136

UNIVERSIDAD NACIONAL AUTONOMA DE MÉXICO

FACULTAD DE INGENIERIA

GENERACIÓN DE UN ALGORITMO PARA DISEÑO DE OBSERVADORES LINEALES MULTIVARIABLES

T E S I S

PARA OBTENER EL GRADO DE :

DOCTOR EN INGENIERIA

P R E S E N T A

JOSE HORACIO SANDOVAL RODRIGUEZ

MÉXICO

GENERACION DE UN ALGORITMO PARA DISEÑO DE OBSERVADORES LINEALES MULTIVARIABLES

EXAMEN DE GRADO DE MAESTRIA

Propuso: Ismael Espinosa E.

Realizó: J. Horacio Sandoval R.

ENERO DE 1978

0096

UNIVERSIDAD NACIONAL AUTONOMA

DIVISION DE ESTUDIOS SUPERIORES
FACULTAD DE INGENIERIA

CIUDAD UNIVERSITARIA MEXICO 20, D. F.
APDO. POSTAL 70.256 TEL. 548.58.27

01149
136

MEMORANDO

Para: M. en C. Luis Marcial Hernández Ortega
De: Ismael Espinosa E.

El tema de examen de grado de maestría en control para el ingeniero

JOSE HORACIO SANDOVAL RODRIGUEZ

es el siguiente:

GENERACION DE UN ALGORITMO PARA DISEÑO DE OBSERVADORES LINEALES MULTIVARIABLES.

Condiciones:

i) Lenguaje: FORTRAN IV Versión B6700
ii) Que acepte sistemas hasta de orden 10 con n entradas y m salidas
iii) Que verifique controlabilidad y observabilidad
iv) Que utilice un método sencillo (por ej. el de Fallside) para asignar los polos al observador
v) Que permita cambiar fácilmente la matriz de ubicación de polos, para así poder escoger el mejor observador
vi) Que grafique las variables de estado, tento las disponibles como las estimadas
vii) El programa deberá estar disponible en disco y verificado su funcionamiento con algunos ejemplos.

El plazo máximo considerado adecuado es de un mes.

Atentamente

Cd. Universitaria, a 4 de Noviembre de 1977

1. INTRODUCCION

En el análisis de sistemas dinámicos uno de los objetivos principales es lograr que el sistema responda de alguna forma determinada. Para un sistema dado esto puede lograrse en ocasiones. La respuesta a cuando es posible llevar un sistema o a las variables de estado que lo definen a ciertos valores predeterminados, se encuentra en el concepto de *Controlabilidad*, el cual fue introducido por Kalman en la década de los 60.

Una vez que la condición de Controlabilidad Absoluta se ha satisfecho, surge la pregunta de cómo lograr el control necesario para poder llevar el sistema a un cierto estado. Una de las técnicas más comunes es la realimentación de estado, es decir, a la excitación original se agrega el vector de estado.

Esta técnica presenta un nuevo problema cuando una o más de las variables de estado no puede medirse directamente, y por tanto no puede realimentarse.

Esta dificultad implica la necesidad de contar al menos con alguna estimación de las variables de estado que faltan, o en general, de todo el estado.

El concepto de *Observabilidad* resuelve cualitativamente esta dificultad, ya que permite saber si dado el sistema y sus salidas es posible recuperar la información de todo el vector de estado.

Debe notarse que los conceptos de Controlabilidad y Observabilidad indican si el sistema se puede controlar y observar, pero *no* indican cómo lograr ese control y esa observación, respectivamente.

En el presente escrito se tratará únicamente de los sistemas necesarios para estimar el estado, también llamados, *Observadores*. Se analizará un método en particular, el de Luenberger. También se describirá el programa para computadora digital que genera los observadores con el método indicado además de resolver la ecuación de estado del sistema y del observador.

tico de su matriz de transferencia G(s), donde

$$G(s) \triangleq c(sI - A)^{-1} B + D$$

o incluyendo la definición de la matriz de transición en el dominio complejo s

$$G(s) \triangleq c\ \phi(s)\ B + D$$

Este resultado se obtiene de tomar la transformada de Laplace de las ecs 1 y 2, y eliminar el vector de estado x para llegar a la expresión

$$y(s) = G(s)\ u(s)$$

II) Criterio de Wolovich

Si se representa un sistema dinámico con operadores diferenciales de la siguiente manera

$$P(D)\ x(t) = Q(D)\ u(t)$$
$$y(t) = R(D)\ z(t) + T(D)\ u(t)$$

será controlable, si y solo si, P(D) y Q(D) son matrices primo izquierda relativas, y además será observable, si y solo si, P(D) y R(D) son primo derecho relativas.

III) Criterio de Gilbert

Para aplicar este criterio se requiere llevar las ecs 1 y 2 del sistema a la forma normal

2. CONTROLABILIDAD Y OBSERVABILIDAD

2.1 Los planteamientos formales de controlabilidad y observabilidad son los siguientes:

Controlabilidad.- Si un sistema se caracteriza por la ecuación de estado

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

donde

- A — matriz $(n \times n)$, llamada matriz de coeficiente
- B — matriz $(n \times r)$, llamada matriz de distribución
- $u(t)$ — vector de excitaciones $(r \times 1)$
- $x(t)$ — vector de estado $(n \times 1)$

se dice que es completamente controlable si el estado $x(t)$ para $t = t_o$ puede llevarse mediante alguna entrada $u(t)$ a un estado final cualquiera $x(t_f)$ en un tiempo finito $(t_f - t_o) \geq 0$.

Observabilidad.- Si además de la ec (1) se tiene la ecuación de salida

$$y(t) = cx(t) + Du(t) \qquad (2)$$

donde

C matriz $(m \times n)$, llamada matriz de salida

D matriz $(m \times r)$, llamada matriz de transmisión

$y(t)$ vector de salida $(m \times 1)$

se dice que el estado $x(t_o)$ es completamente observable si para cualquier entrada $u(t)$ existe un tiempo finito $t_f \geq t_o$ tal que es posible deducir $x(t_o)$ a partir de la información dada por $y(t)$ en ese intervalo.

2.2 Existen varias formas para definir la controlabilidad y observabilidad de un sistema a partir de las ecs (1) y (2), aquí se indicaran solo cuatro de ellas

I) Análisis de la Observabilidad y Controlabilidad empleando la matriz de transferencia. (Método de Chen)

El sistema lineal e invariante en el tiempo definido por las ecuaciones (1) y (2) será controlable y observable, si y solo si, el orden de la matriz A es igual al grado de la matriz de transferencia G(s), lo anterior equivale a decir que el sistema será controlable y observable, si y solo si, el polinomio característico de la matriz A es igual al polinomio caracterís tico de su matriz de transferencia G(s), donde

$$G(s) \triangleq c(sI - A)^{-1} B + D$$

o incluyendo la definición de la matriz de transición en el dominio complejo s

$$G(s) \triangleq c\, \phi(s)\, B + D$$

Este resultado se obtiene de tomar la transformada de Laplace de las ecs 1 y 2, y eliminar el vector de estado x para llegar a la expresión

$$y(s) = G(s)\, u(s)$$

II) Criterio de Wolovich

Si se representa un sistema dinámico con operadores diferenciales de la siguiente manera

$$P(D)\, x(t) = Q(D)\, u(t)$$
$$y(t) = R(D)\, z(t) + T(D)\, u(t)$$

será controlable, si y solo si, P(D) y Q(D) son matrices primo izquierda relativas, y además será observable, si y solo si, P(D) y R(D) son primo derecho relativas.

III) Criterio de Gilbert

Para aplicar este criterio se requiere llevar las ecs 1 y 2 del sistema a la forma normal

$$\dot{z}(t) = \Delta\ z(t) + \beta\ u(t)$$

$$y(t) = \gamma\ z(t) + D\ u(t)$$

al transformar de coordenadas mediante la matriz modal asociada a la matriz A, entonces el sistema será controlable si β no tiene renglones nulos y será observable si γ no tiene columnas nulas.

IV) Criterio de Kalman

Para la aplicación de este criterio se requiere la ecuación de estado y la de salida con las cuales se forman las siguientes matrices

$$P \triangleq [B\ AB\ A^2B\ \ldots\ldots\ A^{n-1}\ B]$$

$$Q \triangleq [C^T\ A^TC^T\ (A^T)^2\ C^T\ \ldots\ldots\ (A^T)^{n-1}\ C^T]$$

donde

P matriz (n x nr) y es llamada matriz de controlabilidad

Q matriz (n x mn) y es llamada matriz de observabilidad

Este criterio indica que el sistema será completamente controlable, si y solo si, el rango de la matriz P es igual al orden del sistema, en este caso n.

Así mismo, el sistema será completamente observable, si y solo si, el rango de la matriz Q es igual al orden del sistema, en este caso n.

Como se ve el criterio más simple de aplicar es el de Kalman, aunque el más claro conceptualmente es el de Gilbert. Debe mencionarse que estos son algunos de los criterios existentes solamente, y que el criterio de Kalman es el que se usa con mayor frecuencia, y es el que se incluyó en el programa.

3. ALGORITMO DE LUENBERGER PARA LA CONSTRUCCION DEL OBSERVADOR

Existen en la literatura bastantes algoritmos para la generación de los observadores, cada uno de ellos tiene sus propias características y en la mayoría de los casos son de aplicación particular, es decir, han sido desarrollados para una necesidad específica.

Aquí se describirá el algoritmo desarrollado por Luenberger el cual no es, necesariamente el óptimo en procesamiento o en resultados, aunque si es el más divulgado.

3.1 Algoritmo de Luenberger

Este algoritmo requiere que el sistema analizado sea completamente observable, y a partir de las entradas y salidas estima el vector de estado.

De lo anterior se tiene que el observador es un sistema lineal e invariante en el tiempo cuyos controles son las entradas y salidas del sistema que observa y cuya salida es la estimación del vector de estado, tal como se muestra en la figura 1.

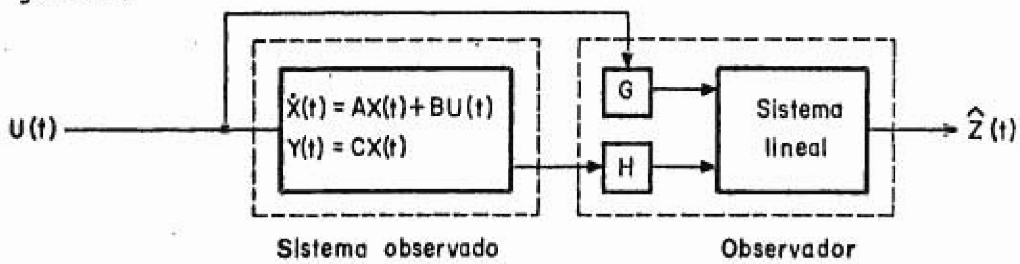

Figura 1

Dado que el observador es un sistema dinámico lineal, se puede escribir su ecuación de estado como

$$\frac{d\hat{z}(t)}{dt} = \tilde{A}z(t) + \tilde{B}\tilde{u}(t) \tag{4}$$

donde

$\tilde{A}$ matriz $(n \times n)$

$\tilde{B}$ matriz $[n \times (r + m)]$

$\tilde{u}(t)$ vector de $(r + m) \times 1$

$\hat{z}$ vector $n \times 1$

$\tilde{u}(t) = \begin{bmatrix} u(t) \\ y(t) \end{bmatrix}$

$\tilde{B} = [G\ H]$

o de manera equivalente

$$\frac{d\hat{z}(t)}{dt} = \tilde{A}\hat{z}(t) + Gu(t) + Hcx(t) \tag{5}$$

donde

G matriz n x r

H matriz n x m

Se desea llegar a un vector z(t) cuyos elementos sean combinaciones lineales de las variables de estado del sistema, de manera que

$$z(t) = T\,x(t) \tag{6}$$

Por tanto, $\hat{z}(t)$ es la estimación de z(t), lo cual involucra un error asociado en la estimación $\Delta z(t)$ tal que

$$\Delta z(t) = \hat{z}(t) - z(t) \tag{7}$$

Ahora el problema consiste en definir las matrices $\tilde{A}$, G y H que minimicen el error $\Delta z(t)$. Luenberger formó el siguiente procedimiento:

Derivando la ec 6

$$\frac{dz(t)}{dt} = T\,\frac{dx(t)}{dx}$$

$$= T\,[Ax(t) + Bu(t)]$$

$$= T\,Ax(t) + T\,Bu(t)$$

restando esta última igualdad a la ec 5, se tiene

$\tilde{A}\hat{z}$

$$\frac{d\hat{z}(t)}{dt} - \frac{dz(t)}{dt} = \cancel{Ax(t)} + [G - TB]u(t) + [HC - TA]x(t) \qquad (8)$$

recordando la ec 7 se tiene

$= \tilde{A}\hat{z} - \tilde{A}z$

$$\tilde{A}\Delta z(t) = \tilde{A}\cancel{\hat{x}}(t) - \tilde{A}x(t)$$

$$\therefore \tilde{A}\hat{z}(t) = \tilde{A}\Delta z(t) + \tilde{A}z(t)$$

$$= \tilde{A}\Delta z(t) + \tilde{A}Tx(t)$$

sustituyendo en la ec 8

$$\frac{d}{dt}\Delta z(t) = \tilde{A}\Delta z(t) + [G - TB]u(t) + [HC - TA + \tilde{A}T]x(t) \qquad (9)$$

si en esta última ecuación se obliga que

$$\tilde{A}T + HC = TA$$
$$G = TB \qquad (10)$$

resulta $\quad \dfrac{d}{dt}\Delta z(t) = \tilde{A}\Delta z(t)$

que es un sistema de ecuaciones diferenciales de primer orden en $\Delta z(t)$, y cuya solución esta dada por

$$\Delta z(t) = e^{\tilde{A}t}\,\Delta z(t = t_o)$$

de manera que si la parte real de todos los valores propios de $\tilde{A}$ es negativa se tiene

$$\Delta z(t) \to 0 \text{ cuando } t \to \infty$$

lo que implica que

$$\tilde{z}(t) \to z(t) = T\,x(t) \qquad (11)$$

Dado que los polos del observador deben tener parte real negativa surge la pregunta ¿qué tan negativa? Analizando el comportamiento del observador y del sistema en el dominio del tiempo, vemos que es conveniente seleccionar los polos del observador a la izquierda de la región donde están los polos del sistema, de manera que el transitorio provocado por el observador, no se confunda o afecte apreciablemente el transitorio del sistema observado.

La limitación para no ir más hacia la izquierda, usualmente es de índole física, en el programa solo se considera la posición relativa de los polos del sistema y los del observador.

3.2 *Elección de los polos del observador*

La ecuación característica del observador es

$$\det\,(\lambda I - \tilde{A}) = |\lambda I - \tilde{A}| = 0 \qquad (12)$$

y haciendo la matriz T igual a la identidad, las ecuaciones 10 resultan

$$\tilde{A} = A - HC \qquad (13)$$

$$G = B$$

sustituyendo en la ecuación 12

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

y tomando como factor común la matriz $(\lambda I - A)$ se tiene

$$|(\lambda I - A)\ [I + (\lambda I - A)^{-1}HC]| = 0$$

$$|(\lambda I - A)||I + (\lambda I - A)^{-1}HC| = 0$$

haciendo $\phi(\lambda) \triangleq (\lambda I - A)^{-1}$

resulta $|\lambda I - A|\ |In + \phi(\lambda)\ HC| = 0$

y empleando la siguiente igualdad de determinantes

$$|In + KG| = |Im + GK|$$

donde

- In identidad de orden n
- Im identidad de orden m
- K matriz n x m
- G matriz m x n

se llega a

$$|\lambda In - A||Im + c\phi(\lambda)\ H| = 0$$

o en su lugar

$$|\lambda In - A||Im + s(\lambda)\ H| = 0 \qquad (14)$$

donde

$$s(\lambda) \triangleq c\ \phi\ (\lambda)$$

Dado que los polos del observador son diferentes a los del sistema, entonces

$$|\lambda In - A| \neq 0$$

por tanto la única posibilidad para que se anule la ecuación es que el segundo factor sea cero es decir

$$|Im + s(\lambda)\ H| = 0 \qquad (15)$$

para cuando λ adquiere el valor de cualquier polo del observador.

Una manera de lograr que el determinante sea nulo, es mediante una columna o un renglón de ceros obtenidos a partir de la matriz H.

Sea el renglón j-ésimo el que se desea anular, y llamando

ϕ_j al j-ésimo renglón de la matriz identidad
$s_j(\lambda)$ al j-ésimo renglón de la matriz $s(\lambda)$

entonces el j-ésimo renglón de la ec 15 queda

$$\phi_j - s_j(\lambda)\ H = 0$$

Esta expresión no es suficiente para determinar H, que posee n renglones, pero es posible obtener a partir de los polos del observador.

n renglones linealmente independientes uno para cada uno. Es
to es una consecuencia de la observabilidad del par (A, C), es decir de que sus modos linealmente independientes.

De esta manera se obtiene la matriz regular W.

$$W \triangleq [\, s_j(\lambda_1) \;\; s_j(\lambda_2) \; . \; . \; . \; s_j(\lambda_n)]^T$$

en donde $s_j(\lambda_i)$ es el j-ésimo renglón de $s(\lambda)$ asociado con λ_i

Luego para el determinante 15 tenemos

$$[\, \phi_{j1} \;\; \phi_{j2} \; \cdots \; \phi_{jn}] - [\, s_j(\lambda_1) \;\; s_j(\lambda_2) \; \cdots \; s_j(\lambda_n)] = 0$$

donde ϕ_{ji} es el j-ésimo renglón de I_r asociado con λ_i

llamando $\quad \Phi \triangleq [\, \phi_{j1} \;\; \phi_{j2} \; \cdots \; \phi_{j3}] \quad (n \times m)$

queda $\quad \Phi + W\,H = 0$

y dado que W es invertible (a menos que la estructura de $s_j(\lambda_i)$ tenga elementos nulos y este renglón se use para todos los polos)

$$H = -\,W^{-1}\,\Phi$$

Una vez definida la matriz H, se pueden obtener las matrices $\tilde{A}$ para formar la ecuación de estado del observador y verificar si los polos del observador quedaron bien ubicados.

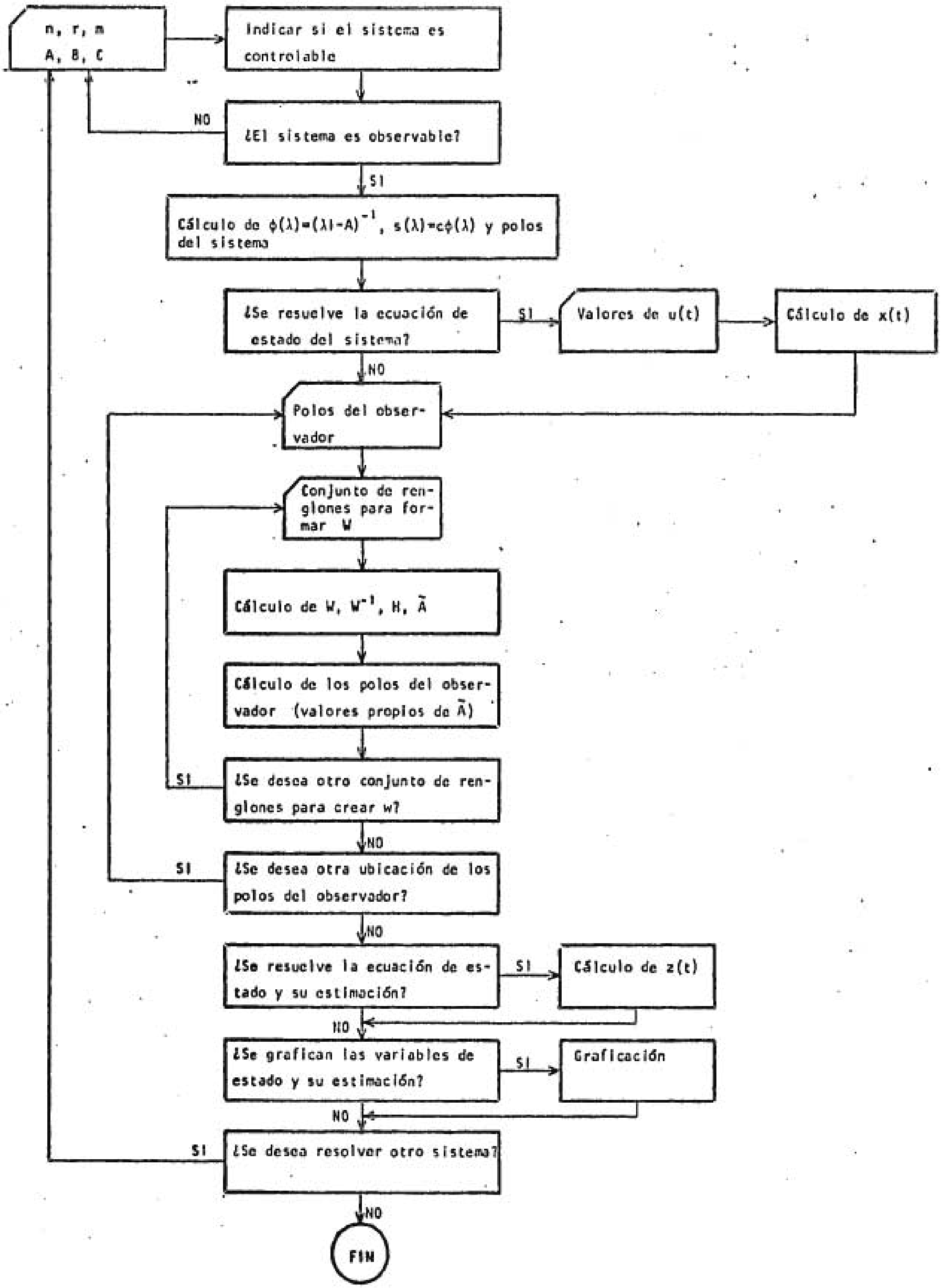

Fig 2. Diagrama de bloques del programa

4. BASE TEORICA DEL PROGRAMA (METODOS NUMERICOS)

El programa OBSERVADOR/UNO que efectúa todas las operaciones indicadas en la fig 2, se creó tomando algunas partes ya hechas, desarrollando totalmente otras e integrando todas ellas.

Dada la magnitud de OBSERVADOR/UNO, 2300 proposiciones, no se describirán los diversos artificios empleados para reducir el tiempo de procesamiento o las localidades de memoria, ya que eso resultaría muy largo y tedioso. En su lugar, se indicará brevemente en que consiste la mayoría de los subprogramas y en ocasiones el método numérico empleado. Así mismo las entradas necesarias, aunque resulte redundante con las indicaciones que da el propio programa en el momento de su ejecución. Esto último se debe a que el programa se desarrolló para usarse en forma interactiva con la computadora a través de algún teletipo.

4.1 Controlabilidad y Observabilidad

Siguiendo el diagrama de bloques encontramos como primer etapa el análisis de la controlabilidad y observabilidad del sistema. Este análisis se hizo de acuerdo con el criterio de Kalman y se efectúa en la subrutina RANTOT la cual sirve para controlabilidad y observabilidad indistintamente.

A RANTOT se alimentan las matrices A y B o las transpuestas de A y C, así como sus dimensiones y se obtiene el rango de la matriz ampliada de controlabilidad u observabilidad.

RANTOT emplea las subrutinas MULMAT, LLAMA y MFGR.

La subrutina MULMAT multiplica dos matrices reales, se usa en bastantes puntos del programa por lo que no se volverá a describir.

La subrutina LLAMA reordena las localidades de memoria para hacer compatibles las subrutinas que funcionan en el sistema IBM, con el sistema Burroughs.

La subrutina MFGR es una parte primordial en esta etapa y también cuando se analiza la degeneración de raíces multiples. Fue tomada del paquete de subrutinas científicas de IBM citada en las referencias.

El método numérico en que se basa es eliminación Gaussiana. Acepta matrices rectangulares, y aun cuando se analice una matriz cuadrada singular, la respuesta es, entre otros datos, el rango. Maneja un error interno en función de la tolerancia del subprograma que la llamó.

Dado que para construir el observador no se requiere que el sistema sea controlable, solo se imprime si el sistema satisface esta condición o no, sin tener ninguna consecuencia en el resto del programa.

4.2 Inversión de la matriz polinomial $(\lambda I - A)$

Esta etapa consta de un conjunto de subrutinas tomadas de H. Elliot citado en las referencias. Este paquete requiere como entrada la matriz polinomial almacenada en un arreglo tridimensional y regresa como resultado de matriz polinomial adjunta (la transpuesta de la matriz de cofactores) y un polinomio, que en esta aplicación es el determinante. En otras aplicaciones será el determinante afectado por un escalar.

De este paquete se tomaron las ideas para construir la subrutina MUMCMP que multiplica una matriz de elementos constantes por otra de elementos polinomiales.

Del determinante de (S I - A) se obtienen los valores propios del sistema mediante la subrutina POLRT tomada del conjunto de IBM. Esta subrutina recibe como dato el polinomio característico y calcula las raíces del mismo por el método de Newton-Raphson. Debe mencionarse la eficiencia la eficiencia de este subprograma, ya que no importa que existan raíces múltiples, complejas o nulas, en la gran mayoría de los casos las localiza sin problemas.

Antes de hacer los cálculos necesarios para formar la matriz W, se revisa si los renglones solicitados no son mayores que los disponibles (el límite esta dado por el número de salidas M), y si la ordenación que se pide en ese momento no se ha empleado anteriormente.

Después se sustituyen los valores de los polos con los renglones indicados y se valúan los polinomios de $s(\lambda)$ para formar W.

La evaluación de los polinomios para un valor determinado de la variable independiente se hace mediante la función HOR, que emplea el método de Horner.

Para calcular la inversa de la matriz W, se emplean las subrutinas LLAMA (explicada antes) y MINV.

La subrutina MINV también fue tomada del grupo de subrutinas de IBM. Efectúa la inversión mediante el método de Gauss-Jordan.

El cálculo de H y $\tilde{A}$, resulta evidente.

4.4 Cálculo de los polos del observador

En esta etapa se emplea la subrutina DETERM, que es parte del paquete de H. Elliot. Se alimenta a la subrutina la matriz polinomial $(\lambda I - \tilde{A})$ y se obtiene como resultado los valores propios de $\tilde{A}$, estos valores se recuperan mediante la única proposición COMMON del programa.

DETERM contiene un llamado a la subrutina POLRT explicada anteriormente.

4.5 Solución de la ecuación de estado del sistema y del observador

De los diferentes métodos que existen para resolver la ecuación

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

se decidió buscar algún método cerrado, es decir, no emplear los métodos aproximados de integración tipo Pickard. Con esta condición, fue necesario calcular la ecuación solución

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

es decir, fue necesario calcular e^{At}. De las diversas formas que hay para calcular la matriz exponencial, se decidió efectuar la diagonalización de la matriz A. La causa principal para elegir este camino fue que en las etapas 4.2 y 4.4 ya se calculaban los valores propios, solo faltaba generar los vectores propios.

Pero un problema previo a la generación de los vectores propios es determinar si la matriz es diagonalizable o no. Esto se superó analizando la degeneración de la matriz al sustituir los polos repetidos (cuando existan) mediante la subrutina MFGR que ya se mencionó.

Afortunadamente los casos de polos repetidos son raros, y es más raro aún que al sustituirlos en la matriz $(\lambda I - A)$ provoquen degeneración incompleta (matriz no diagonalizable).

Después de este análisis, se llama a la subrutina VECPRO, para generar los vectores propios asociados a un valor propio dado, de multiplicidad uno o mayor.

Esta subrutina se formó depurando y adaptando un programa publicado por James, Smith y Wolford.

Actualmente la subrutina VECPRO no acepta valores propios complejos, a pesar de que la base teórica y el método numérico es el mismo, esto se debe a que todo el programa se lleva en doble precisión (salvo la etapa de graficación) y el sistema Burroughs 6700 no maneja variables complejas en doble precisión. Es posible sacrificar la doble precisión con el fin de trabajar en el campo de los complejos, y los cambios necesarios son pocos y sencillos.

Una vez localizados los vectores propios y formada la matriz modal, se procede a su inversión, inversa que teóricamente siempre existe y que puede no encontrarse debido a errores de redondeo principalmente en matrices mal condicionadas.

De aquí la necesidad de comprobar el proceso mediante la pre y postmultiplicación para encontrar la matriz diagonal,

Otro resultado importante de esta diagonalización es el orden de los valores propios sobre la diagonal.

Antes de continuar, es útil notar que aunque existen métodos que calculan simultáneamente los valores y vectores propios de una matriz usualmente requieren que la matriz sea, de alguna forma, particular (en banda, simétrica, positiva definida, etc) y el único método (que se encontró) aplicable a cualquier tipo de matriz fue el de las potencias o iteraciones. Pero este método falla cuando los valores propios son parecidos o iguales (independientemente del signo) y al invertir la matriz modal y multiplicar con la matriz original, se presentan términos no nulos fuera de la diagonal, y mucho mayores que la tolerancia aceptada en el cálculo de los valores y vectores propios, aún con doble precisión.

Para mayor detalle ver los libros de Wilkinson y Faddeeva.

Una vez diagonalizada la matriz de coeficientes, se tiene que el vector de estado esta dado por

$$x(t) = Pe^{Dt} P^{-1} x(o) + Pe^{Dt} \int_{o}^{t} e^{-D\tau} P^{-1} Bu(\tau)\, d\tau$$

donde $A = PDP^{-1}$

para la implantación de esta solución en el programa, se hacen dos hipótesis:

a) Las condiciones inicial son nulas: $x(t=0) = 0$

b) Las excitaciones son conjuntos de funciones escalón de diferentes amplitudes: $u(t) = K\, v(t)$ siendo K un vector y $v(t)$ la función escalón.

de aquí se obtiene

$$x(t) = Pe^{Dt} P^{-1} A^{-1} BK - A^{-1} BK$$

Análogamente se tiene, de la ecuación de estado del observador

$$\dot{z}(t) = \tilde{A}x(t) + Gu(t) + Hc\, x(t)$$

la siguiente solución

$$z(t) = Rse^{Dt}\,H + Re^{\Delta t}(L - SH - M) + R(M - L)$$

donde $\tilde{A} = R\Delta R^{-1}$

Δ matriz diagonal asociada con $\tilde{A}$

R y R^{-1} matriz modal y su inversa.

las demás matrices así como el desarrollo completo para obtener $x(t)$ y $z(t)$ se encuentra en los apéndices.

Una de las características más importantes del programa es que los vectores de estado, del sistema y del observador, se obtienen en expresiones *continuas*, de manera que para calcular el estado en un tiempo t solo hay que valuar las funciones correspondientes. Esto es una ventaja notable porque permite definir el comportamiento de las variables de estado para cualquier tiempo y con cualquier incremento, sin aumentar la memoria necesaria y con variaciones despreciables en el tiempo de procesamiento.

4.6 Graficación

En esta última etapa se emplean las subrutinas FINAL, VALUA y GRAFIC. El subprograma FINAL requiere las matrices que contienen la solución del sistema y del observador, y los tiempos en los que hay que valuar y graficar los estados. Estos tiempos se dan a través del tiempo inicial, el número de puntos que se desea (51 o menos) y el espaciamiento entre ellos.

Estos datos son enviados al subprograma VALUA que calcula dichos valores para todos los tiempos solicitados en todas las variables de estado.

Por último estos datos son enviados a la subrutina GRAFIC, previa consulta con el usuario, para su presentación final a través de una gráfica con los valores numéricos del tiempo, variable del sistema y estimación del observador correspondiente en la margen izquierda.

El límite de 51 puntos para la gráfica, obedece a que la salida proyectada es por teletipo. Es posible, y también deseable, pedir menos de 51 puntos cuando se desea analizar solo un intervalo pequeño.

Al finalizar la etapa de graficación, se ofrece la posibilidad de analizar otro sistema. En caso afirmativo, el programa regresa a la primer etapa, en caso contrario la ejecución termina.

5. BIBLIOGRAFIA

Ogata Katsuhiko, "State space analysis of control systems", Prentice-Hall, New Jersey; 1967

"System/360 Scientific Subroutine Package", Versión III, Programmer's Manual

Elliot, H, "Implementation of computer algorithms related to multivariable system theory" NSF-ENG73-0384601/3

Espinosa, I., "Apuntes del curso. Dinámica de sistemas lineales", DESFI, UNAM, 1978

Faddeeva V. N., "Computational methods of linear algebra" Ed. Dover, 1959

Carnahan B, Luther H A, Wilkes J O, "Applied numerical methods", Ed. John Wiley, 1969

Apostol T M, "Calculus", Vol III, Xerox College Publishing, 1969

Wolovich W. A, "Linear multivariable systems", Ed. Springer-Verlag, 1974

James M. L, Smith G M, Wolford J C, "Métodos numéricos aplicados a la computación digital con Fortran", Ed. Iris, 1967

Frazer, Duncan, Collar, "Elementary matrices", Ed. Cambridge at the University Press, 1965

Chen C T, "Introduction to linear system theory", Ed. Holt, Rinehart, Winston, 1970

Gantmacher, F, "The theory of matrices", Ed. Chelsea, 1960

Wilkinson, J. H, "The algebraic eigenvalue problem", London, Osford, 1965

Acton F. S, "Numerical methods that work", Ed. Harper-Row, 1970

Luenberger, D G, "Observers for multivariable systems", IEEE Trans on Automatic Control, Vol AC-11, No 2, Abril 1966

Gilbert, E. G, "Controllability and observability in multivariable control systems", I.S.I.A.M. Control, Vol 2, No 1, 1963

A1. SOLUCION DE LA ECUACION DE ESTADO (SISTEMA Y OBSERVADOR)

A1.1 Ecuación de estado del sistema

Sea la ecuación

$$\dot{x}(t) = Ax(t) + Bu(t) \tag{A.1}$$

cuya solución está dada por

$$x(t) = e^{At}\, x(o) + e^{At} \int_{o}^{t} e^{-A\tau}\, Bu(\tau)\, d\tau \tag{A.2}$$

considerando las siguientes hipótesis

a) $x(t=0) = 0$

b) $u(t) = Kv(t)$; $v(t)$ = función escalón

K vector de constantes $(n \times 1)$

c) $e^{At} = Pe^{Dt}\, P^{-1}$; D matriz diagonal

P, P^{-1} matriz modal y su inversa

se llega a

$$x(t) = Pe^{Dt} \int_{o}^{t} e^{-D\tau} \, d\tau \; P^{-1} \, BK \qquad (A.3)$$

dado que

$$\int_{o}^{t} e^{-D\tau} \, d\tau = D^{-1}(I - e^{-Dt}) = (I - e^{-Dt}) \, D^{-1}$$

se tiene

$$x(t) = Pe^{Dt}(I - e^{-Dt})D^{-1} \, P^{-1} \, BK$$

$$x(t) = Pe^{Dt} \, D^{-1}P^{-1} \, BK - PD^{-1} \, P^{-1} \, BK \qquad (A.4)$$

$$x(t) = Pe^{Dt} \, P^{-1} \, A^{-1} \, BK - A^{-1} \, BK$$

A1.2 Ecuación de estado del Observador

Sea la ecuación

$$\dot{z}(t) = \tilde{A} \, z(t) + Gu(t) + HC \, x(t) \qquad (A.5)$$

cuya solución es

$$z(t) = e^{\tilde{A}t} \, x(o) + e^{\tilde{A}t} \int_{o}^{t} e^{-\tilde{A}\tau} [\, Gu(\tau) + Hcx(\tau)] \; d\tau \qquad (A.6)$$

considerando las siguientes hipótesis

a) $z(t = 0) = 0$

b) $u(t) = Kv(t)$ igual que antes

c) $e^{\tilde{A}t} = R \, e^{\Delta t} \, R^{-1}$; Δ matriz diagonal

R, R^{-1} matriz modal y su inversa

se llega a

$$z(t) = Re^{\Delta t}\int_0^t e^{-\Delta\tau}\, d\tau\; R^{-1}GK + Re^{\Delta t}\int_0^t e^{-\Delta\tau}R^{-1}Hcx(\tau)\,d\tau$$

$$z(t) = \alpha(t) + \beta(t) \qquad (A.7)$$

donde

$$\alpha(t) = Re^{\Delta t}\int_0^t e^{-\Delta\tau}\, d\tau\; R^{-1}\; GK$$

que es una expresión igual a la ecuación A.3, y efectuando las sustituciones correspondientes

$$\alpha(t) = Re^{\Delta t}\,[I - e^{-\Delta t}]\;\Delta^{-1}\;R^{-1}\;GK$$

$$= Re^{\Delta t}\;\Delta^{-1}R^{-1}GK - R\Delta^{-1}R^{-1}\cdot GK \qquad (A.8)$$

para $\beta(t)$ se tiene:

$$\beta(t) = Re^{\Delta t}\int_0^t e^{-\Delta\tau}\;R^{-1}\;Hcx(\tau)\;d\tau$$

y sustituyendo la ecuación A.4

$$\beta(t) = Re^{\Delta t}\int_0^t e^{-\Delta\tau}R^{-1}HC\,Pe^{D\tau}P^{-1}A^{-1}BK\,d\tau - Re^{\Delta t}\int_0^t e^{-\Delta\tau}R^{-1}HC\,A^{-1}BK\,d\tau$$

$$= \beta_1(t) + \beta_2(t) \qquad (A.9)$$

por comodidad se define la matriz Q (n x n)

$$Q \triangleq R^{-1}\;H\;C\;P$$

$$\therefore\; \beta_1(t) = Re^{\Delta t}\int_0^t e^{-\Delta\tau}\;Qe^{D\tau}\;d\tau\;P^{-1}\;A^{-1}\;BK$$

$$= Re^{\Delta t}\cdot\gamma(t)\cdot P^{-1}\;A^{-1}\;BK \qquad (A.10)$$

donde $\gamma(t) = \int_{o}^{t} e^{-\Delta\tau} Q e^{D\tau} d\tau$

En el integrando de gama se tiene el producto de dos matrices diagonales y una matriz cualquiera, esto se puede aprovechar recordando que cuando una matriz diagonal premultiplica a otra matriz, el renglón i-ésimo de la matriz producto es igual al renglón de la matriz que postmultiplica escalado por el elemento i-ésimo de la matriz diagonal. Analógamente, sucede lo mismo con las columnas cuando la matriz diagonal postmultiplica.

Tomando en cuenta estas propiedades y definiendo los elementos de cada matriz

$$[e^{-\Delta\tau}] = (e^{-\delta i\tau}) \quad \text{para } i = j$$
$$= 0 \quad \text{para } i \neq j$$

$$[Q] = (q_{ij})$$

$$[e^{-D\tau}] = (e^{-di\tau}) \quad \text{para } i = j$$
$$= 0 \quad \text{para } i \neq j$$

se llega a que el término general del integrando es:

$$q_{ij} \cdot e^{-\delta i\tau} \cdot e^{dj\tau} = q_{ij}\, e^{-(\delta i-dj)\tau}$$

integrando este término general se tiene

$$\int_o^t q_{ij}\, e^{-(\delta_i - d_j)\tau}\, d\tau = [\, q_{ij} \frac{e^{-(\delta_i - d_j)\tau}}{-(\delta_i - d_j)}]_o^t$$

$$= \frac{q_{ij}}{d_j - \delta_i} [\, e^{(d_j - \delta_i)t} - 1] \text{ para toda } i, j$$

y haciendo $s_{ij} = \dfrac{q_{ij}}{d_j - \delta_i}$

se llega a

$$\gamma_{ij}(t) = \int_o^t q_{ij}\, e^{-(\delta_i - d_j)\tau}\, d\tau = s_{ij}\, e^{(d_j - \delta_i)t} - s_{ij} \quad \text{(A.11)}$$

analizando el primer sumando de la ecuación A.11 se observa que es posible descomponerlo en un producto, es decir,

$$\gamma_{ij}(t) = e^{-\delta_i t}\, s_{ij}\, e^{d_j t} - s_{ij}$$

colocado en forma matricial se tiene

$$\gamma(t) = e^{-\Delta t} \cdot S \cdot e^{Dt} - S$$

donde $S = (s_{ij})$

sustituyendo $\gamma(t)$ en la ecuación A.10

$$\beta_1(t) = RSe^{Dt}P^{-1}A^{-1}\, BK - Re^{\Delta t}\, SP^{-1}A^{-1}\, BK$$

Para el segundo sumando de la ec A.9 se tiene

$$\beta_2(t) = -\, Re^{\Delta t} \int_o^t e^{-\Delta\tau} d\tau\; Q\, D^{-1}\, P^{-1}\, BK$$

donde se encuentra otra vez la estructura de la ec A.3

YA/SALIU (10/09/78) DSL/OBSERV 1:12 PM TUESDAY, FEB

27 Febrero 1979.

```
* RESET FREE
*RESET SINGLE
*SET LIST
FILE  6=SALL,UNIT=REMOTE,RECORD=22
      INTEGER R
      LOGICAL FLAG/
      REAL * 8 F,FI,T1,T2,DI,TOL ,T3,VALS,VECTS,CT,C,A,T4,RR,RI,DET,AG
     *,IH,W,PR,RRC,RIO,FS,PMENOR,PMAYOR,AT,MA,MO,P,WI,H,HC
     * ,T5,MAI,AD,MOI,AGD,DHP,DMA,Q,S
     *,AK,BK,X,Y,DIPIBK,XH,ZH,DZH
      DIMENSION A(10,10),B(10,10),C(10,10),CT(10,10),FI(10,10,20),T3(10)
     *,F(10,10,20),T1(10,10,20),T2(10,10,20),DI (20) ,VALS(10),VECTS(10)
     *,T4(20),RR(20),RI(20),IPERM(10),IT1(10),IT2(10),IH(10,10),AG(10,10)
     *,W(10,10),PR(10),RRC(20),RIO(20),FS(10,10,20),XH(10),ZH(10)
     *,AT(10,10),MA(10,10),MO(10,10),WI(10,10),H(10,10),HC(10,10)
     * ,T5(10,10),MAI(10,10),AD(10,10),MOI(10,10),AGD(10,10),DZH(10)
     * ,AK(10),BK(10),X(10,12),Y(10,23),DIPIBK(10),Q(10,10),S(10,10)
      COMMON /POLOS/RRC,RIO
      WRITE (6,97)
   97 FORMAT (///1X,130("*")/" *",128X,"*"/" *",41X,
     *"DISENO DE OBSERVADORES LINEALES MULTIVARIABLES",
     *41X,"*"/" *",106X,"J. HORACIO SANDOVAL F *"/
     *" *",106X,"NOV 1977  -  ENE 1978 *"/
     *" *",128X,"*"/1X,130("*")///
     *20X,"SE USA EL METODO DE LUENBERGER PARA CREAR EL OBSERVADOR"/
     *20X,"LOS POLOS DE ESTE SE ASIGNAN POR EL PROCEDIMIENTO DE BROGAN"/
     *20X,"EL PROGRAMA ACEPTA SISTEMAS HASTA DE ORDEN 10, CON 10 ENTRADA
     *S Y 10 SALIDAS"///
     *20X,"ECUACION DE ESTADO: DX(T)/DT = A * X(T) + B * U(T)"//
     *20X,"ECUACION DE SALIDA: Y(T) = C * X(T)"//
```

```
     *33X,"DONDE:   A ES MATRIZ (N*N)"/
     *42X,"B ES MATRIZ (N*R)"/
     *42X,"C ES MATRIZ (M*N)"///)
C
C          LECTURA DE LAS DIMENSIONES DE LAS MATRICES A,B, Y  C.
C
    3 WRITE (6,125)
  125 FORMAT (/2X,"DAME EL ORDEN DEL SISTEMA (N), EL NUMERO DE EXCITACIO
     *NES (R) Y EL DE SALIDAS (M) Y LA APROXIMACION PARA EL PROCESO (TOL
     *)"/)
      READ(5,/) N,R,M,TOL
      NE = 0
      IF (N .LE. 0) CALL EXIT
        SI = 6HSI
C
C          REVISION DE LAS DIMENSIONES
C
      IF (N .GE. R  .AND.  N .GE. M) GO TO 4
      WRITE (6,101) N,R,M
  101 FORMAT (/10X,"MATRICES MAL DEFINIDAS"/
     *10X,"LOS VALORES DE N,R,  Y  M SON",3I5,"  RESPECTIVAMENTE"//
     *10X,"EJECUCION TERMINADA")
      GO TO 3
C
C          LECTURA  (POR RENGLONES) E IMPRESION DE LA  MATRIZ "A"
C
    4 WRITE (6,126)
  126 FORMAT (/2X,"DAME LOS ELEMENTOS DE LA MATRIZ DE COEFICIENTES DEL S
     *ISTEMA (MATRIZ A), POR RENGLONES"/)
      READ (5,/) ((A(I,J),J=1,N),I=1,N)
      WRITE (6,105)
```

```
  105 FORMAT (//20X,"MATRIZ A"//)
      CALL SACAM (A,N,N)
  100 FORMAT (2X,10F9.3)
C
C         LECTURA (POR RENGLONES) E IMPRESION DE LA MATRI  "B"
C
      WRITE (6,127)
  127 FORMAT (/2X,"DAME LOS ELEMENTOS DE LA MATRIZ DE DISTRIBUCION (MATR
     *IZ B), POR RENGLONES"/)
      IF (R .LE. 0) GO TO 5
      READ (5,/) ((B(I,J),J=1,R),I=1,N)
      WRITE (6,102)
  102 FORMAT (//20X,"MATRIZ B"//)
       CALL SACAM (B,N,R)
C
C         LECTURA (POR RENGLONES) E IMPRESION DE LA MATRIZ  "C"
C
    5 WRITE (6,126)
  126 FORMAT (/2X,"DAME LOS ELEMENTOS DE LA MATRIZ DE SALIDA (MATRIZ C),
     * POR RENGLONES"/)
      READ (5,/) ((C(I,J),J=1,N),I=1,M)
      WRITE (6,103)
  103 FORMAT (//20X,"MATRIZ C"//)
      CALL SACAM (C,M,N)
C
C         ASIGNACION DE LA TOLERANCIA (TOL) PARA TODO EL PROCESO
C
      IF (TOL .EQ.0.0) TOL = 1.D-6
C
C         ANALISIS DE LA CONTROLABILIDAD DEL SISTEMA
C
```

```
      IF (R .EQ. 0) GO TO 31
      CALL RANTOT (A,B,N,R,TOL,IRANG)
      IF (IRANG .EQ. N) GO TO 30
      WRITE (6,129) IRANG ,N
  129 FORMAT (//20X,"EL RANGO DE LA MATRIZ AMPLIADA (",I2,") ES MENOR QU
     *E EL ORDEN DEL SISTEMA (",I2," )."/
     *20X,"EL SISTEMA NO ES COMPLETAMENTE CONTROLABLE"/)
      GO TO 31
   30 WRITE (6,130)
  130  FORMAT (//20X,"EL SISTEMA ES COMPLETAMENTE CONTROLABLE"//)
C
C          ANALISIS DE LA OBSERVABILIDAD DEL SISTEMA
C
   31 DO 11 I = 1 , M
      DO 11 J = 1,N
   11 CT(J,I) = C (I,J)
       DO 29 I=1,N
      DO 29 J=1,N
   29 AT(I,J) = A(J,I)
      CALL RANTOT (AT,CT,N,M,TOL,IRANGO)
      IF (IRANGO .EQ. N) GO TO 14
      WRITE (6,106) IRANGO,N
  106 FORMAT (//20X,"EL RANGO DE LA MATRIZ AMPLIADA (",I2,") ES MENOR QU
     *E EL ORDEN DEL SISTEMA (",I2,")."/
     *20X,"EL SISTEMA NO ES COMPLETAMENTE OBSERVABLE"/
     *20X," EJECUCION TERMINADA"/)
      CALL EXIT
   14 WRITE (6,104)
  104 FORMAT (//20X,"EL SISTEMA ES COMPLETAMENTE OBSERVABLE"//)
C
C          FORMACION DE LA MATRIZ POLINOMIAL (SI-A) PARA SU USO EN EL
```

```
C          PROCESO DE LUENBERGER Y PARA OBTENER LOS VALORES PROPIOS DEL
C          SISTEMA ORIGINAL
C
      CALL LIMPIA(F,10,10,20)
      CALL LIMPIA (F1,10,10,20)
      DO 12 J=1,N
      DO 12 I=1,N
   12 F(J,I,1) = -A(I,J)
      DO 13 I=1,N
   13 F(I,I,2) = 1.0
      WRITE (6,111)
  111 FORMAT(/10X,"ELEMENTOS DE LA MATRIZ CUADRADA (SI-A)")
C
C          IMPRESION DE LA MATRIZ POLINOMIAL (SI-A)
C
      CALL PRNT (F,N,N,20,TOL)
C
C          INVERSION DE (SI-A)
C
      CALL INVERT (F,FI,T1,T2,DI,N,GRADIV,20,TOL,ISIN)
      IF (ISIN .NE. 1) GO TO 17
      WRITE (6,144)
  144 FORMAT (///30X,"LA MATRIZ (SI-A) ES SINGULAR"//)
      GO TO 3
   17 WRITE (6,112)
  112 FORMAT (/10X,"ELEMENTOS DE LA MATRIZ INVERSA DE (SI-A)")
C
C          IMPRESION DE LA MATRIZ INVERSA DE (SI-A) Y DE SU DETERMINANTE
C
      CALL PRNT (FI,N,L,20,TOL)
      WRITE (6,113)
```

```
  113 FORMAT (/10X,"DETERMINANTE DE (SI-A)"/
     *10X,"(POLINOMIO DE GRADO N)")
      CALL PRNT3 (DI,20,TOL)
C
C          VALORES PROPIOS DEL SISTEMA ORIGINAL O RAICES DEL DETERMINANTE
C          DE (SI-A)
C
      CALL  POLRT (DI,T4,N,RR,RI,IER)
      GO TO (1,50,51,52,50),IER + 1
   50 WRITE (6,108) N
  108 FORMAT (//20X,"EL VALOR DE N (",I3," ) ES MENOR QUE 1"/
     *20X,"EJECUCION TERMINADA")
      CALL EXIT
   51 WRITE (6,109) N
  109 FORMAT (//20X,"EL VALOR DE N (",I3," ) ES MAYOR QUE 36 (CAPACIDAD
     *DEL POLRT)"/20X," EJECUION TERMINADA")
      CALL EXIT
   52 WRITE (6,110)
  110 FORMAT (//20X,"EL METODO PARA LOCALIZAR LAS RAICES NO CONVERGE"/
     *20X,"EJECUCION TERMINADA")
      CALL EXIT
    1 WRITE (6,107) (RR(I),RI(I),I=1,N)
  107 FORMAT (/20X,"VALORES PROPIOS DEL SISTEMA"//
     *20X,"PARTE REAL",10X,"PARTE IMAG."/( 6X,2E25.5))
      WRITE (6,145)
      FLAG5 =0H
  145 FORMAT (/10X,"DESEAS RESOLVER LA ECUACION DE ESTADO DEL SISTEMA (S
     *I) (NO)"/)
      READ (5,120) FLAG5
      IF (FLAG5  .NE. SI) GO TO 16
      CALL MODAL (A,RR, RI,N,TOL,MA,IND)
```

```
      IF (IND .NE. 3) GO TO 3
C
C           MA ES LA MATRIZ MODAL DEL SISTEMA (ASOCIADA CON A)
C
      WRITE (6,133)
  133 FORMAT (//20X"MATRIZ MODAL DE A (MA)"//)
      CALL SACAM (MA,N,N)
      DO 8 I=1,N
      DO 8 J=1,N
    8 MAI ( J,I)=MA(J,I)
      CALL LLAMA (MAI,N,N,10,10,1)
      CALL MINV (MAI,N,DMA,IT1,IT2)
      CALL LLAMA (MAI,N,N,10,10,2)
      WRITE (6,134)
  134 FORMAT (//20X,"INVERSA DE MA (MAI)"//)
      CALL SACAM (MAI,N,N)
      CALL MULMAT (MAI,A,T5,N,N,N)
      CALL MULMAT (T5,MA,AD,N,N,N)
      WRITE (6,135)
  135 FORMAT (//20X,"MATRIZ DIAGONAL ASOCIADA CON A (AD)"//)
      CALL SACAM (AD,N,N)
   38 WRITE (6,160)
  160 FORMAT (/2X,"DAME LOS VALORES INICIALES DEL SISTEMA"/)
      READ (5,*) (XH(I),I=1,N)
      WRITE (6,161)
  161 FORMAT (/20X,"VALORES INICIALES DEL VECTOR DE ESTADO EL SISTEMA"/)
      CALL SACAM (XH,N,1)
C
C     P**(-1) *X(0) = T3
C
      CALL MULMAT(MAI,XH,T3,N,N,1)
```

```
C      ASIGNACION DE LOS POLOS DEL SISTEMA A LAS MATRICES SOLUCION
      DO 39 J=1,N
      X(J,11) =AD(J,J)
      Y(J,21) = AG (J,J)
C    CALCULO DE LA MATRIZ DE COEFICIENTES DE LA SOLUCION HOMOGENEA
      DO 39 I=1,N
   39 T5(I,J) =MA(I,J) *T3(J)
      FLAG9 = 6H
      WRITE (6,162)
  162 FORMAT (/10X,"DESEAS IMPRIMIR LA SOLUCION HOMOGENEA DEL SISTEMA  (
     *SI) (NO)"/)
      READ (5,120) FLAG9
      IF (FLAG9 .NE. SI) GO TO 37
      WRITE (6,163)
  163 FORMAT (//20X,"SOLUCION HOMOGENEA DE LA ECUACION DE ESTADO DEL SIS
     *TEMA  X(T) = X1 * EXP(Y2*T)"/)
      WRITE (6,149)
      CALL SACAM (T5,N,N)
      WRITE (6,150)
      CALL SACAM (X(1,11),N,1)
   37 WRITE (6,164)
  164 FORMAT (/10X,"DESEAS OTRO CONJUNTO DE VALORES INICIALES DEL SISTEM
     *A   (SI) (NO)"/)
      FLAG10 =6H
      READ (5,120)FLAG10
      IF (FLAG10 .EQ. SI) GO TO 38
        FLAG7 = .FALSE.
   15 WRITE (6,146)
  146 FORMAT (/2X,"DAME LAS AMPLITUDES DE LAS FUNCIONES ESCALON DE LA EX
     *CITACION"/)
      READ (5,/) (AK(I),I=1,P)
```

```
      WRITE (6,147)
  147 FORMAT (/20X,"VECTOR DE AMPLITUDES DE LAS FUNCIONES ESCALON DE LA
     *EXCITACION"/)
      CALL SACAM (AK,R,1)
      CALL MULMAT (B,AK,BK,N,R,1)
C           P**(-1) * B * K
      CALL MULMAT (MAI,BK,T3,N,N,1)
      IF (FLAG7) GO TO 53
C             FLAG7=TRUE, INDICA  QUE DEBE SALTAR PARA CUANDO SE PIDE MAS
C           DE UN CONJUNTO DE ENTRADAS (EVITA DOBLE INVERSION DE AD)
      FLAG7 = .TRUE.
      DO 7 I=1,N
    7 AD(I,I) =1.0/AD(I,I)
C          D**(-1) * P**(-1) * B * K
   53 CALL MULMAT (AD,T3,DIPIBK,N,N,1)
C
C           TERMINO INDEPENDIENTE DE X(T):  P * D**(-1) *P**(-1)*B*K
C
      CALL MULMAT (MA,DIPIBK,X(1,12),N,N,1)
      DO 6 J=1,N
      X(J,12) = - X(J,12)
      DO 6 I=1,N
    6 X(I,J) =MA(I,J) * DIPIBK(J) + T5(I,J)
      WRITE (6,148)
  148 FORMAT (//20X,"SOLUCION GENERAL DE LA ECUACION DE ESTADO DEL SISTE
     *MA: X(T) = X1 * EXP(X2*T) + X3"/)
      WRITE (6,149)
  149 FORMAT(//20X,"MATRIZ DE COEFICIENTES DE LA SOLUCION X(T)  (X1)"/)
      CALL SACAM (X,N,N)
      WRITE (6,150)
  150 FORMAT (//20X,"VECTOR DE COEFICIENTES DEL EXPONENTE DE LA SOLUCION
```

```
     * X(T)  (X2)"/)
      CALL SACAM (X(1,11),N,1)
      WRITE (6,151)
  151 FORMAT(//20X,"VECTOR DE TERMINOS CONSTANTES DE LA SOLUCION X(T)  (
     *X3)"/)
      CALL SACAM (X(1,12),N,1)
C
C          MULTIPLICACION DE LA MATRIZ DE ELEMENTOS CONSTANTES  C POR LA
C          INVERSA DE LA MATRI  POLINOMIAL (SI-A)
C
      WRITE (6,152)
      FLAG6 =6H
  152 FORMAT (/10X,"DESEAS OTRO CONJUNTO DE ENTRADAS  (SI) (NO)"/)
      READ (5,120) FLAG6
      IF (FLAG6 .EQ.SI) GO TO 15
   14 CALL MUMCMP (C,FI,F,M,L,NA,NB,N,20,TOL)
      WRITE (6,114)
  114 FORMAT (//10X,"ELEMENTOS DE LA MATRIZ POLINOMIAL  S")
      CALL PRNT (F,M,N,20,TOL)
C
C          LECTURA DE LA POSICION DE LOS POLOS DEL OBSERVADOR (ESPERADA)
C
   23 WRITE(6,115)
      NL =0
  115 FORMAT(/2X,"DAME LA UBICACION DE LOS POLOS DEL OBSERVADOR"/)
      READ(5,/) (PN(I),I=1,N)
C
C          IMPRESION Y ANALISIS DE LOS POLOS DEL OBSERVADOR
C
      PMENOR = RR (1)
      DO 26 I = 2,N
```

```
   26 PMENOR = DMIN1 (PMENOR,PR(I))
      PMAYOR = PR(1)
      DO 27 I=2,N
   27 PMAYOR =DMAX1 (PMAYOR,PR(I))
      IF (PMENOR .GT. PMAYOR ) GO TO 22
      WRITE (6,123)  PMENOR,PMAYOR
  123 FORMAT (/10X,"ALGEBRAICAMENTE,EL MENOR POLO DEL SISTEMA ES",F8.4/
     *10X,"Y EL MAYOR DEL OBSERVADOR ES",F8.4/)
      GO TO 23
C
C         LECTURA DE LOS RENGLONES SELECCIONADOS PARA FORMAR LA MATRIZ
C         CALCULAR W, AL SUSTITUIR LOS VALORES DE LOS NUEVOS POLOS
C
   22 WRITE (6,116)
  116 FORMAT (2X,"DAME EL ORDEN DE LOS RENGLONES PARA CREAR  W"/)
      READ (5,/) (IREN(I),I=1,N)
      CALL LIMPIA (W,10,10,1)
      CALL LIMPIA (IM,10,10,1)
      CALL CREAT (F,M,N,DI ,IREN,PR,W,IR,TOL,KK)
      DO 32 I=1,10
      DO 32 J=1,10
   32 WI(I,J)=W(I,J)
      GO TO (22,23,28),KK
   28 WRITE (6,132)
  132 FORMAT (//20X,"MATRIZ W"//)
      CALL SACAM (W,N,N)
C
C         CALCULO DE LA MATRIZ INVERSA DE W (WI).
C
      CALL LLAMA (WI,N,N,10,10,1)
      CALL MINV (WI,N,DET,IT1,IT2)
```

```
      IF (DABS(DET) .GT. 0.  ) GO TO 24
      WRITE (6,117)
  117 FORMAT (/10X,"LA MATRIZ  W  RESULTO SINGULAR"//)
      GO TO 25
   24 CALL LLAMA (WI,N,N,10,10 ,2)
      WRITE (6,131)
  131 FORMAT (//20X,"MATRIZ INVERSA DE W (WI)"//)
      CALL SACAM (WI,N,N)
      CALL MULMAT (WI,IH,H,N,N,R)
      CALL MULMAT (H,C,HC,N,R,N)
      DO 2 I = 1,N
      DO 2 J = 1,N
      HC (I,J) = - HC (I,J)
    2 AG(I,J) = A(I,J) - HC (I,J)
      WRITE (6,141)
  141 FORMAT (///30X,"ECUACION DE ESTADO DEL OBSERVADOR: DZ(T)/DT = AG *
     *  (T) +  G * U(T)  +  HC * X(T)"//)
      WRITE (6,136)
  136 FORMAT (//20X,"MATRIZ DE COEFICIENTES DEL OBSERVADOR (AG)"//)
       CALL SACAM (AG,N,N)
      WRITE (6,142)
  142 FORMAT (//20X,"MATRIZ DE DISTRIBUCION ASOCIADA CON U(T)  (G)"//)
      CALL SACAM (B,N,R)
      WRITE (6,143)
  143 FORMAT (//20X,"MATRIZ DE DISTRIBUCION ASOCIADA CON X(T)  (HC)"//)
      CALL SACAM (HC,N,N)
C
C         CALCULO DE LOS VALORES PROPIOS DE AG
C
      CALL LIMPIA (F3,10,10,20)
      CALL LIMPIA (T2,10,10,20)
```

```
      CALL LIMPIA (RRO,20,1,1)
      CALL LIMPIA (RIO,20,1,1)
      DO 20 I = 1,10
      DO 21 J = 1,10
   21 FS(J,I,1) = -AG(I,J)
   20 FS (I,I,2) = 1.0
      CALL DLTERM (FS,T2,N,20,TOL)
      WRITE (6,124) (RRO(I) ,RIO(I),I=1,N)
  124 FORMAT (/20X,"VALORES PROPIOS DEL OBSERVADOR"//
     *  30X,"PARTE REAL",10X,"PARTE IMAG."/ ,(15X,2E25.5))
   25 WRITE (6,118)
  118 FORMAT(/10X,"SELECCIONAS UN NUEVO CONJUNTO DE RENGLONES (SI) (NO)"
     */)
      READ (5,120) FLAG1
  120 FORMAT (A6)
      IF (FLAG1 .EQ. SI  ) GO TO 22
      IF (FLAG5 .NE. SI ) GO TO 10
      WRITE (6,137)
  137  FORMAT(/10X,"DESEAS RESOLVER LA ECUACION DE ESTADO DEL OBSERVADOR
     * (SI) (NO)"/)
      READ (5,120) FLAG4
      IF (FLAG4 .NE. SI) GO TO 10
C
C         MO ES LA MATRIZ MODAL DEL OBSERVADOR (ASOCIADA CON AG)
C
      CALL MODAL (AG,RRO,RIO,N,TOL,MO,IND)
      WRITE (6,138)
  138 FORMAT (//20X,"MATRIZ MODAL DE AG (MO)"//)
      CALL SACAM (MO,N,N)
      DO 9 I=1,N
      DO 9 J=1,N
```

```
    9 MDI (J,I)= MD(J,I)
      CALL LLAMA (MDI,N,N,10,10,1)
      CALL MINV (MDI,N,DMD,IT1,IT2)
      CALL LLAMA (MDI,N,N,10,10,2)
      WRITE (6,139)
  139 FORMAT (//20X,"INVERSA DE MD (MDI)"//)
      CALL SACAM (MDI,N,N)
      CALL MULMAT (MDI,AG,T5,N,N,N)
      CALL MULMAT (T5,MD,AGD,N,N,N)
      WRITE (6,140)
  140 FORMAT (//20X,"MATRIZ DIAGONAL ASOCIADA CON AG (AGD)"//)
      CALL SACAM (AGD,N,N)
      DO 40 I=1,N
      Y(I,23) = X(I,12)
   40 Y(I,22) =AGD(I,I)
   19 WRITE (6,165)
  165 FORMAT (/2X,"DAME LOS VALORES INICIALES DEL OBSERVADOR"/)
      READ (5,/) (ZH(I),I=1,N)
      WRITE (6,166)
  166 FORMAT (/20X,"VALORES INICIALES DEL VECTOR DE ESTADO DEL OBSERVADO
     *R"/)
      CALL SACAM (ZH,N,1)
      DO 41 I=1,N
   41 DZH(I) = ZH(I) - XH(I)
      CALL MULMAT (MDI,DZH,T3,N,N,1)
      DO 42 J=1,N
      DO 42 I=1,N
      Y(I,10+J) =MD (I,J) * T3(J)
   42 Y(I,J) = X(I,J)
      WRITE (6,154)
  154 FORMAT (//20X,"SOLUCION DE LA ECUACION DE ESTADO DEL OBSERVADOR  Z
```

```
     *(T) = Z1 * EXP(Z2*T) + Z3 * EXP(Z4*T) + Z5"/)
      WRITE (6,153)
  153 FORMAT (//20X,"MATRIZ DE COEFICIENTES DE LA SOLUCION Z(T)  (Z1)"/)
      CALL SACAM (Y,N,N)
      WRITE (6,155)
  155 FORMAT (//20X,"VECTOR DE COEFICIENTES DEL EXPONENTE DE LA SOLUCION
     * Z(T)  (Z2)"/)
      CALL SACAM (Y(1,21),N,1)
      WRITE (6,156)
  156 FORMAT (//20X,"MATRIZ DE COEFICIENTES DE LA SOLUCION Z(T)  (Z3)"/)
      CALL SACAM (Y(1,11),N,N)
      WRITE (6,157)
  157 FORMAT (//20X,"VECTOR DE COEFICIENTES DE LA SOLUCION Z(T)  (Z4)"/)
      CALL SACAM (Y(1,22),N,1)
      WRITE (6,158)
  158 FORMAT(//20X,"VECTOR DE TERMINOS CONSTANTES DE LA SOLUCION Z(T)"/)
      CALL SACAM (Y(1,23),N,1)
      FLAG11 = 6H
      WRITE (6,167)
  167 FORMAT (/10X,"DESEAS OTRO CONJUNTO DE VALORES INICIALES DEL OBSERV
     *ADOR  (SI) (NO)"/)
      READ ( 5,120) FLAG11
      IF ( FLAG11 .EQ. SI) GO TO 18
        FLAG9=6H
      WRITE (6,159)
  159 FORMAT (/10X,"DESEAS GRAFICAR LAS SOLUCIONES (SI) (NO)"/)
      READ (5,120) FLAG9
      IF (FLAG9 .EQ. SI) CALL FINAL (X,Y,N)
   18 WRITE (6,119)
  119 FORMAT (/10X,"SELECCIONAS OTRO CONJUNTO DE POLOS (SI) (NO)"/)
      READ (5,120) FLAG2
```

```
      IF (FLAG2 .EQ. SI ) GO TO 23
      FLAG3 = 6H
      WRITE (6,121)
  121 FORMAT (/10X,"DESEAS RESOLVER OTRO SISTEMA (SI) (NO)"/)
      READ (5,120) FLAG3
      IF (FLAG3 .EQ. SI ) GO TO 3
      WRITE (6,122)
  122 FORMAT (20X,"EJECUCION TERMINADA"/)
      CALL EXIT
      END
      SUBROUTINE FINAL (X,Y,M)
      REAL * 8 X,Y,TI,DT
      DIMENSION X(10,12),Y(10,23),A(10,51),B(10,51),CT(51),V1(51),V2(51)
      SI = 6HSI
      WRITE (6,100)
  100 FORMAT (//20X,"SE GRAFICAN LAS VARIABLES DE ESTADO POR PARES (LAS
     *DEL SISTEMA Y LA ESTIMACION"/
     *20X,"CORRESPONDIENTE DEL OBSERVADOR) A INTERVALO CONSTANTE"/
     *20X,"LAS GRAFICAS PUEDEN SER DE 51 PUNTOS O MENOS."/
     *20X,"EL EJE DE LOS TIEMPOS X(T)=Z(T)=0.0  ESTA INDICADO POR LA LET
     *RA I"/)
    1 WRITE (6,101)
  101 FORMAT (/10X,"DAME EL TIEMPO INICIAL ,EL INCREMENTO Y EL NUMERO DE
     * PUNTOS QUE DESEAS"/)
      READ (5,/) TI,DT,N
      IF (TI .GE. 0.0  .AND.  DT .GT. 0.0) GO TO 5
      WRITE (6,106)
  106 FORMAT (30X,"ERROR: EL TIEMPO INICIAL ES NEGATIVO O EL INCREMENTO
     *NO ES POSITIVO"/)
      GO TO 1
    5 CALL VALUA (X,Y,N,TI,DT,A,B, CT,MP)
```

```
      DO 2 I=1,N
      DO 3 J=1,NP
    3 V1(J) = A(I,J)
      DO 4 J=1,NP
    4 V2(J) = B(I,J)
      FLAG2 = 6H
      WRITE (6,103) I,I
  103 FORMAT (/10X,"DESEAS GRAFICAR LAS VARIABLES   X",I2,"   Y   Z"
     *,I2,"       (SI) (NO)"/)
      READ (5,120) FLAG2
      IF (FLAG2 .NE. SI ) GO TO 2
      WRITE (6,105) I,I
  105 FORMAT (//55X,"VARIABLES   X",I2,"(*) DEL SISTEMA Y   Z",I2,
     *"(O) DEL OBSERVADOR")
      WRITE (6,104) I,I
  104 FORMAT (" TIEMPO        Y",I2,8X,"Z",I2)
      CALL GRAFIC (V1,V2,CT,MP)
    2 CONTINUE
      FLAG1 = 6H
      WRITE (6,102)
  102 FORMAT (/10X,"DESEAS CALCULAR PARA OTROS TIEMPOS (SI) (NO)"/)
      READ (5,120) FLAG1
  120 FORMAT (A6)
      IF (FLAG1 .EQ. SI) GO TO 1
      RETURN
       END
      SUBROUTINE SACAR (A,N,M)
      REAL * 8 A(10,10)
      DO 1 I=1,N
    1 WRITE (6,10)(A(I,J),J=1,M)
   10 FORMAT (10F12.4)
```

```
      RETURN
      END
      SUBROUTINE VALUA (X,Y,N,TI,DT,V,W,CT,NP)
      REAL * 8 X,Y,TI,DT,T,E1,E2,A
      DIMENSION X(10,12),Y(10,23),V(10,51),W(10,51),CT(51),E1(10),E2(10)
      T = TI
      DO 1 K=1,NP
      DO 2 I=1,N
      E1(I) = DEXP(Y(I,21)*T)
    2 E2(I) = DEXP (Y(I,22)*T)
      DO 3 I=1,N
      A = 0.0
      DO 4 J=1,N
    4 A = A+ X(I,J) * E1(J)
      V(I,K) = SNGL(A+X(I,12))
      A = 0.0
      DO 5 J=1,N
    5 A = A + Y(I,J) *E1(J) + Y(I,10+J) * E2(J)
    3 W(I,K) = SNGL (A + Y(I,23))
      CT (K) =SNGL (T)
    1 T = T +DT
      RETURN
      END
      SUBROUTINE GRAFIC (X,Y,Z,N)
       DIMENSION X(1),Y(1),Z(1),FILA (101)
      C = X(1)
      G = C
      DO 1 I=1,N
       C = AMIN1 (C,X(I),Y(I))
    1 G =  AMAX1 (G,X(I),Y(I))
      WRITE (6,101) C,G
```

```
  101 FORMAT (30X,"(MINIMO)",F11.4,64X,F11.4,"(MAXIMO)")
      ES = (G-C)/100.
      EJE = G*C
      L = ABS (C/ES) + 1.5
      RES = 1.0/ES
      DO 3 I=1,101
    3 FILA(I) = 1H
      K=1
      J=1
      DO 4 I=1,N
      FILA (J) =1H
      FILA (K) =1H
      FILA (1) = 1H+
      FILA (101) = 1H+
       IF (EJE .LE. 0.0 ) FILA (L) = 1HI
      J = (X(I) -C) *RES +1.5
      K = (Y(I)-C) *RES +1.5
      FILA (J) =1H*
      FILA (K) =1HO
    4 WRITE (6,100) Z(I),X(I),Y(I),FILA
  100 FORMAT (1X,F6.3,2F11.4,2X,101A1)
      RETURN
      END
      SUBROUTINE MODAL (A,VALR,VALI,N,TOL,Y,IND)
      REAL *8 A,VALR,VALI,TOL,VEC,T1,T2,T3,T4,Y,X   ,AA
C
C         ESTA SUBROUTINE GENERA LA MATRIZ MODAL DE UNA MATRIZ A .
C         REQUIERE LA MATRIZ A Y LOS VALORES PROPIOS DE LA MISMA.
C         DETECTA SI HAY VALORES REPETIDOS, COMPLEJOS, Y SI ES POSIBLE
C         GENERAR TODO EL ESPACIO DE N DIMENSIONES CUANDO HAY
C         MULTIPLICIDAD EN LAS RAICES.
```

```
C
      DIMENSION A(10,10),VALR(10), VALI(10),AA(10,10),T2(10),T3(10) ,
     $M(10),T4(10),X(10,10),Y(10,10)
      IND =0
C
C     DETECCION DE LA PARTE IMAGINARIA DE LAS RAICES.
C
      DO 1 I=1,N
      IF (DABS(VALI(I)).GE. TOL ) GO TO 10
    1 CONTINUE
C
C     ORDENACION DE LAS RAICES.
C
      NN = N-1
      DO 2 I = 1, NN
      T1 = VALR(I)
      DO 3 J = I+1,N
      IF (T1 .LE. VALR(J)) GO TO 3
      VALR(I)= VALR(J)
      VALR(J) = T1
      T1 = VALR(I)
    3 CONTINUE
    2 CONTINUE
C
C     DEFINICION DE LA MULTIPLICIDAD DE CADA RAIZ Y ELIMINACION DE
C     LOS VALORES REPETIDOS.
C
      T2(1) = VALR(1)
      DO 5 I=1,N
    5 M(I) = 1
      K = 1
```

```
      DO 6 I = 2,N
      IF (DABS(VALR(I-1)-VALR(I)) .LT. TOL ) GO TO 4
      K = K + 1
      T2 (K) = VALR (I)
      GO TO 6
    4 M(K) = M(K) + 1
    6 CONTINUE
C
C         SI K ES IGUAL A N, IMPLICA QUE NO HAY VALORES REPETIDOS.
C
      IF ( K .GE. N) GO TO 11
      DO 7 I=1,K
      IF (M(I ) .LE. 1 ) GO TO 7
C
C         ANALISIS DE LA DEGENERACION EN POLOS REPETIDOS.
C
      DO 8 J= 1,N
      DO 9 L=1,N
    9 AA(J,L) = -A(J,L)
    8 AA(J,J) = AA(J,J)    + T2(I)
      CALL LLAMA (AA,N,I ,10,10,1)
      CALL MEGR (AA,N,N,TOL,IR,T3,T4)
      IF (IR + M(I) .EQ. N) GO TO 7
      WRITE (6,100) T2(I),M(I),IR
  100 FORMAT  (///40X,"MATRIZ NO DIAGONALIZABLE"/
     +45X,"EL VALOR PROPIO",F12.4," TIENE MULTIPLICIDAD",I4/
     $45X,"Y LA MATRIZ (SI-A) ASOCIADA A ESTE VALOR ES DE RANGO",I5/
     $45X,"ES DECIR, POSEE DEGENERACION INCOMPLETA."/)
      IER = 1
      RETURN
    7 CONTINUE
```

```
C
C          GENERACION DE LOS VECTORES PROPIOS Y FORMACION DE LA MATRI
C          MODAL Y.
C
    11 J2 = 0
       J1 = 1
       DO 12 I= 1,K
       J2 = J2 + M(I)
       DO 13 J=1,N
       DO 13 L=1,N
    13 AA(J,L) = A(J,L)
C
C          X ES LA MATRIZ QUE TRANSPORT LOS VECTORES PROPIOS  DE VECPRO
C          A MODAL.
C
       CALL VECPRO (AA,T2(I),N,TOL,X )
       J3 = 0
C
C          ALMACENAMIENTO DE LOS VECTORES PROPIOS EN LA MATRIZ MODAL  Y.
C
       DO 15 J = J1,J2
       J3 = J3 + 1
       DO 15 L=1,N
    15 Y(L,J) = X(L,J3)
    12 J1 = J2 + 1
       IND = 3
       RETURN
    10 WRITE (6,101)
   101 FORMAT (//40X,"VALORES PROPIOS COMPLEJOS"/
      *45X,"EL PROGRAMA AUN NO CONTEMPLA ESTA POSIBILIDAD"//)
       IND = 2
```

```
      RETURN
      END
      SUBROUTINE VECPRO (A,VAL,N,TOL,X)
C
C         DETERMINA LOS VECTORES PROPIOS LINEALMENTE INDEPENDIENTES ASO-
C         CIADOS A UN VALOR PROPIO DADO.
C                                           JAMES,SMITH,WOLFORD (PAG 278)
C                                           MODIF E IMPLANT POR H.SANDOVAL
C                                                             DIC 1977
C
      REAL *8 A,VAL,TOL,GDE ,X ,B ,AUX
      DIMENSION A(10,10),B(10,10),X(10,10),IX(10)   ,AUX(11,10)
C
C      CALCULO DE LOS ELEMENTOS EN LA DIAGONAL DE LA MATRIZ DE COEFICIENT
C
      DO 7 I=1,N
    7 A(I,I) = A(I,I) -VAL
C
C         REGISTRO DEL ORDEN INICIAL DE LAS COMPONENTES DEL VECTOR
C         CARACTERISTICO.
C
      DO 8 I = 1,N
    8 IX(I) = I
      K= N
    9 M= N-K+1
C
C         LOCALIZACION DEL MAYOR ELEMENTO PIVOTE.
C
      JJ= 1
      II=1
      GDE =DABS(A(1,1))
```

```
      DO 11 J=1,K
      DO 11 I=1,K
      IF (GDE.GE. DABS(A(I,J))) GO TO 11
      JJ= J
      II= I
      GDE = DABS (A(I,J))
   11 CONTINUE
C
C           VERIFICACION Y EN SU CASO EJECUCION, DEL INTERCAMBIO DE
C           RENGLONES Y/O COLUMNAS.
C
      IF ( EPS .EQ. 0.0 ) EPS = TOL * GDE
      IF ( II .LE. 1) GO TO 14
      DO 13 J=1,K
      TEMP = A(II,J)
      A(II,J) = A(1,J)
   13 A(1,J) = TEMP
   14 IF (JJ.LE. 1) GO TO 17
C           INTERCAMBIO DE COLUMNAS Y REGISTRO DEL MISMO.
      TEMP = IX(M)
      L = JJ + N - K
      IX(M) = IX(L)
      IX(L) = TEMP
      DO 16 I = 1,K
C
C           ELIMINACION DE GAUSS-JORDAN.
C
      TEMP = A(I,JJ)
      A(I,JJ) = A(I,1)
   16 A(I,1) = TEMP
   17 IF (DABS(A(1,1)) .LE. EPS) GO TO 23
```

```
      TEMP = 1.0/A(1,1)
   18 DO 20 J=2,K
      DO 20 I=2,N
      B(I-1,J-1)= A(I,J) - A(1,J)*A(I,1) * TEMP
      IF (DABS(B(I-1,J-1)) .GE. EPS *DABS(A(I,J)))GO TO 20
      B(I-1,J-1) = 0.0
   20 CONTINUE
      DO 21 J= 2,K
   21 B(N,J-1)= A(1,J) *TEMP
      N=N-1
      IF ( N.LE. 0) GO TO 31
      DO 22 J=1,K
      DO 22 I=1,N
   22 A(I,J) = B(I,J)
      GO TO 9
C
C         OBTENCION DE LOS VECTORES LINEALMENTE INDEPENDIENTES A PARTIR
C         DE LA MATRIZ A.
C
   23 DO 24 J=1,K
   24 A(J,J) = -1.
      IF ( N.LT. 10) GO TO 59
      DO 57 J = 1,10
      DO 57 I = 1,10
   57 AUX (I,J) = A(I,J)
      DO 60 MM = 1,K
      DO 60 J = 1,K
      AUX(N+1,J) = AUX(1,J)
      DO 60 I=1,N
   60 AUX(I,J) = AUX(I+1,J)
      DO 62 J=1,10
```

```
      DO 62 I=1,N
   62 A(I,J) = AUX(I,J)
      GO TO 61
   59 DO 25 NN= 1,K
      DO 25 J=1,K
      A(N+1,J) = A(1,J)
      DO 25 I=1,N
   25 A(I,J) = A(I+1,J)
   61 DO 27 J=1,K
      DO 27 I=1,N
      DO 27 NUM=1,N
      IF (IX(NUM) .NE.I) GO TO 27
      X(I,J) = A (NUM,J)
   27 CONTINUE
C
C            IMPRESION DE LOS VECTORES PROPIOS LINEALMENTE INDEPENDIETSTES.
C
C     WRITE (6,100)
  100 FORMAT (/10X,"EL(LOS) VECTOR(ES) PROPIO(S) LINEALMENTE INDEPENDIEN
     *TE(S) ASOCIADO(S) CON",F10.4," ES(SON):"/)
C     DO 29 I=1,N
C  29 WRITE(6,101) (A(I,J),J=1,K)
  101 FORMAT (10X,10F12.4)
      RETURN
   31 WRITE(6,102)
  102 FORMAT (//20X,"EL RANGO DE (SI-A) ES IGUAL AL ORDEN DE A"/
     *20X,"EJECUCION TERMINADA")
      CALL EXIT
      END
      FUNCTION HON (A,N,X)
C
```

```
C      EVALUA EL POLINOMIO A(1)+ A(2)*X+ ..... +A(N)*X**(N+1)
C      POR EL METODO DE HORNER
C
C      A    VECTOR DE COEFICIENTES. EL PRIMER ELEMENTO ES EL TERMINO
C           INDEPENDIENTE.
C      N    GRADO DEL POLINOMIO MAS UNO.
C      X    VALOR DE LA VARIABLE INDEPENDIENTE
C
      REAL *8 A,  X,HOR
      DIMENSION A(1)
      NT = N
      HOR = 0.0
      DO 1 I = 1,N
      HOR = HOR * X + A(NT)
    1 NT = NT -1
      RETURN
      END
      SUBROUTINE CREAT (F,N,M,DIV,IREN,RN,T,IM,TOL,NK)
C
C      F(10,10,20)       MATRIZ POLINOMIAL DE N*M
C      N          NUMERO DE RENGLONES DE F
C      M         NUMERO DE COLUMNAS DE F
C      IREN(I)   RENGLONES SELECCIONADOS PARA FORMAR LA MATRIZ A
C       RN(10)   UBICACION DE LOS NUEVOS POLOS.
C       T         MATRIZ (M*N) DE ELEMENTOS CONSTANTES GENERADA AL SU
C                  BSTITUIR LOS POLOS DE RN(I), EN LOS RENGLONES ELEGI
C                  DOS
C       MI      MATRIZ IDENTIDAD EXPANDIDA DE ACUERDO CON LA SELECCION
C               EMPLEADA EN T
C       NK=1   INDICA QUE ALGUN POLO NUEVO ES IGUAL A UNO VIEJO.
      REAL *8 HOR,F,RN,T,TOL,DIV,DENOM,TEL ,IM
```

```
      DIMENSION K(10,10,20),IREN(10), N(10),T(10,10),IN(10,10),TEL(20),
     *NE    (150),DIV(20)
      DATA NE/150*0/,IK/0/
      IF (KK .EQ. 0) IK = 0
      NL  = 1
      IF (IK .LE. 0) GO TO 5
      DO 7 I = 1, IK
      IA = (I-1) * N + 1
      DO 6 J = 1, N
      IF ( NE(IA) .NE. IREN(J)) GO TO 7
      IA = IA + 1
    6 CONTINUE
      WRITE (6,104)  I
  104 FORMAT (//10X,"ESTA ORDENACION DE RENGLONES YA SE PROBO EN LA ITER
     *ACION",I3//)
      RETURN
    7 CONTINUE
    5 NIT =N**N
      NIT = MINO (10,NIT)
      IF (K    .LE.  NIT) GO TO 10
      WRITE (6,102) (I,I=1,N)
  102 FORMAT (//10X,"YA SE HICIERON  DIEZ  INTENTOS,QUE ES EL LIMITE DE
     *ITERACIONES"/10X,"SE EMPLEARON LAS SIGUIENTES ORDENACIONES:"/
     *5X,"ITER",10("  REN",I2))
      DO 9 I =1,15
      IB = I*N
      IA = IB-N+1
    9 WRITE (6,103) I,(NE(J) ,J=IA,IB)
  103 FORMAT (/2X,11I7)
      WRITE (6,105)
  105 FORMAT (/10X,"ASIGNAR NUEVOS POLOS O REPETIRLOS EN CASO DE SER NE
```

```
     *CLSALIDA)"/)
      KL = 2
      RETURN
   12 WRITE (6,106) I,M,(J,IREN(J),J=1,N)
  106 FORMAT (//10X,"EL RENGLON SOLICITADO PARA LA POSICION",I3," ES MAY
     *OR QUE",I3,"  (NUMERO DE SALIDAS)"//
     *(10X,"PARA EL RENGLON",I3," SE PIDIO EL RENGLON",I3/)/)
      RETURN
   10 IA = IK * N
      IK = IK + 1
      WRITE (6,107) IK,(IREN(J),J=1,N)
  107 FORMAT (/ 2X,"INTENTO NUMERO",I3," PARA FORMAR LA MATRIZ  W  CON L
     *OS RENGLONES :"     , 10I6/)
      WRITE (6,121) (PN(J),J=1,N)
  121 FORMAT (/37X,"Y LOS POLOS DEL OBSERVADOR :",10F6.1/)
      DO 6 I = 1,N
      IA = IA + 1
      IF (IREN(I) .GT. M) GO TO 12
    6 WL(IA) = IREN (I)
      DO 11 I = 1,N
      DO 11 J = 1,N
   11 IL(I,J) = 0
      DO 1 I = 1,N
      IR = IREN(I)
      IL(I,IR) =1
      DENOM = HOR(DIV,N+1,PN(I))
      IF (ABS(DENOM) .GE. TOL) GO TO 3
      WRITE (6,100) PN(I)
  100 FORMAT (//10X,"EL VALOR DE UN NUEVO POLO",F10.4,"  COINCIDE CON AL
     *GUNO DE LOS ORIGINALES"//)
      KL =2
```

```
      RETURN
    3 DO 2 J = 1,N
      NG = NZEL (K,J,IR,20,TEL)
      IF (NG .GT. 0) GO TO 13
      T(I,J) = 0.0
      GO TO 2
   13 DO 4 K = 1,NG
    4 TEM(K) = F(J,IR,K)
      T(I,J) = HOR(TEM,NG,PN(I))/DENOM
    2 CONTINUE
    1 CONTINUE
      KL = 3
      RETURN
      END
      SUBROUTINE LIMPIA(A,N,M,K)
      REAL *8 A
      DIMENSION A(1)
      NT = N*M*K
      DO 1 I=1 ,NT
    1 A(I) = 0.0
      RETURN
      END
      SUBROUTINE MUMCMP (A1,A2,A3,L1,M1,L2,M2,L3,M3,ID1,TCL)
      IMPLICIT REAL*8 (A,T)
      DIMENSION A1(10,10),A2(10,10,20),A3(10,10,20),TEMP(20)
      IF (M1 .NE. L2) GO TO100
      L3 = L1
      M3 = M2
      DO 50 I = 1, M3
      DO 50 J = 1,L3
      DO 25 K1 = 1,M1
```

```
   25 TEMP (K1)= 0.0
      DO 20 N = 1,M1
      NA1 = 1
      IF (DABS (A1(J,N)) .LE. TOL ) NA1=0
      NA2 = NZEL (A2,I,N,ID1,TOL)
      IF ( NA1 .EQ. 0 .OR. NA2 .EQ. 0) GO TO 20
      TT = A1 (J,N)
      DO 10 K2 = 1,NA2
   10 TEMP (K2) = TEMP (K2) + TT * A2(I,N,K2)
   20 CONTINUE
      DO 30 K1 = 1,ID1
      A3 (I,J,K1)= TEMP (K1)
   30 CONTINUE
   50 CONTINUE
      RETURN
  100 WRITE (6,900)
  900 FORMAT (//2X,"ILLEGAL MATRIX MULTIPLICATION (CONFORMABILITY)")
      RETURN
      END
      SUBROUTINE RANGT(X,Y,M,N,EPS,IR)
C
C         DESCRIPCION :
C
C         SE DETERMINA EL RANGO DEL PAR DE MATRICES (X,Y) DE ACUERDO CON
C         EL CRITERIO DE KALMAN , O SEA, T= (Y.XY.XXY.XXXY.......)
C         --- SI Y ES LA MATRIZ B DE LA ECUACION DE ESTADO, SE INFIERE
C             ACERCA DE LA CONTROLABILIDAD DEL SISTEMA.
C         --- SI Y ES LA TRANSPUESTA DE LA MATRIZ C DE LA ECUACION DE
C             SALIDA,SE INFIERE ACERCA DE LA OBSERVABILIDAD DEL SISTEMA
C
C         ARGUMENTOS
```

```
C              X     MATRIZ DE N*N,ES LA MATRIZ A EN AMBOS CASOS
C              Y     MATRIZ DE N*M
C              N     NUMERO DE RENGLONES (VARIABLES DE ESTADO)
C              M     NUMERO DE COLUMNAS (EXCITACIONES O SALIDAS)
C              EPS   TOLERANCIA RELATIVA DE LOS ELEMENTOS DE LAS MATRICES EN
C                    FUNCION DEL MAXIMO VALOR ABSOLUTO PIV.
C                    SI ABS(T(I,J)) ES MENOR  QUE ABS(PIV)*EPS,IMPLICA T(I,J)
C                    IGUAL A CERO DURANTE EL PIVOTEO.
C              IR    RANGO DE LA MATRIZ T
C                    SI IR = -1 ERROR EN LAS DIMENSIONES DE LA MATRIZ
C                    SI IR = 0 LA MATRI  ES NULA.
C
C                                                          R.SANDOVAL R.
C                                                          DICIEMBRE 1977
C
      REAL *8 X,Y,XT,YT,T,TT,EPS
      DIMENSION X(10,10),Y(10,10),XT(10),YT(100),T(10,100),TT(10,100)
      DO 11 I2= 1,M
      DO 11 I1 = 1,N
   11 TT (I1,I2) = Y(I1,I2)
      CALL LLAMA (TT,N,M,10,10,1)
      CALL MFGR (TT,N,M,EPS,IR,XT,YT)
      IF (IR) 1,2,3
    1 WRITE (6,100)
  100 FORMAT (//10X, "ERROR EN LAS DIMENSIONES DE LA MATRIZ Y"//)
      RETURN
    2 WRITE(6,101)
  101 FORMAT (//10X,"LA MATRIZ Y ES UNA MATRIZ NULA"//)
      RETURN
    3 IF (IR .LT.N) GOTO 4
      WRITE (6,102)
```

```
  102 FORMAT (//10X,"LA MATRIZ  Y  ES CUADRADA Y DE RANGO N"//)
      RETURN
    4 NP = N-1
    5 DO 6 J = 1,M
      DO 6 I =1,N
    6 T(I,J) = Y(I,J)
      DO 10 I = 1,NP
      II = I*M
      IIM = II+M
      CALL MULMAT (X,Y,T(1,II+1),N,N,M)
      DO 7 I1 = 1,IIM
      DO 7 I2=1,N
    7 TT(I2,I1) = T(I2,I1)
      IF (IIM .GE. N) CALL LLAMA (TT,N,IIM,10,100,1)
      IF (IIM .GE. N) CALL MEGR(TT,N,IIM,EPS,IR,XT,YT)
C     WRITE (6,104) IR,I
  104 FORMAT (//10X,"EL RANGO ES",I4," HASTA LA POTENCIA",I4,"A. DE A"/)
      IF (IR .LT. N) GO TO 8
C     WRITE (6,105) IR,N
  105 FORMAT (//10X,"EL VALOR DEL RANGO",I4," ES IGUAL AL ORDEN",I4," DE
     *L SISTEMA"//)
      RETURN
    8 JJ = 0
      DO 9 J = II +1,IIM
      JJ = JJ + 1
      DO 9 IN= 1,N
    9 Y(IN,JJ) = T(IN,J)
   10 CONTINUE
      WRITE (6,106)
  106 FORMAT (//"EL RANGO DE LA MATRIZ AMPLIADA ES MENOR QUE EL ORDEN DE
     *L SISTEMA"//)
```

```
      RETURN
      END
      SUBROUTINE MULMAT (X,Y,Z,N,L,M)
      REAL *8 X,Y,Z,T
      DIMENSION X(10,10),Y(10,10),Z(10,10)
      DO 2 I = 1,N
      DO 2 J = 1,M
      T = 0.0
      DO 1 K = 1,L
    1 T = T  + X(I,K)*Y(K,J)
    2 Z(I,J) = T
      RETURN
      END
      SUBROUTINE LLAMA (A,N,M,ND,MD,IND)
C
C    "LLAMA"  ORGANIZA LOS ELEMENTOS DE UN ARREGLO DE ACUERDO CON EL MANE-
C             JO MATRICIAL DE IBM O BURROUGHS.
C
C      A       MATRIZ
C      N       ORDEN DE LA MATRIZ (DEBE SER MENOR O IGUAL A ND)
C      M       ORDEN DE LA MATRIZ (DEBE SER MENOR O IGUAL A MD)
C      ND      DIMENSION  DECLARADA PARA LA MATRIZ A EN EL PROGRAMA QUE
C      MD      DIMENSION  DECLARADA PARA LA MATRIZ A EN EL PROGRAMA QUE
C              LLAMA A "LLAMA"
C      IND = 1      TRANSFORMA UNA MATRIZ A UN VECTOR CONTINUO
C      IND = 2      TRANSFORMA UN VECTOR CONTINUO A UNA MATRIZ
C
C                                              HORACIO SANDOVAL R.
C                                           INST. ING. UNAM,NOV 1977.
C
      REAL *8 A
```

```
      DIMENSION A(1)
C
C         CUANDO N = ND  NO SE REQUIERE NINGUNA TRANSFORMACION
C
      IF (N .GT. ND .OR. M .GT. MD) GO TO 5
      IF ( N.EQ. ND) RETURN
      GO TO (2,3),IND
C
C       TRANSFORMACION DE MATRIZ A VECTOR CONTINUO
C
    2 I = 0
      DO 1 NC = 1,M
      K =(NC-1) * ND
      DO 1 L = 1,N
      I = I + 1
    1 A(I) = A (K+L)
      RETURN
C
C       TRANSFORMACION DE VECTOR CONTINUO A MATRIZ
C
    3 NN = M*N +1
      DO 4 I = 1,M
      NC = (M-I)*ND +N+1
      DO 4 J=1,N
      NN = NN -1
      NC = NC -1
    4 A(NC) = A (NN)
      RETURN
    5 WRITE (6,100)
  100 FORMAT (//10X,"LA MATRIZ A EXCEDE LAS DIMENSIONES PREVISTAS")
      CALL EXIT
```

```
      END
      SUBROUTINE MFGR (A,M,N,EPS,IRANK,IROW,ICOL)
      REAL *8 A,IROW,ICOL,PIV,HOLD,EPS,TOL,SAVE
      DIMENSION A(1) ,IROW(1), ICOL(1)
      IF (M) 2,2,1
    1 IF (N) 2,2,4
    2 IRANK = -1
    3 RETURN
    4 IRANK = 0
      PIV = 0.0
      JJ = 0
      DO 6 J = 1,N
      ICOL(J) = J
      DO 6 I = 1,M
      JJ = JJ + 1
      HOLD = A (JJ)
      IF (DABS(PIV) -DABS(HOLD)) 5,6,6
    5 PIV = HOLD
      IR = I
      IC = J
    6 CONTINUE
      DO 7 I = 1,M
    7 IROW(I) = I
      TOL =DABS (EPS*PIV)
      MM = M*N
      DO 19 NCOL =M,MM,M
    8 IF(DABS(PIV) - TOL) 20,20,9
    9 IRANK= IRANK + 1
      JJ = IR - IRANK
      IF ( JJ)12,12,10
   10 DO 11 J = IRANK,MM,M
```

```
      I = J + JJ
      SAVE = A(J)
      A(J) = A(I)
   11 A(I) = SAVE
      JJ = IROW (IR)
      IROW (IR) = IROW(IRANK)
      IROW (IRANK) = JJ
   12 JJ = (IC - IRANK) * M
      IF (JJ) 15,15,13
   13 KK = NCOL
      DO 14 J = 1, M
      I = KK + JJ
      SAVE = A(KK)
      A(KK) = A(I)
      KK= KK+1
   14 A(I) = SAVE
      JJ = ICOL(IC)
      ICOL(IC)= ICOL(IRANK)
      ICOL(IRANK) = JJ
   15 KK = IRANK + 1
      KL = IRANK - M
      LL = NCOL + MM
      IF(KL)16,25,25
   16 JJ = LL
      SAVE = PIV
      PIV = 0.0
      DO 19 J = KK,M
      JJ = JJ + 1
      HOLD = A(JJ)/SAVE
      A(JJ) = HOLD
      L = J-IRANK
```

```
      IF (IRANK -N) 17,19,19
   17 II=JJ
      DO 19 I = KK,N
      II = II +M
      LL = II -L
      A(II) = A(II) -HOLD * A(LL)
      IF (DABS(A(II))-DABS(PIV))19,19,18
   18 PIV = A(II)
      IR = J
      IC = I
   19 CONTINUE
   20 IF (IRANK-1)3,25,21
   21 IR = LL
      DO 24 J = 2,IRANK
      II = J -1
      IR = IR - M
      JJ = LL
      DO 23 I = KK,M
      HOLD = 0.0
      JJ = JJ +1
      MM = JJ
      IC = IR
      DO 22 L = 1,II
      HOLD = HOLD + A(MM) * A(IC)
      IC = IC -1
   22 MM = MM -M
   23 A(MM) = A(MM) - HOLD
   24 CONTINUE
   25 IF (M-IRANK)3,3,26
   26 IR = LL
      LL = LL + M
```

```
      DO 30 J = 1, IRANK
      DO 29 I =KK,NM,M
      JJ=IR
      LL = I
      HOLD = 0.0
      II = J
   27 II= II-1
      IF (II) 29,29,28
   28 HOLD =HOLD-A(JJ) *A(LL)
      JJ = JJ -M
      LL = LL -1
      GO TO 27
   29 A(LL) = (HOLD -A(LL))/A(JJ)
   30 IR = IR -1
      RETURN
      END
      SUBROUTINE GCRD(A,A1,L,M,ID1,TOL,IER)
C     *
C     **********************************************************************
C     *                                                                    *
C     * THE SUBROUTINE 'GCRD' FINDS THE GREATEST COMMON RIGHT DIVISION *
C     * IN UPPER TRIANGULAR FORM OF ALL THE ROWS OF THE INPUT MATRIX.  *
C     * THIS IS DONE BY PERFORMING ELEMENTARY ROW OPERATIONS ON THE     *
C     * INPUT MATRIX. CONSEQUENTLY THE INPUT MATRIX IS ALTERED BY       *
C     * THE SUBROUTINE. IN ADDITION A SECOND MATRIX IS PASSED TO THIS  *
C     * ROUTINE WHICH IS CONVERTED TO THE IDENTY MATRIX OF THE ROW      *
C     * DIMENSION OF THE INPUT MATRIX. THE ELEMENTARY ROW OPERATIONS   *
C     * ARE ALSO PERFORMED ON THIS MATRIX, THE RESULT BEING THE UNI-    *
C     * MODULAR PRE-MULTIPLICATION MATRIX NECESSARY TO CREATE THE GCRD *
C     * DETERMINED. IF THE INPUT MATRIX IS UNIMODULAR THEN THIS MATRIX *
C     * WILL ALSO BE THE INVERSE OF THE INPUT MATRIX.                  *
```

```
C     *                                                                  *
C     *     VARIABLES PASSED:                                            *
C     *     A(1,2,3)       -   3-D DOUBLE PRECISION ARRAY STORING POLY   *
C     *                        MATRIX WHOSE GCRD IS TO BE DETERMINED     *
C     *                        INDICES REP COLS, ROWS, AND COEFS RESP-   *
C     *                        ECTIVELY. GCRD IS RETURNED IN THIS ARRAY  *
C    *      A1(1,2,3)      - ARRAY WHICH RETURNS UNIMODULAR PRE-        SS
C     *                        MULTIPLICATION MAT. OF INVERSE OF INPUT   *
C     *                        MATRIX                                    *
C     *     L,N                ROW AND COLUMN DIMENSIONS OF INPUT A MAT  *
C     *     IB1            -   UPPER BOUND POLYNOMIAL SIZE               *
C     *     TOL            -   ROUNDOFF TOLERANCE(DOUB PREC)             *
C     *     IER            -   RETURNS A 1 IF INPUT MATRIX HAS RANK      *
C     *                        LESS THEN ITS MINIMUM DIMENSION           *
C     *                                                                  *
C     *     OTHER ROUTINES CALLED:                                       *
C     *     MNROW          -   DETERMINES A PROPER MAIN ROW FOR ROW      *
C     *                        OPERATIONS                                *
C     *     ROWOP          -   PERFORMS ACTUAL ROW OPERATIONS ON 3-D     *
C     *                        A, AND A1 ARRAYS (POLY MATRICES)          *
C     *           RSHFT         - EFFECTIVELY SWITCHES TWO ROWS OF A POLY *
C     *                        MATRIX (ACTUALLY 3-D EQUIVALENT)          *
C     *     NZEL           -   EXPLAINED WITH SUBROUTINE                 *
C     *                                                                  *
C     ******************************************************************
      IMPLICIT REAL*8 (A,T,X)
      DOUBLE PRECISION DABS
      DIMENSION A(10,10,20),A1(10,10,20)
      INTEGER POSK,POSJ
C     * CREATE IDENTITY MATRIX
      DO 2 I=1,L
```

```
      DO 2 J=1,L
      DO 2 K = 1, ID 1
    2 A1(I,J,K)=0.0
      DO 3 I=1,L
    3 A1(I,I,1)=1.0
C
C     * MAIN LOOP FOR MATRIX MANIPULATION
C
      ML=M
      IF (L.LT.M) ML=L
      DO 100 I=1,ML
C
C     *FIND NEW MAIN ROW SUCH THAT LOWEST DEGREE
C     * POLYNOMIAL IS IN DIAGONAL POSITION
C
    7 CALL MNROW(A,I,L,ID1,MROW,IMAX,TOL)
      IF (IMAX.EQ.-1) GO TO 500
      IF (I.EQ.L) GO TO 56
      IF (I.NE.MROW) CALL RSHFT(A,A1,L,M,ID1,I,MROW)
      MROWL=I+1
      POSM=NZLL(A,I,I,ID1,TOL)
C
C     *  PERFORM ROW OPERATIONS ON ROWS BELOW MAIN ROW
C
      DO 50 J=MROWL,L
      POSJ=NZLL(A,I,J,ID1,TOL)
      IF (POSJ.EQ.0) GO TO 50
      CALL ROWOP(A,A1,I,J,POSJ,POSM,L,M,ID1,TOL)
   50 CONTINUE
C
C     * SEE IF SUBMATRIX IS REDUCED BELOW MAIN ROW
```

```
      DO 55 J=MROWB,L
      K=NZEL(A,I,J,ID1,TOL)
      IF (K.NE.0) GO TO 7
   55 CONTINUE
C
C     * REDUCE ORDER OF POLYNOMIAL ABOVE MAIN ROW IF POSSIBLE
C
   56 IF (I.EQ.1) GO TO 100
      POSM=NZEL(A,I,1,ID1,TOL)
      MROWA=I-1
      DO 95 J=1,MROWA
   57 POSJ=NZEL(A,I,J,ID1,TOL)
      IF (POSJ.LT.POSM) GO TO 95
      CALL ROWOP(A,A1,I,J,POSJ,POSM,L,M,ID1,TOL)
      GO TO 57
   95 CONTINUE
  100 CONTINUE
C
C     * NORMALIZE ROWS SO THAT LEADING COEFFICIENT OF DIAGONAL
C     * ELEMENTS EQUALS ONE
C
      DO 25 J=1,ML
      K3=NZEL(A,J,J,ID1,TOL)
      IF (K3.EQ.0) GO TO 25
      X=A(J,J,K3)
      DO 20 I=J,M
      K2=NZEL(A,I,J,ID1,TOL)
      IF (K2.EQ.0) GO TO 20
      DO 15 K=1,K2
      IF (DABS(A(I,J,K)).GE.TOL) GO TO 13
      A(I,J,K)=0.0
```

```
      GO TO 15
   13 A(I,J,K)=A(I,J,K)/X
   15 CONTINUE
   20 CONTINUE
      DO 24 I=1,L
      K2=NZEL(A1,I,J,ID1,TOL)
      IF (K2.EQ.0) GO TO 24
      DO 23 K=1,K2
      IF (DABS(A1(I,J,K)).GE.TOL) GO TO 22
      A1(I,J,K)=0.0
      GO TO 23
   22 A1(I,J,K)=A1(I,J,K)/X
   23 CONTINUE
   24 CONTINUE
   25 CONTINUE
C
      IER=0
      RETURN
  500 IER=1
      RETURN
  980 FORMAT(4I3)
      END
      SUBROUTINE NNROW(A,I,L,ID1,NROW,IMAX,TOL)
      IMPLICIT REAL*8 (A,T,X)
      DOUBLE PRECISION DABS
      DIMENSION A(10,10,20)
      IMAX=-1
      DO 10 J=I,L
      N=0
      DO 5 K=1,ID1
      K1=ID1+1-K
```

```
      X=DABS(A(I,J,K1))
      IF (X.GT.TOL) GO TO 6
      A(I,J,K1)=0.0
      N=N+1
    5 CONTINUE
      GO TO 10
    6 IF (N.LE.IMAX) GO TO 10
      IMAX=N
      NROW=J
   10 CONTINUE
      RETURN
      END
      SUBROUTINE RSHFT(A,A1,L,M,IP1,IROW1,IROW2)
      IMPLICIT REAL*8 (A,T,X)
      DOUBLE PRECISION DABS
      DIMENSION A(10,10,20),A1(10,10,20),TMP1(20),TMP2(20)
      DO 5 I=1,M
      DO 5 K=1,IP1
      TMP1(K)=A(I,IROW1,K)
      A(I,IROW1,K)=A(I,IROW2,K)
      A(I,IROW2,K)=TMP1(K)
    5 CONTINUE
      DO 10 I=1,L
      DO 10 K=1,IP1
      TMP2(K)=A1(I,IROW1,K)
      A1(I,IROW1,K)=A1(I,IROW2,K)
      A1 (I,IROW2,K) = TMP2 (K)
   10 CONTINUE
      RETURN
      END
      SUBROUTINE ROWOP(A,A1,I,J,POSJ,POSK,L,M,IP1,TOL)
```

```
      IMPLICIT REAL*8 (A,T,X)
      DOUBLE PRECISION DABS
      DIMENSION A(10,10,20),A1(10,10,20),TMP1(20),TMP2(20)
      INTEGER POSJ,POSM
      NSHFT=POSJ-POSM
      X=A(I,J,POSJ)
      DO 55 I1=1,M
      IF (NSHFT.EQ.0) GO TO 18
      DO 15 K=1,NSHFT
   15 TMP1(K)=0.0
   18 K1=IQ1-NSHFT
      DO 20 K=1,K1
   20 TMP1(K+NSHFT)=A(I1,I,K)
   45 DO 50 K=1,IQ1
      IF (DABS(TMP1(K)).GE.TOL) GO TO 48
      TMP1(K)=0.0
      GO TO 50
   48 TMP1(K)=TMP1(K)*X/A(I,I,POSM)
   50 A(I1,J,K)=A(I1,J,K)-TMP1(K)
   55 CONTINUE
      DO 75 I1=1,L
      IF (NSHFT.EQ.0) GO TO 60
      DO 56 K=1,NSHFT
   56 TMP2(K)=0.0
   60 K1=IQ1-NSHFT
      DO 65 K=1,K1
   65 TMP2(K+NSHFT)=A1(I1,I,K)
      DO 70 K=1,IQ1
      IF (DABS(TMP2(K)).GE.TOL) GO TO 68
      TMP2(K)=0.0
      GO TO 70
```

```
   68 TMP2(K)=TMP2(K)*X/A(I,I,POSN)
   70 A1(I1,J,K)=A1(I1,J,K)-TMP2(K)
   75 CONTINUE
      RETURN
      END
      SUBROUTINE FACT (R,A,X,M,N,ID1,TOL)
C     *
C     *********************************************************************
C     *                                                                   *
C     * THE SUBROUTINE ≥FACT≥ DETERMINES THE POLYNOMIAL MATRIX ≥X≥ IN    *
C     * THE EQUATION:      R = (X) (A)
C     * WHERE ≥R≥ IS A POLYNOMIAL MATRIX OF FULL COLUMN RANK, AND ≥A≥) *
C     * IS A GREATEST COMMON RIGHT DIVISOR OF THE ROWS OF ≥R≥ IN UPPER *
C     * TRIANGULAR FORM.  IT COMPUTES ≥X≥ BY TAKING ADVANTAGE OF THE    *
C     * UPPER TRIANGULAR FORM OF ≥A≥. THE UPPER TRIANGULAR FORM ALLOWS *
C     * YOU TO UNIQUELY DETERMINE THE COLUMNS OF ≥X≥ STARTING WITH COL-*
C     * UMN 1 AND GOING FROM LEFT TO RIGHT.                             *
C     *                                                                 *
C     *   VARIABLES PASSED:                                             *
C     *   R(1,2,3),A(1,2,3)                                             *
C     *   X(1,2,3)      -     3-D DOUBLE PRECISION ARRAYS CON-          *
C                             TAINING POLY. MATRICES AS EX-             *
C     *                       PLAINED ABOVE.  R AND A ARE PASSED        *
C     *                       AND REMAIN UNCHANGED, X IS RE-            *
C     *                       TURNED WITH THE RESULT                    *
C     *    M,N          -     NO. OF ROWS AND COLS. OF R                *
C        ID1            -     UPPER BOUND ON POLYNOMIAL SIZE            *
C     *    TOL          -     ROUNDOFF TOLERANCE (DOUB PREC)            *
C     *                                                                 *
C     *   OTHER ROUTINES USED:                                          *
C     *   SUMPR,DIV     -     PERFORM ALGEBRAIC MANIPULATIONS ON        *
```

```
C                                   THE POLYNOMIAL ELEMENTS AS RE-     *
C                                   QUIRED BY THE ALGORITHM            *
C     *                                                                *
C     ******************************************************************
C     *
      IMPLICIT REAL*8 (A,B,T,X)
      DIMENSION A(10,10,20),B(10,10,20),X(10,10,20),TEMP(2,20)
      INTEGER ERR
C
C
      DO 5 I=1,N
      DO 5 J=1,M
      DO 5 K=1,ID1
    5 X(I,J,K)=0.0
C
C     * DETERMINE COLUMN 1 OF X(S)
C
      I=1
      DO 20 J=1,M
      DO 15 K=1,ID1
      TEMP(1,K)=B(I,J,K)
   15 TEMP(2,K)=A(1,1,K)
      CALL DIV(A,X,TEMP,I,J,ID1,TOL,ERR)
      IF (ERR.EQ.1) RETURN
   20 CONTINUE
C
C     * DETERMINE REMAINING COLUMNS OF X(S)
C
      DO 25 I=2,N
      DO 25 J=1,M
      CALL SUBPR(A,X,TEMP,I,J,ID1,TOL)
```

```
      DO 22 K=1,ID1
      TEMP(1,K)=R(1,J,K)-TEMP(1,K)
      TEMP(2,K)=A(1,1,K)
   22 CONTINUE
      CALL DIV(A,X,TEMP,I,J,ID1,TOL,ERR)
      IF (ERR.EQ.1) RETURN
   25 CONTINUE
      RETURN
      END
         SUBROUTINE DETERM (R, A, M, ID1, TOL)
         IMPLICIT REAL*8 (A, R, T, X,D)
      COMMON /POLOBS/DROOTR,DROOTI
         DIMENSION A(10,10,20),R(10,10,20),IDR(10),DET(20),X(20),
     *   DET1(20),DROOTR(20),DROOTI(20)
C
C     TRANSFORM MATRIX TO UPPER TRIANGULAR EQUIVALENT
C
        CALL LIMPIA (DROOTR,20,1,1)
        CALL LIMPIA (DROOTI,20,1,1)
         CALL GCRD (R, A, M, M, ID1, TOL, IER)
         IF (IER.EQ.1) GO TO 60
         ICD = 0
C
C     DETERMINE IF MATRIX IS UNIMODULAR
C
         DO 10 I = 1,M
         IDR(I) = NZEL (R, 1, I, ID1, TOL)
   10    ICD = ICD + IDR(I)
         IF (ICD.GT.M) GO TO 20
         WRITE (6,900)
         RETURN
```

```
C
C     MULTIPLY DIAGONAL ELEMENTS OF UPPER TRIANGULAR
C     EQUIVALENT TO FORM NORMALIZED DETERMINANT
C
   20 I = 1
      KX = IDR(I)
      CALL VLOAD (R, X, I, I, KX, ID1, TOL)
      IF (N.EQ.1) GO TO 45
      DO 40 I = 2,N
      KR = IDR(I)
      CALL MVMULT (R, X, I, I, KR, KX, ID1, TOL)
   40 CONTINUE
   45 WRITE (6,905)
      CALL PRNT3 (X, ID1, TOL)
     GO TO 1
C     WRITE (6,910)
C
C     FIND AND PRINT ROOTS OF DIAGONAL ELEMENTS INDIVIDUALLY
C
C       DO 55 I=1,N
      IF (IDR(I).EQ.1) GO TO 55
      IDEG = IDR(I)
      DO 50 K = 1,IDEG
   50 DET(K) = R(I, I, K)
      IDEG = IDEG - 1
    1 CALL POLRT (X,DET1,N,DROOTR,DROOTI,IER)
      GO TO (200,300,400,500),IER
C     DO 100 K = 1,IDEG
C 100 WRITE (6,920) DROOTR(K),DROOTI(K)
      GO TO 55
  200 WRITE (6,950)
```

```
              GO TO 55
        300   WRITE (6,960)
              GO TO 55
        400   WRITE (6,970)
              GO TO 55
        500   WRITE (6,980)
              GO TO 55
         55   CONTINUE
              RETURN
         60   WRITE (6,907)
              RETURN
        900 FORMAT (1X//1X,"THIS MATRIX IS UNIMODULAR")
      C 905 FORMAT (1X//1X,"THE NORMALIZED DETERMINANT POLYNOMIAL",
      C     * " OF THIS MATRIX IS: ")
        905 FORMAT (/1X," POLINOMIO CARACTERISTICO DEL OBSERVADOR :"/)
        907 FORMAT (1X//1X,"THIS MATRIX IS SINGULAR")
        910 FORMAT (1X//1X,"THE ROOTS OF THIS POLYNOMIAL MATRIX ARE: ")
        920 FORMAT (1X//15X,"S =  ",F13.6," + ",F13.6," J ")
        950 FORMAT (1X//1X,"THE INPUT DEGREE IS LESS THEN ONE")
        960 FORMAT (1X//1X,"THE INPUT DEGREE IS GREATER  THAN 36")
        970 FORMAT (1X//1X,"UNABLE TO DETERMINE ROOT WITHIN 500",
           * "ITERATIONS ON 5 STARTING VALUES")
        980 FORMAT (1X//1X,"THE HIGHEST DEG. COEFFICIENT INPUT IS ZERO")
              END
            SUBROUTINE DIV(A,X,TEMP,I,J,ID1,TOL,ERR)
            IMPLICIT REAL*8 (A-H,T,X)
            DOUBLE PRECISION DABS
            DIMENSION A(10,10,20),X(10,10,20),TEMP(2,20),IPOS(2)
            INTEGER ERR
            DO 10 I1=1,2
            DO 5 L=1,ID1
```

```
      IPOS(I1)=ID1+1-K
      IF (DABS(TEMP(I1,IPOS(I1))).GT.TOL) GO TO 10
    5 CONTINUE
      IF (I1-1) 25,25,30
   10 CONTINUE
      NSHFT=IPOS(1)-IPOS(2)+1
      IP2=IPOS(2)
      IF (NSHFT.LE.0) GO TO 30
      DO 20 I1=1,NSHFT
      N1=NSHFT+1-I1
      IP1=IPOS(1)+1-I1
      X(I,J,N1)=TEMP(1,IP1)/TEMP(2,IP2)
      DO 15 I2=1,IP2
   15 TEMP(1,I2+N1-1)=TEMP(1,I2+N1-1)-X(I,J,N1)*TEMP(2,I2)
   20 CONTINUE
      IP1=IPOS(1)
      DO 22 K=1,IP1
      IF (DABS(TEMP(1,K)).LE.TOL) GO TO 22
      WRITE (6,901) I,J
      IER=0
      RETURN
   22 CONTINUE
   25 IER=0
      RETURN
   30 WRITE (6,903) I,J
      IER=1
      RETURN
C
C
  901 FORMAT (1X,1X,27HWARNING REMAINDER ABOVE TOL,2I3)
  903 FORMAT (1X,1X,27HZERO DIVIDE OR DEGREE ERROR,2I3)
```

```
      END
         SUBROUTINE PMREAD (A, N, M)
C     *
C     ***********************************************************************
C     *
C     * THE SUBROUTINE "PMREAD" READS IN A POLYNOMIAL MATRIX INTO A       *
C     * 3-D DOUBLE PREC. ARRAY. THE FIRST LINE OF INPUT SHOULD BE A       *
C     * LINE OF (I3) INTEGERS WHICH ARE THE ROW, COLUMN MATRIX DIM-       *
C     * ENSIONS, AND THE DEGREES OF THE HIGHEST DEGREE POLYNOMIAL IN      *
C     * EACH COLUMN OF THE MATRIX TO BE INPUT. THE MATRIX POLYNOMIAL      *
C     * COEFFICIENTS ARE THEN READ IN COLUMNWISE ASSUMING EACH POLY-      *
C     * NOMIAL IS OF THE SAME DEGREE OF THE MAXIMUM DEGREE POLYNOMIAL     *
C     * FOR THAT COLUMN. CONSEQUENTLY ZERO COEFFICIENTS SHOULD BE         *
C     * ADDED APPROPRIATELY. IN ADDITION COEFFICIENTS FOR EACH COLUMN     *
C     * START ON A NEW LINE WITH FORMAT (F8.2). FOR EACH POLYNOMIAL       *
C     * COEFFICIENTS ARE ORDERED FROM THE SCALAR TERM TO THE HIGHER       *
C     * ORDERED TERMS. NOTE: THE FORTRAN READ STATEMENTS USE DATA SET     *
C     * REFERENCE NUMBER 1. THIS MEANS THAT FOR TERMINAL USE INPUT        *
C     * MUST BE PLACED IN THE FILE "FILE FT01F001". FOR BATCH USE THE     *
C     * REFERENCE NUMBER MUST BE CHANGED FROM 1 TO 5.                     *
C     *              A(1,2,3)    -  3-D DOUBLE PREC ARRAY TO BE FILLED    *
C     *          VARIABLES PASSED:                                        *
C     *                                                                   *
C     *                                  WITH POLY. MATRIX                *
C     *                                                                   *
C     *              N,M         -  ROW COLUMN DIMENSIONS OF MATRIX       *
C     ***********************************************************************
C     *
         IMPLICIT REAL*8 (A)
         DIMENSION A(10,10,20), IDA(10)
C
```

```
C       READ (5,600) N, M, (IDA(I), I = 1, M)
        READ (5,/  ) N, M, (IDA(I), I = 1, M)
C
        DO 5 I = 1,M
          IDA1 = IDA(I) + 1
C    5    READ (5,610) ((A(I,J,K),K = 1,IDA1),J = 1,N)
     5    READ (5,/  ) ((A(I,J,K),K = 1,IDA1),J = 1,N)
        RETURN
  600   FORMAT (24I3,24I3)
  610   FORMAT (9F8.2)
        END
      SUBROUTINE SUMPR(A,X,TEMP,I,J,ID1,TOL)
      IMPLICIT REAL*8 (A,P,T,X)
      DIMENSION A(10,10,20),X(10,10,20),TEMP(2,20),TEMP1(10,20)
      I1=I-1
      DO 20 K=1,I1
      NX=NZER(X,K,J,ID1,TOL)
      NA=NZER(A,I,K,ID1,TOL)
      DO 5 K1=1,ID1
    5 TEMP1(K,K1)=0.0
      IF (NX.EQ.0) GO TO 20
      IF (NA.EQ.0) GO TO 20
      DO 10 I1=1,NA
      DO 10 K1=1,NX
      TEMP1(K,K1+I1-1)=TEMP1(K,K1+I1-1)+X(K,J,K1)*A(I,K,I1)
   10 CONTINUE
   20 CONTINUE
      DO 30 K1=1,ID1
      TEMP(1,K1)=0.0
      DO 25 K=1,I1
   25 TEMP(1,K1)=TEMP(1,K1)+TEMP1(K,K1)
```

```
   30 CONTINUE
      RETURN
      END
      SUBROUTINE INVERT(P,PI,T1,T2,X,N,KX,IP1,TOL,IER)
C     ******************************************************************
C     *                                                                *
C     * THE SUBROUTINE 2INVERT2 TAKES THE INVERSE OF ANY NONSINGULAR  *
C     * SQUARE POLYNOMIAL MATRIX P. THE RESULTS IS RETURNED IN THE     *
C     * FORM OF A POLYNOMIAL MATRIX PI AND A DIVISOR POLYNOMIAL X      *
C     * SUCH THAT:                                                     *
C     *                           -1                                   *
C     *                          P   = PI/X
C     * NORMALLY X WOULD BE THE DETERMINANT OF P AND PI WOULD BE THE   *
C     * ADJOINT OF P, BUT IN THIS CASE THE POLYNOMIAL X IS NORMALIZED  *
C     SO THAT ITS LARGEST COEFFICIENT IS 1. THEREFORE X DIFFERS        *
C     * FROM DET P AND PI FROM ADJ P BY SCALAR FACTORS                 *
C     *                                                                *
C     *    VARIABLE PASSED:                                            *
C     *    P(1,2,3)     -    3-D ARRAY (DOUB PREC) CONTAINING          *
C                            MATRIX TO BE INVERTED. PROGRAM RETURNS    *
C                            AN UPPER TRIANGULAR COLUMN PROPER         *
C     *                      EQUIVALENT TO THE MATRIX PASSED.          *
C     *    PI(1,2,3)    -     CONTAINS RESULTANT POLY MATRIX AS        *
C     *                       EXPLAINED ABOVE (3-D DOUB PREC)          *
C         T1(1,2,3)     -    3-D DOUB PREC ARRAYS USED FOR TEMP        *
C     *    T2(1,2,3)         ORARY MANIPULATION WITHIN ROUTINE         *
C         X(1)          -    VECTOR CONTAINING RESULTANT DIVISOR       *
C                            POLYNOMIAL AS EXPLAINED ABOVE (DOUB       *
C     *                      PRECISION)
C     *    N            -    SQUARE MATRIX DIMENSION OF P              *
C     *    KX           -    (DEGREE+1) OF RESULTANT POLY  X           *
```

```
C          ID1          -     UPPER BOUND OF POLYNOMIAL SIZE          *
C     *    TOL          -     ROUNDOFF TOLERANCE (DOUB PREC)          *
C     *    IER          -     RETURNS 1 IF INPUT MATRIX IS SINGULAR   *
C     *                                                               *
C     *    OTHER ROUTINES CALLED:                                     *
C     *                PMMULT,FACT,MVMULT,VLOAD,NZEL                  *
C     *                (REFER TO ROUTINE FOR DESCRIPTION)             *
C     *                                                               *
C     *****************************************************************
C     *
      IMPLICIT REAL*8 (P,T,X)
      DIMENSION P(10,10,20),PI(10,10,20),T1(10,10,20),T2(10,10,20),
     1 X(20),IDIM(10)
C
C
C     * CASE WHERE MATRIX IS 1X1
C
      IF (M.NE.1) GO TO 10
      DO 5 K=1,ID1
      PI(M,M,K)=0.0
    5 X(K)=P(M,M,K)
      PI(M,M,1)=1.0
      NX=NZEL(P,M,M,ID1,TOL)
      IER=0
      RETURN
C
C     * FIND G(S) AN UPPER TRIANGULAR EQUIVALENT TO P(S)
C
   10 CALL GORD(P,T1,M,M,ID1,TOL,IER)
      IF (IER.EQ.1) RETURN
      N=0
```

```
      DO 15 I=1,M
      ILIN(I)=NZEL(P,I,I,ID1,TOL)
   15 K=K+ILIN(I)
      IF (K.GT.M) GO TO 25
C
C     * CASE WHERE INPUT MATRIX IS UNIMODULAR (G(S)=I)
C
      DO 20 K=2,ID1
   20 X(K)=0.0
      X(1)=1.0
      KX=1
      KO=1
      CALL SPLIT (X1,PI,KO,N,MO,N,M,M,ID1)
      RETURN
C
C     * CASE WHERE INPUT MATRIX HAS NO SPECIAL FORM
C
   25 K1=ILIN(1)
      I=1
      CALL VLOAD(P,X,I,I,K1,ID1,TOL)
      DO 30 I=2,M
      K2=ILIN(I)
      CALL MVMULT(P,X,I,I,K2,K1,ID1,TOL)
   30 CONTINUE
      KX=K1
C
C     * CREATE DIAGONAL MATRIX OF G(S) DETERMINANTS
C
      DO 35 I=1,M
      DO 35 J=1,M
      DO 35 K=1,ID1
```

```
   35 PI(I,J,K)=0.0
      DO 40 I=1,M
      DO 40 K=1,KX
   40 PI(I,I,K)=X(K)
C
C     * DETERMINE ADJOINT(G(S)) AND THEN  I(S)
C
      CALL FACT (FI,F,T2,M,M,ID1,TOL)
      CALL PMMULT(T2,T1,PI,M,M,M,M,M,K,ID1,TOL)
      RETURN
      END
      FUNCTION NZEL(A,I,J,ID1,TOL)
C     *
C     ******************************************************************
C     *                                                                *
C     * THE FUNCTION 2NZEL2 RETURNS AN INTEGER ONE GREATER THAN THE    *
C     * DEGREE OF THE POLYNOMIAL MATRIX ELEMENT PASSED. IF THIS EL-    *
C     * EMENT HAS THE VALUE ZERO, THEN ZERO IS RETURNED ELSE ONE OR    *
C     * GREATER IS RETURNED.                                           *
C     *                                                                *
C     *    VARIABLES PASSED:                                           *
C     *    A(1,2,3)    -    3-D DOUBLE PRECISION ARRAY WHICH STORES    *
C     *                     POLYNOMIAL MATRIX WHOSE ELEMENT IS OF      *
C     *                     INTEREST                                   *
C     *    I,J         -    COLUMN AND ROW INDICES OF DESIRED ELEMENT  *
C     *    ID1         -    UPPER BOUND ON POLYNOMIAL SIZE             *
C     *    TOL         -    ROUNDOFF TOLERANCE (DOUBLE PREC)           *
C     *                                                                *
C     ******************************************************************
C     *
C
```

```
C
      IMPLICIT REAL*8 (A,T,X)
      DOUBLE PRECISION DABS
      DIMENSION A(10,10,20)
      DO 5 K=1,IL1
      K1=IL1+1-K
      X=DABS(A(I,J,K1))
      IF (X.GT.TOL) GO TO 7
    5 CONTINUE
    6 K1=0
    7 NZEL=K1
      RETURN
      END
      SUBROUTINE PMMULT(A1,A2,A3,L1,M1,L2,M2,L3,M3,ID1,TOL)
C
C     ***************************************************************
C     *                                                             *
C     * THE SUBROUTINE ≥PMMULT≥ PERFORMS MATRIX MULTIPLICATION ON POLY- *
C     * NOMIAL MATRICES. IN TERMS OF THE ARRAYS PASSED IT COMPUTES:   *
C     *                    A3 = (A1) (A2)                           *
C     *                                                             *
C     *    VARIABLES PASSED:                                        *
C     *          A1(1,2,3),A2(1,2,3)
C     *       A3(1,2,3)        -   3-D DOUBLE PREC. ARRAYS STORING  *
C     *                            POLY. MATRICES. A3 CONTAINS      *
C     *                            RESULT. A1,A2 REMAIN UNCHANGED   *
C     *       L1,L2,L3         -   NO. OF ROWS IN A1,A2,A3 RESPECT. *
C     *                            L3 IS RETURNED,L1 AND L2 RE-     *
C                                  MAIN UNCHANGED                   *
C     *    M1,M2,M3     -   NO. OF COLS. IN A1,A2,A3, RESP.         *
C                           L3 IS RETURNED, M1 AND M2 RE-           *
```

```
C                          MAIN UNCHANGED                                  *
C     *    ID1      -      UPPER BOUND ON POLYNOMIAL SIZE                  *
C     *    TOL      -      ROUNDOFF TOLERANCE (DOUB PREC)                  *
C                                                                          *
C     **********************************************************************
C     *
      IMPLICIT REAL*8 (A,T)
      DIMENSION A1(10,10,20),A2(10,10,20),A3(10,10,20),TEMP(10,20)
      IF (N1.NE.L2) GO TO 100
      L3=L1
      M3=M2
      DO 50 I=1,M3
      DO 50 J=1,L3
      DO 20 N=1,M1
      NA1=NZEL(A1,N,J,ID1,TOL)
      NA2=NZEL(A2,I,N,ID1,TOL)
      DO 5 K1=1,ID1
    5 TEMP(N,K1)=0.0
      IF (NA1.EQ.0) GO TO 20
      IF (NA2.EQ.0) GO TO 20
      DO 10 K2=1,NA2
      DO 10 K1=1,NA1
   10 TEMP(N,K1+K2-1)=TEMP(N,K1+K2-1)+A1(N,J,K1)*A2(I,N,K2)
   20 CONTINUE
      DO 30 K1=1,ID1
      A3(I,J,K1)=0.0
      DO 25 N=1,M1
   25 A3(I,J,K1)=A3(I,J,K1)+TEMP(N,K1)
   30 CONTINUE
   50 CONTINUE
      RETURN
```

```
  100 WRITE(6,900)
      RETURN
  900 FORMAT(1X//1X,'ILLEGAL MATRIX MULTIPLICATION')
      END
      SUBROUTINE SPLIT (A1,A2,L0,L1,M0,M1,L2,M2,IL1)
C     *
C     ****************************************************************************
C     *                                                                          *
C     * THE SUBROUTINE 'SPLIT' TAKES A SECTION OF ROWS AND COLUMNS OUT *
C     * OF A 3-D REP OF A POLYNOMIAL MATRIX AND PLACES IT INTO AN-      *
C     * OTHER ARRAY.                                                     *
C     *                                                                  *
C     *    VARIABLES PASSED:                                             *
C     *    A1(1,2,3),A2(1,2,3) - 3-D DOUBLE PREC REP OF POLY MAT-        *
C     *                          RICES. A1 CONTAINS CONTAINS OR-         *
C     *                          IGINAL MATRIX AND REMAINS UN-           *
C     *                          CHANGED. SUBSECTION OF A1 IS            *
C     *                          PLACED INTO A2.                         *
C     *    L0,L1        -        BEGINNING AND FINAL ROWS TO BE          *
C     *                          TAKEN OUT OF A1. 1 =L0 =L1.             *
C     *    M0,M1        -        BEGINNING AND FINAL COLS. TO BE         *
C     *                          TAKEN OUT OF A1. 1 = M0 =M1.            *
C     *    L2,M2        -        RESULTING ROW AND COLUMN DIM-           *
C     *                          ENSIONS OF A2                           *
C     *    IL1          -        UPPER BOUND ON POLYNOMIAL SIZE          *
C     *                                                                  *
C     ****************************************************************************
C     *
C
      IMPLICIT REAL*8 (A)
      DIMENSION A1(10,10,20),A2(10,10,20)
```

```
      DO 10 I=N0,N1
      DO 10 J=L0,L1
      DO 10 K=1,ID1
   10 A2(I+1-N0,J+1-L0,K)=A1(I,J,K)
      L2=L1-L0+1
      N2=N1-N0+1
      RETURN
      END
      SUBROUTINE VLOAD(A,X,I,J,KX,ID1,TOL)
C     *
C     ******************************************************************
C     *                                                                *
C     * THE SUBROUTINE ≥VLOAD≥ COPIES A POLYNOMIAL MATRIX ELEMENT FROM *
C     A DOUBLE PRECISION ARRAY INTO A DOUBLE PRECISION VECTOR          *
C     *                                                                *
C     *    VARIABLES PASSED:                                           *
C     *    A(1,2,3)    -    3-D DOUBLE PREC ARRAY WHOSE ELEMENT        *
C     *                     IS DESIRED.                                *
C     *    X(1)        -    DOUB PREC VECTOR WHICH RECEIVES            *
C     *                     POLYNOMIAL.                                *
C     *    I,J         -    COL,ROW INDICES OF DESIRED MATRIX          *
C     *                     ELEMENT                                    *
C     *    KX          -    (DEGREE+1) OF POLYNOMIAL RETURNED          *
C     *                     IN X
C     *    ID1         -    UPPER BOUND OF POLYNOMIAL SIZE             *
C     *    TOL         -    ROUNDOFF TOLERANCE (DOUB PREC)             *
C     *                                                                *
C     ******************************************************************
C     *
      IMPLICIT REAL*8(A,X,T)
      DIMENSION A(10,10,20),X(20)
```

```
C
      DO 5 K=1,KX
    5 X(K)=A(I,J,K)
      IF (KX.EQ.ID1) RETURN
      K1=KX+1
      DO 10 K=K1,ID1
   10 X(K)=0.0
      RETURN
      END
      SUBROUTINE MVMULT(A,X,I,J,KA,KX,ID1,TOL)
C     *
C     ***************************************************************************
C     *                                                                         *
C     * THE SUBROUTINE ≥MVMULT≥ MULTIPLIES A POLYNOMIAL MATRIX ELEMENT *
C     * BY ANOTHER POLYNOMIAL IN A VECTOR X AND RETURNS THE PRODUCT    *
C     * POLYNOMIAL IN THE VECTOR X.                                     *
C     *                                                                 *
C     *    VARIABLES PASSED:                                            *
C     *    A(1,2,3)     -    3-D DOUB PREC ARRAY CONTAINING POLY        *
C                            MATRIX WHOSE ELEMENT IS BEING MULTIPL≥D    *
C     *    X(1)         -    VECTOR INITIALLY CONTAINING POLYNOMIAL     *
C     *                      TO BE MULTIPL≥D AND RETURNING RESULTANT    *
C     *                      POLYNOMIAL                                 *
C     *    KA           -    (DEGREE+1) OF MATRIX ELEMENT TO BE MULT    *
C     *    KX           -    (DEGREE+1) OF OTHER POLY TO BE MULT.       *
C     *                      AND RETURNING SAME INFO ABOUT RESULTANT    *
C     *                      PRODUCT POLY.                              *
C     *    I,J          -    COL,ROW INDICIES OF MATRIX ELEMENT         *
C     *    ID1          -    UPPER BOUND OF POLYNOMIAL SIZE             *
C     *    TOL          -    ROUNDOFF TOL (DOUB PREC)                   *
C     *                                                                 *
```

```
C     ***********************************************************************
C     *
      IMPLICIT REAL*8 (A,R,T,X)
      DIMENSION A(10,10,20),X(20),R(20)
C
      IF (KX.EQ.0) RETURN
      IF (KA.EQ.0) KA=1
      DO 5 K1=1,ID1
    5 R(K1)=0.0
      DO 10 K1=1,KA
      DO 10 K2=1,KX
   10 R(K1+K2-1)=R(K1+K2-1)+A(I,J,K1)*X(K2)
      DO 15 K1=1,ID1
   15 X(K1)=R(K1)
      KX=KA+KX-1
      RETURN
      END
      SUBROUTINE PRNT3(A,ID1,TOL)
C     *
C     ***********************************************************************
C     *                                                                     *
C     * 'PRNT3' IS A MODIFIED VERSION OF THE 'PRNT' SUBROUTINE, IT          *
C     * TAKES IN A SINGLE POLYNOMIAL AND PRINTS IT OUT. THIS ROUTINE        *
C     * IS DOUBLE PRECISION, AND CAN PRINT UP TO A 19TH DEGREE POLY.        *
C     *                                                                     *
C     *   VARIABLES PASSED:                                                 *
C              A(I)      -   A VECTOR OF POLYNOMIAL COEFFICIENTS            *
C                            ORDERED FROM THE SCALER TO THE HIGHEST         *
C                            DEGREE TERMS                                   *
C     *         ID1      -    UPPER BOUND ON POLYNOMIAL SIZE                *
C     *    TOL           -    ROUNDOFF TOLERANCE (DOUB PREC)                *
```

```
C     *                                                                   *
C     *********************************************************************
C     *
C
      IMPLICIT REAL*8 (A,T,X)
      DIMENSION A(20),TEMP(20),ITEMP(20)
      DO 2 K=1,ID1
      K1=ID1+1-K
      X=DABS(A(K1))
      IF (X.GT.TOL) GO TO 4
    2 CONTINUE
C
C     * BRANCH FOR PRINTING ACCORDING TO POLYNOMIAL
C     * (DEGREE+1)
    4 DO 5 K=1,K1
      K2=K1+1-K
      ITEMP(K)=K2-1
      TEMP(K)=A(K2)
    5 CONTINUE
C
C
C    BRANCH FOR PRINTING ACCORDING TO POLINOMIAL (DEGREE+1)
C
      IF (K1.EQ.1) GO TO 7
      IF (K1.LE.6) GO TO 9
      IF (K1.LE.13) GO TO 11
      WRITE (6,910) (ITEMP(K),K=1,6)
      WRITE (6,911)(TEMP(K),K=1,6)
      WRITE (6,912) (ITEMP(K),K=7,13)
      WRITE (6,913) (TEMP(K),K=7,13)
      WRITE (6,912) (ITEMP(K),K=14,K1)
```

```
      WRITE (6,913) (TEMP (K),K=14,K1)
      RETURN
    7 WRITE (6,915) TEMP(1)
      RETURN
    9 WRITE (6,916) (ITEMP(K),K=1,K1)
      WRITE (6,917)(TEMP(K),K=1,K1)
      RETURN
   11 WRITE (6,916) (ITEMP(K),K=1,6)
      WRITE (6,917)(TEMP(K),K=1,6)
      WRITE (6,912)(ITEMP(K),K=7,K1)
      WRITE (6,913)(TEMP(K),K=7,K1)
      RETURN
C     *
  912 FORMAT(1X,7(15X,I2))
  913 FORMAT(1X,1X,7(F13.7,4H S +))
  915 FORMAT (1X//1X,15X,F13.7)
  916 FORMAT (1X/1X,14X,6(15X,I2))
  917 FORMAT (1X,15X,6(F13.7,4H S +))
C
      END
        SUBROUTINE PRNT (A, L, M, ID1, TPL)
C    *
C    *************************************************************************
C    *                                                                       *
C    *  THE SUBROUTINE "PRNT" PRINTS OUT THE ELEMENTS OF A POLYNOMIAL        *
C    *  MATRIX IN A 3-D DOUBLE PRECISION ARRAY REPRESENTATION.               *
C    *  EACH POLYNOMIAL CAN BE UP TO 19TH DEGREE.                            *
C    *                                                                       *
C    *            VARIABLES PASSED:                                          *
C    *                  A(1,2,3)    -  3-D DOUBLE PREC. ARRAY CONTAIG PLY.   *
C    *                                 MATRIX TO BE PRINTED-REMAINS UNCHANGED*
```

```
C     *              L,M        -  ROW AND COLUMN DIMENSIONS OF A            *
C     *              ID1        -  UPPER BOUND ON POLYNOMIAL SIZE            *
C     *              TOL        -  ROUNDOFF TOLERANCE (DOUB PREC)            *
C     *                                                                      *
C     ************************************************************************
C     *
            IMPLICIT REAL*8 (A, T, X)
            DOUBLE PRECISION DABS
            DIMENSION A(10,10,20), TEMP(20), ITEMP(20)
        DO 20 I = 1,M
                DO 20 J = 1,L
C
C     * DETERMINE (DEGREE + 1) OF A MATRIX ELEMENT
C
                    DO 2 K = 1,ID1
                      K1 = ID1 + 1 - K
                       X = DABS(A(I,J,K1))
                       IF (X.GT.TOL) GO TO 4
      2             CONTINUE
C
C     * REORDER COEFFICIENTS FOR PRINTING
C
      4             DO 5 K = 1,K1
                       K2 = K1 + 1 - K
                       ITEMP(K) = K2 - 1
                       TEMP(K) = A(I,J,K2)
      5             CONTINUE
C
C     * SEARCH FOR PRINTING ACCORDING TO POLYNOMIAL
C     * (DEGREE + 1)
C
```

```
      IF (K1.EQ.1) GO TO 7
      IF (K1.LE.6) GO TO 9
      IF (K1.LE.13) GO TO 11
      WRITE (6,916) (ITEMP(K),K = 1,6)
      WRITE (6,917) J, I, (TEMP(K),K = 1,6)
      WRITE (6,912) (ITEMP(K),K = 7,13)
      WRITE (6,913) (TEMP(K),K = 7,13)
      WRITE (6,912) (ITEMP(K),K = 14,K1)
      WRITE (6,913) (TEMP(K),K = 14,K1)
      GO TO 20
    7 WRITE (6,915) J, I, TEMP(1)
      GO TO 20
    9 WRITE (6,916) (ITEMP(K),K = 1,K1)
      WRITE (6,917) J, I, (TEMP(K),K = 1,K1)
      GO TO 20
   11 WRITE (6,916) (ITEMP(K),K = 1,6)
      WRITE (6,917) J, I, (TEMP(K),K = 1,6)
      WRITE(6,912) (ITEMP(K),K = 7,K1)
      WRITE(6,913) (TEMP(K),K = 7,K1)
   20 CONTINUE
      RETURN
C     E
  912 FORMAT (1X,7(15X,I2))
  913 FORMAT (1X,1X,7(F13.7,4H S +))
  915 FORMAT (1X//1X,7H ELEM (,I2,1H,,I2,3H) :,F13.7)
  916 FORMAT (1X/1X,14X,6(15X,I2))
  917 FORMAT (1X,7H ELEM (,I2,1H,,I2,3H) : ,6(F13.7,4H S +))
C
      END
      SUBROUTINE POLRT(XCOF,COF,M,ROOTR,ROOTI,IER)
C        SUBROUTINE POLRT
```

```
C     PURPOSE
C        COMPUTES THE REAL AND COMPLEX ROOTS OF A REAL POLYNOMIAL
C     USAGE
C        CALL POLRT(XCOF,COF,M,ROOTR,ROOTI,IER)
C     DESCRIPTION OF PARAMETERS
C        XCOF -VECTOR OF M+1 COEFFICIENTS OF THE POLYNOMIAL
C              ORDERED FROM SMALLEST TO LARGEST POWER
C        COF  -WORKING VECTOR OF LENGTH M+1
C        M    -ORDER OF POLYNOMIAL
C        ROOTR-RESULTANT VECTOR OF LENGTH M CONTAINING REAL ROOTS
C             OF THE POLYNOMIAL
C        ROOTI-RESULTANT VECTOR OF LENGTH M CONTAINING THE
C             CORRESPONDING IMAGINARY ROOTS OF THE POLYNOMIAL
C        IER  -ERROR CODE WHERE
C              IER=0 NO ERROR
C              IER=1 M LESS THAN ONE
C              IER=2 M GREATER THAN 36
C              IER=3 UNABLE TO DETERMINE ROOT WITH 500 INTERATIONS
C                    ON 5 STARTING VALUES
C              IER=4 HIGH ORDER COEFFICIENT IS ZERO
C     REMARKS
C        LIMITED TO 36TH ORDER POLYNOMIAL OR LESS.
C        FLOATING POINT OVERFLOW MAY OCCUR FOR HIGH ORDER
C     POLYNOMIALS BUT WILL NOT AFFECT THE ACCURACY OF THE RESULTS.
C     SUBROUTINES AND FUNCTION SUBPROGRAMS REQUIRED
C        NONE
C     METHOD
C        NEWTON-RAPHSON ITERATIVE TECHNIQUE. THE FINAL ITERATIONS
C        ON EACH ROOT ARE PERFORMED USING THE ORIGINAL POLYNOMIAL
C        RATHER THAN THE REDUCED POLYNOMIAL TO AVOID ACCUMULATED
C        ERRORS IN THE REDUCED POLYNOMIAL.
```

```
      DIMENSION XCOF(1),COF(1),ROOTR(1),ROOTI(1)
      DOUBLE PRECISION XO,YO,X,Y,XPR,YPR,UX,UY,V,YT,XT,U,XTZ,YTZ,SUMSQ,
     1DX,DY,TEMP,ALPHA
C        IF A DOUBLE PRECISION VERSION OF THIS ROUTINE IS DESIRED, THE
C        C IN COLUMN 1 SHOULD BE REMOVED FROM THE DOUBLE PRECISION
C        STATEMENT WHICH FOLLOWS.
      DOUBLE PRECISION XCOF,COF,ROOTR,ROOTI
C        THE C MUST ALSO BE REMOVED FROM DOUBLE PRECISION STATEMENTS
C        APPEARING IN OTHER ROUTINES USED IN CONJUNCTION WITH THIS
C        ROUTINE.
C        THE DOUBLE PRECISION VERSION MAY BE MODIFIED BY CHANGING THE
C        CONSTANT IN STATEMENT 78 TO 1.0D-12 AND IN STATEMENT 122 TO
C        1.0D-10. THIS WILL PROVIDE HIGHER PRECISION RESULTS AT THE
C        COST OF EXECUTION TIME
C        SET ERROR CODE TO 1
C        SET ERROR CODE TO 4
C        SET ERROR CODE TO 2
C        SET INITIAL VALUES
C        ZERO INITIAL VALUE COUNTER
C        SET X AND Y TO CURRENT VALUE
C        EVALUATE POLYNOMIAL AND DERIVATIVES
C        STEP ITERATION COUNTER
C        SET ERROR CODE TO 3
      IFIT=0
      N=M
      IER=0
      IF (XCOF (N+1) .EQ. 0.0) GO TO  25
   10 IF(N) 15,15,32
   15 IER=1
   20 RETURN
   25 IER=4
```

```
      GO TO 20
   30 ILR=2
      GO TO 20
   32 IF(N-36) 35,35,30
   35 NX=N
      NXX=N+1
      N2=1
      KJ1= N+1
      DO 40 L=1,KJ1
      KT=KJ1-L+1
   40 COF(KT)=XCOF(L)
   45 X0=.00500101
      Y0=0.01000101
      IN=0
   50 X=X0
      X0=-10.0*Y0
      Y0=-10.0*X
      X=X0
      Y=Y0
      IN=IN+1
      GO TO 59
   55 IFIT=1
      XPR=X
      YPR=Y
   59 ICT=0
   60 UX=0.0
      UY=0.0
      V =0.0
      YT=0.0
      XT=1.0
      U=COF(N+1)
```

```
      IF (U) 65,130,65
   65 DO 70 I=1,N
      L =N-I+1
      TEMP=COF(L)
      XT2=X*XT-Y*YT
      YT2=X*YT+Y*XT
      U=U+TEMP*XT2
      V=V+TEMP*YT2
      FI=I
      UX = UX+FI*XT*TEMP
      UY=UY-FI*YT*TEMP
      XT=XT2
   70 YT=YT2
      SUMSQ=UX*UX+UY*UY
      IF(SUMSQ) 75,110,75
   75 DX=(V*UY-U*UX)/SUMSQ
      X=X+DX
      DY=-(U*UY+V*UX)/SUMSQ
      Y=Y+DY
   78 IF(DABS(DY)+DABS(DX)-1.0D-05) 100,80,80
   80 ICT=ICT+1
      IF(ICT-500) 60,85,85
   85 IF(IFIT)100,90,100
   90 IF(IN-5) 50,95,95
   95 IER=3
      GO TO 20
  100 DO 105 L=1,NXX
      MT=KJ1-L+1
      TEMP=XCOF(MT)
      XCOF(MT)=COF(L)
  105 COF(L)=TEMP
```

```
      ITEMP=N
      N=NX
      NX=ITEMP
      IF(IFIT) 120,50,120
  110 IF(IFIT) 115,50,115
  115 X=XPN
      Y=YPN
  120 IFIT=0
  122 IF(DABS(Y)-1.0D-4*DABS(X))135,125,125
  125 ALPHA=X+X
      SUMSQ=X*X+Y*Y
      N=N-2
      GO TO 140
  130 X=0.0
      NX=NX-1
      NXX=NXX-1
  135 Y=0.0
      SUMSQ=0.0
      ALPHA=X
      N=N-1
  140 COF(2)=COF(2)+ALPHA*COF(1)
  145 DO 150 L=2,N
  150 COF(L+1)=COF(L+1)+ALPHA*COF(L)-SUMSQ*COF(L-1)
  155 ROOTI(N2)=Y
      ROOTR(N2)=X
      N2=N2+1
      IF(SUMSQ) 160,165,160
  160 Y=-Y
      SUMSQ=0.0
      GO TO 155
  165 IF (N) 20,20,45
```

```
      END
      SUBROUTINE MINV (A,N,D,L,M)
C        SUBROUTINE MINV
C        PURPOSE
C           INVERT A MATRIX
C        USAGE
C           CALL MINV(A,N,D,L,M)
C        DESCRIPTION OF PARAMETERS
C           A -INPUT MATRIX, DESTROYED IN COMPUTATION AND REPLACED BY
C              RESULTANT INVERSE.
C           N - ORDER OF MATRIX A
C           D - RESULTANT DETERMINANT
C           L - WORK VECTOR OF LENGTH N
C           M - WORK VECTOR OF LENGTH N
C        REMARKS
C           MATRIX A MUST BE A GENERAL MATRIX
C        SUBROUTINES AND FUNCTIONS SUBPROGRAMS REQUIRED
C           NONE
C        METHOD
C           THE STANDARD GAUSS-JORDAN METHOD IS USED. THE DETERMINANT
C           IS ALSO CALCULATED. A DETERMINANT OF ZERO INDICATES THAT
C           THE MATRIX IS SINGULAR.
C           IF A DOUBLE PRECISION VERSION OF THE ROUTINE IS DESIRED,THE
C           C IN COLUMN 1 SHOULD BE REMOVED FROM THE DOUBLE PRECISION
        DOUBLE PRECISION A,D,BIGA,HOLD
C           THE C MUST ALSO BE REMOVED FROM DOUBLE PRECISION STATEMENTS
C           APPEARING IN OTHER ROUTINES USED IN CONJUNCTION WITH THIS
C           ROUTINE.
C           THE DOUBLE PRECISION VERSION OF THIS SUBROUTINE MUST ALSO
C           CONTAIN DOUBLE PRECISION FORTRAN FUNCTIONS, ABS STATEMENT
C              MUST BE CHANGED TO DABS.
```

```
C                 SEARCH FOR LARGEST ELEMENT
      DIMENSION      L(1),M(1)  ,A(1)
      D=1.0
      NK=-N
      DO 80 K=1,N
      NK=NK+N
      L(K)=K
      M(K)=K
      KK = NK + K
      BIGA = A(KK)
      DO 20 J = K,N
      IZ = N*(J-1)
      DO 20 I=K,N
      IJ = IZ + I
   10 IF (DABS(BIGA)-DABS(A(IJ))) 15,20,20
   15 BIGA=A(IJ)
      L(K)=I
      M(K)=J
   20 CONTINUE
C                 INTERCHANGE ROWS
      J=L(K)
      IF(J-K) 35,35,25
   25 KI=K-N
      DO 30 I=1,N
      KI=KI+N
      HOLD=-A(KI)
      JI = KI - K +J
      A(KI) =A(JI)
   30 A(JI) =HOLD
C                 INTERCHANGE COLUMNS
   35 I=M(K)
```

```
      IF(I-K) 45,45,38
   38 JP=N*(I-1)
      DO 40 J=1,N
      JK=NK+J
      JI=JP+J
      HOLD=-A(JK)
      A(JK)=A(JI)
   40 A(JI) =HOLD
C             DIVIDE COLUMN BY MINUS PIVOT (VALUE OF PIVOT ELEMENT IS
C             CONTAINED IN BIGA)
   45 IF(BIGA) 48,46,48
   46 D=0.0
      RETURN
   48 DO 55 I=1,N
      IF(I-K) 50,55,50
   50 IK=NK+I
      A(IK)=A(IK)/(-BIGA)
   55 CONTINUE
C             REDUCE MATRIX
      DO 65 I=1,N
      IK=NK+I
      HOLD=A(IK)
      IJ=I-N
      DO 65 J=1,N
      IJ=IJ+N
      IF(I-K) 60,65,60
   60 IF(J-K) 62,65,62
   62 KJ=IJ-I+K
      A(IJ)=HOLD*A(KJ)+A(IJ)
   65 CONTINUE
C             DIVIDE ROW BY PIVOT
```

```
      KJ=K-N
      DO 75 J=1,N
      KJ=KJ+N
      IF(J-K) 70,75,70
   70 A(KJ)=A(KJ)/BIGA
   75 CONTINUE
C          PRODUCT OF PIVOTS
      D=D*BIGA
C          REPLACE PIVOT BY RECIPROCAL
      A(KK)=1.0/BIGA
   80 CONTINUE
C          FINAL ROW AND COLUMN INTERCHANGE
      K=N
  100 K=(K-1)
      IF(K) 150,150,105
  105 I=L(K)
      IF(I-K) 120,120,108
  108 JQ=N*(K-1)
      JR=N*(I-1)
      DO 110 J=1,N
      JK=JQ+J
      HOLD=A(JK)
      JI=JR+J
      A(JK)=-A(JI)
  110 A(JI) =HOLD
  120 J=M(K)
      IF(J-K) 100,100,125
  125 KI=K-N
      DO 130 I=1,N
      KI=KI+N
      HOLD=A(KI)
```

```
      JI=KI-K+J
      A(KI)=-A(JI)
  130 A(JI) =HOLD
      GO TO 100
  150 RETURN
      END
```