1
            i=i+1
            j=j+1
         TERMINA
      TERMINA
      SI raices[i].tipo = 3 ENTONCES
      COMIENZA
         c= -2*raices[i].r
         d= raices[i].r*raices[i].r+raices[i].i*raices[i].i
         a= mat[j,n+1]
         b = mat[j+1,n+1]
         j = j+2
         r = √d
         c1 = cr
         d1 = √(c1² - a1²)
         con2 = (-bc/2 + a)/(rsen(angulo)
         imprime(b,'(',r,')**k cos',angulo,'+',con2,'(',r,')**k
```

```
                sen ',angulo)
        TERMINA
    TERMINA
TERMINA
```

c. Modulo despliega: Desarrolla un algoritomo para desplegar simbolicamente el valor de la transformada z inversa de un termino de la forma:

$$\frac{z}{(z - raiz)^n}$$

Parametro del modulo:

Cont: Valor de n.

Pseudocodigo:

```
COMIENZA
    si cont > 1 ENTONCES
    COMIENZA
       j=1
       Escribe ('K')
       Para i = 1 hasta cont HAZ
       COMIENZA
          Escribe('(k-',j,')')
          j=j+1
       TERMINA
    TERMINA
TERMINA
```

f. Modulo polinomio_ci: Captura las condiciones iniciales del sistema y encuentra un polinomio auxiliar para posteriormente calcular la respuesta libre del sistema.

Parametros del modulo:

Utiliza los mismos parametros que los descritos para el caso continuo.

8. Modulo valor_num: Calcula numericamente el valor de la transformada z inversa.

Pseudocodigo:

COMIENZA
Divide el numerador de x(s) entre el denominador
Obtén los coeficientes del polinomio del resultado de la division
Almacena los resultados en archivos
TERMINA

3.6 Objeto disc_escalon

Modulo disc_escalón: Multiplica a la funcion de transferencia por z/(z-1) para asi formar la ecuación (6).

Pseudocodigo:

COMIENZA
Multiplica el numerador por z
Incrementa el número de polinomios en el numerador
Multiplica el denominador por (z-1)
Incrementa el número de polinomios en el denominador
TERMINA

3.7 Objeto disc_rampa

Módulo disc_rampa: Multiplica a la función de transferencia por $z/(z-1)^2$ para asi formar la ecuacion (7).

Pseudocódigo:

COMIENZA
Multiplica el numerador por z
Incrementa el número de polinomios en el numerador

Multiplica el denominador por $(z-1)^2$

Incrementa el número de polinomios en el denominador

TERMINA

III.8 Objeto disc_otros

1. Módulo despliega: Despliega un menú con posibles entradas al sistema discreto.

Parámetros del módulo:

Utiliza los mismos parámetros que los descritos para el caso continuo.

2. Módulo desp_func: Obtiene la transformada z de la opción elegida por el usuario.

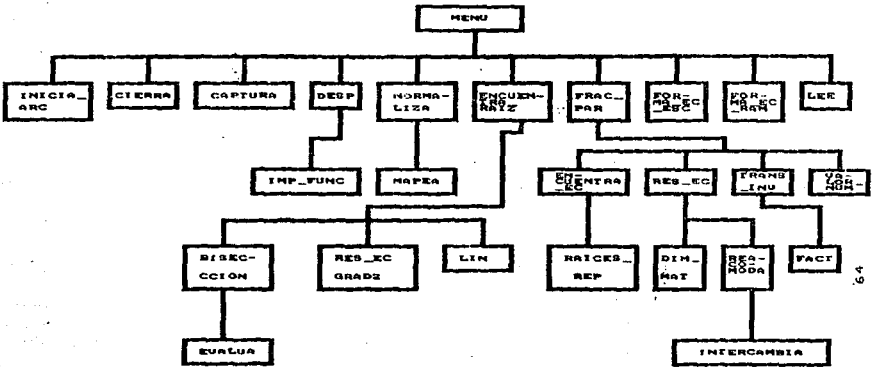
3. Módulo lee: Calcula la transformada z de la señal de entrada y la multiplica por la función de transferencia del sistema.

Parámetros del módulo:

Utiliza los mismos parámetros que en el caso continuo.

IV Diagrama de estructura

En la siguiente página se muestra el diagrama de estructura del programa


MENU
INICIA_ARC
CIERRA
CAPTURA
DESP
NORMALIZA
ENCUENTRA RAIZ
FRAC_PAR
FOR_RES_EC
FOR_RAM_EC
LEE
IMP_FUNC
MAPEA
ENCUENTRA
RES_EC
TRANS_INV
VANDER
BISEC-CCION
RES_EC GRAD2
LIN
RAICES_REP
DIM_MAT
INTERCAMBIA
FACT
EVALUA
INTERCAMBIA

V Métodos virtuales

Fué necesario utilizar el concepto de métodos virtuales para el desarrollo de los procedimientos trans_inv y valor_num, para que no fuera sino hasta tiempo de ejecución cuando se resolvieran las referencias externas a ellos. Es decir, para que el método frac_par tuviara elementos para decidir llamar a los métodos trans_inv y valor_num bien del objeto sal_discreta para el caso contínuo o bien del objeto sal_discreta para el caso discreto, lo cual sólo podría determinarse hasta al tiempo de ejecución.

VI Unidades desarrolladas

El programa para cálculo de la respuesta en el tiempo de sistemas dinámicos se codificó dentro de cuatro unidades: respuesta, resp2, resp3 y principal. La unidad respuest está integrada por el objeto transfer, que se utiliza para obtener la respuesta impulso para el caso contínuo. En la unidad resp2 se codificaron los objetos sal_escalón, sal_rampa y sal_otros, que se utilizan para obtener la respuesta escalón, rampa y para obtener la respuesta del sistema a cualquier entrada indicada por el usuario en el dominio del tiempo contínuo. En la unidad resp3 se codificaron los objetos sal_discareta, disc_escalón, disc_rampa y disc_otros, que son utilizados para obtener la respuesta del sistema a diversas excitaciones en el dominio del tiempo discreto. Finalmente dentro de principal se controla la activación de los diversos objetos de que consta el programa dependiendo de la opción con la que elija trabajar el usuario.

VII Descripción del método de solución

A continuación se describirá con un ejemplo el método de solución empleado.

Sea la función de transferencia:

$$G(s) = \frac{3s - 2}{s^3(s^2 + 4)} \quad \ldots (1)$$

Su representacion interna dentro del objeto transfer estara dada por:

```
Num_poln   = 1          coef_d[1] = 1 }
Num_pold   = 2          coef_d[2] = 0 }  Representa a s^3, que es el primer
grado_n[1] = 1          coef_d[3] = 0 }  polinomio del
grado_d[1] = 3          coef_d[4] = 0 }  denominador
grado_d[2] = 2          coef_d[5] = 1 }
coef_n[1]  = 3          coef_d[6] = 0 }  Representa a s^2+ 4 que corresponde al
coef_n[2]  = 2          coef_d[7] = 4 }  segundo polinomio del denominador
                        cte = 1
```

Primeramente se llevará a cabo la captura de los datos a traves del metodo denominado captura. Despues se llamara a un metodo que normalizara a los polinomios que formarán al denominador de la funcion de transferencia.

Si se quisiera encontrar la respuesta impulso del sistema representado en (1) se procederia de la siguiente manera:

El metodo encuentra_raiz llamaria a otros metodos (biseccion, lin y res_ec_grad2) con el fin de obtener los polos de la funcion de transferencia. Si no es posible encontrar todos los polos, en este momento termina el programa con un mensaje para el usuario. En caso contrario se procede a clasificar a las raices en tipo 1, 2, 3 o 4 según corresponda. Para este caso:

Arreglo raices

.r	.i	tipo
0	0	2
0	0	2
0	0	2
0	2	3

A partir de este arreglo se puede conocer la forma de las fracciones equivalentes a la función de transferencia. Para el ejemplo que se está desarrollando se tendría:

$$\frac{A}{s} + \frac{B}{s^2} + \frac{C}{s^3} + \frac{D + Es}{s^2 + 4} = \frac{3s - 2}{s^3(s^2+4)} \quad \text{--- (2)}$$

Este desarrollo lo llevaría a cabo el método frac_par. Los valores de las constantes A, B, C, D y E se determinarán a través del método encuentra_ec y res_ec. Para este ejemplo, de la ecuación (2) se tiene:

$$A\,s^2(s^2+4) + B\,s\,(s^2+4) + C(s^2+4) + (D + Es)\,s^3 = 3s - 2$$

$$= A(s^4 + 4s^2) + B(s^3 + 4s) + C(s^2 + 4) + Ds^3 + Es^4$$

Comparando los coeficientes de las potencias de s en ambos miembros de la igualdad, se puede formar un sistema de cinco ecuaciones con cinco incógnitas A, B, C, D y E:

$$\begin{array}{c} s^0 \\ s \\ s^2 \\ s^3 \\ s^4 \end{array} \begin{bmatrix} 0 & 0 & 4 & 0 & 0 & -2 \\ 0 & 4 & 0 & 0 & 0 & 3 \\ 4 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 \end{bmatrix} \begin{array}{c} \\ \\ \\ \\ \\ A \quad B \quad C \quad D \quad E \end{array}$$

El método encuentra_ec se encargará de formar este sistema de ecuaciones, y el método res_ec de encontrar su solución aplicando el método de Gauss - Jordan, el cual transformaría a la matriz anterior en:

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0.125 \\ 0 & 1 & 0 & 0 & 0 & 0.75 \\ 0 & 0 & 1 & 0 & 0 & -0.5 \\ 0 & 0 & 0 & 1 & 0 & -0.75 \\ 0 & 0 & 0 & 0 & 1 & -0.125 \end{bmatrix}$$

Matriz MAT

Con el arreglo raices y la matriz MAT ya es posible encontrar la respuesta impulso. A partir de estos datos se puede obtener:

$$y(s) = \frac{0.125}{s} + \frac{0.75}{s^3} - \frac{0.5}{s^3} + \frac{-.75 - .125}{s^2 + 4}$$

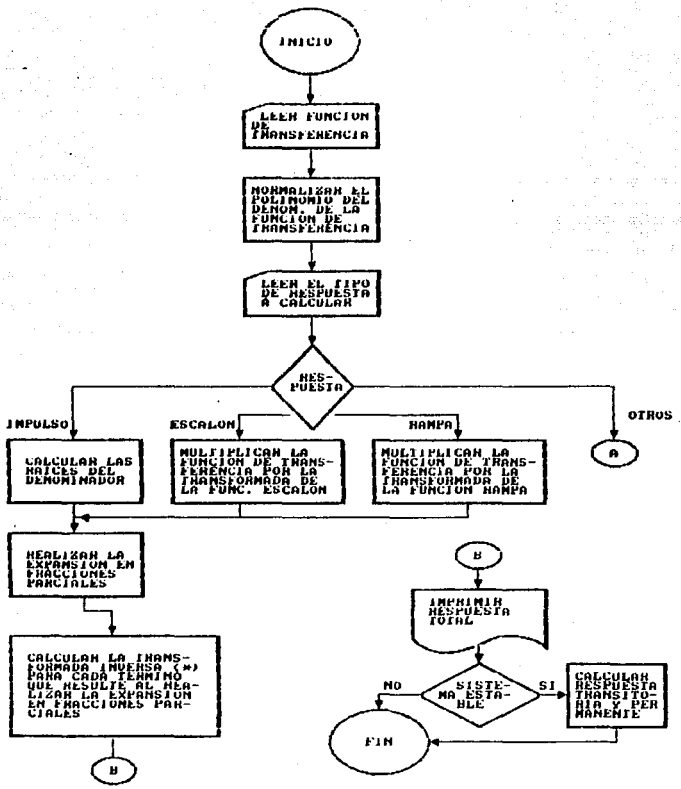
El método trans_inv se encargará de desplegar simbólicamente la respuesta en el tiempo de la siguiente manera:

$$y(t) = 0.125 + 0.75t - 0.25\ t^2 - 0.125 \cos 2t - 0.375 \operatorname{sen} 2t$$

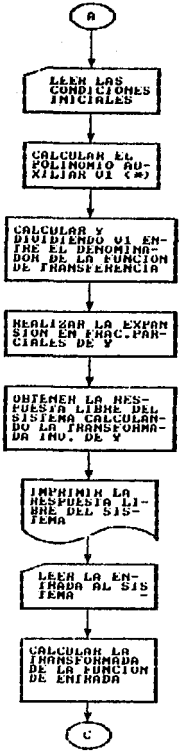
El método valor_num se encargará de evaluar numéricamente esta expresión y de almacenarla en un archivo.

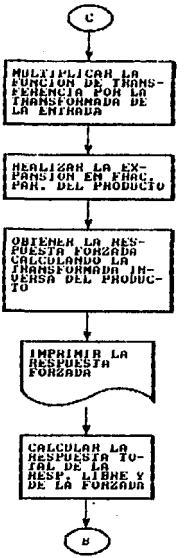
Posteriormente se encontrará la respuesta transitoria y permanente a partir de la respuesta total.

Para el caso de señales de entrada típicas (impulso, escalón y rampa) no se calculará el valor de la respuesta forzada ni de la respuesta libre, ya que en estos casos se consideran condiciones iniciales nulas. A continuación se describirá con un diagrama de bloques el procedimiento general que se sigue para el cálculo de los diferentes tipos de respuestas de sistemas dinámicos:



(*) Para el caso continuo la transformada inversa de Laplace, mientras que para el discreto la transformada Z inversa.





(*) Este polinomio se describio en el modulo denominado polinomio_01

CAPITULO IV CASOS DE PRUEBA

I. EJEMPLOS

A continuación se mostrará la salida del programa de cálculo de la respuesta en el tiempo de sistemas dinámicos ante diversas señales de entrada. Los casos de prueba serán tomados de ejercicios resueltos de algunos libros para así poder validarlos.

CASO 1. Para

$$G(s) = \frac{10\ 000}{s^2 + 200\ s + 10\ 000}$$

$$y(t)_{IMPULSO} = 10\ 000\ t\ e^{-100t}$$

El programa desplegó el siguiente resultado:

10000 t exp(-100t)

Respuesta transitoria:
10000 t exp(-100t)

Respuesta permanente:
0

CASO 2. Para: (16)

$$G(s) = \frac{s^2 + s + 7}{(s + 0.5)(s^2 + 2s + 3)}$$

$$y(t)_{ESCALON} = 4.66 - 6e^{-0.5t} + e^{-t}(1.34 \cos 1.414t + 0.47 \text{ sen } 1.414t)$$

De donde:

(16) Rodríguez, Fco. op.cit p.342

$$y(t)_{TRANSITORIA} = -6e^{-0.5t} + e^{-t}(1.34 \cos 1.414t + 0.47 \text{ sen } 1.414t)$$

$$y(t)_{PERMANENTE} = 4.66$$

El programa desplegó los siguientes resultados:

-6exp(-0.500t) - 0.471 exp(-t) sen 1.414t + 1.333 exp(-t)cos1.414t + 4.667

Respuesta transitoria:
-6exp(-0.500t) - 0.471 exp(-t) sen 1.414t + 1.333 exp(-t)cos1.414t

Respuesta permanente:
+ 4.667

CASO 3. Para:

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

$$y(t)_{RAMPA} = t - 1 + e^{-2t}(1 + t) \quad (17)$$

$$y(t)_{TRANSITORIA} = e^{-2t}(1 + t)$$

$$y(t)_{PERMANENTE} = t - 1$$

El programa desplegó los siguientes resultados:

exp(-2t) + texp(-2t) - 1 + t

Respuesta transitoria:
exp(-2t) + texp(-2t)

(17) Ogata, Op. cit p. 296-297

Respuesta permanente:

-1 + t

Caso 4. Para

$$G(s) = \frac{1}{s^2 + 200s + 10\,000}$$

Señal de entrada: 1 200 000 cos 377t

Condiciones iniciales: y(0) = 50

y'(0) = 500

Los diferentes tipos de respuesta del sistema son:

$$y(t)_{\text{LIBRE}} = 50\, e^{-100t} + 5500\, t\, e^{-100t}$$

$$y(t)_{\text{FORZADA}} = 6.85\, e^{-100t} - 788.8\, te^{-100t} - 6.85 \cos 377t + 3.9 \operatorname{sen} 377t$$

$$y(t)_{\text{TOTAL}} = 56.85\, e^{-100t} + 4\,711.7\, t\, e^{-100t} - 6.85 \cos 377t + 3.9 \operatorname{sen} 377t$$

$$y(t)_{\text{TRANSITORIA}} = 56.85\, e^{-100t} + 4\,711.7\, t\, e^{-100t} \quad (18)$$

$$y(t)_{\text{PERMANENTE}} = 3.9 \operatorname{sen} 377t - 6.85 \cos 377t$$

Los resultados que desplego el programa fueron:

Respuesta libre:

50exp(-100t) + 5500texp(-100t)

Respuesta forzada:

6.851exp(-100t) - 788.804texp(-100t) + 3.910sen377t

-6.851cos377t

(18) Ibidem, p. p. 254-272

Respuesta total:

56.851exp(-100t) + 4711.196texp(-100t) + 3.910sen377t
-6.851cos377t

Respuesta transitoria:

56.851exp(-100t) + 4711.196texp(-100t)

Respuesta permanente:

+ 3.910sen377t - 6.851cos377t

CASO 5. Para:

$$G(z) = \frac{0.632z}{(z-1)(z-0.368)} \qquad (19)$$

$$y(k)_{ESCALON} = -0.582\ (1)^k + k + 0.582\ (0.368)^k$$

De donde:

$$y(k)_{TRANSITORIA} = 0.582\ (.368)^k$$

$$y(k)_{PERMANENTE} = -0.582\ (1)^k + k$$

Los resultados que desplego el programa fueron:

Respuesta total:

-0.582(1)**k + k(1)**k + 0.582 (0.368)**k

Respuesta transitoria:

0.582 (0.368)**k

Respuesta permanente:

-0.582(1)**k + k(1)**k

(19) Ismael Espinoza, Análisis de sistemas discretos, p. 48

CASO 6

$$G(z) = \frac{0.632z}{(z-0.368)}$$

$$y(k)_{\text{RAMPA}} = -0.582\ (1)^k + k + 0.582\ (0.368)^k$$

De donde:

$$y(k)_{\text{TRANSITORIA}} = 0.582\ (.368)^k$$

$$y(k)_{\text{PERMANENTE}} = -0.582\ (1)^k + k$$

Los resultados que despliego el programa fueron:

```
 0.582 ( 0.368 )**k - 0.582(1)**k + k(1)**k

Respuesta transitoria:
0.582 ( 0.368 )**k

Respuesta permanente::
-0.582(1)**k + k(1)**k
```

CASO 7

$$G(z) = \frac{z^2}{z + 2}$$

El programa despliega el siguiente resultado:

La funcion de transferencia no corresponde a un sistema fisico realizable.

Con lo que se cumple el punto 1 del requerimiento 3.0.

CASO 8

G(s) = 0/0

El programa despliega el siguiente mensaje:

Error: G(s) = 0/0. Existe una indeterminación.

Con lo que se cumple el punto 2 del requerimiento 3.0

CASO 9

En el momento de capturar la función de transferencia si se da un 0 como número de polinomios en el numerador; el programa despliega el mensaje:

Entrada inválida: El número de polinomios debe de ser mayor a cero.

Con lo que se cumple el punto 4 del requerimiento 3.0

II. RESULTADOS

Los casos de prueba presentados anteriormente muestran que el programa para cálculo de la respuesta en el tiempo de sistemas dinámicos cumple con los objetivos y requerimientos planteados inicialmente.

El programa calcula analítica y numéricamente las respuestas libre, forzada, total, transitoria y permantente de sistemas dinámicos.

Se puede trabajar tanto en el dominio del tiempo contínuo como en el discreto.

El programa se activa tal como se señaló en el. requerimiento 2.0.

La validación de los datos de entrada se lleva a cabo tal como se indicó en el requerimiento 3.0.

III. DISCUSION DE RESULTADOS

Tal como se esperaba al iniciar el desarrollo del sistema que he mostrado a lo largo de la presente tesis, se logró obtener eficiencia en el tiempo de ejecución del programa, aunque obviamente este tiempo se va incrementando a medida que el sistema dinámico sea más complejo. Se logró también un producto final de fácil manejo para el usuario.

El programa se sometió a diversas pruebas por medio de las cuales fué posible detectar algunos errores en los algoritmos empleados, sin embargo será necesario continuar somentiendo al programa con más casos de prueba en una etapa posterior de mantenimiento. Se espera que sean los mismos usuarios de este sistema los que indiquen si para algún caso específico el programa no despliega los resultados esperados.

Una vez que se considere que este sistema ha sido sometido a suficientes pruebas sin que presente problemas, se integrará junto con otros sistemas para formar así un paquete de control que se espera sea poderoso y que al mismo tiempo sea de fácil manejo para el usuario final.

CONCLUSIONES

El programa para cálculo de la respuesta en el tiempo de sistemas dinámicos presenta la ventaja de ser eficiente en cuanto a tiempo de ejecución. Es un programa de fácil manejo. No requiere del empleo de un coprocesador matemático, como es el caso del paquete CC.

Con respecto al empleo de la programación orientada a objetos en el desarrollo del programa presentado, se concluye que efectivamente esta metodología de diseño es un medio efectivo de manejar problemas. Sus características de polimorfismo y herencia

añadidas con la reusabilidad de código existente, llevan a la disminución del tiempo de desarrollo de un programa y al manejo más eficiente de problemas complejos.

APENDICE A

Listado del programa fuente para el cálculo de la respuesta en el tiempo de sistemas dinámicos

```
{En esta unidad se codificaron una serie de procedimientos para el
cálculo de la respuesta impulso}
unit respuesta;
interface
   uses crt;
type
   arreglo = array [1..100] of INTEGER;
   vector_r= array [1..100] of REAL;
   vec_20 =  array [0..20] of REAL;
   vec_23 =  array [-2..30] of REAL;
   mat_20 = array[1..20,1..20] of REAL;
   reg= RECORD
          R,I: REAL;
          tipo: integer;
        end;
   reg2 = RECORD
             grado:integer;
             pol: vec_20;
          end;
   arreglo2 = array[1..20] of reg2;
   con= set of 1..30;
   transfer = object
      Num_poln, num_pold :INTEGER;
      grado_n, grado_d: arreglo;
      coef_n, coef_d: vector_r;
      cte: REAL;
      constructor inicia_arc;
      procedure cierra;
      procedure captura(var indice,indice2:integer);
      procedure desp(var grad: arreglo; var coef: vector_r; var num_pol
      : integer);
      procedure imp_func;
      procedure multiplica(var v1,v2, resultado:vec_20; grado1, grado2
      :integer);
      procedure evalua(var f:real; x:real; func:vec_20; grado:integer);
      procedure dim_mat(var n:integer);
      procedure intercambia(i,j,n:integer; var mat:mat_20);
      procedure reacomoda(var mat:mat_20);
      procedure res_ec(var mat:mat_20; n: integer);
      procedure mapea(var k: integer; indice: integer);
      procedure raices_rep( var repetidas: integer; indice:integer);
      procedure normaliza;
      procedure res_ec_grad2( a,b,c:real);
      function sgn(x:real):integer;
      function fact(n:integer):integer;
      procedure lin(var func: vec_23; n:integer);
      procedure biseccion (grado,i:integer);
      procedure divide(var numerador:vector_r; var grado:integer;
                       var t:real);
      procedure suma_quebrados(mat1,mat2: arreglo2; apuntador:integer;
                               var vec: vec_20; var mayor:integer);
```

```
        procedure encuentra_raiz;
        Procedure encuentra_ec (conjunto:con; i, num_ra:integer;
                                var mat:mat_20; col:integer);
        Procedure imprime(coef:real; j:integer);
        Procedure trans_inv(mat:mat_20; opcion:integer); virtual;
        Procedure eleva(t:real; cont:integer; var elev:real);
        Procedure frac_par(var inestable:boolean; vec:vec_20;opcion:integer);
        Procedure valor_num(mat: mat_20; opcion:integer); virtual;

    end;
    var
        aP_raiz:integer;
        raices: array[1..21] of reg;
        falla:boolean;
        cons2:real;
        f1,f2,f3,f4,f5:text;
        genera:char;

{ Num_poln, num_pold: En estas variables se almacena el número
      de polinomios del numerador y del denominador que confor-
      man a la función de transferencia.
  Grado_n, grado_d: En estos vectores se almacenan los grados
      de los polinomios que forman al numerador y denominador
      de la función de transferencia
  Coef_n, coef_d: Contienen los coeficientes de los polinomios
      que forman a la función de transferencia }


implementation

{*****************************************************************}
{ El siguiente procedimiento tiene por objetivo capturar a la funcion}
{ de transferencia                                                }
{*****************************************************************}

procedure transfer_captura(var indice,indice2:integer);
var
   opcion,i,j, pol: integer;

BEGIN
   num_poln:=0;
   while num_poln<=0 DO
   begin
      write('Numero de polinomios en el numerador: ');
      readln(num_poln);
      if num_poln<=0 then
         writeln('Entrada inválida: El número de polinomios debe de ser
                  mayor a cero');
   end;
   indice2:=1;
   for i:= 1 to num_poln do
   BEGIN
      write('Grado del ',i,'o: ');
      readln(grado_n[i]);
      writeln('Introducir los coeficientes del polinomio');
      FOR j:= 0 to grado_n[i] DO
      BEGIN
         readln(coef_n[indice2]);
```

```
            indice2:= indice2+1;
        end;
    end;
    num_pold:=0;
    while num_pold<= 0 do
    begin
        write('Numero de polinomios en el denominador: ');
        readln(Num_pold);
        if num_pold<=0 then
            writeln('Entrada inválida el número de polinomios debe de ser
            mayor a cero');
    end;
    indice:= 1;
    FOR i:= 1 to num_pold DO
    BEGIN
        write('Grado del ',i,'o: ');
        readln(grado_d[i]);
        writeln('Introducir coeficientes');
        FOR j:= 0 to grado_d[i] do
        BEGIN
            readln(coef_d[indice]);
            indice:= indice+1;
        end;
    end;
    for i:= 1 to 21 do
    begin
        raices[i].r:=0;
        raices[i].i:=0;
        raices[i].tipo:=0;
    end;
end;

{*********************************************************************}
{ El siguiente metodo prepara los archivos que van a utilizarse      }
{*********************************************************************}

constructor transfer.inicia_arc;
begin
    assign(f1,'resp_libre.rtc');
    assign(f2,'resp_forz.rtc');
    assign(f3,'resp_total.rtc');
    assign(f4,'resp_trans.rtc');
    assign(f5,'resp_perm.rtc');
    rewrite(f1);
    rewrite(f2);
    rewrite(f3);
    rewrite(f4);
    rewrite(f5);
    writeln(f1,'RESPUESTA LIBRE');
    writeln(f1,'INICIO: ');
    writeln(f2,'RESPUESTA FORZADA');
    writeln(f2,'INICIO: ');
    writeln(f3,'RESPUESTA TOTAL');
    writeln(f3,'INICIO: ');
    writeln(f4,'RESPUESTA TRANSITORIA');
    writeln(f4,'INICIO: ');
    writeln(f5,'RESPUESTA PERMANENTE');
    writeln(f5,'INICIO: ');
```

```
end;

{*************************************************************}
{ El siguiente metodo cierra los archivos utilizados          }
{*************************************************************}

procedure transfer.cierra;
begin
   close(f1);
   close(f2);
   close(f3);
   close(f4);
   close(f5);
end;

{*************************************************************}
{ El metodo transfer.desp despliega la funcion de transferencia }
{*************************************************************}

procedure transfer.desp(Var grad : arreglo;
                Var coef: vector_r;
                Var num_pol: INTEGER);

VAR
   i, grado, j, terminos, part_ent
   :INTEGER;

BEGIN
   if (num_pol=1) and (grad[1]=0) and (coef[1]=0) then write('0');
   j:= 1;
   FOR i:= 1 to num_pol do
   BEGIN
      grado:= grad[i];
      terminos:= grad[i] + 1;
      WHILE terminos > 0 DO
      BEGIN
         IF ((num_pol <> 1) and (terminos = grad[i]+1)) THEN
            write ('(');
         IF (grado = 0) THEN
         BEGIN
            part_ent:= trunc(coef[j]);
            if (coef[j]<0) THEN
            begin
               if ((part_ent-coef[j])=0) THEN write (part_ent)
               else
                  write(coef[j]:6:3);
            end;
            if num_pol <> 1 then write ( ')' );
            j:= j+1;
         end
         ELSE
         BEGIN
            IF (coef[j]<>0) THEN
            BEGIN
               if coef[j]<>1 then
               begin
                  part_ent := trunc (coef[j]);
                  if((part_ent-coef[j])=0) THEN write (part_ent)
```

```
                else
                   write (coef[j]:6:3);
             end;
             write('s');
             if grado>1 then write ('**',grado);
          end;
          j:= j+1;
          IF ((coef[j]>0) and (grado <> 0)) THEN
             write('+');
       end;
       terminos:= terminos-1;
       grado:=grado-1;
       IF ((coef[j]=0) and (grado>0)) THEN
       BEGIN
          j:= j+1;
          if ((coef[j]>0) and (grado<>0)) then
             write('+');
          terminos:= terminos-1;
          grado:= grado-1;
       end;
    end; (while)
 end; (for)
end;

{*************************************************************************}
{ El procedimiento imp_func tiene por objetivo imprimir la función      }
{ de transferencia del sistema                                          }
{*************************************************************************}

procedure transfer.imp_func;

begin
   clrscr;
   gotoxy(5,2);
   write('G(s) = ');
   desp(grado_n,coef_n, num_poln);
   write(' / ');
   desp(grado_d,coef_d, num_pold);
end;

{*************************************************************************}
{ Salida: resultado, que contendra los coeficientes del polinomio que   }
{ resulte al multiplicar los polinomios v1 por v2                       }
{*************************************************************************}

procedure transfer.multiplica (Var v1, v2, resultado: vec_20;
                               grado1, grado2: integer);

VAR
   i, j, producto
   : INTEGER;

BEGIN
   FOR I := 0 to 20 do
      resultado[i]:= 0;
   FOR I := 0 to grado1 DO
   BEGIN
      FOR j:= 0 to grado2 DO
```

```
      begin
         resultado[i+j] := resultado[i+j] + v1[i]*v2[j];
      end;
   end;
end;

{*****************************************************************}
{ Dado un pllinomio cuyoes coeficientes estan en el arreglo func, y dado}
{ el valor de x, el metodo transfer evalua,  deja  en  f  el  reslutado
(de f(x))
{*****************************************************************}

procedure transfer.evalua (VAR f: real;
                           x: real;
                        func: vec_20;
                       grado: integer);

VAR
   term: integer;
   expon: real;

BEGIN
   expon:= 1;
   term:= 0;
   f:= func[0];
   while term<grado DO
   BEGIN
      term:= term+1;
      expon:= expon * x;
      f:= f+ expon*func[term];
   end;
end;

{*****************************************************************}
{ El siguiente metodo calcula el numero de variables cuyo valor sera)
{ necesario determinar por el metodo de expancion en fracciones)
{ parciales)
{*****************************************************************}


procedure transfer.dim_mat(VAR N: integer);
VAR
   i: integer;

BEGIN
   n:= 0;
   FOR i := 1 to num_pold DO
      N:= N+ grado_d[i];
end;

{*****************************************************************}
{ El siguiente metodo intercambia el renglon i por el j de la matriz mat)
{*****************************************************************}

procedure transfer.intercambia(i,j,n: integer;
                               VAR mat:mat_20);

VAR
```

```
   temporal: array [1..20] of real;
   k: integer;

BEGIN
   FOR k:= 1 to n+1 DO
   BEGIN
      temporal[k]:= mat[j,k];
      mat[j,k]:= mat[i,k];
   end;
   FOR k:= 1 to N+1 DO
      mat[i,k]:= temporal[k];
end;

(*************************************************************)
( Si al tratar de resolver un sistema de ecuaciones se encuentra que )
( alguno de los valores de la diagonal principal es cero, se llamara )
(al siguiente proedimiento para reacomodar los renglones de la matriz)
(que contiene un sistema de ecuacione                        s )
(*************************************************************)

procedure transfer.reacomoda(VAR mat: mat_20);
VAR
   N, INDICE, I, J, dif_cero
   : integer;
   conj: set of 1..30;
   bandera: boolean;

BEGIN
   conj:= [];
   DIM_MAT(N);
   j:=n;
   while j>0 DO
   BEGIN
      dif_cero:=0;
      FOR i:= 1 to N DO
      BEGIN
         IF mat[i,j]<>0 THEN
         BEGIN
            dif_cero:= dif_cero+1;
            indice:=i;
         end;
      end;
      IF dif_cero = 1 THEN
      BEGIN
         IF indice <> j THEN
            intercambia(indice,j,n,mat);
         conj:= conj + [j];
      end;
      j:= j-1;
   end;
   FOR i:= 1 to n DO
   BEGIN
      bandera:= true;
      if mat[i,i] = 0 THEN
      BEGIN
         j:= 1;
         WHILE (bandera) and (j<=n) DO
         BEGIN
```

```
            if (mat[j,i]<>0) and (mat[i,j]<>0) and (not(j in conj)) THEN
            BEGIN
               intercambia (i,j,n,mat);
               bandera:= false;
            end;
            j:= j+1;
         end;
         IF (bandera) THEN
         BEGIN
            j:= 0;
            WHILE (bandera) and (j<= N) DO
            BEGIN
               j:= j+1;
               if (mat[j,i]<>0) and (not(j in conj)) then
               BEGIN
                  intercambia(i,j,n,mat);
                  bandera:= false;
               end;
            end;
         end;
      end;
   end;
end;

(*****************************************************************)
( El siguiente metodo resuelve un sistema de ecuaciones         )
(*****************************************************************)

Procedure transfer.res_ec(VAR mat: mat_20;
                              n: INTEGER);

VAR
   m, i, k, j
   : INTEGER;
   pivote,c
   : REAL;
   bandera: boolean;

BEGIN
   bandera:= true;
   j:=1;
   m:= n+1;
   while ( (j<10) and (bandera) ) DO
   BEGIN
      bandera:= false;
      for i:= 1 to n DO
         if mat[i,i]=0 then bandera:= true;
      if bandera THEN
         reacomoda(mat);
      j:= j+1;
   end;
   FOR i:= 1 to n do
   BEGIN
      pivote:= mat[i,i];
      if pivote = 0 then
      begin
         writeln('- El sistema sale fuera de los alcances de este programa')
         falla:= true;
```

```
        i:=n;
      end
      else
      BEGIN
         FOR j:= 1 to m DO
            mat[i,j]:= mat[i,j]/Pivote;
         FOR k:= 1 to n DO
         BEGIN
            if k<>i then
            BEGIN
               c:= mat[k,i];
               FOR j:= 1 to m DO
                  mat[k,j]:= mat[k,j]-c*mat[i,j];
            end;
         end;
      end;
   end;
   for j:= 1 to n do
      if (abs(mat[j,m])<1.0E-9) then mat[j,m]:=0;
end;

{**********************************************************************}
{ Dado un valor de indice,  que corresponde al numero asignado a algun }
{ polinomio del denominador que se quiera localizar, el procedimiento  }
{ mapea regres un apuntador al arreglo coef_d, en donde se localizan   }
{ los coeficientes del polinomio a localizar                           }
{**********************************************************************}

procedure transfer.mapea (Var k: integer;
                          indice:integer);

VAR
   i: INTEGER;

BEGIN
   k:= 0;
   FOR i:= 1 to (indice-1) DO
      k:= grado_d[i] + k + 1;
   k:= k+1;
end;

{**********************************************************************}
{ Salida: La variable repetidas, que contiene el numero de veces que se}
{ repite el polo dentro de la funcion                                  }
{**********************************************************************}

procedure transfer.raices_rep(Var repetidas: INTEGER;
                              INDICE: INTEGER);

VAR
   i: INTEGER;
   registro: reg;

BEGIN
   i:= indice;
   repetidas:= 0;
   registro:= raices[i];
   WHILE ((raices[i].r = registro.r) and (raices[i].i = registro.i)
```

```
    and (raices[i].tipo=registro.tipo)) DO
    BEGIN
       repetidas:= repetidas + 1;
       i:= i+1;
    end;
end;

{*************************************************************}
{ El siguiente metodo toma al conjunto de polinomios del denominador }
{ de la funcion de transferencia y deja al coeficente del grado mayor}
{de cada  polinomio con valor de uno                            }
{*************************************************************}


procedure transfer.normaliza;
VAR
   i,j,grad
   : INTEGER;
   coef: REAL;

BEGIN
   cte:= 1;
   if (num_poln=1) and (grado_n[1]=0) and (coef_n[1]<>0) then
   begin
      cte:=1/coef_n[1];
      coef_n[1]:=1;
   end;
   j:= 1;
   FOR i:= 1 to num_pold DO
   BEGIN
      IF coef_d[j] <> 1 then
      BEGIN
         coef:= coef_d[j];
         grad:= grado_d[i];
         coef_d[j]:= 1;
         WHILE grad>0 DO
         BEGIN
            j:= j+1;
            coef_d[j]:= coef_d[j]/coef;
            grad:= grad-1;
         end;
         cte:= cte*coef;
         j:= j+1;
      end
      else j:= j+grado_d[i]+1;
   end;
end;

{*************************************************************}
{ El metodo transfer.res_ec_grad2 resuelve una ecuacion de segundo   }
{grado dejando las raices del polinomio en un arreglo llamado raices }
{*************************************************************}

procedure transfer.res_ec_grad2( a,b,c: real);
VAR
   delta: real;

BEGIN
```

```
   delta:= b*b-4*a*c;
   if (delta<0) and (delta>= -0.001) then delta:=0;
   if delta<0 THEN
   BEGIN
      raices[ap_raiz].R := -b/(2*a);
      raices[ap_raiz].tipo:=3;
      raices[ap_raiz].i:= sqrt(-delta)/(2*a);
      ap_raiz:= ap_raiz+1;
   end
   else
   BEGIN
      raices[ap_raiz].R := (-b+sqrt(delta))/(2*a);
      raices[ap_raiz].I:= 0;
      raices[ap_raiz+1].R := (-b-sqrt(delta))/(2*a);
      raices[ap_raiz+1].I:=0;
      ap_raiz:= ap_raiz+2;
   end;
end;


function transfer.sgn(x: real):integer;
BEGIN
   IF x<0 then sgn:=-1
   else
      sgn:= 1;
end;

(*****************************************************************)
( El siguiente es un metodo numerico para descomponer un polinomio en )
( polinomios de orden dos, el cual se conoce como metodo de LIN       )
(*****************************************************************)

procedure transfer.LIN (Var func: vec_23; n:integer);

VAR
   b: vec_23;
   p,q,r,s,delta_p, delta_q: real;
   CONV: BOOLEAN;
   it,k: integer;
BEGIN
   falla:= false;
   b[-1]:= 0;
   b[-2]:= 0;
   p:=-0.1;
   q:=-0.1;
   delta_p:= 0.1;
   delta_q:= 0.1;
   it:=0;
   conv:= true;
   while ( ( (abs(delta_p)>0.0001) or
           (abs(delta_q)>0.0001) ) and (it<2000) ) DO
   BEGIN
      p:= p+delta_p;
      q:= q+delta_q;
      FOR k:= 0 to n-2 DO
         b[k]:= func[k]-p*b[k-1]-q*b[k-2];
      r:= func[n-1]-p*b[n-2]-q*b[n-3];
      s:= func[n]-q*b[n-2];
```

```
      if b[n-2] <> 0 then
      BEGIN
         delta_q:= s/b[n-2];
         delta_p:= r/b[n-2];
      end;
      it:=it+1;
   end;
   if it<2000 then
   begin
      func:= b;
      res_ec_grad2(1,p,q);
      n:=n-2;
      if n<=2 then
      begin
         if n=1 then
         begin
            raices[ap_raiz].r:= -b[1]/b[0];
            raices[ap_raiz].i:=0;
            ap_raiz:= ap_raiz+1;
         end
         else res_ec_grad2(b[0],b[1],b[2]);
      end
      else
         lin(b,n);
   end
   else falla:= true;
end;


{*********************************************************************}
{ El siguiente procedimiento calcula las raices de un polinomio a     }
{ través del método numérico de bisección                             }
{*********************************************************************}

procedure transfer.biseccion (grado,il:integer);

VAR
   Li, Ls, H,f,g,a, raiz, ev_a: REAL;
   iteraciones,i, indice, w, j, k:integer;
   converge: boolean;
   func,res: vec_20;
   func2: vec_23;

BEGIN
   converge:= true;
   falla:= false;
   mapea(k,il);
   for j:= 0 to grado DO
   begin
      func[grado-j]:= coef_d[k];
      k:= k+1;
   end;
   j:=0;
   while (func[0]=0) and (grado>2) do
   BEGIN
      raices[ap_raiz].r:= 0;
      raices[ap_raiz].i:= 0;
      ap_raiz:= ap_raiz+1;
```

```
    grado:= grado-1;
    for i:= 0 to grado DO
       func[i]:=func[i+1];
end;
while (grado>2) and (converge) DO
BEGIN
    Li:= -50;
    Ls:= 50;
    evalua(F,Li, func, grado);
    evalua(G,Ls,func, grado);
    while ( ((F>0) and (G>0)) or ( (F<0) and (G<0) ) ) and (Li<Ls) DO
    BEGIN
       Li:= Li+0.2;
       evalua(F, Li, func, grado);
    end;
    if (Li<Ls) then
    BEGIN
       if f=0 then raiz:= Li;
       if g=0 then raiz:= Ls;
       if (f<>0) and (g<>0) then
       BEGIN
          iteraciones:= 0;
          ev_a:=1;
          while (abs(ev_a) > 0.0000001) and (iteraciones < 1000) DO
          BEGIN
             a:= (Li+Ls)/2;
             evalua(ev_a,a,func,grado);
             w:= sgn(ev_a);
             if sgn(f) = w then
                Li:= a
             else
                Ls:= a;
             iteraciones:= iteraciones+1;
          end;
          raiz:= A;
       end;
       if (iteraciones < 1000) then
       BEGIN
             raices[ap_raiz].R := raiz;
             raices[ap_raiz].i:= 0;
             ap_raiz:= ap_raiz + 1;
             res[grado-1]:= func[grado];
             indice:=2;
             while indice <=grado do
             BEGIN
                res[grado-indice]:= raiz*res[grado-indice+1]
                 + func[grado-indice+1];
                indice := indice + 1;
             end;
             func:= res;
             grado:= grado-1;
       end
       else
       begin
          converge:=false;
          falla:= true;
       end;
    end
```

```
      else
      begin
            CONVERGE := false;
            k:= 0;
            for j:=0 to grado do
            begin
               func2[grado-j]:= func[k];
               k:=k+1;
            end;
            lin(func2,grado);
      end;
   end;
   if (converge) then res_ec_grad2(func[2], func[1], func[0]);
end;

{*****************************************************************}
{ El siguiente procedimiento realiza la división de un polinomio  }
{ entre un número                                                 }
{*****************************************************************}

procedure transfer.divide(var numerador:vector_r;
                          var grado:integer;var t:real);

VAR
   num,den,res,num2,den2,num3,den3: vec_20;
   grado_num, grado_den,indice,grado1,grado2,i,j,elementos:integer;

BEGIN
   grado1:= grado_n[1];
   for i:=0 to grado1 DO
      num[i]:= coef_n[grado1-i+1];
   indice:= grado1+2;
   for j:=2 to num_poln DO
   BEGIN
      grado2:= grado_n[j];
      for i:= 0 to grado2 DO
         num2[i]:= coef_n[indice+grado2-i];
      multiplica(num,num2,res,grado1,grado2);
      grado1:= grado1+grado2;
      num:=res;
      indice:= indice+grado1+1;
   end;
   grado_num:=grado1;
   grado1:= grado_d[1];
   for i:= 0 to grado1 DO
      den[i]:= coef_d[grado1-i+1];
   indice:= grado1+2;
   for j:= 2 to num_pold DO
   BEGIN
      grado2:= grado_d[j];
      for i:= 0 to grado2 DO
         den2[i]:= coef_d[indice+grado2-i];
      multiplica(den,den2,res,grado1,grado2);
      grado1:= grado1+grado2;
      den:=res;
      indice:= indice+grado2+1;
   end;
   grado_den:= grado1;
```

```
    for i:= 0 to 20 do
    begin
       num3[i]:=0;
       den3[i]:=0;
    end;
    for i:= 0 to grado_num DO
       num3[i]:= num[grado_num-i];

    for i:= 0 to grado_den DO
       den3[i]:=  den[grado_den-i];
    t:= num3[0]/den3[0];
    for j:= 1 to grado_den DO
       num3[j-1]:= num3[j]-t*den3[j];
    grado_num:= grado_num-1;
    while (num3[0]=0) and (grado_num>0) DO
    begin
       grado_num:=grado_num-1;
       for i:=0 to grado_num DO
          num3[i]:= num3[i+1];
    end;
    for i:= 0 to grado_num DO
       numerador[i+1]:=num3[i];
    grado:= grado_num;
end;

procedure transfer.suma_quebrados(mat1,mat2: arreglo2; apuntador:integer;
                                  var vec: vec_20; var mayor:integer);
var
    v1,v2,res: vec_20;
    i,j,k,grad_vec,grado1,grado2:integer;

begin
    for i:= 0 to 20 do
       vec[i]:=0;
    mayor:=0;
    FOR i:= 1 to apuntador-1 DO
    BEGIN
       v1:= mat2[i].pol;
       grado1:= mat2[i].grado;
       if apuntador>2 then
       BEGIN
          j:=1;
          while j <= (  apuntador -1 ) DO
          BEGIN
             if i=j THEN j:= j+1;
             if j<=(apuntador-1) THEN
             BEGIN
                v2:= mat1[j].pol;
                grado2:=mat1[j].grado;
                multiplica(v1,v2,res,grado1,grado2);
                v1:= res;
                grado1:=grado1+grado2;
                if grado1>mayor then
                   mayor:= grado1;
             end;
             j:=j+1;
          end;
          for k:= 0 to 20 do
```

```
            vec[k]:= v1[k]+vec[k];
          grad_vec:= mayor;
       end
       else
       BEGIN
          mayor:= grado1;
          vec:=v1;
       end;
    end;
end;

(*************************************************************)
( Para un polinomio de grado menor o igual que dos, el  siguiente  )
( método coloca los polos de la funcion en el arreglo raices )
( Una vez que encontro todos los polos los clasifica en clases:)
(clase 1 en caso de que se trate de raiz real no repetida. )
( Clase 2 para raices reales repetidas. )
( Clase 3 para las raices complejas )
(*************************************************************)

procedure transfer.encuentra_raiz;
VAR
   i,k,m,j,comp_rep
   :INTEGER;
   delta,a,b,c,diferencia
   :REAL;
   registro: REG;

BEGIN
   ap_raiz:=1;
   falla:= false;
   FOR i:= 1 to num_pold DO
   BEGIN
      IF grado_d[i]=1 THEN
      BEGIN
         mapea(k,i);
         raices[ap_raiz].R:= -coef_d[k+1];
         raices[ap_raiz].I:= 0;
         ap_raiz:= ap_raiz + 1;
      end
      else
      if grado_d[i]=2 THEN
      BEGIN
         mapea(k,i);
         a:= coef_d[k];
         b:= coef_d[k+1];
         c:= coef_d[k+2];
         res_ec_grad2(a,b,c);
      end;
      if grado_d[i] <> 2 THEN
         biseccion(grado_d[i],i);
   end;
   if (not(falla)) THEN
   BEGIN
      i:= 0;
      while i<(ap_raiz-1) DO
      BEGIN
         i:= i+1;
```

```
   registro:= raices[i];
   while ((registro.r = raices[i+1].r)
         and (registro.i = raices[i+1].i) and (i<(ap_raiz-1))) DO
      begin i:= i+1;  end;
   j:= i+1;
   while j<=(ap_raiz-1) DO
   BEGIN
      IF( (abs(registro.r-raices[j].r)<0.002)
      and (abs(registro.i-raices[j].i)<0.002))
      THEN
      BEGIN
         M:= j-1;
         WHILE m <> i DO
         BEGIN
            raices[m+1]:= raices[m];
            m:= m-1;
         end;
         raices[i+1]:=raices[i];
         i:= i+1;
      end;
   j:= j+1;
   end;
end;
for i:=1 to ap_raiz-1 do
begin
   if ((raices[i].r >=-0.0005) and (raices[i].r<=0.0005)) then
      raices[i].r := 0;
end;
i:= 1;
falla:= false;
while i<=(ap_raiz-1) DO
BEGIN
   registro:= raices[i];
   IF ((registro.r=raices[i+1].r) and
      (registro.i = raices[i+1].i) and  (i< (ap_raiz-1)) ) THEN
   BEGIN
      IF (raices[i].tipo=3) THEN
      BEGIN
         raices[i].tipo:=4;
         comp_rep:=1;
         i:= i+1;
         WHILE ((registro.r=raices[i].r)
               and (registro.i = raices[i].i) and
                  (raices[i].tipo = 3)) DO
         BEGIN
            raices[i].tipo := 4;
            comp_rep:= comp_rep+1;
            if comp_rep>2 then falla:= true;
            i:= i+1;
         end;
      end
      else
      BEGIN
         raices[i].tipo:=2;
         i:=i+1;
         while ((registro.r=raices[i].r)
         and (registro.i=raices[i].i) and (i<ap_raiz)) DO
         BEGIN
```

```
                    raices[i].tipo:=2;
                    i:=i+1;
                end;
            end;
        end
        else
        BEGIN
            if raices[i].i=0 then
                raices[i].tipo:=1;
            i:=i+1;
        end;
    end;
end;

{*************************************************************************}
( El metodo transfer.encuentra_ec tiene por objetivo formar un sistema)
( de ecuaciones para encontrar el valor de las variables que resultan )
( al aplicar el metodo de desarrollo en fracciones parciales)
{*************************************************************************}

Procedure transfer.encuentra_ec (conjunto: con;
                                 i, num_ra: INTEGER;
                                 VAR mat: mat_20;
                                 col: INTEGER);

VAR
    j,grado1,grado2,grado_acum,k,cont_ra,m: INTEGER;
    v1,v2,res: vec_20;

BEGIN
    j:= 1;
    WHILE ( j in conjunto ) DO
        j:= j+1;
    IF (i=j) THEN j:= j+1;
    if (j>= ap_raiz) then
    begin
        v1[0]:= 1;
        grado_acum:= 0;
    end
    else
    begin
        IF (( raices[j].tipo = 1) or (raices[j].tipo =2) )THEN
        BEGIN
            v1[0] := -(raices[j].R);
            v1[1] := 1;
            grado1:= 1;
            grado_acum:=1;
            res:=v1;
        end;
        if (raices[j].tipo = 3) or (raices[j].tipo = 4) then
        begin
            v1[2]:=1;
            v1[1]:= -2*raices[j].r;
            v1[0]:= raices[j].r*raices[j].r +raices[j].i*raices[j].i;
            grado1:=2;
            grado_acum:=2;
        end;
```

```
      if j<> num_ra then
      begin
         cont_ra := num_ra;
         WHILE ((cont_ra-2 > 0) and (j<num_ra)) DO
         BEGIN
            WHILE (( j+1) IN conjunto) DO
               j:= j+1;
            IF (( j+1)=1) THEN
               j:= j+1;
            if(j<num_ra) then
            begin
               IF (raices[j+1].tipo=1) or ( raices[j+1].tipo = 2) THEN
               BEGIN
                  v2[0]:= -(raices[j+1].R);
                  v2[1]:= 1;
                  m:= 1;
                  grado2:= 1;
               end;
               IF (raices[j+1].i<>0) then
               BEGIN
                  grado2:= 2;
                  v2[2]:= 1;
                  v2[1]:= -2*raices[j+1].r;
                  v2[0]:= raices[j+1].r*raices[j+1].r
                  + raices[j+1].i*raices[j+1].i;
               end;
               multiplica (v1, v2, res, grado1, grado2);
               j:= j+1;
               grado_acum:= grado1 + grado2;
               v1:= res;
               grado1:= grado_acum;
            end;
            cont_ra:= cont_ra-1;
         end;
      end
      else
      begin
         res:=v1;
         grado_acum:= grado1;
      end;
   end;
   res:=v1;
   FOR j:= 1 to (grado_acum+1) DO
   mat[j,col]:= res[j-1];
end;


procedure transfer.imprime(coef:real; j:integer);
var
   part_ent:integer;

begin
   part_ent:= trunc(coef);
   if (j<>1) and (coef>0)
   then write('+');
   if coef <> 0 then
   begin
      if (abs(coef)<1.0004) and (abs(coef)>=0.9996) then
```

```
      begin
         if coef<0 then write ('-');
      end
      else
      begin
         if (part_ent-coef)=0 then write (part_ent)
         else
         begin
            if (abs(coef)<0.001) then write (coef:6)
            else
               write (coef:6:3);
         end;
      end;
   end;
end;

(*****************************************************************)
{ El siguiente metodo imprime el valor de la transformada de Laplace }
{ inversa }
(*****************************************************************)

procedure transfer.trans_inv(mat: mat_20; opcion:integer);
VAR
   coef,con1,con2,con3,coef2,valor_abs,b,c,d,c2,c3,ra: real;
   ent,ent2,aux,comp_rep,cont,fac,i,j ,part_ent,imp, n,rep:integer;

BEGIN
   write('y(t)= ');
   dim_mat(n);
   imp:=0;
   aux:=0;
   for i:= 1 to n do
      if mat[i,n+1]<>0 then aux:=1;
   if (aux=0) and (cons2=0) then
      writeln('0')
   else
   begin
   j:=1;
   comp_rep:= 0;
   for i:= 1 to aP_raiz-1 DO
   BEGIN
      if raices[i].tipo = 1 then
      begin
         if (raices[i].r>=0) and ((opcion=5) or (opcion<=3))
         or (raices[i].r<0) and ((opcion=4) or (opcion<=3)) then
         begin
            coef:= mat[j,n+1]/cte;  { En coef se guarda el valor
                                      de la constante A, }
            imprime(coef,j); { del tipo A/(s-r)}

            if coef<>0 then
            begin
               imp:=1;
               if (raices[i].R <> 0)  then
               begin
                  part_ent:= trunc(raices[i].r);
                  if not( (abs(raices[i].r)>= 0.9996)
                  and (abs(raices[i].r)<=1.0004)) then
```

```
            begin
               if (abs(part_ent-raices[i].r)<=0.0004) then
                  write(' exp( ',part_ent, 't) ')
               else
               begin
                  aux:= round(raices[i].r);
                  if (abs(aux-raices[i].r)<0.0001) then
                  begin
                     if abs(aux)=1 then
                     begin
                        if aux = 1 then write('exp(t)');
                        if aux = -1 then write('exp(-t)');
                     end
                     else
                     if aux<>0 then
                        write('exp(',aux,'t)');
                  end
                  else
                  begin
                     if(abs(raices[i].r)<0.001 )then
                        write('exp(',raices[i].r:6,'t)')
                     else
                        write('exp(',raices[i].r:6:3,'t)');
                  end;
               end;
            end
            else
            begin
               if raices[i].r > 0 then
                  write(' exp(t)');
               if raices[i].r < 0 then
                  write (' exp(-t)');
            end;
         end
         else
         if (abs(coef)<1.0004) and (abs(coef)>=0.9996) then
         write('1');
      end;
   end;
   j:= j+1;
end
else
if raices[i].tipo = 2 then
begin
   if (raices[i].r>=0) and ((opcion=5) or(opcion<=3))
    or (raices[i].r<0) and ((opcion=4) or (opcion<=3)) then
   begin
      cont:=0;
      raicez_rep(rep,i);
      fac:= 1;
      while rep>0 DO
      BEGIN
         coef:= mat[j,n+1]/(cte*fac);  { En coef se guarda el valor
                                         de la constante B}
         if (j<>1) and (coef>0) then write ('+');   (      n )
         if coef <> 0 then              (  del tipo B/(s-r) )
         begin
            imp:=1;
```

```
if (abs(coef)<1.0009) and ( abs(coef) >=1 ) then
begin
   if coef<0 then write ('-');
end
else
begin
   part_ent:= trunc(coef);
   if (part_ent-coef)=0 then
      write(part_ent)
   else
   begin
      aux:= round(coef);
      if (abs(aux-coef)<0.0001) then
      begin
         if aux<>0 then
            write(aux);
      end
      else
      begin
         if(abs(coef)<0.001 )then
            write(coef:6)
         else
            write(coef:6:3);
      end;
   end;
end;
if cont <> 0 then write ('t');
if cont>1 then write('**',cont);
if raices[i].R <> 0 then
begin
   part_ent:= trunc(raices[i].r);
   if not( (abs(raices[i].r)>= 0.9996)
   and (abs(raices[i].r)<=1.0004)) then
   begin
      if (abs(part_ent-raices[i].r)<=0.0004) then
         write('exp(',part_ent,'t)')
      else
      begin
         aux:= round(raices[i].r);
         if (abs(aux-raices[i].r)<0.0001) then
         begin
            if abs(aux)=1 then
            begin
               if aux = 1 then write('exp(t)');
               if aux = -1 then write('exp(-t)');
            end
            else
            if aux<>0 then
               write('exp(',aux,'t)');
         end
         else
         begin
            if(abs(raices[i].r)<0.001 )then
               write('exp(',raices[i].r:6,'t)')
            else
               write('exp(',raices[i].r:6:3,'t)');
         end;
      end;
```

```
                end
                else
                begin
                   if (raices[i].r>0) then
                      write('exp(t)');
                   if (raices[i].r<0) then
                      write('exp(-t)');
                end;
             end
             else
             if (cont=0) and (abs(coef)<1.0004) and
                (abs(coef)>=0.9996) then write('1');
          end;
          i:= i+1;
          rep:= rep-1;
          cont:= cont+1;
          fac:= fac*cont;
          j:= j+1;
       end;
       i:=i-1;
    end
    else
    begin
       raices_rep(rep,i);
       i:= i+rep-1;
       j:=j+rep;
    end;
end;
if (raices[i].tipo = 3) or ((raices[i].tipo=4) and (comp_rep = 0))
then
begin       ( Si se tiene un termino del tipo (Cs+D)/(s*s + A*s +B) :
    if raices[i].tipo =4 then
       comp_rep:= comp_rep+1;
    if (raices[i].r=0) and (opcion=4) then  j:=j+2
    else
    if (raices[i].r=0) and ((opcion=5)or(opcion<=3)) then
    begin
       coef:= mat[j,n+1]/cte;
       coef:= coef/raices[i].i;
       if raices[i].i<0 then coef:=-coef;
       if (j>1) and (coef>0) then write('+');
       part_ent:= trunc(coef);
       if coef<0 then
       begin
          imp:=1;
          if (abs(coef -1.0009) and ( abs(coef) >=1 ) then
          begin
             if coef<0 then write ('-');
          end
          else
          begin
             if (part_ent-coef)=0 then
                write (part_ent)
             else
             begin
                if (abs(coef)<0.001) then write(coef:6)
                else write (coef:6:3);
             end;
```

```
    end;
    if (abs(raices[i].i) >= 0.99) and (abs(raices[i].i)<1.0009
    then
       write('sent')
    else
    begin
       part_ent:= trunc(raices[i].i);
       if (abs(part_ent-raices[i].i)<=0.0004) then
          write ('sen',part_ent,'t')
       else
       begin
          aux:= round(raices[i].i);
          if aux<0 then aux:=-aux;
          if (abs(aux-raices[i].i)<0.0001) then
          begin
             if (abs(aux)=1) then
                write('sen t')
             else
                write('sen(',aux,'t)');
          end
          else
          begin
             valor_abs:= abs(raices[i].i);
             if(abs(raices[i].i)<0.001 )then
                write('sen(',valor_abs:6,'t)')
             else
                write('sen(',valor_abs:6:3,'t)');
          end;
       end;
    end;
end;
j:= j+1;
coef:= mat[j,n+1]/cte;
if coef>0 then write ('+');
part_ent:= trunc (coef);
if coef <> 0 then
begin
    imp:=1;
    if (abs(coef)<1.0009) and ( abs(coef) >=1 ) then
    begin
       if coef<0 then write ('-');
    end
    else
    begin
       if  (part_ent-coef)=0 then
          write (part_ent)
       else
       begin
          if (abs(coef)<0.001) then write(coef:6)
          else write (coef:6:3);
       end;
    end;
    if (abs(raices[i].i)>=0.9996 ) and (abs(raices[i].i)<=1.0004
    then
       write ('cost')
    else
    begin
       part_ent:= trunc(raices[i].i);
```

```
        if part_ent<0 then part_ent:=-part_ent;
        if (abs(part_ent-raices[i].i)<=0.0004) then
           write ('cos',part_ent,'t')
        else
        begin
           valor_abs:= abs(raices[i].i);
           write ('cos',valor_abs:6:3,'t');
        end;
     end;
  end;
  j:= j+1;
end
else
if raices[i].r<>0 then
begin
  b:= -2*raices[i].r;
  c:= raices[i].r*raices[i].r+raices[i].i*raices[i].i;
  c2:= b/2;
  c3:=-c2;
  if ((c3>=0) and((opcion=5)or(opcion<=3))
  or (c3<0) and ((opcion=4)or(opcion<=3)))
  then
  begin
     d:= c-c2*c2;
     ra:= sqrt(abs(d));
     coef:= (mat[j,n+1]-mat[j+1,n+1]*c2)/(ra*cte);
     if (j>1) and(coef>0) then
        write ('+');
     if (coef <> 0) then
     begin
        imp:=1;
        if (abs(coef)>=0.9996) and (abs(coef)<=1.0004) then
        begin
           if coef < 0 then
              write('-exp(');
           if coef > 0 then
              write('exp(');
        end
        else
        begin
           part_ent:= trunc(coef);
           if (part_ent-coef)=0 then
              write(part_ent,'exp(')
           else
           begin
              if (abs(coef)<0.001) then write(coef:6,'exp(')
              else write(coef:6:3,'exp(');
           end;
        end;
        c3:= -c2;
        part_ent:= trunc(c3);
        if (abs(c3)<> 1 then
        begin
           if (abs(part_ent-c3)<=0.0004) then write(part_ent,'t)')
           else
           write(c3:6:3,'t)');
        end;
        if c2 = 1 then write('t)');
```

```
    if c3= -1 then write('-t)');
    part_ent:= trunc(ra);
    if d>0 then
    begin
       if (abs(part_ent-ra)<=0.0004) then
       begin
          if part_ent<0 then part_ent:= - part_ent;
          if abs(part_ent) = 1 then write ('cent')
          else
             write ('sen',part_ent,'t');
       end
       else write('sen',ra:6:3,'t');
    end
end;
c3:= -c2;
j:= j+1;
coef:= mat[j,n+1]/cte;
if coef > 0 then write ('+');
if coef<>0 then
begin
    imp:=1;
    if (abs(coef)>=0.9996) and (abs(coef)<=1.0004) then
    begin
       if (coef > 0)  then
          write('exp(');
       if coef < 0 then
          write('-exp(');
    end
    else
    begin
       part_ent:= trunc(coef);
       if (part_ent-coef)= 0 then write(part_ent,'exp(')
       else
       begin
          if(abs(coef)<0.001) then write (coef:6,'exp(')
          else  write (coef:6:3,'exp(');
       end;
    end;
    if abs(c3)<>1 then
    begin
       part_ent:= trunc(c3);
       if (abs(part_ent-c3)<= 0.0004) then write(part_ent,'t)'
       else write(c3:6:3,'t)');
    end;
    if c3 = 1 then write('t)');
    if c3 = -1 then write ('-t)');
    part_ent:= trunc(ra);
    if d>0 then
    begin
       if (abs (ra)>=0.9996) and (abs(ra)<=1.0004) then
          write ('cost')
       else
       begin
          if (abs(part_ent-ra)<=0.0004) then
             write('cos',part_ent,'t')
          else write('cos',ra:6:3,'t');
       end;
    end
```

```
            end;
            j:= j+1;
         end
         else
            j:=j+2;
      end
   end
   else
   if raices[i].tipo = 4 then
   begin
      comp_rep:=0;
      coef2:= mat[j+1,n+1]/cte;
      b:= -2*raices[i].r;
      c:= raices[i].r*raices[i].r+raices[i].i*raices[i].i;
      c2:= b/2;
      c3:= -c2;
      d:= c-c2*c2;
      ra:= sqrt(abs(d));
      coef:= (mat[j,n+1]-mat[j+1,n+1]*c2)/(ra*cte);
      con1:= coef/(2*ra*ra);
      con2:= -coef/(2*ra);
      con3:= coef2/(ra*2);
      if ((c3<0) and (opcion<>5)) or ((c3>=0) and (opcion<>4)) then
      begin
         imp:=1;
         if c3<>0 then
         begin
            part_ent:= trunc(c3);
            if (abs(c3))<>1 then
            begin
               if (abs(part_ent-c3)<=0.0004) then
                  write('+exp(',part_ent,'t)')
               else
               write('+exp(',c3:6:3,'t)');
            end;
            if c3 = 1 then write('+exp(t)');
            if c3 = -1 then write ('+exp(-t)');
         end
         else
            write('+');
         write('(');
         if con1<>0 then
         begin
            write(con1:6:3);
            aux:= round(ra);
            if (abs(aux-ra)<0.0001) then
            begin
               if (abs(aux)>=0.9996) and (abs(aux)<=1.0004) then
                  write('sen t')
               else
               if aux<>0 then
                  write('sen(',aux,'t)');
            end
            else
            begin
               if(abs(ra)<0.001 )then
                  write('sen(',ra:6,'t)')
               else
```

```
                    write('sen(',ra:6:3,'t)');
                end;
            end;
            if con2<>0 then
            begin
                if con2>0 then write('+');
                if (abs(con2)<0.001) then write (con2:6)
                else
                    write(con2:6:3);
                write(' t cos',ra:6:3,'t');
            end;
            if con3<>0 then
            begin
                if (con3>0) then write('+');
                if (abs(con3)<0.001) then write (con3:6)
                else
                    write(con3:6:3);
                write('t sen');
                aux:= round(ra);
                if (abs(aux-ra)<0.0001) then
                begin
                    if abs(aux)=1 then
                    begin
                        if aux = 1 then write('t');
                        if aux = -1 then write('-t');
                    end
                    else
                    if aux<>0 then
                        write(aux,'t)');
                end
                else
                begin
                    if(abs(ra)<0.001 )then
                    write(ra:6,'t)')
                    else
                        write(ra:6:3,'t)');
                end;
            end;
            if (con1=0) and (con2=0) and (con3=0) then write ('0');
            write(')');
        end;
        j:= j+2
    end;
end;
end;
if (cons2<>0) and ((opcion=4)or(opcion=3)) then
begin
    imprime(cons2,2);
    write('δ(t)');
    imp:=1;
end;
if (imp=0) and (opcion=3) then write('0');
writeln;
writeln;
end;

{**********************************************************************
{ El metodo transfer_frac_par aplica el metodo de desarrollo en
```

```
{ fracciones parciales a una funcion }
{*************************************************************}

procedure transfer.frac_par(var inestable:boolean;
                            vec:vec_20; opcion:integer);
VAR
   terminos,indice,i,j,grado1,grado2, grado_acum,
   num_ra, repetidas, n, col,op
   : INTEGER;
   termina: char;
   conj: con;
   mat,mat2: mat_20;
   v1,v2,res: vec_20;
   r:real;

BEGIN
   if falla= true then
   writeln(' Falla: El sistema sale fuera de los alcances de este programa')
   else
   begin
      num_ra:= ap_raiz-1;
      FOR i:= 1 to 20 DO
         FOR j:= 1 to 20 DO
            mat[i,j]:=0;
      dim_mat(N);
      i:=0; { A continuacion se procede a establecer un sistema
      de ecuaciones, en donde se comparan los coeficientes de las }
      FOR col:= 1 to n DO  { potencias de s      }
      BEGIN
         i:=i+1;
         IF (raices[i].tipo=1) THEN
         BEGIN
            conj:= [];
            encuentra_ec(conj,i,num_ra,mat,col);
         end;
         IF (raices[i].tipo = 2) THEN
         BEGIN
            raices_rep (repetidas,i);
            conj:= [];
            while repetidas>0 DO
            BEGIN
               encuentra_ec(conj,i,num_ra,mat,col);
               conj:= conj + [i];
               i:= i+1;
               col:= col+1;
               repetidas:= repetidas-1;
            end;
            i:= i-1;
            col:= col-1;
         end;
         IF (raices[i].tipo = 3) THEN
         BEGIN
            conj:= [];
            encuentra_ec (conj,i,num_ra,mat,col);
            col:= col+1;
            FOR j:= 1 to N-1 DO begin
               mat[j+1,col]:= mat[j,col-1]; end;
         end;
```

```
    if (raices[i].tipo= 4) then
    begin
       conj:=[];
       for repetidas:=1 to 2 DO
       BEGIN
          encuentra_ec(conj,i,num_ra,mat,col);
          col:= col+1;
          conj:= conj+[i];
          FOR j:= 1 to n-1 DO
             mat[j+1,col]:= mat[j,col-1];
          col:= col+1;
          i:= i+1;
       end;
       i:= i-1;
       col:= col-1;
    end;
end;
i:= grado_n[1] + 1;
j:= 0;
while (j<i) DO
BEGIN
   v1[j]:= coef_n[i-j];
   j:= j+1;
end;
res:=v1;
indice:= i;
grado1:= grado_n[1];
terminos:= num_poln -1;
while (terminos > 0 ) DO
BEGIN
   j:= 0;
   i:= grado_n[num_poln-terminos+1];
   i:= indice + i + 1;
   while (j+indice) < i DO
   BEGIN
      v2[j]:= coef_n[i-j];
      j:= j+1;
   end;
   grado2:=grado_n[num_poln-terminos+1];
   multiplica(v1,v2,res,grado1,grado2);
   v1:= res;
   grado1:= grado1 + grado2;
   terminos:= terminos - 1;
   indice := i;
end;
FOR j:= 1 to grado1+1 DO
   mat[j,N+1]:= res[j-1];
mat2:= mat;
res_ec(mat,n);
if (opcion=1) or (opcion=3) then
begin
   i:=1;
   for j:=1 to ap_raiz-1 DO
   begin
      if raices[j].r >0.0004 then
      begin
         if (raices[j].tipo=1) or (raices[j].tipo=2) then
         begin
```

```
            if mat[i,n+1]<>0 then inestable:= true;
            i:=i+1;
         end
         else
         begin
            if not( (mat[i,n+1]=0) and (mat[i+1,n+1]=0) ) then
               inestable:=true;
            i:=i+2;
         end;
      end;
   end;
end;
if (opcion=6) or (opcion=8) then
begin
   i:=1;
   for j:=1 to ap_raiz-1 DO
   begin
      r:= sqrt(raices[j].r*raices[j].r + raices[j].i*raices[j].i);
      if r>1.0004 then
      begin
         if (raices[j].tipo=1) or (raices[j].tipo=2) then
         begin
            if mat[i,n+1]<>0 then inestable:= true;
         end
         else
            if not( (mat[i,n+1]=0) and (mat[i+1,n+1]=0) ) then
               inestable:=true;
      end;
      if (raices[j].tipo=1) or (raices[j].tipo=2) then i:= i+1
      else i:= i+2;
   end;
end;
if falla= false then trans_inv(mat,opcion);
if falla= false then
begin
   repeat
      falla:= false;
      termina:= 'n';
      valor_num(mat,opcion);
      if falla=true then
      begin
         write ('Desea dar otro valor para t? (s/n)');
         readln(termina);
         if termina='n' then falla:=false;
      end;
   until((falla=false) or (termina='n'));
   if (opcion = 2) or (opcion=7) then
   begin
      writeln('Respuesta total: ');
      for i:=0 to n DO
         mat2[i+1,n+1]:= mat2[i+1,n+1]+vec[i];
      res_ec(mat2,n);
      if opcion=2 then op:=3
      else op:=8;
      if falla = false then trans_inv(mat2,op);
      repeat
         falla:= false;
         termina:= 'n';
```

```
      if opcion=2 then
         valor_num(mat2,3)
      else
         valor_num(mat2,8);
      if falla=true then
      begin
         write ('Desea dar otro valor para t? (s/n)');
         readln(termina);
         if termina='n' then falla:=false;
      end;
   until((falla=false) or (termina='n'));
end;
if (inestable) and (opcion<>1) and (opcion<>6) then
writeln('No se puede obtener la respuesta transitoria,
      ya que el sistema es inestable')
else
begin
   if (opcion<>1)  then
   begin
      res_ec(mat2,n);
      termina:= 's';
      if (opcion=3) or (opcion=2) then
      begin
         if cons2<>0 then termina:='n';
         i:= 1;
         for j:=1 to ap_raiz-1 DO
         begin
            if raices[j].r<0 then
            begin
               if (raices[j].tipo=1) or (raices[j].tipo=2) then
               begin
                  if mat2[i,n+1]<>0 then
                     termina:='n';
               end
               else
               if not((mat2[i,n+1]=0) and (mat2[i+1,n+1]=0)) then
                  termina:='n';
            end;
            if (raices[j].tipo=1) or (raices[j].tipo=2) then
               i:=i+1
            else i:=i+2;
         end;
      end;
      if (opcion=8) or (opcion=7) then
      begin
         i:=1;
         for j:= 1 to ap_raiz-1 DO
         begin
            r:= sqrt(raices[j].r*raices[j].r +
            raices[j].i*raices[j].i);
            if r<1 then
            begin
               if (raices[j].tipo=1) or (raices[j].tipo=2) then
               begin
                  if mat2[i,n+1]<>0 then
                     termina:='n';
               end
               else
```

```
                    if not((mat2[i,n+1]=0) and (mat2[i+1,n+1]=0)) then
                       termina:='n';
                 end;
                 if (raices[j].tipo=1) or (raices[j].tipo=2) then
                  i:=i+1
                 else i:=i+2;
              end;
           end;
           if termina='n' then
           begin
              if opcion<6 then op:=4
              else op:=9;
              writeln('Respuesta transitoria: ');
              trans_inv(mat2,op);
              repeat
                 falla:= false;
                 termina:= 'n';
                 valor_num(mat2,op);
                 if falla=true then
                 begin
                    write ('Desea dar otro valor para t? (s/n)');
                    readln(termina);
                    if termina='n' then falla:=false;
                 end;
              until((falla=false) or (termina='n'));
              writeln('Respuesta permanente: ');
              if opcion<6 then op:=5
              else op:=10;
              trans_inv(mat2,op);
              repeat
                 falla:= false;
                 termina:= 'n';
                 valor_num(mat2,op);
                 if falla=true then
                 begin
                    write ('Desea dar otro valor para t? (s/n)');
                    readln(termina);
                    if termina='n' then falla:=false;
                 end;
              until((falla=false) or (termina='n'));
           end;
        end;
     end;
  end;
end;

(*************************************************************)
{ El siguiente metodo deja en elev el valor que resulte al elevar t  }
{ al valor contenido en cont }
(*************************************************************)

procedure transfer.elava(t:real; cont:integer; var elev:real);
var
   i:integer;
begin
   elev:=1;
   for i:= 1 to cont do
```

```
      elev:= elev*t;
end;

{*****************************************************************}
{ El siguiente procedimiento calcula en forma numerica el valor de la}
{ respuesta en el tiempo }
{*****************************************************************}

procedure transfer.valor_num(mat:mat_20; opcion:integer);
var
   j,i,fac,comp_rep,cont,rep, indice,n, num_puntos:integer;
   func: array[0..501] of real;
   aux,t,coef, coef2,con1,con2,con3,a,b,c,c2,c3,d,ex,ra,Ls,
   incremento,elev : real;

begin
   if (genera='s') or (genera='S') then
   begin
   falla:= false;
   comp_rep:=0;
   writeln;
   write( 'Hasta que valor del tiempo desea conocer la respuesta
   del sistema? ');
   readln(Ls);
   for indice:= 1 to 501 do
      func[indice]:=0;
   num_puntos:= 500;
   incremento := Ls/num_puntos;
   Ls:= Ls+incremento;
   t:= 0;
   j:=1;
   dim_mat(n);
   aux:=0;
   for i:= 1 to n do
      if mat[i,n+1]<>0 then aux:=1;
   if aux=1 then
   begin
   for i:= 1 to ap_raiz-1 do
   begin
      if raices[i].tipo=1 then
      begin
         if (raices[i].r=0) and ((opcion=5) or (opcion<=3)) or
         (raices[i].r<>0) and ((opcion=4) or (opcion<=3)) then
         begin
            coef:= mat[j,n+1]/cte;
            if coef <> 0 then
            begin
               ex:= raices[i].r;
               t:=0;
               indice:= 1;
               while t<=ls do
               begin
                  aux:= ex*t;
                  if abs(aux)>=88 then
                  begin
                     falla:=true;
                     t:=ls+1;
                     i:= ap_raiz-1;
```

```
                writeln('Valor de t demasiado grande');
            end
            else
            begin
               func[indice]:= func[indice] + coef * exp(ex*t);
               t:= t+ incremento;
               indice:= indice+1;
            end;
         end;
      end;
   end;
   j:=j+1;
end;
if (raices[i].tipo = 2) and (not(falla)) then
begin
 if (raices[i].r=0) and ((opcion=5) or (opcion<=3)) or
 (raices[i].r<>0) and ((opcion=4) or (opcion<=3)) then
 begin
   cont:= 0;
   fac:=1;
   raices_rep(rep,i);
   ex:= raices[i].r;
   while (rep>0) do
   begin
      coef:= mat[j,n+1]/(fac*cte);
      indice:= 1;
      t:= 0;
      while t<=ls do
      BEGIN
         aux:= ex*t;
         if abs(aux)>=88 then
         begin
            falla:=true;
            rep:=0;
            t:=ls+1;
            i:= ap_raiz-1;
            writeln('Valor de t demasiado grande');
         end
         else
         begin
            eleva(t,cont,elev);
            func[indice]:= func[indice]+coef*(elev)*exp(ex*t);
            t:= t+incremento;
            indice:= indice+1;
         end;
      end;
      rep:= rep-1;
      j:= j+1;
      i:= i+1;
      cont:= cont+1;
      fac:= fac*cont;
   end;
 i:= i-1;
 end
 else
 begin
    raices_rep(rep,i);
    i:= i+rep-1;
```

```
      j:=j+rep;
   end;
end;
if ((raices[i].tipo = 3) or ((raices[i].tipo=4) and
(comp_rep=0))) and (not(falla)) then
begin
   if (raices[i].tipo=4) then comp_rep:= comp_rep+1;
   if (raices[i].r=0) and (opcion=4) then  j:=j+2
   else
   if (raices[i].r=0) and ((opcion=5)or(opcion<=3)) then
   begin
      coef:= mat[j,n+1]/(cte*raices[i].i);
      a:= raices[i].i;
      t:= 0;
      indice:= 1;
       while t<=ls do
       begin
          func[indice]:= func[indice] + coef * sin(a*t);
          t:= t+incremento;
          indice:= indice+1;
       end;
       j:= j+1;
       coef:= mat[j,n+1]/cte;
       t:= 0;
       indice:= 1;
       while t<=Ls do
       begin
          func[indice]:= func[indice]+coef*cos(a*t);
          t:= t+incremento;
          indice:= indice+1;
       end;
       j:= j+1;
   end
   else
   if raices[i].r<>0 then
   BEGIN
      b:= -2*raices[i].r;
      c:= raices[i].r*raices[i].r + raices[i].i*raices[i].i;
      c2:= b/2;
      c3:= -c2;
      if ((c3=0) and((opcion=5)or(opcion<=3)) or
      (c3<>0) and ((opcion=4)or(opcion<=3))) then
      begin
         d:= c-c2*c2;
         ra:= sqrt(abs(d));
         coef:= (mat[j,n+1]-mat[j+1,n+1]*c2)/(ra*cte);
         if d>0 then
         begin
            t:= 0;
            indice:= 1;
            while t<=Ls DO
            BEGIN
               aux:= c3*t;
               if abs(aux)>=88 then
               begin
                  falla:=true;
                  t:=ls+1;
                  i:= ap_raiz-1;
```

```
                writeln('Valor de t demasiado grande');
            end
            else
            begin
                func[indice]:= func[indice]+
                (coef*exp(c3*t))*sin(ra*t);
                t:= t+incremento;
                indice:= indice+1;
            end;
        end;
    end
    else
    begin
        t:=0;
        indice:= 1;
        while t<=Ls DO
        BEGIN
            aux:= c3*t;
            if abs(aux)>=88 then
            begin
                falla:=true;
                t:=ls+1;
                i:= ap_raiz-1;
                writeln('Valor de t demasiado grande');
            end
        end;
    end;
    j:= j+1;
    coef:= mat[j,n+1]/cte;
    if (d>0) and (not(falla)) then
    begin
        t:=0;
        indice:=1;
        while t<=Ls DO
        BEGIN
            aux:= c3*t;
            if abs(aux)>=88 then
            begin
                falla:=true;
                t:=ls+1;
                i:= ap_raiz-1;
            end
            else
            begin
                func[indice]:= func[indice]+(coef*exp(c3*t))*
                cos(ra*t);
                t:= t+ incremento;
                indice:= indice+1;
            end;
        end;
    end;
    j:= j+1;
  end
  else
    j:=j+2;
 end;
end
else
```

```
    if not(falla) then
    begin
       if raices[i].tipo=4 then
       begin
          comp_rep:=0;
          coef2:= mat[j+1,n+1]/cte;
          b:= -2*raices[i].r;
          c:= raices[i].r*raices[i].r+raices[i].i*raices[i].i;
          c2:= b/2;
          c3:= -c2;
          d:= c-c2*c2;
          ra:= sqrt(abs(d));
          coef:= (mat[j,n+1]-mat[j+1,n+1]*c2)/(ra*cte);
          con1:= coef/(2*ra*ra);
          con2:= -coef/(2*ra);
          con3:= coef2/(ra*2);
          if ((c3<>0) and (opcion<>5)) or ((c3=0) and (opcion<>4))
          then
          begin
             t:=0;
             indice:=1;
             while t<=ls DO
             BEGIN
                aux:= c3*t;
                if abs(aux)>=88 then
                begin
                   falla:=true;
                   t:=ls+1;
                   i:= ap_raiz-1;
                   writeln('Valor de t demasiado grande');
                end
                else
                begin
                   func[indice]:= func[indice]+
                   exp(c3*t)*(con1*sin(ra*t)+
                   con2*t*cos(ra*t)+con3*t*sin(ra*t));
                   t:= t+incremento;
                   indice:= indice+1;
                end;
             end;
          end;
          j:=j+2;
       end;
    end;
end;
end;
if not(falla) then
begin
   t:= 0;
   for i:= 1 to 501 do
   begin
      case opcion of
         1: writeln(f1,t,'      ',func[i]);
         2: writeln(f2,t,'      ',func[i]);
         3: writeln(f3,t,'      ',func[i]);
         4: writeln(f4,t,'      ',func[i]);
         5: writeln(f5,t,'      ',func[i]);
      end;
```

```
        t:= t + incremento;
      end;
   end;
   writeln;
   end;
end;

{*************************************************************}
{ El siguiente metodo calcula el factorial de un numero       }
{*************************************************************}
function transfer.fact(n:integer):integer;
var
   valor: integer;
begin
   valor:=1;
   while  (n>1) do
   begin
      valor:= valor*n;
      n:= n-1;
   end;
   fact:=valor;
end;


end.
```

```
{ En la siguiente unidad se encuentran los procedimientos necesarios}
{ para calcular la respuesta escalón, la respuesta rampa y la       }
{ respuesta a cualquier señal de entrada introducida por el usuario }
{ en el dominio del tiempo continuo                                 }

unit resp2;

interface
uses respuesta,
     crt,
     tldef,
     menus;
type
   sal_escalon = object(transfer)
      constructor forma_ec_esc(indice:integer);
   end;
   sal_rampa = object(transfer) .
      constructor forma_ec_ram(indice:integer);
   end;
   sal_otros = object(transfer)
      cond_inic: vec_20;
      constructor inicializa;
      procedure men(VAR respuesta:integer);
      procedure limpia(x,y,renglones:integer);
      procedure despliega(var respuesta:integer);
      procedure desp_func(respuesta:integer;var v1,num: vec_20;
                                    var grado1,grado_num:integer;
                                    var mat1,mat2:arreglo2;
                                    var apuntador,ap2:integer);
      procedure lee(var indice,indice2:integer; var vector1:vec_20;
arr:vec_20; grad2:integer);
      procedure polinomio_ci(var grado1: integer; var v1:vec_20);
   end;


implementation


constructor sal_otros.inicializa;
var
   i:integer;
begin
   for i:= 1 to 20 do
   cond_inic[i]:=0;
end;
procedure sal_otros.limpia(x,y,renglones:integer);
var
   i:integer;
begin
   for i:= 1 to renglones DO
   begin
      gotoxy(x,y);
      writeln('                                                ');
      y:=y+1;
   end;
end;
```

```
{*****************************************************************}
{El procedimiento despliega las opciones para que el usuario intro }
{duzca una entrada                                                 }
{*****************************************************************}

procedure sal_otros.despliega(VAR respuesta:integer);

Begin
    limpia(1,8,10);
    VpullDownMenu ('Entrada_al_sistema',
                'A(t**n) A_exp(Bt) A_sankt A_coskt A_exp(Bt)*(t**n)
A_exp(Bt)*sankt A_exp(Bt)*coskt',
                2,8,Simple,
                31,112,
                FormAtributeByte(White,Black),
                FormAtributeByte(Black,Cyan),
                FormAtributeByte(Black,Red),
                FormAtributeByte(White,Black),
                MenuNotInWindow,AcceptWithLetter,
                AcceptWithLetter,respuesta);
End;

{*****************************************************************}
{En el siguiente menú se despliegan las opciones + y Termina entrada }
{*****************************************************************}

procedure sal_otros.mer(VAR respuesta:integer);

Begin
    VpullDownMenu ('',
                '+ Termina_entrada',
                15,12,Simple,
                31,112,
                FormAtributeByte(White,Black),
                FormAtributeByte(Black,Cyan),
                FormAtributeByte(Black,Red),
                FormAtributeByte(White,Black),
                MenuNotInWindow,AcceptWithLetter,
                AcceptWithLetter,respuesta);
End;

{*****************************************************************}
{ El metodo desp_func introduce en el vector v1 la transformada de }
{ Laplace de la funcion seleccionada por el usuario                }
{*****************************************************************}

procedure sal_otros.desp_func(respuesta:integer;var v1,num: vec_20;
                              var grado1,grado_num:integer;
                              var mat1,mat2:arreglo2;
                              var apuntador,ap2:integer);
VAR
    a,b,k:real;
    i,j,n,grado2:integer;
    v2,res: vec_20;

BEGIN
    gotoxy(1,13);
    case respuesta of
```

```
1: begin
      write('Introducir A: ');
      readln(a);
      write( 'Introducir n: ');
      readln(n);
      num[0]:= fact(n)*a;
      v1[n+1]:=1;
      for j:=0 to n DO
         v1[j]:=0;
      grado1:=n+1;
      grado_num:=0
   end;
2: begin
      write('Introducir A: ');
      readln(A);
      write('Introducir B: ');
      readln(b);
      num[0]:= a;
      grado_num:= 0;
      v1[0]:= -b;
      v1[1]:= 1;
      grado1:= 1;
   end;
3: begin
      write('Introducir A: ');
      readln(a);
      write('Introducir k: ');
      readln(k);
      num[0]:= k*a;
      grado_num:= 0;
      v1[0]:= k*k;
      v1[1]:= 0;
      v1[2]:= 1;
      grado1:= 2;
   end;
4: begin
      write('Introducir A: ');
      readln(a);
      write('Introducir k: ');
      readln(k);
      num[0]:= 0;
      num[1]:= a;
      grado_num:=1;
      v1[0]:= k*k;
      v1[1]:= 0;
      v1[2]:= 1;
      grado1:=2;
   end;
5: begin
      write('Introducir A: ');
      readln(a);
      write('Introducir B: ');
      readln(b);
      write('Introducir n: ');
      readln(n);
      num[0]:= fact(n)*a;
      grado_num:=0;
      v1[0]:= -b;
```

```
            v1[1]:=1;
            v2[0]:= -b;
            v2[1]:= 1;
            grado1:=1;
            grado2:= 1;
            for i:= 1 to n DO
            BEGIN
               multiplica(v1,v2,res,grado1,grado2);
               v1:= res;
               grado1:= grado1+grado2;
            end;
         end;
      6: begin
            write('Introducir A: ');
            readln(a);
            write('Introducir B: ');
            readln(b);
            write('Introducir k: ');
            readln(k);
            num[0]:=k*a;
            grado_num:= 0;
            v1[0]:= k*k+b*b;
            v1[1]:= -2*b;
            v1[2]:= 1;
            grado1:= 2;
         end;
      7: begin
            write('Introducir A: ');
            readln(a);
            write('Introducir B: ');
            readln(b);
            write('Introducir k: ');
            readln(k);
            num[0]:= -b*a;
            num[1]:= a;
            grado_num:=1;
            v1[0]:= k*k+b*b;
            v1[1]:= -2*b;
            v1[2]:= 1;
            grado1:=2;
         end;
   end;
end;



(*****************************************************************)
{ El siguiente metodo encuentra la transformada de Laplace de la funcion}
{ de entrada :
(*****************************************************************)

procedure sal_otros.lee(var indice,indice2:integer; var vector1:vec_20;
                            arr:vec_20; grad2:integer);
VAR
   mat1,mat2: arreglo2;
   mayor,apuntador,i,j,k, ap2,grado1,grado2,
   grad1,grad_vec,grado_num,grado_num2,respuesta: integer;
```

```
    v1,v2,vec,num2,num,res:vec_20;
    sal:boolean;

BEGIN
    despliega(respuesta);
    grado1:=0;
    grado2:= 0;
    grado_num:=0;
    apuntador:=1;
    ap2:=1;
    sal:= false;
    desp_func(respuesta,v1, num, grado1, grado_num,mat1,mat2,apuntador,ap2);
    while(not(sal)) DO
    BEGIN
       men(respuesta);
       if ((respuesta=1) or (respuesta=2)) THEN
       BEGIN
          mat1[apuntador].pol:= v1;
          mat1[apuntador].grado:= grado1;
          mat2[ap2].pol:= num;
          mat2[ap2].grado:= grado_num;
          apuntador:= apuntador+1;
          grado1:= 0;
          ap2:= ap2+1;
          grado_num:= 0;
          if respuesta = 1 then
          BEGIN
             despliega(respuesta);

desp_func(respuesta,v1,num,grado1,grado_num,mat1,mat2,apuntador,ap2);
             respuesta:=1;
          end;
          if respuesta = 2 then sal:= true
       end;
    end; (while)
    grad1:= 0;
    for i:= 1 to 20 DO
       vector1[i]:=0;
    vector1[0]:= 1;
    For i:= 1 to apuntador-1 DO
    BEGIN
       num_pold:= num_pold+1;
       grado_d[num_pold]:= mat1[i].grado;
       j:=mat1[i].grado;
       multiplica(vector1,mat1[i].pol,res,grad1,mat1[i].grado);
       vector1:=res;
       grad1:=grad1+mat1[i].grado;
       while j>= 0 DO
       BEGIN
          coef_d[indice]:= mat1[i].pol[j];
          indice:= indice+1;
          j:= j-1;
       end;
    end;
    multiplica(vector1,arr,res,grad1,grad2);
    vector1:=res;
    suma_quebrados(mat1,mat2,apuntador,vec,mayor);
    num_poln:= num_poln+1;
```

```
    grado_n[num_poln]:= mayor;
    while mayor>=0 DO
    BEGIN
       coef_n[indice2]:= vec[mayor];
       mayor:=mayor-1;
       indice2:=indice2+1;
    end;
end;

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

procedure sal_otros.polinomio_c( VAR grado1: integer; var v1:vec_20);
VAR
     deriv,v2,res: vec_20;
     i,indice,j,grado2,n,apunta:integer;
BEGIN
   for i:= 1 to 20 DO
      deriv[i]:=0;
   for i:= 0 to 20 DO
      cond_inic[i]:=0;
   indice:=1;
   grado1:= grado_d[1];
   for j:= 0 to grado1 DO
   begin
      deriv[grado1-j]:= coef_d[indice];
      indice:= indice+1;
   end;
   for i:= 2 to num_pold DO
   BEGIN
      grado2:= grado_d[i];
      for j:= 0 to grado2 DO
      BEGIN
         v2[grado2-j]:= coef_d[indice];
         indice:= indice+1;
      end;
      multiplica(deriv,v2,res,grado1,grado2);
      grado1:= grado1+grado2;
      deriv:=res;
   end;
   gotoxy(1,1);
   if grado1>=1 then writeln('Introducir las condiciones iniciales a partir
la derivada de mayor orden: ');
   writeln;
   for i:= 0 to grado1-1 do
   begin
      readln(cond_inic[i]);
   end;
   v1:= cond_inic;
   v2:=v1;
   n:= grado1;
   apunta:= n-1;
   for i:= 0 to n-2 DO
   begin
      for j:= 1 to apunta DO
         v2[j-1]:= v2[j];
      if deriv[grado1-1-i]<>0 then
      begin
```

```
        for j:= 0 to apunta -1 DO
          v1[j]:= v1[j]+v2[j]*deriv[grado1-i-1]
      end;
      apunta:= apunta-1;
    end;
    grado1:= grado1-1;
    while (v1[grado1]=0) and (grado1>=0) DO
       grado1:=grado1-1;
  if grado1<0 then grado1:=0;
end;


(*************************************************************************)
( El siguiente metodo multiplica a la funcion de transferencia por 1/s   )
(*************************************************************************)

constructor sal_escalon.forma_ec_esc(indice:integer);
begin
    num_pold:= num_pold+1;
    grado_d[num_pold]:= 1;
    coef_d[indice]:=1;
    coef_d[indice+1]:= 0;
end;

(*************************************************************************)
: El metodo forma_ec_ram multiplica la funcion de transferencia por      )
; 1/(s*s)                                                                )
(*************************************************************************)

constructor sal_rampa.forma_ec_ram(indice: integer);
BEGIN
    num_pold:= num_pold+2;
    grado_d[num_pold-1]:= 1;
    grado_d[num_pold]:= 1;
    coef_d[indice]:=1;
    coef_d[indice+1]:=0;
    coef_d[indice+2]:=1;
    coef_d[indice+3]:=0;
end;

(*************************************************************************)

end.
```

```
{ En esta unidad se encuentran los procedimientos para calcular la  }
{ respuesta en el tiempo para sistemas en el dominio del tiempo     }
{ discreto}

unit resp3;
interface
   uses respuesta,crt,menuz,tldef;
   type
   vec_1000 = array[0..200] of real;
   sal_discreta = object(transfer)
      procedure abre;
      constructor init(indice:integer);
      procedure desp(var grad:arreglo; var coef: vector_r;
                     var num_pol:integer);
      procedure imp_func;
      procedure trans_inv(mat:mat_20; opcion:integer); virtual;
      procedure encuentra_cte(cont:integer;
                     var cons:real; raiz:real);virtual;
      procedure despliega_func(cont:integer);virtual;
      procedure valor_num(mat:mat_20;opcion:integer); virtual;
      procedure cierra;
   end;
   disc_escalon = object(sal_discreta)
      procedure forma_ec_esc(indice,indice2:integer);
   end;
   disc_rampa = object(sal_discreta)
      procedure forma_ec_ram(indice,indice2:integer);
   end;
   disc_otros = object(sal_discreta)
      cond_inic:vec_20;
      procedure limpia(x,y,renglones:integer);
      procedure men(VAR respuesta:integer);
      procedure lee(var indice,indice2:integer; var vector1:vec_20;
                                                arr:vec_20; grad2:integer);
      procedure despliega(var respuesta:integer);
      procedure desp_func(respuesta:integer;VAR v1,
                num:vec_20;VAR grado1,grado_num:integer;
                var mat1,mat2:arreglo2; var apuntador, ap2:integer);
      procedure polinomio_ci( VAR grado1: integer; var v1:vec_20);
   end;
   var
      arch,arch2,arch3,arch4,arch5: text;


implementation;

{*************************************************************}
{ El siguiente metodo inicializa los archivos de salida       }
{*************************************************************}

procedure sal_discreta.abre;
begin
   assign(arch,'resp_lib.rtd');
   rewrite(arch);
   writeln(arch,'RESPUESTA LIBRE');
   writeln(arch,'INICIO: ');
   assign(arch2, 'resp_for.rtd');
   rewrite(arch2);
```

```
    writeln(arch2,'RESPUESTA FORZADA');
    writeln(arch2,'INICIO: ');
    assign(arch3, 'resp_tot.rtd');
    rewrite(arch3);
    writeln(arch3,'RESPUESTA TOTAL');
    writeln(arch3,'INICIO:');
    assign(arch4, 'resp_tran.rtd');
    rewrite(arch4);
    writeln(arch4, 'RESPUESTA TRANSITORIA');
    writeln(arch4, 'INICIO:');
    assign(arch5, 'resp_per.rtd');
    rewrite(arch5);
    writeln(arch5, 'RESPUESTA PERMANENTE');
    writeln(arch5, 'INICIO: ');
end;

constructor sal_discreta.init(indice:integer);
begin
   num_pold:= num_pold+1;
   grado_d[num_pold]:=1;
   coef_d[indice]:=1;
   coef_d[indice+1]:= 0;
end;

{*************************************************************}
{ El siguiente metodo cierra los archivos utilizado           }
{*************************************************************}

procedure sal_discreta.cierra;
begin
   close(arch);
   close(arch2);
   close(arch3);
   close(arch4);
   close(arch5);
end;

{*************************************************************}
{ El metodo transfer.desp despliega la funcion de transferencia }
{*************************************************************}

procedure sal_discreta.desp(Var grad : arreglo;
                Var coef: vector_r;
                Var num_pol: INTEGER);

VAR
   i, grado, j, terminos, part_ent
   :INTEGER;

BEGIN
   j:= 1;
   FOR i:= 1 to num_pol do
   BEGIN
      grado:= grad[i];
      terminos:= grad[i] + 1;
      WHILE terminos > 0 DO
      BEGIN
         IF ((num_pol <> 1) and (terminos = grad[i+1])) THEN
```

```
            write ('(');
         IF (grado = 0) THEN
         BEGIN
            part_ent:= trunc(coef[j]);
            if (coef[j]<>0) THEN
            begin
               if ((part_ent-coef[j])=0) THEN write (part_ent)
               else
                  write(coef[j]:6:3);
            end;
            if num_pol <> 1 then write (')');
            j:= j+1;
         end
         ELSE
         BEGIN
            IF (coef[j]<>0) THEN
            BEGIN
               if coef[j]<>1 then
               begin
                  part_ent := trunc (coef[j]);
                  if((part_ent-coef[j])=0) THEN write (part_ent)
                  else
                     write (coef[j]:6:3);
               end;
               write('z');
               if grado>1 then write ('**',grado);
            end;
            j:= j+1;
            IF ((coef[j]>0) and (grado <> 0)) THEN
               write('+');
         end;
         terminos:= terminos-1;
         grado:=grado-1;
         IF ((coef[j]=0) and (grado>0)) THEN
         BEGIN
            j:= j+1;
            if ((coef[j]>0) and (grado<>0)) then
               write('+');
            terminos:= terminos-1;
            grado:= grado-1;
         end;
      end; {while}
   end; {for}
end;

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

procedure sal_discreta.imp_func;

begin
   clrscr;
   write('G(z) = ');
   desp(grado_n,coef_n, num_poln);
   write(' / ');
   desp(grado_d,coef_d, num_pold);
end;
```

```
{*****************************************************************}
{ El siguiente metodo imprime el valor de la transformada de Laplace }
{ inversa }
{*****************************************************************}

procedure sal_discreta.trans_inv(mat: mat_20; opcion:integer);
VAR
    coef,con1,cons2,con2,con3,coef2,a,b,c,d,r,a1,b1,c1,c2,c3,cons,angulo:
     real;
    imp,aux,comp_rep.cont,i,j ,part_ent,des, n,rep:integer;

BEGIN
    write('y(k)= ');
    j:=1;
    imp:=0;
    comp_rep:= 0;
    dim_mat(n);
    for i:= 1 to ap_raiz-1 DO
    BEGIN
       if raices[i].tipo = 1 then
       begin
          if ( (abs(raices[i].r)<1) and (opcion<=9) )
          or ( (abs(raices[i].r)>=1) and (opcion<>9) ) then
          begin
             coef:= mat[j,n+1]/cte;
             part_ent:= trunc(coef);
             if (j<>1) and (coef>0)
                then write('+');
             if (coef <> 0)  then
             begin
                imp:=1;
                if (abs(coef)<=1.0004) and (abs(coef)>=0.9996) then
                begin
                   if coef<0 then write ('-');
                end
                else
                begin
                   if (part_ent-coef)=0 then write (part_ent)
                   else
                   begin
                      if (abs(coef)<0.001) then write (coef:6)
                      else
                         write (coef:6:3);
                   end;
                end;
                if raices[i].r <> 0 then
                begin
                   part_ent:= trunc(raices[i].r);
                   if not( (abs(raices[i].r)>=0.9996
                    and (abs(raices[i].r)<=1.0004)) then
                   begin
                      if (abs(part_ent-raices[i].r)<=0.0004) then
                         write(' (',part_ent,')**k ')
                      else
                      begin
                         aux:= round(raices[i].r);
                         if (abs(aux-raices[i].r)<0.0001) then
                         begin
```

```
                    if abs(aux)=1 then
                    begin
                       if aux = 1 then write(' ( 1 )**k');
                       if aux = -1 then write(' ( -1 )**k');
                    end
                    else
                    if aux<>0 then
                       write(' (',aux,' )**k');
                 end
                 else
                 begin
                    if(abs(raices[i].r)<0.001 )then
                       write(' (',raices[i].r:6,' )**k ')
                    else
                       write(' (',raices[i].r:6:3,' )**k ');
                 end;
              end;
           end
           else
           begin
              if raices[i].r > 0 then
                 write(' (1)**k ' );
              if raices[i].r < 0 then
                 write (' (-1)**k ');
           end;
        end
        else
           write(' δ(k) ');
     end;
  end;
  j:= j+1;
end
else
if raices[i].tipo = 2 then
begin
   if ( (abs(raices[i].r)<1) and (opcion<=9) )
   or ( (abs(raices[i].r)>=1) and (opcion<>9) ) then
   begin
      cont:=1;
      des:=0;
      raices_rep(rep,i);
      while rep>0 DO
      BEGIN
         if raices[i].r<>0 THEN
         BEGIN
            coef:= mat[j,n+1]/(cte);
            encuentra_cte(cont,cons.raices[i].r);
            coef:= coef/cons;
            if (j<>1) and (coef>0) then write ('+');
            if coef <> 0 then
            begin
               imp:=1;
               if (abs(coef)<1.0009) and ( abs(coef) >=1 ) then
               begin
                  if coef<0 then write ('-');
               end
               else
               begin
```

```
      part_ent:= trunc(coef);
      if (part_ent-coef)=0 then
         write(part_ent)
      else
      begin
         aux:= round(coef);
         if (abs(aux-coef)<0.0001) then
         begin
            if aux<>0 then
               write(aux);
         end
         else
         begin
            if(abs(coef)<0.001 )then
               write(coef:6)
            else
               write(coef:6:3);
         end;
      end;
   end;
   despliega_func(cont);
   part_ent:= trunc(raices[i].r);
   if not( (abs(raices[i].r)>=0.9996)
   and (abs(raices[i].r)<=1.0004)) then
   begin
      if (abs(part_ent-raices[i].r)<=0.0004) then
         write( '( ',part_ent,' )**k ')
      else
      begin
         aux:= round(raices[i].r);
         if (abs(aux-raices[i].r)<0.0001) then
         begin
            if abs(aux)=1 then
            begin
               if aux = 1 then write('(1)**k ');
               if aux = -1 then write('(-1)**k ');
            end
            else
            if aux<>0 then
               write(' ( ',aux,' )**k ');
         end
         else
         begin
            if(abs(raices[i].r)<0.001 )then
               write(' ( ',raices[i].r:6,' )**k ')
            else
               write(' ( ',raices[i].r:6:3,' )**k ');
         end;
      end;
   end
   else
   begin
      if (raices[i].r>0) then
         write('(1)**k');
      if (raices[i].r<0) then
         write('(-1)**k ');
   end;
end;
```

```
end
else
begin
   coef:= mat[j,n+1]/cte;
   if (abs(coef)>=0.9996) and (abs(coef)<=1.0004) then
   begin
      if (des<>0) then
      begin
         if coef<0 then write('-δ(k-',des,')');
         if coef>0 then write('+δ(k-',des,')');
      end
      else
      begin
         if coef<0 then write('-δ(k)');
         if coef>0 then write('δ(k)');
      end;
   end
   else
   begin
      if (j<>1) and (coef>0) then write ('+');
      if coef <> 0 then
      begin
         if (abs(coef)<1.0009) and ( abs(coef) >=1 ) then
         begin
            if coef<0 then write ('-');
         end
         else
         begin
            part_ent:= trunc(coef);
            if (part_ent-coef)=0 then
               write(part_ent)
            else
            begin
               aux:= round(coef);
               if (abs(aux-coef)<0.0001) then
               begin
                  if aux<>0 then
                     write(aux);
               end
               else
               begin
                  if(abs(coef)<0.001 )then
                     write(coef:6)
                  else
                     write(coef:6:3);
               end;
            end;
         end;
         if des<>0 then
            write('δ(k -',des,')')
         else write('δ(k)');
      end;
   end;
   des:=des+1;
end;
rep:= rep-1;
i:= i +1;
cont:= cont+1;
```

```
            j:=j+1;
         end;
         i:=i-1;
      end
      else
      begin
         raices_rep(rep,i);
         i:= i+rep-1;
         j:=j+rep;
      end;
   end;
   if (raices[i].tipo = 3)  then
   begin
      c:= -2*raices[i].r;
      d:= raices[i].r*raices[i].r+raices[i].i*raices[i].i;
      a:=mat[j,n+1]/cte;
      b:= mat[j+1,n+1]/cte;
      j:=j+2;
      r:=sqrt(d);
      if ( (abs(r)<1) and (opcion<=9) ) or
      (( r>=1) and (opcion<>9) ) then
      begin
         a1:= -c;
         c1:= 2*r;
         b1:= sqrt(c1*c1-a1*a1);
         if a1<>0 then
         begin
            angulo:= arctan(b1/a1);
            if a1<0 then angulo:= 3.141593+angulo;
         end
         else
         begin
            if b1>0 then angulo:= 1.57079;
         end;
         if(b<>0) and(r<>0) and (cos(angulo)<>0) then
         BEGIN
            imp:=1;
            if (j>1) and (b>0) then write('+');
            if abs(b) = 1 then
            begin
               if b=-1 then write ('-');
            end
            else
            BEGIN
               part_ent:= trunc(b);
               if (part_ent-b)=0 then
                  write(part_ent)
               else
               begin
                  if abs(b)<0.001 then write(b:6)
                  else write(b:6:3);
               end;
            end;
            if abs(r)=1 then
            BEGIN
               if r=-1 then write('(-1)**k');
            end
            else
```

```
    begin
       part_ent:= trunc(r);
       if (abs(part_ent-r)<=0.0004) then
         write(' ( ',part_ent,' )**k ')
       else
       begin
          if abs(r)<0.001 then
            write(' ( ',r:6,' )**k')
          else
            write(' ( ',r:6:3,' )**k');
       end;
    end;
    part_ent:= trunc(angulo);
    write(' ');
    if (abs (angulo)=1) then
       write ('cosk')
    else
    begin
       if (abs(part_ent-angulo)<=0.0004) then
        write('cos',part_ent,'k')
       else write('cos',angulo:6:3,'k');
    end;
end;
con2:=0;
if (sin(angulo))<>0 then
   con2:= (-b*c*0.5+a)/(r*sin(angulo));
if con2<>0 then
begin
   imp:=1;
   if (con2>0) and (j>1) then write('+');
   if abs(con2) = 1 then
   begin
      if con2=-1 then write ('-');
   end
   else
   BEGIN
      part_ent:= trunc(con2);
      if (part_ent-con2)=0 then
        write(part_ent)
      else
      begin
         if abs(con2)<0.001 then write(con2:6)
         else write(con2:6:3);
      end;
   end;
   if abs(r)=1 then
   BEGIN
      if r=-1 then write('(-1)**k');
   end
   else
   begin
      part_ent:= trunc(r);
      if (abs(part_ent-r)<=0.0004) then
      write(' ( ',part_ent,' )**k ')
      else
      begin
         if abs(r)<0.001 then
           write(' ( ',r:6,' )**k')
```

```
                    else
                      write(' ( ',r:6:3,' )**k');
                  end;
               end;
               write(' ');
               part_ent:= trunc(angulo);
               if (abs (angulo)=1) then
                  write ('senk')
               else
               begin
                  if (abs(part_ent-angulo)<=0.0004) then
                     write('sen',part_ent,'k')
                  else write('sen',angulo:6:3,'k');
               end;
            end;
         end;
      end;
      if raices[i].tipo = 4 then
      begin
         falla:= true;
         i:= ap_raiz-1;
      end;
   end;
   if (opcion=10) and (imp=0) then write('0');
   writeln;
end;


procedure sal_discreta.encuentra_cte
      (cont:integer; var cons:real; raiz:real);
VAR
   i,exp,factor:integer;

BEGIN
   exp:= cont-1;
   cons:=1;
   for i:=1 to exp DO
      cons:= cons*raiz;
   factor:= exp;
   for i:= 1 to exp DO
   BEGIN
      cons:= cons*factor;
      factor:= factor-1;
   end;
end;

{*************************************************************************}
{ El siguiente procedimiento se utiliza para encontrar la transformada z}
{ inversa cuando se tienen raices repetidas                               }
{*************************************************************************}

procedure sal_discreta.despliega_func(cont:integer);
VAR
   i,j:integer;


BEGIN
   if cont<1 THEN
```

```
   BEGIN
      j:=1;
      write('k');
      for i:= 3 to cont DO
      BEGIN
         write('(k-',j,')');
         j:= j+1;
      end;
   end;
end;

{*************************************************************}
( El siguiente procedimiento calcula un polinomio auxiliar para )
( calcular la respuesta libre del sistema                       )
{*************************************************************}

procedure disc_otros.polinomio_ci( VAR grado1: integer; var v1:vec_20);
VAR
     y,v2,res: vec_20;
     aux,i,indice,j,grado2,n,apunta:integer;
BEGIN
   for i:= 1 to 20 DO
       y[i]:=0;
   indice:=1;
   grado1:= grado_d[1];
   for j:= 0 to grado1 DO
   begin
      y[grado1-j]:= coef_d[indice];
      indice:= indice+1;
   end;
   for i:= 2 to num_pold DO
   BEGIN
      grado2:= grado_d[i];
      for j:= 0 to grado2 DO
      BEGIN
         v2[grado2-j]:= coef_d[indice];
         indice:= indice+1;
      end;
      multiplica(y,v2,res,grado1,grado2);
      grado1:= grado1+grado2;
      y:=res;
   end;
   grado1:= grado1-1;
   for i:= 0 to grado1 DO
      y[i]:=y[i+1];
   writeln;
   if grado1<>0 then writeln('Introducir las condiciones iniciales ');
   writeln;
   cond_inic[0]:=0;
   for i:= 0 to grado1-1 do
   begin
      aux:= grado1-1-i;
      write('v[ ',aux,' ]= ');
      readln(cond_inic[i+1]);
   end;
   v1:= cond_inic;
   v2:=v1;
   n:= grado1;
```

```
    apunta:= n;
    for i:= 0 to n-2 DO
    begin
       for j:= 2 to apunta DO
          v2[j-1]:= v2[j];
       if y[grado1-1-i]<>0 then
       begin
          for j:= 1 to apunta -1 DO
             v1[j]:= v1[j]+v2[j]*y[grado1-i-1];
       end;
       apunta:= apunta-1;
    end;
    grado1:= grado1-1;
    while (v1[grado1]=0) and (grado1>=0) DO
       grado1:=grado1-1;
end;


(*************************************************************************)
( El siguiente procedimiento calcula el valor de la transformada z inversa)
( numericamente                                                           )
(*************************************************************************)

procedure sal_discreta.valor_num(mat:mat_20; opcion:integer);

VAR
   num,den,res,num2,den2,v1: vec_20;
   num3,den3: vec_1000;
   ap,n,grado_num,k, grado_den,k2,i2,i3,indice,
   grado1,grado2,grado3,i,j,elementos:integer;
   t,c,d,r:real;
   mat1,mat2:arreglo2;
   repetidas:boolean;

BEGIN
   if (genera='s') or (genera='S') then
   begin
      num[0]:=0;
      grado_num:=0;
      if (opcion=9) or (opcion=10) then
      begin
         grado1:=0;
         grado2:=0;
         den[0]:=1;
         den2[0]:=1;
         j:= 1;
         ap:=1;
         dim_mat(n);
         repetidas:=false;
         for i:= 1 to ap_raiz-1 DO
         BEGIN
            if (raices[i].tipo=1)  then
            begin
               if ( (abs(raices[i].r)<0.9996) and (opcion=9)) or
                  ( (abs(raices[i].r))=0.9996) and (opcion=10)) then
               begin
                  den2[1]:= 1;
                  den2[0]:= - raices[i].r;
```

```
        grado2:=1;
        mat2[ap].grado:=0;
        mat2[ap].pol[0]:= mat[j,n+1];
        mat1[ap].grado:=1;
        mat1[ap].pol:=den2;
        ap:=ap+1;
     end;
     j:=j+1;
  end;
  if (raices[i].tipo=2)  then
  begin
     if ( (abs(raices[i].r)<0.9996) and (opcion=9)) or
        ( (abs(raices[i].r)>=0.9996) and (opcion=10)) then
     begin
        if repetidas=false then
        begin
           raices_rep(i2,i);
           k:=i2;
           k2:=i2;
           repetidas:=true;
        end;
        den2[1]:= 1;
        den2[0]:= -raices[i].r;
        grado2:=1;
        grado3:=1;
        v1:=den2;
        while k2<i2 DO
        begin
           multiplica(den2,v1,res,grado2,grado3);
           grado2:= grado2+grado3;
           den2:=res;
           k2:=k2+1;
        end;
        k:=k-1;
        k2:=k;
        if k=0 then repetidas:=false;
        mat2[ap].grado:=0;
        mat2[ap].pol[0]:= mat[j,n+1];
        mat1[ap].grado:=grado2;
        mat1[ap].pol:=den2;
        ap:=ap+1;
     end;
     j:=j+1;
  end;
  if (raices[i].tipo=3) then
  begin
     c:= -2*raices[i].r;
     d:=raices[i].r*raices[i].r+raices[i].i*raices[i].i;
     r:= sqrt(d);
     if ( ( abs(r)<0.9996) and (opcion=9)) or
      ( (abs(r)>=0.9996) and (opcion=10)) then
     begin
        den2[2]:=1;
        den2[1]:=c;
        den2[0]:=d;
        grado2:=2;
        mat2[ap].grado:=1;
        mat2[ap].pol[0]:= mat[j,n+1];
```

```
            mat2[ap].pol[1]:= mat[j+1,n+1];
            mat1[ap].grado:=2;
            mat1[ap].pol:=den2;
            ap:=ap+1;
          end;
          j:=j+2;
        end;
        multiplica(den,den2,res,grado1,grado2);
        grado1:=grado1+grado2;
        den2[0]:= 1;
        grado2:=0;
        den:=res;
    end;
    grado_den:= grado1;
    suma_quebrados(mat1,mat2,ap,num,grado_num);
end
else
begin
    grado1:= grado_n[1];
    for i:=0 to grado1 DO
       num[i]:= coef_n[grado1-i+1];
    indice:= grado1+2;
    for j:=2 to num_poln DO
    BEGIN
       grado2:= grado_n[j];
       for i:= 0 to grado2 DO
          num2[i]:= coef_n[indice+grado2-i];
       multiplica(num,num2,res,grado1,grado2);
       grado1:= grado1+grado2;
       num:=res;
       indice:= indice+grado2+1;
    end;
    grado_num:=grado1;
    grado1:= grado_d[1];
    for i:= 0 to grado1 DO
       den[i]:= coef_d[grado1-i+1];
    indice:= grado1+2;
    for j:= 2 to num_pold DO
    BEGIN
       grado2:= grado_d[j];
       for i:= 0 to grado2 DO
          den2[i]:= coef_d[indice+grado2-i];
       multiplica(den,den2,res,grado1,grado2);
       grado1:= grado1+grado2;
       den:=res;
       indice:= indice+grado2+1;
    end;
    grado_den:= grado1;
end;
if not ((num[0]=0) and (grado_num=0) ) then
begin
    grado2:=1;
    num2[0]:=0;
    num2[1]:=1;
    multiplica(num,num2,res,grado_num,grado2);
    num:=res;
    grado_num:= grado_num+1;
end;
```

```
for i:= 0 to 200 do
begin
   num3[i]:=0;
   den3[i]:=0;
end;
write('Cuantos valores de la respuesta en el tiempo desea? ');
readln (elementos);
if not( (num[0]=0) and (grado_num=0) ) then
begin
   for i:= 0 to grado_num DO
      num3[i]:= num[grado_num-i];
   for i:= 0 to grado_den DO
      den3[i]:= den[grado_den-i];
   FOR i:= 0 to elementos-1 DO
   BEGIN
      t:= num3[i]/den3[0];
      FOR j:= 1 to grado_den DO
         num3[i+j]:= num3[i+j]-t*den3[j];
      num3[i]:= t;
   end;
end;
j:=0;
t:=0;
case opcion of
   6: begin
         for i:= grado_num to grado_den-1 do
         begin
            writeln(arch,j,'      ',t);
            j:=j+1;
         end;
         t:=j;
         for i:= 0 to elementos-j-1 DO
         BEGIN
            writeln(arch,j,'      ',num3[i]);
         j:=j+1;
      end;
   end;
   7: begin
         for i:= grado_num to grado_den-1 do
         begin
            writeln(arch2,j,'      ',t);
            j:=j+1;
         end;
         t:=j;
         for i:= 0 to elementos-j-1 DO
         BEGIN
            writeln(arch2,j,'      ',num3[i]);
            j:=j+1;
         end;
      end;
   8: begin
         for i:= grado_num to grado_den-1 do
         begin
            writeln(arch3,j,'      ',t);
            j:=j+1;
         end;
         t:=j;
         for i:= 0 to elementos-j-1 DO
```

```
            BEGIN
               writeln(arch3,j,'      ',num3[i]);
               j:=j+1;
            end;
         end;
      9: begin
            for i:= grado_num to grado_den-1 do
            begin
               writeln(arch4,j,'      ',t);
               j:=j+1;
            end;
            t:=j;
            for i:= 0 to elementos-j-1 DO
            BEGIN
               writeln(arch4,j,'      ',num3[i]);
               j:=j+1;
            end;
         end;
      10: begin
            for i:= grado_num to grado_den-1 do
            begin
               writeln(arch5,j,'      ',t);
               j:=j+1;
            end;
            t:=j;
            for i:= 0 to elementos-j-1 DO
            BEGIN
               writeln(arch5,j,'      ',num3[i]);
               j:=j+1;
            end;
         end;
   end;
end;
writeln;
end;

{*****************************************************************}
{ El siguiente procedimiento multiplica la funcion de transferencia por
  z/(z-1)}
{*****************************************************************}

procedure disc_escalon.forma_ec_esc(indice,indice2:integer);
begin
   num_pold:=num_pold+1;
   grado_d[num_pold]:=1;
   coef_d[indice]:=1;
   coef_d[indice+1]:= -1;
   num_poln:= num_poln+1;
   grado_n[num_poln]:=1;
   coef_n[indice2]:= 1;
   coef_n[indice2+1]:= 0;
end;
{*****************************************************************}
{ El procedimiento forma_ec_ram multiplica la funcion de transferencia}
{ por z/(z-1)(z-1)                                                  }
{*****************************************************************}
procedure disc_rampa.forma_ec_ram(indice,indice2:integer);
begin
```

```
   num_pold:= num_pold+2;
   grado_d[num_pold-1]:=1;
   grado_d[num_pold]:=1;
   coef_d[indice]:=1;
   coef_d[indice+1]:= -1;
   coef_d[indice+2]:= 1;
   coef_d[indice+3]:=-1;
   num_poln:= num_poln+1;
   grado_n[num_poln]:=1;
   coef_n[indice2]:= 1;
   coef_n[indice2+1]:= 0;
end;

procedure disc_otros.limpia(x,y,renglones:integer);
var
   i:integer;
begin
   for i:= 1 to renglones DO
   begin
      gotoxy(x,y);
      writeln('                                                    ');
      y:=y+1;
   end;
end;


(*********************************************************************)
{El procedimiento desp despliega las opciones para que el usuario intro }
{duzca una entrada                                                    }
(*********************************************************************)

procedure disc_otros.despliega(VAR respuesta:integer);

Begin

   limpia(1,8,10);
   VpullDownMenu ('Entrada_al_sistema',
                  'B_exp(-a*k) B_(a**k) B_cos_wkt B_sen_wkt B_(r**k)sen_wkt
B_(r**k)cos_wkt',
                  2,8,Simple,
                  31,112,
                  FormAtributeByte(White,Black),
                  FormAtributeByte(Black,Cyan),
                  FormAtributeByte(Black,Red),
                  FormAtributeByte(White,Black),MenuInWindow,AcceptWithLetter,
                  AcceptWithLetter.respuesta);
End;

(*********************************************************************)
{En el siguiente menú se despliegan las opciones + y Termina entrada  }
(*********************************************************************)

procedure disc_otros.men(VAR respuesta:integer);

Begin
   VpullDownMenu ('',
                  '+ Termina_entrada',
                  15,12,Simple,
```

```
            31,112,
            FormAtributeByte(White,Black),
            FormAtributeByte(Black,Cyan),
            FormAtributeByte(Black,Red),
            FormAtributeByte(White,Black),MenuInWindow,AcceptWithLetter,
            AcceptWithLetter,respuesta);
End;


procedure disc_otros.desp_func(respuesta:integer; var v1,num:vec_20;
                               var grado1,grado_num:integer;
                               var mat1, mat2:arreglo2;
                               var apuntador, ap2:integer);

var
   a,b,w,r:real;
BEGIN
   gotoxy(1,18);
   case respuesta of
      1: begin
           write('Intruducir b: ');
           readln(b);
           write('Intruducir a: ');
           readln(a);
           grado_num:=1;
           num[0]:= 0;
           num[1]:= b;
           v1[0]:= -exp(-a);
           v1[1]:=1;
           grado1:=1;
         end;
      2: begin
           write('Intruducir b: ');
           readln(b);
           write('Intruducir a: ');
           readln(a);
           grado_num:=1;
           num[0]:= 0;
           num[1]:= b;
           v1[0]:= -a;
           v1[1]:= 1;
           grado1:=1;
         end;
      3: begin
           write('Introducir b');
           readln(b);
           write ('Introducir w ');
           readln(w);
           grado_num:=2;
           num[0]:= 0;
           num[1]:= -cos(w)*b;
           num[2]:= b;
           grado1:=2;
           v1[0]:= 1;
           v1[1]:= -2*cos(w);
           v1[2]:= 1;
         end;
```

```
      4: begin
            write('Introducir b');
            readln(b);
            Write ('Introducir w ');
            readln(w);
            grado_num:=1;
            num[0]:= 0;
            num[1]:= sin(w)*b;
            v1[0]:= 1;
            v1[1]:= -2*cos(w);
            v1[2]:= 1;
            grado1:=2;
         end;
      5: begin
            write('Introduzca b ');
            readln(b);
            write('Introduzca r ');
            readln(r);
            write('Introduzca w ');
            readln(w);
            grado_num:=1;
            num[0]:= 0;
            num[1]:= r*sin(w)*b;
            grado1:=2;
            v1[0]:= r*r;
            v1[1]:= -2*r*cos(w);
            v1[2]:=1;
         end;
      6: begin
            write('Introduzca b ');
            readln(b);
            write('Introduzca r ');
            readln(r);
            write('Introduzca w ');
            readln(w);
            grado_num:=2;
            num[0]:= 1;
            num[1]:= -r*cos(w)*b;
            num[2]:= b;
            grado1:=2;
            v1[0]:= r*r;
            v1[1]:= -2*r*cos(w);
            v1[2]:=1;
         end;
   end;
end;

{*****************************************************************}
{ El siguiente metodo encuentra la transformada de Laplace de la funcion}
{ de entrada }
{*****************************************************************}

procedure disc_otros.lee(var indice,indice2:integer; var vector1:vec_20;
                                 arr:vec_20; grad2:integer);
VAR
   mat1,mat2: arreglo2;
   mayor,apuntador,i,j,k, ap2,grado1,grado2,
   grad1,grad_vec,grado_num,grado_num2,respuesta: integer;
```

```
    v1,v2,vec,num2,num,res:vec_20;
    sal:boolean;

BEGIN
    despliega(respuesta);
    grado1:=0;
    grado2:= 0;
    grado_num:=0;
    apuntador:=1;
    ap2:=1;
    sal:= false;
    desp_func(respuesta,v1, num, grado1, grado_num,mat1,mat2,apuntador,ap2);
    while(not(sal)) DO
    BEGIN
        men(respuesta);
        if ((respuesta=1) or (respuesta=2)) THEN
        BEGIN
            mat1[apuntador].pol:= v1;
            mat1[apuntador].grado:= grado1;
            mat2[ap2].pol:= num;
            mat2[ap2].grado:= grado_num;
            apuntador:= apuntador+1;
            grado1:= 0;
            ap2:= ap2+1;
            grado_num:= 0;
            if respuesta = 1 then
            BEGIN
               despliega(respuesta);
               desp_func(respuesta,v1,num,grado1,
               grado_num,mat1,mat2,apuntador,ap2);
               respuesta:=1;
            end;
            if respuesta = 2 then sal:= true
        end;
    end; (while)
    grad1:= 0;
    for i:= 1 to 20 DO
        vector1[i]:=0;
    vector1[0]:= 1;
    For i:= 1 to apuntador-1 DO
    BEGIN
        num_pold:= num_pold+1;
        grado_d[num_pold]:= mat1[i].grado;
        j:=mat1[i].grado;
        multiplica(vector1,mat1[i].pol,res,grad1,mat1[i].grado);
        vector1:=res;
        grad1:=grad1+mat1[i].grado;
        while j <> 0 DO
        BEGIN
            coef_d[indice]:= mat1[i].pol[j];
            indice:= indice+1;
            j:= j-1;
        end;
    end;
    multiplica(vector1,arr,res,grad1,grad2);
    vector1:=res;
    suma_quebrados(mat1,mat2,apuntador,vec,mayor);
    num_poln:= num_poln+1;
```

```
    grado_n[num_poln]:= mayor;
    while mayor>=0 DO
    BEGIN
       coef_n[indice2]:= vec[mayor];
       mayor:=mayor-1;
       indice2:=indice2+1;
    end;
end;


end.
```

```
{
}
{  ┌──────────────────────────────────────────────────────────────────┐
}
{  │                        Principal.Pas                             │
}
{  │                                                                  │
}
{  │  Unidad para cálculo de la respuesta en el tiempo de sistemas    │
}
{  │  dinámicos                                                       │
}
{  │                     Leticia Anzar Uribe                          │
}
{  │                     México. Agosto. 1991.                        │
}
{  └──────────────────────────────────────────────────────────────────┘
}

{+-------------------------------------------------------------------+
}
{ Para probar la unidad, definir PRUEBA
}
{+-------------------------------------------------------------------+
}

{$Define  PRUEBA }

{$IfDef  PRUEBA }
{$Define UINIT  }
{$EndIf }

{+-------------------------------------------------------------------+}

{$Define UINIT }

{$IfNDef PRUEBA }
Unit
  Principal;

Interface

{$EndIf }

Uses
     respuesta,
     resp2,
     resp3,
     crt,
     tldef,
     menus;



{+-------------------------------------------------------------------+}
{ Encabezados de funciones y procedimientos públicos, debajo del IfNDef }
{ -----------------------------------------------------------------+}

{$IfNDef PRUEBA }
```

```
( Se declara público el procedimiento principal de esta unidad
)
Procedure Main;
Procedure Menu;
Procedure Menu2;

Implementation

{$EndIf}

{+-------------------------------------------------------------------+}
{| Implementación de procedimientos y funciones, tanto públicas      |}
{| como privadas                                                     |}
{| Constantes, tipos y variables privadas                            |}
{+-------------------------------------------------------------------+}

{ Declaramos estas variabes como globales a la unidad, pero privada al resto
 del programa }
Var
    sal1: transfer;
    sal2: sal_escalon;
    sal3: sal_rampa;
    sal4: sal_otros;
    sal_disc1:sal_discreta;
    sal_disc2: disc_escalon;
    sal_disc3: disc_rampa;
    sal_disc4: disc_otros;
    indice,indice2,opcion,entrada:integer;

{+-------------------------------------------------------------------+}
{| Procedimiento para hacer el menú principal                        |}
{+-------------------------------------------------------------------+}

Procedure DoMenu1 ;


Begin
   PullDownMenu ('PRINCIPAL','SISTEMA_CONTINUO SISTEMA_DISCRETO',
              2,2,Simple,
              31,112,
              FormAtributeByte(White,Black),
              FormAtributeByte(Black,Cyan),
              FormAtributeByte(Black,Red),
              FormAtributeByte(White,Black),
              MenuInWindow,AcceptWithLetter,
              AcceptWithLetter,Opcion);
   clrscr;
End;
{+-------------------------------------------------------------------+}
{| Procedimiento para hacer un menú vertical                         |}
{+-------------------------------------------------------------------+}

Procedure DoMenu2(var opcion:integer) ;

Begin

   VpullDownMenu ('Menu',
              'Respuesta_impulso_δ(t) Respuesta_escalón
```

```
                Respuesta_rampa Otro_tipo_de_respuesta',
                2,5,Simple,
                31,112,
                FormAtributeByte(White,Black),
                FormAtributeByte(Black,Cyan),
                FormAtributeByte(Black,Red),
                FormAtributeByte(White,Black),
                MenuInWindow,AcceptWithLetter,
                AcceptWithLetter,Opcion);
   clrscr;
End;
{+-------------------------------------------------------------------------+}
{! Procedimiento para hacer un menú vertical para indicar como se         !}
{!realizará la                                                            !}
{! captura de la función de transferencia.                                !}
{+-------------------------------------------------------------------------+}

Procedure men_ent(var opcion:integer) ;

Begin
   clrscr;
   writeln('Captura de la función de transferencia a través de: ');
   VPullDownMenu ('',
                'Archivo Teclado',
                15,3,Simple,
                31,112,
                FormAtributeByte(White,Black),
                FormAtributeByte(Black,Cyan),
                FormAtributeByte(Black,Red),
                FormAtributeByte(White,Black),
                MenuInWindow,AcceptWithLetter,
                AcceptWithLetter,Opcion);
End;
{+-------------------------------------------------------------------------+}
{! Procedimiento para hacer un menú vertical para el caso discreto       !}
{+-------------------------------------------------------------------------+}

Procedure DoMenu3(var opcion:integer) ;

Begin

   VPullDownMenu ('Menu',
                'Respuesta_impulso_δ(k) Respuesta_escalón
                 Respuesta_rampa Otro_tipo_de_respuesta',
                 2,5,Simple,
                 31,112,
                 FormAtributeByte(White,Black),
                 FormAtributeByte(Black,Cyan),
                 FormAtributeByte(Black,Red),
                 FormAtributeByte(White,Black),
                 MenuInWindow,AcceptWithLetter,
                 AcceptWithLetter,Opcion);
End;


procedure cap_ar(var indice,indice2,num_poln,num_pold:integer;
                 var grado_n,grado_diarreglo;
                 var coef_n,coef_d:vector_r);
```

```
var
   opcion,i,j, pol: integer;
   ar:string;
   nombre:text;

BEGIN
   clrscr;
   write('Nombre del archivo > ');
   readln(ar);
   assign(nombre,ar);
   reset(nombre);
      num_poln:=0;
      while num_poln<=0 DO
      begin
         readln(nombre,num_poln);
         if num_poln<=0 then
            writeln('Entrada inválida:
            El número de polinomios debe de ser mayor a cero');
      end;
      indice2:=1;
      for i:= 1 to num_poln do
      BEGIN
         readln(nombre,grado_n[i]);
         FOR j:= 0 to grado_n[i] DO
         BEGIN
            readln(nombre,coef_n[indice2]);
            indice2:= indice2+1;
         end;
      end;
      num_pold:=0;
      while num_pold<= 0 do
      begin
         readln(nombre,Num_pold);
         if num_pold<=0 then writeln('Entrada inválida
               el número de polinomios debe de ser mayor a cero');
      end;
      indice:= 1;
      FOR i:= 1 to num_pold DO
      BEGIN
         readln(nombre,grado_d[i]);
         FOR j:= 0 to grado_d[i] do
         BEGIN
            readln(nombre,coef_d[indice]);
            indice:= indice+1;
         end;
      end;
      for i:= 1 to 21 do
      begin
         raices[i].r:=0;
         raices[i].i:=0;
         raices[i].tipo:=0;
      end;
end;

{+-------------------------------------------------------------------------+}
{ Implementación de las acciones a realizar al seleccionar un sistema      }
```

```
{ continuo                                                              }
{+----------------------------------------------------------------------+}

Procedure menu;

var
   respuesta,gra_tot, gradn,gradd,i,grado: integer;
   repite:char;
   v1,vec:vec_20;
   bandera,anade: boolean;
   num:vector_r;
   incremento,cons,coef,ls,t: real;

BEGIN
        anade:=false;
        writeln;
        cons2:=0;
        gradn:=0;
        gradd:=0;
        bandera:= false;
        for i:= 1 to sal1.num_pold DO
           gradd:= gradd+sal1.grado_d[i];
        for i:= 1 to sal1.num_poln DO
           gradn:= gradn+sal1.grado_n[i];
        if gradn>gradd then writeln('La función de transferencia
                no corresponde a un sistema físico realizable')
        else
        if gradn<=gradd then
        begin
           domenu2(respuesta);
           case respuesta of
              1: begin
                    if (gradd>=20) or (gradn>=20) then
                    writeln('El sistema es demasiado grande')
                    else
                    begin
                       with sal1 do
                       begin
                          writeln('Respuesta total: ');
                          if (num_poln=1) and( grado_n[1]=0)
                          and (num_pold=1) and (grado_d[1]=0) then
                          begin
                             coef:= coef_n[1]/(coef_d[1]);
                             coef:= coef/cte;
                             cons2:=coef;
                             imprime(coef,0);
                             writeln('δ(t)');
                             if (genera='s') or (genera='S') then
                             begin
                                writeln('Hasta que valor en el tiempo
                                desea conocer la respuesta');
                                readln(LS);
                                incremento:= ls/500;
                                t:=0;
                                while t<=ls DO
                                begin
                                   writeln(f3,t,'      ',0);
                                   t:= t+ incremento;

                                150
```

```
                    end;
                  end;
               end
               else
               begin
                  if gradn = gradd then
                  begin
                     with sal1 DO
                     begin
                        divide(num,gra_tot,cons);
                        num_poln:=1;
                        grado_n[1]:= gra_tot;
                        coef_n:= num;
                        anade:= true;
                     end;
                  end;
                  if anade then
                  begin
                     cons:=cons/cte;
                     cons2:=cons;
                  end;
                  encuentra_raiz;
                  frac_par(bandera,vec,3);
               end;
            end;
         end;
      end;
   2: begin
         with sal2 do
         begin
            if (gradd>=19) or (gradn>=20) then
               writeln('El sistema es demasiado grande')
            else
            begin
               writeln('Respuesta total: ');
               num_poln:= sal1.num_poln;
               num_pold:= sal1.num_pold;
               grado_n:= sal1.grado_n;
               grado_d:= sal1.grado_d;
               coef_n:= sal1.coef_n;
               coef_d:= sal1.coef_d;
               cte:= sal1.cte;
               forma_ec_esc(indice);
               encuentra_raiz;
               frac_par(bandera,vec,3);
            end;
         end;
      end;
   3: begin
         with sal3 do
         begin
            if (gradd>=18) or (gradn>=20) then
               writeln('El sistema es demasiado grande')
            else
            begin
               writeln('Respuesta total;');
               num_poln:= sal1.num_poln;
               num_pold:= sal1.num_pold;
```

```
            grado_n:= sal1.grado_n;
            grado_d:= sal1.grado_d;
            coef_n:= sal1.coef_n;
            coef_d:= sal1.coef_d;
            cte:= sal1.cte;
            forma_ec_ram(indice);
            encuentra_raiz;
            frac_par(bandera,vec,2);
          end;
      end;
    end;
4:  begin
      with sal4 DO
      BEGIN
         inicializa;
         num_pold:= sal1.num_pold;
         grado_d:= sal1.grado_d;
         coef_d:= sal1.coef_d;
         cte:= sal1.cte;
         for i:= 0 to 20 do
            vec[i]:=0;
         polinomio_c(grado,v1);
         for i:= 0 to grado DO
            v1[i]:=v1[i]*cte;
         num_poln:= 1;
         grado_n[1]:= grado;
         for i:= 1 to grado+1 do
         begin
            coef_n[i]:= v1[grado+1-i];
         end;
         gradn:=0;
         gradd:=0;
         for i:= 1 to num_pold DO
            gradd:= gradd+grado_d[i];
         for i:= 1 to num_poln DO
            gradn:= gradn+grado_n[i];
         if (gradd>=20) or (gradn>=20) then
           writeln('El sistema es demasiado grande')
         else
         begin
            encuentra_raiz;
            clrscr;
            writeln('Respuesta libre: ');
            frac_par(bandera,vec,1);
         end;
         num_poln:= sal1.num_poln;
         grado_n:= sal1.grado_n;
         coef_n:= sal1.coef_n;
         gotoxy(1,6);
         writeln('Introducir la entrada al sistema: ');
         limpia(1,7,10);
         lee(indice,indice2,vec,v1,grado);
         writeln;
         limpia(1,6,15);
         gotoxy(1,6);
         writeln('Respuesta forzada: ');
         encuentra_raiz;
         frac_par(bandera,vec,2);
```

```
                end
             end;
          end (case)
       end;
end;

{+-------------------------------------------------------------------------+}
{! Implementación de un menú que se activará al seleccionar un sistema    !}
{! discreto                                                                !}
{+-------------------------------------------------------------------------+}


procedure menu2;


var
   respuesta,gradn,gradd,i,grado: integer;
   repite:char;
   v1,vec:vec_20;
   bandera: boolean;

BEGIN
       writeln;
       gradn:=0;
       gradd:=0;
       bandera:= false;
       for i:= 1 to sal_disc1.num_pold DO
          gradd:= gradd+sal_disc1.grado_d[i];
       for i:= 1 to sal_disc1.num_poln DO
          gradn:= gradn+sal_disc1.grado_n[i];
       if gradn>gradd then writeln('La función de transferencia
          no corresponde a un sistema físico realizable')
       else
       if gradn<=gradd then
       begin
          domenu3(respuesta);
          gotoxy(1,12);
          case respuesta of
             1: begin
                   if (gradd>=20) or (gradn>=20) then
                      writeln('El sistema es demasiado grande')
                   else
                   begin
                      sal_disc1.init(indice);
                      sal_disc1.encuentra_raiz;
                      sal_disc1.frac_par(bandera,vec,8);
                   end;
                end;
             2: begin
                  with sal_disc2 do
                  begin
                     if (gradd>=19) or (gradn>=20) then
                        writeln('El sistema es demasiado grande')
                     else
                     begin
                        num_poln:= sal_disc1.num_poln;
                        num_pold:= sal_disc1.num_pold;
                        grado_n:= sal_disc1.grado_n;
```

```
            grado_d:= sal_disc1.grado_d;
            coef_n:= sal_disc1.coef_n;
            coef_d:= sal_disc1.coef_d;
            cte:= sal_disc1.cte;
            init(indice);
            indice:= indice+2;
            forma_ec_esc(indice,indice2);
            encuentra_raiz;
            frac_par(bandera,vec,8);
          end;
      end;
    end;
  3: begin
      with sal_disc3 do
      begin
        if (gradd>=18) or (gradn>=20) then
          writeln('El sistema es demasiado grande')
        else
        begin
          num_poln:= sal_disc1.num_poln;
          num_pold:= sal_disc1.num_pold;
          grado_n:= sal_disc1.grado_n;
          grado_d:= sal_disc1.grado_d;
          coef_n:= sal_disc1.coef_n;
          coef_d:= sal_disc1.coef_d;
          cte:= sal_disc1.cte;
          init(indice);
          indice:= indice+2;
          forma_ec_ram(indice,indice2);
          encuentra_raiz;
          frac_par(bandera,vec,8);
        end;
      end;
    end;
  4: begin
      with sal_disc4 DO
      begin
        num_pold:= sal_disc1.num_pold;
        grado_d:= sal_disc1.grado_d;
        coef_d:= sal_disc1.coef_d;
        cte:= sal_disc1.cte;
        init(indice);
        indice:= indice+2;
        polinomio_c(grado,v1);
        for i:= 0 to grado DO
            v1[i]:=v1[i]*cte;
        num_poln:= 1;
        grado_n[1]:= grado;
        for i:= 1 to grado+1 do
        begin
          coef_n[i]:= v1[grado+1-i];
        end;
        gradn:=0;
        gradd:=0;
        for i:= 1 to num_pold DO
            gradd:= gradd+grado_d[i];
        for i:= 1 to num_poln DO
            gradn:= gradn+grado_n[i];
```

```
                        if (gradd>=20) or (gradn>=20) then
                           writeln('El sistema es demasiado grande')
                        else
                        begin
                            encuentra_raiz;
                            clrscr;
                            writeln('Respuesta libre: ');
                            frac_par(bandera,vec,6);
                        end;
                        num_poln:= sal_disc1.num_poln;
                        grado_n:= sal_disc1.grado_n;
                        coef_n:= sal_disc1.coef_n;
                        gotoxy(1,6);
                        writeln('Introducir la entrada al sistema: ');
                        limpia(1,7,10);
                        lee(indice,indice2,vec,v1,grado);
                        writeln;
                        limpia(1,6,15);
                        gotoxy(1,6);
                        writeln('Respuesta forzada: ');
                        encuentra_raiz;
                        frac_par(bandera,vec,7);
                     end
                  end;
            end;
      end;
end;
Procedure cuadro;
var
   i:integer;
begin
     writeln('┌──────────────────────────────────────────────────────────
     ──────────────────────┐');
   for i:= 1 to 19 do
   begin
     gotoxy(1,i+2);
     writeln('│');
     gotoxy(78,i+2);
     writeln('│');
   end;
   gotoxy(1,22);
   writeln('└──────────────────────────────────────────────────────────
   ──────────────────────┘');
end;

{+-------------------------------------------------------------------+}
{: Procedimiento principal                                          :}
{+-------------------------------------------------------------------+}
Procedure Main;
begin
   clrscr;
   domenu1;
   clrscr;
   case opcion of
      1: begin
            with sal1 do
            begin
               inicia_arc;
```

```
        men_ent(entrada);
        case entrada of
           1:cap_ar(indice,indice2,num_poln,num_pold,
           grado_n,grado_d,coef_n,coef_d);
           2:begin
                clrscr;
                gotoxy(19,1);
                writeln('Captura de la función de transferencia');
                cuadro;
                window(3,4,77,21);
                captura(indice,indice2);
             end;
        end;
        write('Desea que se generen archivos con la
        información numérica?(s/n) ');
        readln(genera);
        window(1,1,80,25);
        if (num_pold=1) and (grado_d[1]=0) and (coef_d[1]=0) then
        begin
           if (num_poln=1) and (grado_n[1]=0) and (coef_n[1]=0) then
              writeln('Error: G(s) = 0/0. Existe una ideterminación.'
           else
              writeln('Error: No es válida la división por cero');
        end
        else
        begin
           imp_func;
           normaliza;
           menu;
        end;
     end;
  end;
2: begin
     with sal_disc1 DO
     BEGIN
        men_ent(entrada);
        case entrada of
           1:cap_ar(indice,indice2,num_poln,num_pold,grado_n,
           grado_d,coef_n,coef_d);
           2:begin
                clrscr;
                gotoxy(19,1);
                writeln('Captura de la función de transferencia');
                cuadro;
                window(3,4,77,21);
                captura(indice,indice2);
             end;
        end;
        write('Desea que se generen archivos con la información
        numérica?(s/n) ');
        readln(genera);
        window(1,1,80,25);
        if (num_pold=1) and (grado_d[1]=0) and (coef_d[1]=0) then
        begin
           if (num_poln=1) and (grado_n[1]=0) and (coef_n[1]=0) then
              writeln('Error: G(s) = 0/0. Existe una ideterminación.'
           else
              writeln('Error: No es válida la división por cero');
```

```
                end
                else
                begin
                   imp_func;
                   normaliza;
                   abre;
                   menu2;
                   cierra;
                end;
             end;
          end;
    end;
end;


{+--------------------------------------------------------------------+}
{| Variables y tipos que sólo se usan para probar                     |}
{| Las variables que se tenían en el programa principal, aquí son de  |}
{| prueba                                                             |}
{+--------------------------------------------------------------------+}



{+--------------------------------------------------------------------+}
{| I N I C I A L I Z A C I O N   Y / O   P R U E B A                  |}
{+--------------------------------------------------------------------+}
{$IfDef UINIT }
{---------------------------------------------------------------------}
Begin

{$EndIf }

{------------------ Código para inicializar la unidad ----------------}

{ No hay inicialización
}

{------------------ Código para probar la unidad ---------------------}

{$IfDef PRUEBA }
  ;
  Main

{$EndIf}

End.
```

B I B L I O G R A F I A

1. Borland international.- Object Oriented Programming Guide, 1989, 118p.
2. Dorf, Richard.- Sistemas automáticos de control. Editorial Addison Wesley México, 1974, 400p.
3. Espinosa, Ismael.- Análisis de sistemas discretos, UAM Iztapalapa, Méxixo, 1988, 50p.
4. Gerez, Victor et alli.- Desarrollo y administración de programas de computadora, Mexico, Editorial CECSA, 1985, 299p.
5. Harrison, Howard.- Controles automáticos, Editorial Trillas, México, 1979
6. Luthe et alli. Métodos numéricos, Editorial Limusa, México, 1986, 443p
7. Ogata, Katsuhiko.- Ingeniería de control moderna. Editorial Prentice-Hall, Méxixo, 1980, 902p.
8. Ogata, Katsuhiko.- Dinámica de sistemas, Editorial Prentice-Hall, New Jersey, 1978
9. Rodríguez, Francisco.- Apuntes de sistemas dinámicos, Facultad de Ingeniería, UNAM, 1986, 665p.