ESCUELA POLITECNICA NACIONAL

FACULTAD DE INGENIERIA ELECTRICA

TESIS DE GRADO

"SIMULACION DIGITAL DE SISTEMAS DE CONTROL"

TESIS PREVIA LA OBTENCION DEL TITULO DE

INGENIERO EN ELECTRONICA Y CONTROL

FANNY YOLANDA MALDONADO BARRIONUEVO

JULIO, 1993

ANEXOS

ANEXO A

USO DE LOS PAQUETES

A.1. TUTSIM

A.2. KUO

A.3. DYNAMO

A.4. CAD CONTROL

A.5. PC-MATLAB

ANEXO A

USO DE LOS PAQUETES.

En este anexo se hace un estudio introductorio del software para el análisis y diseño de sistemas de control mediante los paquetes: TUTSIM, KUO, DYNAMO, CAD CONTROL, PC-MATLAB. Se desarrolla una guía de información de tipo general enfocado a nivel estructural y de principales comandos de cada uno de estos paquetes de software. Se incluirán ejemplos demostrativos para un mejor entendimiento de dichos paquetes.

A.1. TUTSIM.-

Es un programa que permite la simulación de sistemas del mundo real a través de una estructura en diagrama de bloques. El TUTSIM fue realizado en varios lenguajes de alto nivel, sin embargo el más utilizado es el FORTRAN.

Es un programa que ha re-establecido una analogía entre el mundo real y los lenguajes del computador digital teniendo la conveniencia del computador análogo (simulación análoga) y la velocidad y exactitud del computador digital.

El programa TUTSIM requiere que el usuario disponga del modelo analítico pertinente, y dicho modelo debe estar representado en diagramas de bloques. El programa TUTSIM procesa automáticamente dichos bloques con los correspondientes algoritmos computacionales, de ahí que su procesamiento es transparente al usuario, esto significa que equivale a utilizar un computador análogo. Bajo estas condiciones de simulación se procesa en el esquema:

Problema> Modelo Matemático > Modelo de Bloques > Resultados

Los parámetros del modelo, así como el valor inicial de una integración pueden ser fácilmente ingresados y cambiados. Los resultados son valores dependientes del tiempo y su despliegue puede ser gráfico o de tablas numéricas en la pantalla o en papel.

El TUTSIM es un programa altamente interativo, pudiendo interrumpirse su simulación, para continuar en un modo diferente.

La única respuesta que este programa despliega es en el dominio del tiempo, a través, como ya se dijo de una simulación análoga.

Para ingresar en el programa se lo hace escribiendo

TUTSIM, en caso de tener la tarjeta de gráfico monocromático HERCULES se deberá escribir TUTSIM/D=3, desde el DOS, o subdirectorio donde se encuentre el archivo autoejecutable TUTSIM.EXE.

Las funciones principales de este paquete son:

- Ingreso:

  K Entrada al modelo desde el teclado

  F Entrada al modelo desde un archivo existente en un disco

  N Para continuar con el análisis del modelo

- Comandos de Edición:

  CS Cambio en la estructura del modelo

  CP Cambio de parámetros del modelo

  CT Cambio de parámetros del tiempo

- Comandos de Simulación:

  SD Comienzo de la simulación, despliegue de resultados gráficos

  SN Comienzo de la simulación con resultados numéricos

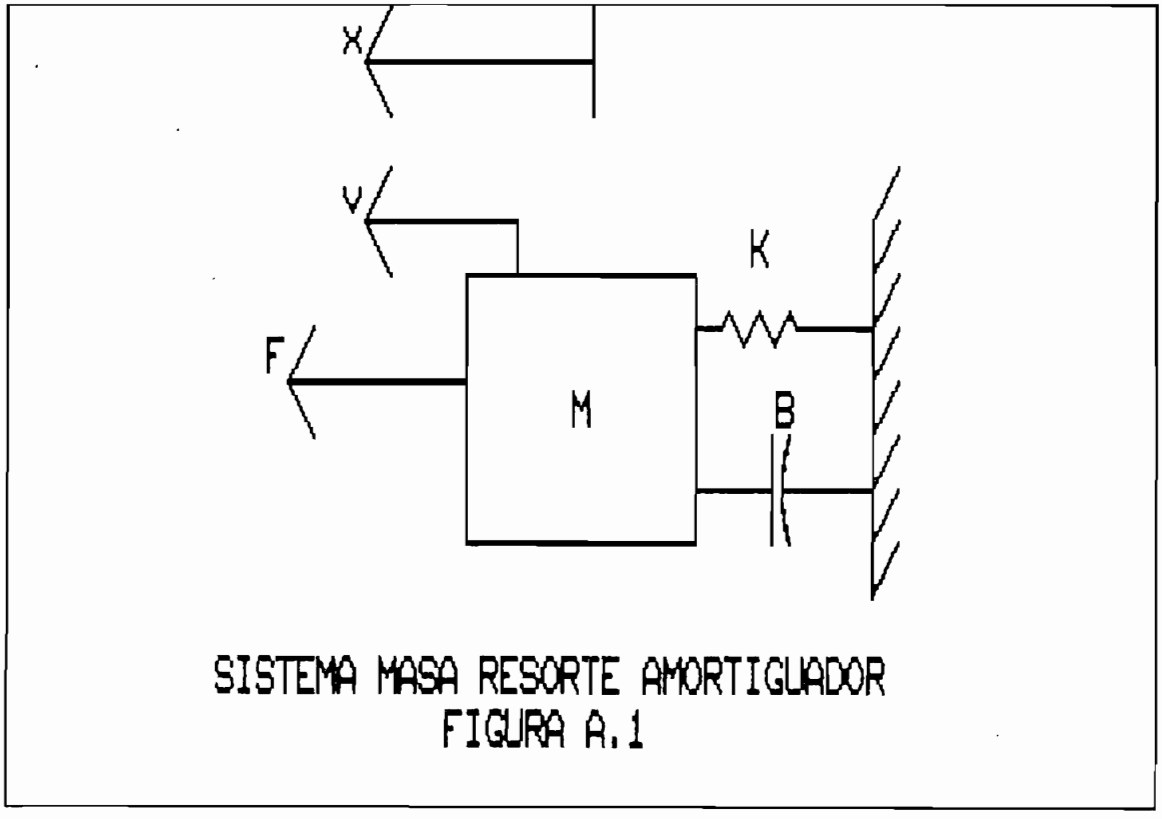
  MR Comienzo de la simulación con archivos Multi-Run

  PD Continuar con los resultados para ser graficados

- Otros:

L	Lista de los registros del modelo
DF	Salvar el modelo en un disco
HC	Impresión
A	Salir del programa
E	Reinicio del programa

A continuación se describe un ejemplo, se considera el modelo físico clásico de control, figura A.1



SISTEMA MASA RESORTE AMORTIGUADOR
FIGURA A.1

donde:

Masa	M = 10 Kg.
Coeficiente de elasticidad	K = 0.1 N/m
Coeficiente de amortiguamiento	B = .5 N/m/s
Fuerza	F = 0 para t < 10
	1 para t ≥ 10
Velocidad	v
Desplazamiento	x

El modelo se obtiene a partir de:

$$F = F_M + F_B + F_K$$

$$F = M\,x'' + B\,x' + K\,x$$

$$x'' = \frac{1}{10}(-.5x' - .1x + F)$$

El diagrama de bloques correspondiente se muestra en la figura A.2.

Se requiere entonces de dos integradores, un sumador, un generador de pulsos para la excitación, un atenuador y dos bloques de ganancia, según los módulos existentes en el TUTSIM. Finalmente el diagrama de bloques del modelo y sus interconexiones se muestran en la figura A.3

Una vez ingresado al paquete, se escoge la opción de

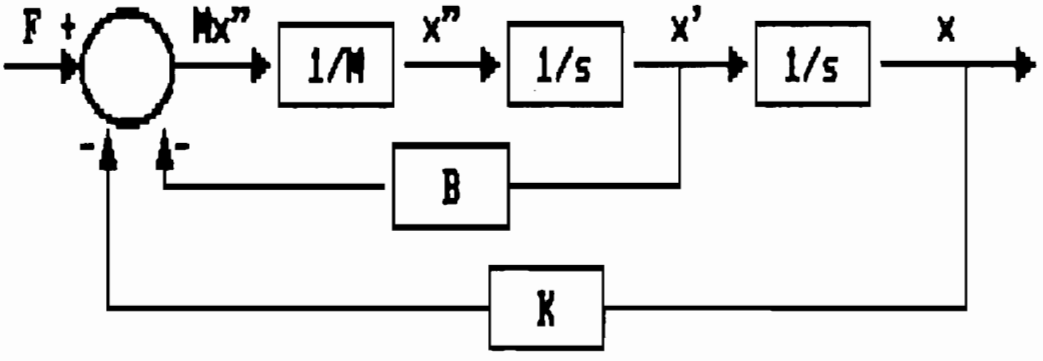


DIAGRAMA DE BLOQUE DEL MODELO
FIGURA A.2

ingreso desde teclado y se continúa de acuerdo con los requerimientos del paquete:

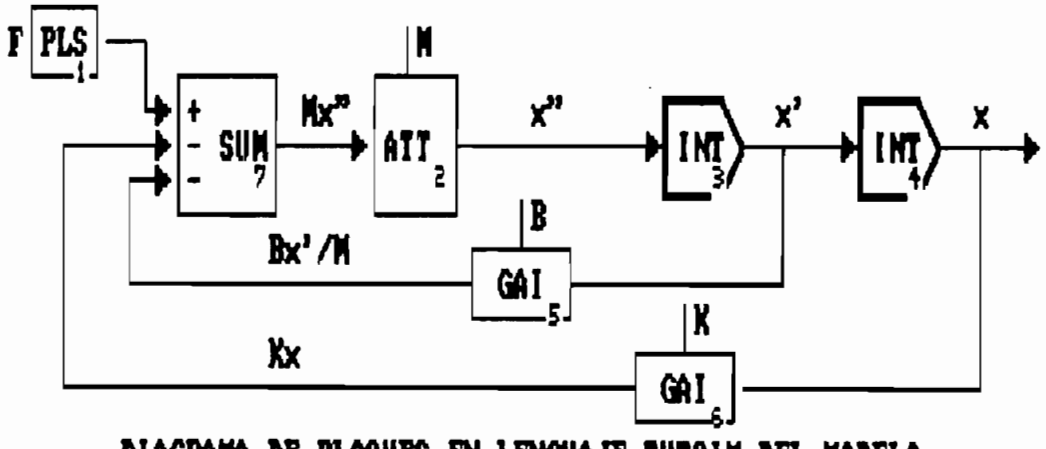


DIAGRAMA DE BLOQUES EN LENGUAJE TUTSIM DEL MODELO
FIGURA A.3

INPUT FORM?:

K = MODEL INPUT WITH KEYBOARD

F = MODEL INPUT FROM FLOPPY

N = TO CONTINUE WITH PRESENT MODEL

K (cr)

KEYBOARD MODEL INPUT

Se procede a ingresar la estructura del modelo:

MODEL STRUCTURES

Format: BLOCKNUMBER,TYPE,INPUT1,INPUT2,....;COMENTARIO

:1,PLS; Bloque pulso

:2,ATT,7; Bloque 2 atenuador, cuya entrada es la salida del bloque 7

:3,INT,2; Bloque 3 integrador, cuya entrada es la salida del bloque 2

:4,INT,3; Bloque 4 integrador, cuya entrada es la salida del bloque 3

:5,GAI,3; Bloque 5 es una ganancia cuya entrada es la salida del bloque 3

:6,GAI,4; Bloque 6 es una ganancia cuya entrada es la salida del bloque 4

:7,SUM,1,-5,-6; Bloque 7 es un sumador cuyas entradas son las salidas del bloque 1, y las salidas de los bloques 5 y 6

negativas

:cr Para finalizar la estructura del modelo.

Luego se ingresa los parámetros del modelo:

MODEL PARAMETERS

Format: BLOCKNUMBER, PARAMETER1,PARAMETER2,..

:1,10,200,1 cr Los parámetros del bloque 1 son 10,200,1 Indican que a t = 10 el bloque de pulsos tendrá una salida de amplitud 1 y al tiempo t = 200 su amplitud será cero.

:2,10 cr El parámetro del bloque 2 es 10. Esto indica que el atenuador del bloque 2 atenúa en un factor de 10.

:5,0.5 cr El parámetro del bloque 5 es 0.5. Esto indica que la ganancia del bloque 5 amplificará en un factor de 0.5.

:6,.1 cr El parámetro del bloque 6 es .1. Esto indica que la ganacia del bloque 6 amplifica en un factor de 0.1.

cr Finaliza los parámetros de entrada.

Luego se ingresa los rangos en el tiempo y para la graficación:

PLOTBLOCKS AND RANGES

Format: BLOCKNUMBER, PLOT-MINIMUN, PLOT-MAXIMUM

Horz: 0,0,200 cr El bloque cero indica el tiempo
El rango es de 0 a 200 seg.

Y1:1,-1,9 cr La salida del bloque 1 es grafi-
cada desde -1 a 9 en el en el eje
Y (Fuerza)

Y2:3,-1,1 cr La salida del bloque 3 es graficada
y escalada desde -1 a 1 en el eje Y
Es la velocidad

Y3:4,0,20 cr La salida del bloque 4 es graficada
y escalada desde 0 a 20 en el eje Y
Es la posición.

Luego se ingresa el formato para el cálculo del tiempo:

TIMING DATA

Format: DELTA TIME, FINAL TIME

:0.5,200 cr Tamaño del paso es 0.5 seg.
Tiempo total de simulación es 200 seg.

Una vez ingresado el sistema, se procede a su simulación:

COMMAND: TUTSIM está en modo de comando

:SD cr El modelo será graficado en pantalla, teniéndose algunas opciones, para mejorar la presentación.

COMMAND:

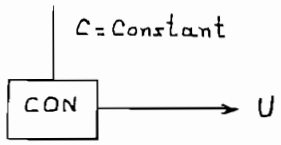
:SN cr Realizará un despliegue de los resultados en forma numérica

COMMAND:

:HC cr Realiza una gráfica de la salida deseada, en impresora.

A modo de ejemplo la figura A.4 muestra la respuesta en el tiempo.

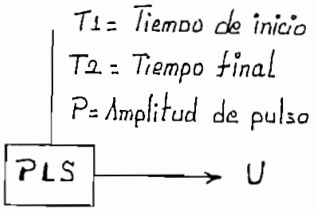
Como complemento se da un esquema simplificado de los principales bloques existentes en este programa:

CODIGO	FUNCION	SIMBOLO
CON	Función constante U(t) = constante	

PLS Función pulso

$U(t) = p \quad T1 \leq t < T2$

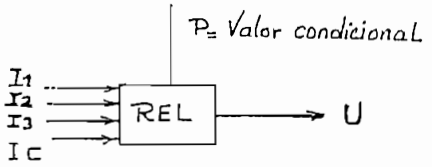
$\emptyset$ en otro caso



REL Relé

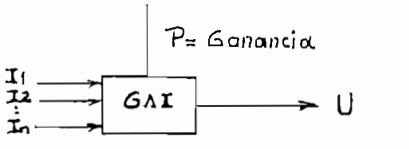
$U(t) = I1, I2, \text{ ó } I3$

dependiendo del valor de Ic



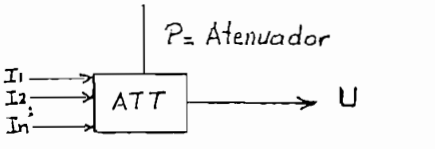
GAI Ganancia

$U(t) = P\ \Sigma In$



ATT Atenuador

$U(t) = (1/P)\ \Sigma In$

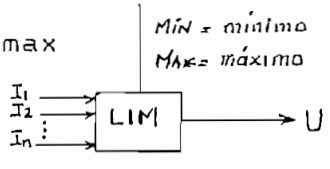


LIM Función Límite

$= \Sigma In, \quad min \leq \Sigma In \leq max$

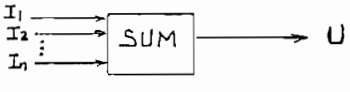
$U(t) = min, \quad \Sigma In < min$

$= max, \quad \Sigma In > max$



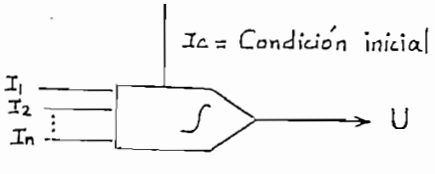
SUM Función suma

$U(t) = \Sigma In$



INT Integrador

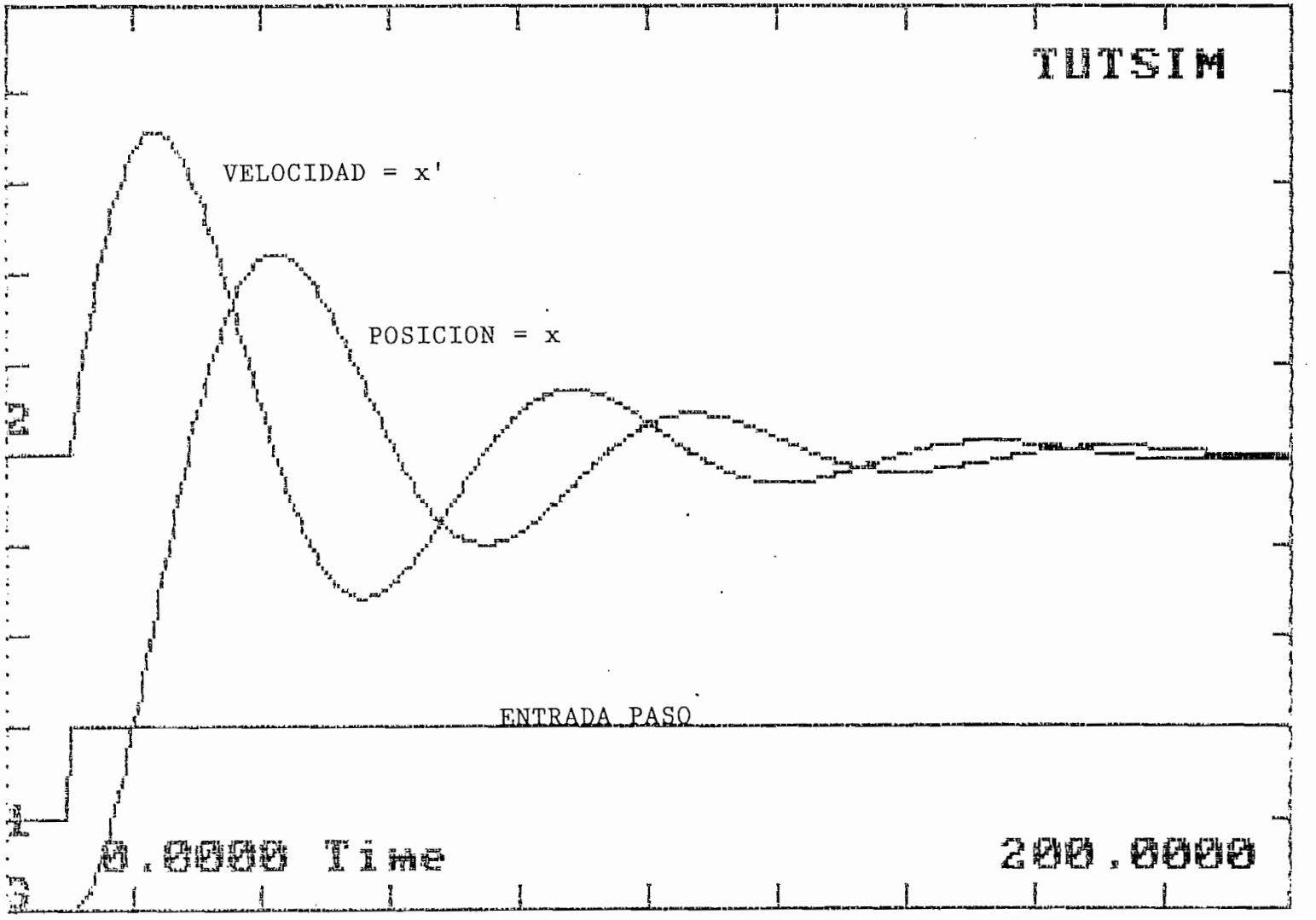
$U(t) = \int \Sigma In$



Finalmente cabe señalar que como limitaciones para este paquete se tiene la imposibilidad de hacer análisis en frecuencia o mediante lugar geométrico de las raíces, así como diseño de controladores. Su versatilidad en cambio se deriva de la facilidad de trabajar en simulación análoga.

PlotBlocks and Scales:
Format:

	BlockNo,	Plot-MINimum,	Plot-MAXimum;	Comment
Horz:	0 ,	0.0000 ,	200.0000 ;	Time
Y1:	1 ,	-1.0000 ,	9.0000 ;	
Y2:	3 ,	-1.0000 ,	1.0000 ;	
Y3:	4 ,	0.0000 ,	20.0000 ;	
Y4:	,	,	;	



RESPUESTA EN EL TIEMPO
FIGURA A.4

A.2. KUO.-

Su objetivo es resolver sistemas altamente complejos, y su principal utilidad es en problemas de control automático, haciendo las tareas más fáciles y confortables; sin embargo la resolución gráfica es de mala calidad.

Algunas limitaciones existen en términos de las magnitudes de los parámetros de los sistemas a ingresarse, que pueden ser resueltos con eficacia, dichas limitaciones son desplegadas al utilizar el paquete, mediante el menú de ayuda. A esto se debe sumar el hecho de que los requerimientos básicos que debe tener el computador personal es que éste debe ser moderadamente configurado, con esto se quiere decir que por lo menos se debe tener dos drivers ó disco duro, 256 Kb de memoria RAM, y un monitor monocromático, su uso está hecho para un IBM, PC, XT, AT, ó algún computador compatible.

Para comenzar el programa se debe escribir **ACS**, desde el sistema operativo, o dentro del subdirectorio donde se encuentre presente el archivo autoejecutable ACS.EXE. Una vez que se encuentra dentro del paquete se despliega un menú de todos los subprogramas que contiene.

El programa **ACS** permite el acceso a todo el contenido

del paquete. Los programas pueden ser clasificados dentro de dos categorías. La primera contiene programas para resolver y almacenar problemas de sistemas continuos y discretos de control lineal (Programas de utilidad). La segunda categoría consiste de una serie de programas que son diseñados con el propósito de revisar y ejercitar el uso juntamente con el texto de los problemas de control automático de sistemas (Programas de revisión). La figura A.5 muestra en un diagrama de bloques las posibilidades de este paquete.

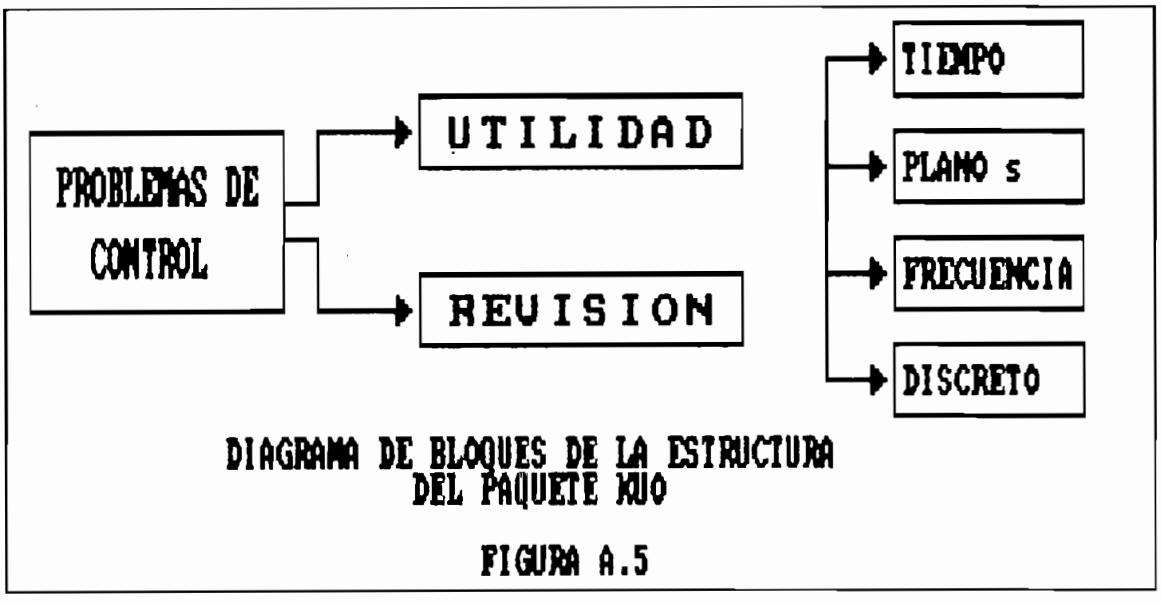


DIAGRAMA DE BLOQUES DE LA ESTRUCTURA DEL PAQUETE KUO

FIGURA A.5

A continuación se listan los programas que están dentro de las dos categorías citadas:

PROGRAMAS DE UTILIDAD.-

(1) LINSYS Análisis general de sistemas lineales

(2) POLYROOT Cálculo de las raíces de los polinomios, $f(x) = 0$

(3) ROOTLOCI Cálculo y graficación del lugar geométrico de las raíces y todos sus parámetros

(4) CLRSP Evaluación a una respuesta paso de un sistema de control en lazo abierto o cerrado

(5) MATRIXM Multiplicación de dos matrices (A * B = C)

(6) PFE Transformación a fracciones parciales de una función racional

(9) FREQRP Cálculo de la respuesta de frecuencia de una función de transferencia en lazo abierto ó en lazo cerrado

(10) FREQCAD Obtención de la respuesta en lazo cerrado o abierto y ejecuta diseños con la función de transferencia. (Redes de compensación)

(11) SDCS Respuesta en el tiempo para sistemas muestreados de control en lazo cerrado, utilizando el cero order hold (ZOH).

PROGRAMAS DE REVISION.-

(7) ZTFORM Revisión general de conceptos y resolución de ejercicios de transformada z.

(8) ROOTL Revisión de las propiedades y ejercicios del lugar geométrico de las raíces (Cálculo de asíntotas, puntos de corte, etc.)

(12) STATE Revisión del análisis de variables de estado

(13) TIMEA Enfoque del análisis en el dominio del tiempo

(14) TIMED Efectos del controlador PID y el método de diseño de un controlador en atraso en el dominio del tiempo

(15) FREQ Revisión y análisis en el dominio de la frecuencia (Criterio de Nyquist, estabilidad relativa).

Se dará un bosquejo general de los programas que se utilizarán en este trabajo:

CLRSP: Evalúa la respuesta a una entrada paso de un sistema en lazo cerrado o lazo abierto de orden máximo 8.

La función de transferencia es de la forma:

$$M(s) = \frac{C(s)}{R(s)} = \frac{G(s)}{[1+G(s)H(s)]}$$

donde:

$$G(s) = \frac{NG(s)}{DG(s)}$$

$$H(s) = \frac{NH(s)}{DH(s)}$$

La función de transferencia puede ser ingresada de la forma polinomial o factorizada.

Forma factorizada:

$$K(a_1 s^2 + a_2 s + a_3) \times \ldots \times [aks^2 + (ak+1)s + (ak+2)]$$

Todos los factores deben tener una forma básica de segundo orden. Si un factor es de primer orden se debe poner $a_1 = 0$. Si el polo o cero es s = 0, se debe poner: $a_1 = 0$, $a_2 = 1$, $a_3 = 0$.

Forma polinomial:

$$b(0)\, s^n + b(1)\, s^{(n-1)} + \ldots + b(n-1)\, s + b(n)$$

El programa descompone a la función de transferencia de lazo cerrado a la forma de variables canónicas y luego resuelve los estados de la ecuación usando la rutina de integración de Runge-Kutta.

ROOTLOCI: Calcula las raíces de un polinomio y grafica el lugar geométrico deseado. Acepta un orden máximo de ocho.

El polinomio está dado de la forma:

FREQCAD: Es usado para obtener las respuestas de frecuencia en lazo cerrado y lazo abierto obteniéndose además el análisis de un sistema en lazo cerrado que es descrito por la siguiente función de transferencia:

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

En la parte de análisis el programa grafica Bode de G(s))H(s) en la forma de módulo en db vs frecuencia. Además determina el margen de ganancia, frecuencia de corte, frecuencia de resonancia y el ancho de banda. La respuesta en lazo cerrado puede ser obtenida como Mag(M) vs frecuencia.

En la parte de diseño ayuda en el cálculo de compensadores en adelanto, atraso o atraso-adelanto. El diseño se basa en especificar un margen de fase o margen de ganancia determinado para una función de transferencia G(s)H(s).

Las funciones G(s) y H(s) pueden ser ingresadas en la forma polinomial o factorada.

La función de transferencia de un controlador en adelanto ó atraso es:

$$\frac{1}{a} \times \frac{1+a \times T \times s}{1+T \times s}$$

Si a > 1 es de adelanto. Si a < 1 es de atraso.

La función de transferencia de un controlador en atraso-adelanto es:

$$\frac{1}{a} x \frac{1+a x T_1 x s}{1+T_1 x s} x \frac{1}{b} x \frac{1+b x T_2 x s}{1+T_2 x s}$$

para: a < 1, b > 1

Para redes de compensación en cascada se tiene:

- Adelanto:

$$\frac{1}{a^N} \left[\frac{1+a \times T \times s}{1+T \times s} \right]^N$$

donde: a > 1

- Atraso:

$$\frac{1}{a^N} \left[\frac{1+a \times T \times s}{1+T \times s} \right]^N$$

$$A = \begin{bmatrix} 0 & 1 \\ -0.01 & -.05 \end{bmatrix}$$

$$B = \begin{bmatrix} 0 \\ .01 \end{bmatrix}$$

$$D = [\ 100 \quad 0]$$

mientras que para lazo cerrado, cuando H = 1, se obtuvo:

$$A = \begin{bmatrix} 0 & 1 \\ -1.1 & -.05 \end{bmatrix}$$

$$B = \begin{bmatrix} 0 \\ .01 \end{bmatrix}$$

$$C = [\ 100 \quad 0]$$

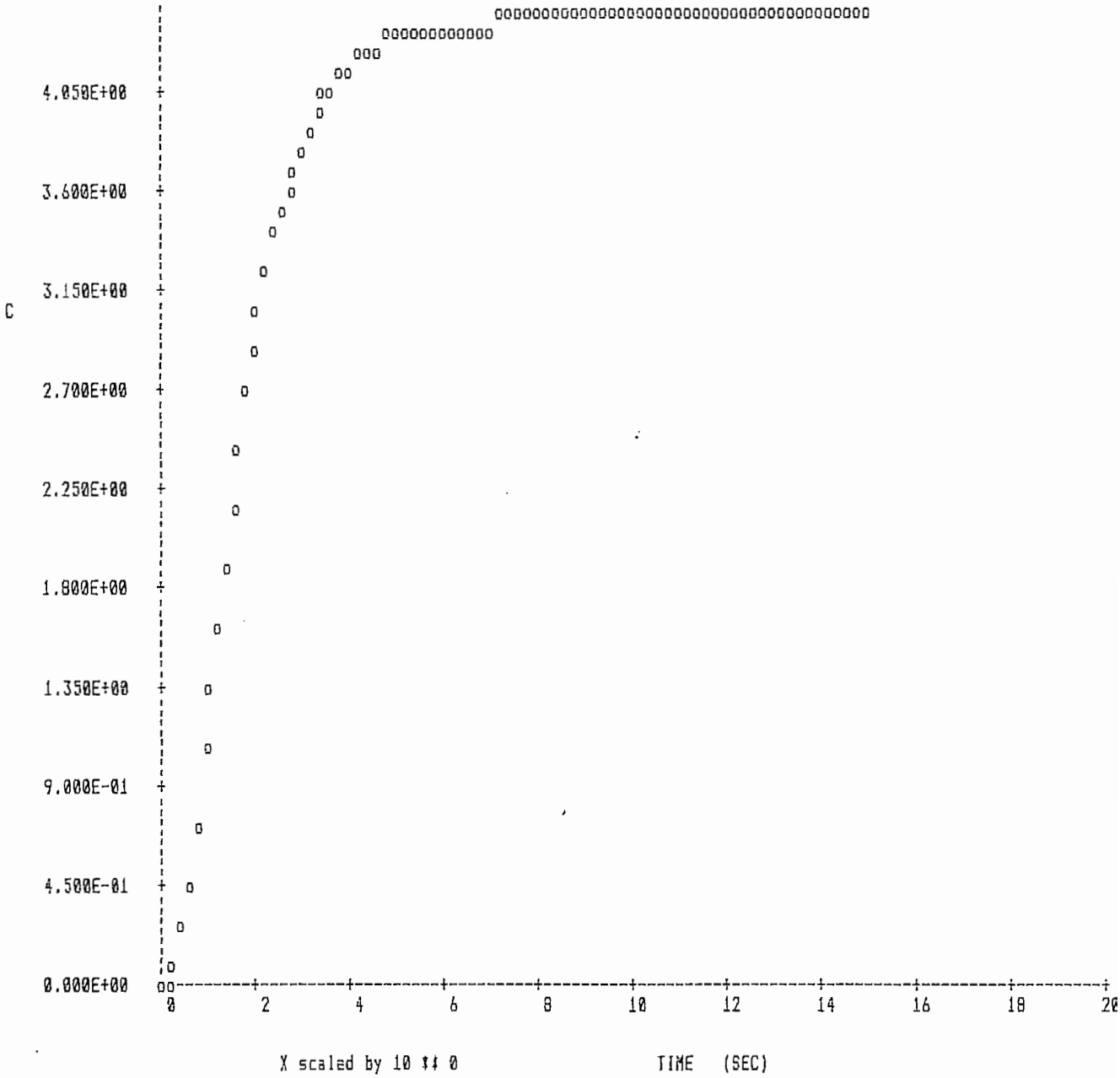
A continuación se muestra los resultados en la figura A.6, para lazo abierto.

La figura A.7 muestra la salida en lazo cerrado que tiene oscilación amortiguada, pero como se puede apreciar la resolución gráfica de este paquete no es buena.

State Equation Simulation Program

Output-Time Response

User Name: FANNY MALDONADO Date: 21/06/93
Problem Identification: RESP. TIEMPO LAZO ABIERTO



RESPUESTA EN EL TIEMPO DEL SISTEMA EN LAZO ABIERTO
FIGURA A.6

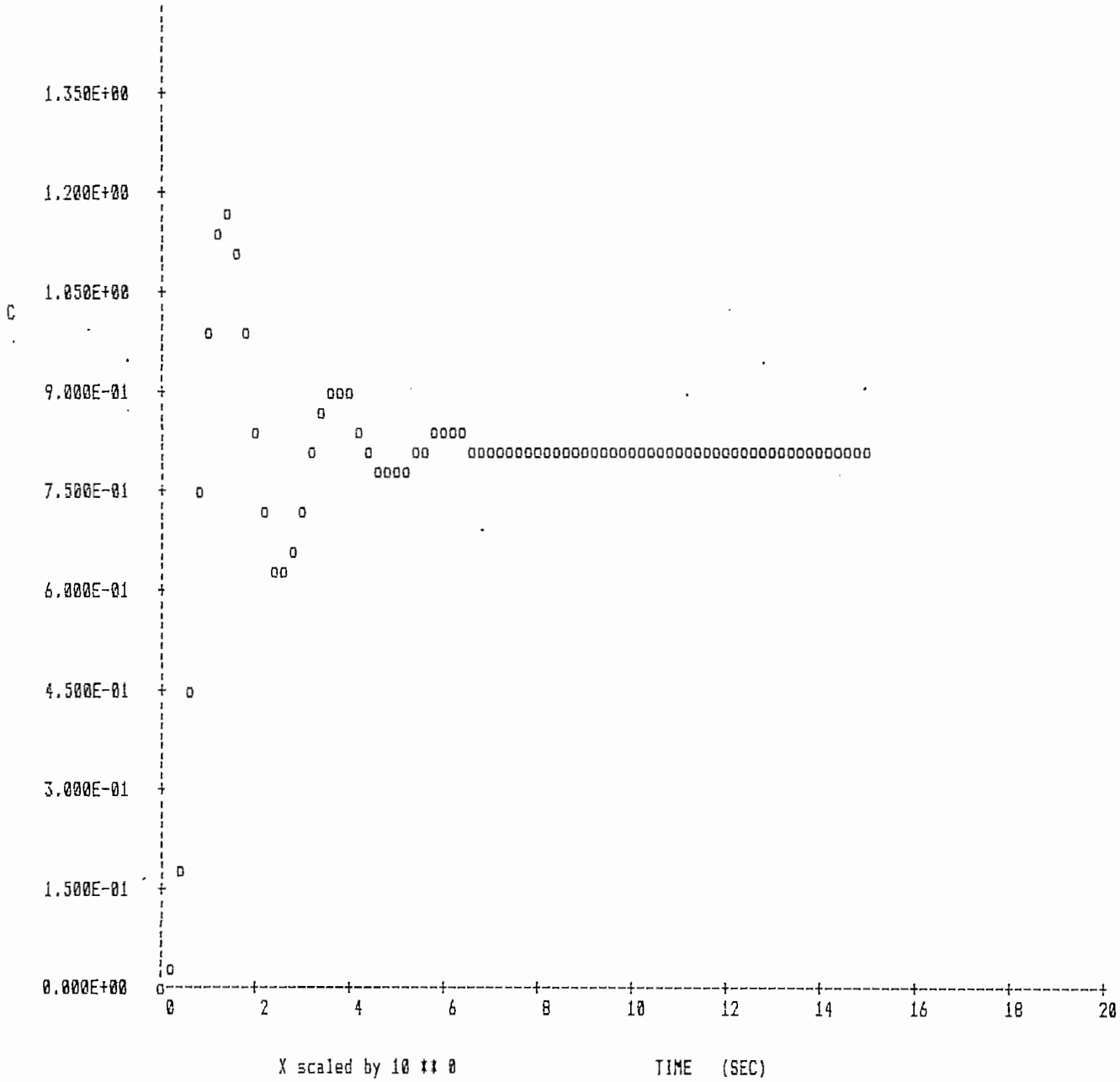
25

State Equation Simulation Program

Output-Time Response

User Name: FANNY MALDONADO Date: 21/06/93

Problem Identification: RESP. TIEMPO LAZO CERRADO



RESPUESTA EN EL TIEMPO DEL SISTEMA EN LAZO CERRADO

FIGURA A.7

A.3. DYNAMO.-

El Professional DYNAMO PLUS es un lenguaje de computación muy útil para trabajar en la simulación de modelos dinámicos de sistemas, es aplicable a modelos determinísticos y que matemáticamente podrían ser representados por un sistema de ecuaciones diferenciales. Además, es un paquete computacional desarrollado específicamente para obtener la respuesta en el tiempo de sistemas dinámicos.

El DYNAMO tiene una interfaz que le permite al usuario trabajar en un ambiente amigable y fácil de manejar. Esta interfaz es manejada por una estructura simple de comandos que aparece al usuario como un set de menús característicos para cada módulo.

El DYNAMO es un programa que da únicamente una respuesta en función del tiempo, siendo una herramienta exclusivamente de análisis, pero su ventaja es su accesibilidad a usuarios sin conocimientos de modelación y aplicable a cualquier disciplina.

Una de las maneras de ingresar al programa y comenzar su correcta utilización es a través del comando **PD**, luego de entrar en el paquete se despliega el siguiente menú:

	Edit
	Compile
	Simulate
PD	View
(PD nombre modelo)	Document
	Help
	Quit

Las cinco primeras opciones son comandos que se usan para seleccionar el módulo que se desea.

El comando Document, es una colección de comandos independientes y cada uno con su propia función. Este módulo ejecuta funciones importantes en el DYNAMO, pero no es indispensable ya que no se requiere su activación para la utilización de este paquete. Presenta información sobre la estructura del programa.

El comando help proporciona una visión general de este paquete computacional, listando los módulos básicos y sus funciones. El comando help puede ser invocado en cualquier momento sin importar donde se esté, dando la ayuda en línea que corresponde a la ubicación.

Al escoger Quit, se sale del DYNAMO a nivel de DOS.

Vale la pena señalar que para que el cursor esté en la línea de comandos se debe presionar la tecla ESC. Las líneas de comandos ocupan una sola línea de la pantalla, excepto para el nivel superior de comandos, que tiene una doble línea dividiendo a los comandos del resto de la pantalla.

Existe la posible de que el usuario trabaje fuera de la interfaz del DYNAMO y que pueda seleccionar cualquier módulo directamente desde el DOS. Esto se logra tecleando el nombre abreviado del módulo seguido del nombre del archivo con el que se desea trabajar. Las abreviaciones son las siguientes:

- **ED** (Editor).- Si se escoge esta opción aparece, un menú de todos los modelos realizados en orden alfabético con extensión .DYN, se sitúa en el modelo deseado y con **ENTER** se ingresa al listado del programa. Para crear un nuevo archivo se presiona **ESC** y se empieza la programación. Permite la edición de un programa, y es aquí donde se pueden realizar los cambios que se desean como permanentes dentro de un determinado modelo.

Una vez concluida la programación se presiona **ESC** y, aparece en la parte inferior de la pantalla opciones las cuales pueden ser activadas por medio de su letra característica o ubicándose con los cursores (teclas de flechas) en el comando deseado. El programa se graba con la

opción Save y con el comando Print se imprime el listado del programa.

Edit Comands: Save Print Return Help Esc Quit

- **CMPL** (Compilador).- Este módulo realiza la tarea de buscar errores de sintaxis. Cuando detecta uno o varios errores, presenta en la pantalla los errores cometidos; en este caso, luego de corregido el error se debe presionar ALT N para ir al siguiente error.

- **SMLT** (Simulador).- Es el módulo que viene luego de que la compilación de un programa haya sido completamente correcta. Aquí se realiza la corrida del programa y la tarea de simulación propiamente dicha. Además permite realizar cambios, con la opción Change, en los valores de los datos de entrada y/o en las especificaciones de la corrida sin necesidad de ir a la edición del programa; pero estos cambios son solamente temporales, ya que una vez que se finalizó la simulación del modelo con los nuevos datos, los datos antiguos toman nuevamente vigencia. El comendo Go inicia la simulación. Las opciones que se presentan son:

Simulate Comands: Change Save Go Preserve Resumen Help Esc Quit

- **VIEW** (Visor).- Es un módulo que permite mirar los resultados de la simulación, ya sea en forma tabular como en

forma gráfica (Plot), permitiendo al usuario escoger las variables que se desea analizar; así con Select_Tabulate se seleccionan las tablas que se desean observar, mientras que con Tabulate_All, se observan todas las variables tabuladas.

View Comands: Plot Select_Tabulate Tabulate_All Old Help Quit

- DOC (Documentador).- Permite realizar reportes en los programas y posee una serie de comandos que realizan funciones específicas. Presenta información sobre la estructura del programa.

Un modelo se construye con variables y constantes que se llamarán cantidades. Las variables, son calculadas durante la simulación, mientras que las constantes pueden ser calculadas una vez y al comienzo de la simulación.

Las proposiciones siempre comienzan con un identificador tal que el dynamo reconozca el tipo de información que contiene la proposición. Este identificador debe estar ubicado en la primera columna. Los diferentes identificadores que existen son:

L indica que las ecuaciones son de nivel, es decir son ecuaciones diferenciales. Cada una de estas ecuaciones representa una variable de estado.

R indica que son ecuaciones de flujo y representa la variación de las variables de estado del sistema.

A son ecuaciones auxiliares y permiten calcular funciones algebraicas simples de variables (en el tiempo) y de constantes.

N se las llama de valor inicial, son calculadas una sola vez y durante la inicialización del modelo.

S se las llama suplementarias, son igual que las auxiliares, pero las variables calculadas deben ser usadas en ecuaciones del mismo tipo .

E se las llama exógenas, los valores de estas variables son aportados por el usuario desde fuera del modelo.

C son valores constante, son asignados para ser utilizados durante todo el programa.

K se las llama de reinicialización de valores iniciales, es decir se asigna un cierto valor a una variable, pero luego puede tomar diferentes valores.

Postíndices temporales.-

K, J, JK, KL, son postíndices temporales, es decir son nombres que deben ser dados para variables temporables subscritas. En el dynamo se entenderá como variables postcritas, de la siguiente manera:

x.K indica que la cantidad x está representada en el tiempo K.

La aplicación de este paquete se ilustra con el mismo ejemplo utilizado para el TUTSIM.

La ecuación diferencial que describe el modelo del sistema de la figura A.1 es:

$$x''(t)+.05x'(t)+0.01x(t)-F(t)$$

donde:

F(t) = u(t)

entonces:

$$x''(t)+.05x'(t)+0.01x(t)-u(t)$$

para mayor facilidad y entendimiento se procede a pasar a variables de estado, entonces:

x1' = x2

x2' = −0.01 x1 − .05 x2 + u(t)

y = x1

Se procede a pasar al lenguaje del paquete, identificando que tipo de variables se tiene:

```
L X1.K=X1.J+DT*VX1.JK
R VX1.KL=X2.K
L X2.K=X2.J+DT*VX2.JK
R VX2.KL=-0.01*X1.K-.05*X2.K+STEP(1,0)
A Y.K=X1.K
N X1=0
N X2=0
SPEC DT=.1,LENGTH=150,SAVPER=1
```

Donde la primera linea del programa representa una ecuación de nivel para calcular X1 al instante K (X1.K) mediante su valor anterior (X1.J), al instante J, con un incremento de tiempo DT, o sea con integración numérica en la cual VX1.JK representa la variación de la variable X1 en el intervalo JK (desde el instante J hasta el instante K).

La segunda línea del programa es una ecuación de flujo que define la variación de la variable X1 en el intervalo KL (desde en instante K hasta el instante L), según la expresión X1' = X2.

Se utiliza el instante K al instante L, pues la derivada, en sistemas discretos implica un adelanto en el tiempo.

La tercera línea reprenta una ecuación de nivel, pero

para la variable X2.

La cuarta línea representa la ecuación de flujo para el cálculo de la ecuación de la variable X2, según la expresión X2' = -.Ø1*X1-.Ø5*X2 + u(t) donde **STEP** es el comando para la función de excitación paso con amplitud unitaria e inicio desde el instante t=Ø.

La quinta línea de programa es una ecuación auxiliar para definir en forma algebraica la salida y en función de la variable X1 en el instante K.

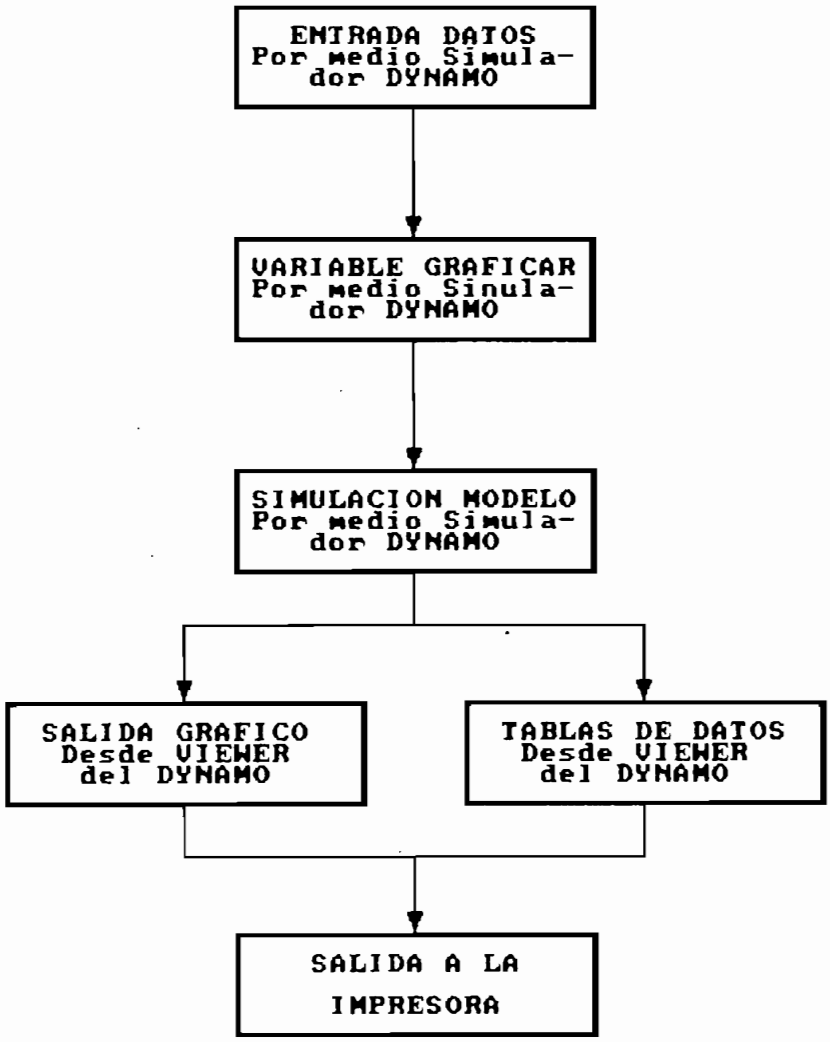
Las líneas 6 y 7 del programa permiten inicializar las variables X1 y X2.

La última línea del programa **SPEC** especifica los parámetros esenciales que se desea que tenga la simulación. Los parámetros que se especifican son:

- DT es el incremento de tiempo, intervalo entre time.J y time.K DT
- LENGTH es el valor del tiempo cuando la corrida termina, es el tiempo de simulación
- SAVPER intervalo de tiempo entre los resultados guardados.

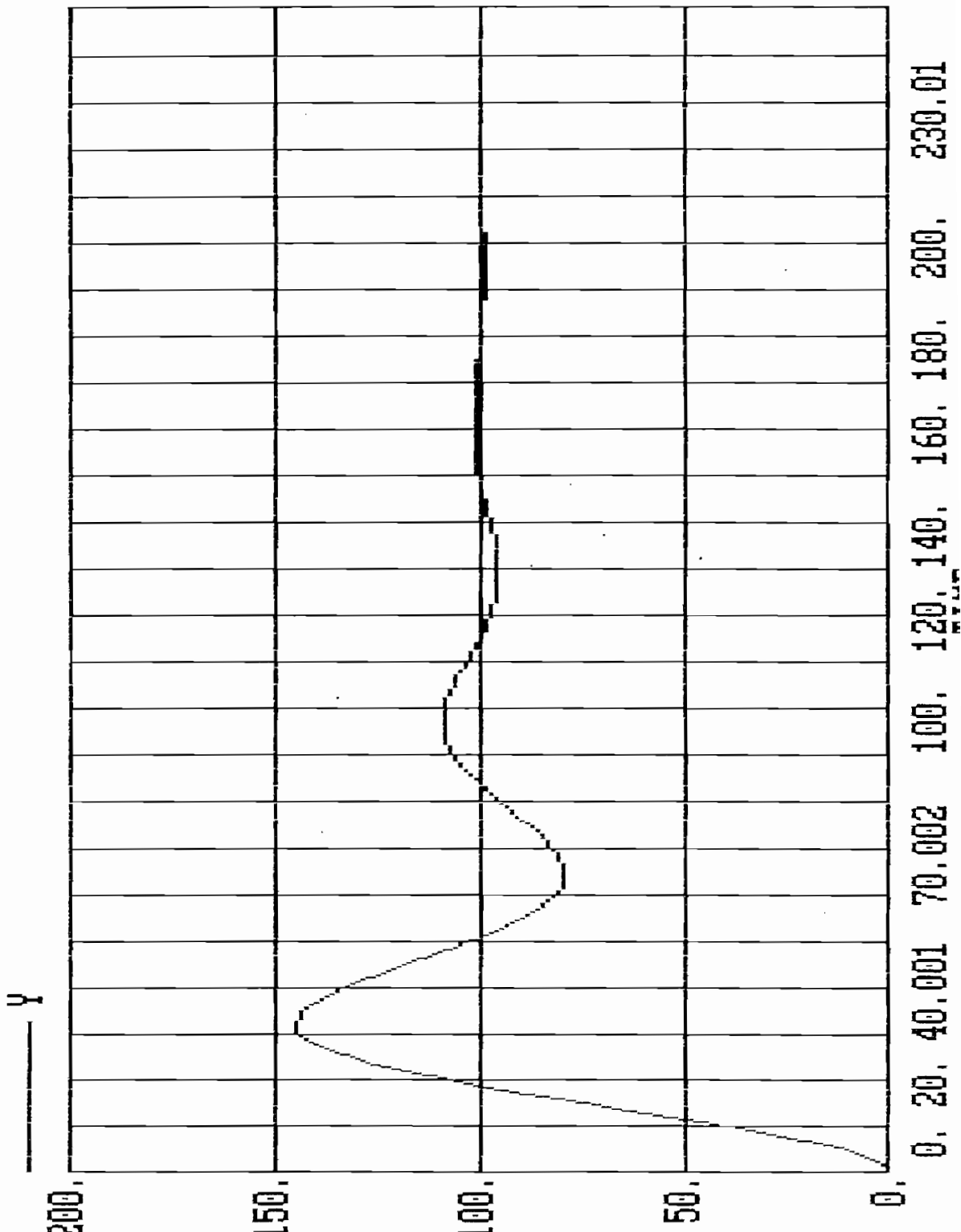
Concluida la secuencia de programación según la

secuencia de la figura A.8, se obtiene como resultado una respuesta en el tiempo como se puede ver en la figura A.9.



ESTRUCTURA BASICA DE LOS PROGRAMAS

FIGURA A.8



RESPUESTA EN EL TIEMPO DE LA SALIDA DEL SISTEMA
FIGURA A.9

A.4. CAD CONTROL.-

Su software es de utilidad para el estudio de varias áreas como son: control clásico, control de datos muestreados, control moderno, control óptimo, etc., siendo una herramienta útil para un mejor entendimiento, adquiriendo destreza en la práctica de ingeniería de control, tanto para análisis como para diseño.

Está escrito en el lenguaje Micro-soft Advanced Basic teniendo la libertad de ejecutarlo en un computador personal IBM ó compatible, con un microprocesador mínimo del tipo 8088, en adelante, requiriéndose como mínimo unos 192 Kb de memoria RAM y un monitor monocromático.

Como parte del software se hace necesario un sistema operativo DOS como un mínimo de versión de 2.0

Para comenzar el programa se debe ejecutar la sentencia CC.

Al ser el CAD CONTROL un programa de análisis y diseño para sistemas de control, hace que tenga la característica de ser altamentente interactivo, siendo capaz de acceder a una amplia variedad de algoritmos.

Consiste en una serie de programas (módulos) que son enlazados teniendo a posibilidad de 230 diferentes comandos, con una estructura jerárquica.

Puede ser utilizado para trabajar en el espacio de estado, en el dominio de la frecuencia, para la conversión de modelos de sistemas continuos o discretos, permite un buen despliegue de gráficos, por lo que es ampliamente utilizado en la simulación y solución de problemas de control.

Otra importante razón para su aceptación es la capacidad de hacer macros y archivos con sorprendente rapidez de ejecución. Además, la habilidad de crear nuevos comandos, estos pueden ser ejecutados para unirlos y poder correrlos en archivos ejecutables (intérprete BASIC) permitiendo la expansión del programa.

Por la característica de los sitemas que puede ser de simple entrada, simple salida (SISO), y múltiple entrada, múltiple salida (MIMO) o sistemas de muestreo de datos, el programa presenta varias alternativas.

Así, los problemas SISO pueden ser modelados por la transformada de Laplace, transformada z o w teniendo las herramientas de análisis de Bode, Nyquist, Nichols, lugar

geométrico de las raíces en el dominio "s" ó "z", con las facilidades correspondientes de simulación.

Para sistemas MIMO se tiene las facilidades que presentan las rutinas para sistemas multivariables en el espacio de estado.

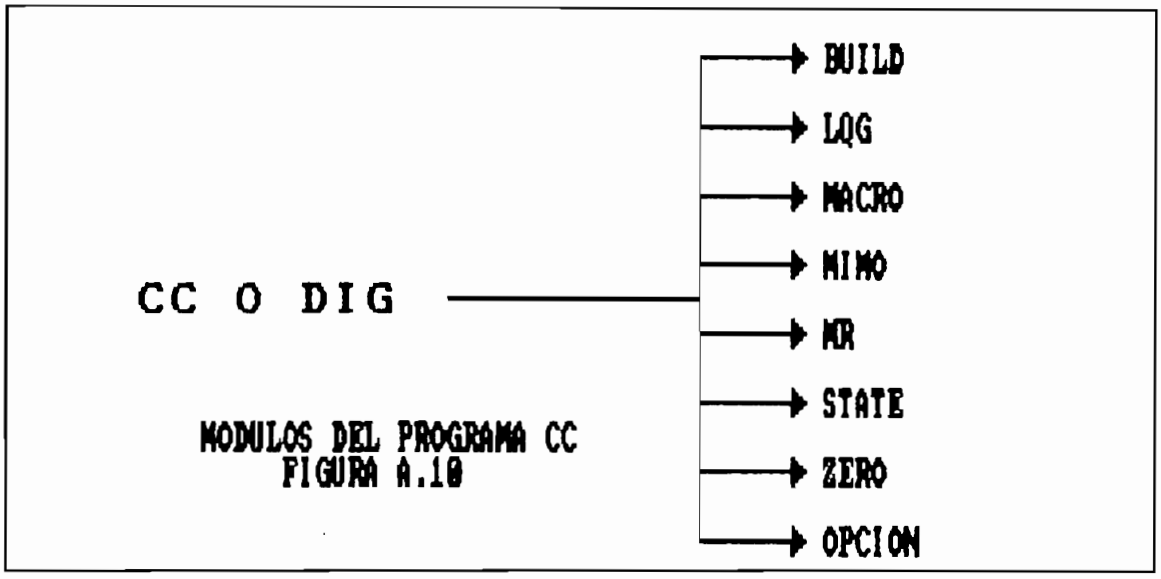
Para sistemas muestreados se requerirá la utilización en el modo DIGITAL que permite varias opciones para este tipo de sistemas, opciones similares a las rutinas CC para sistemas continuos.

El CAD CONTROL es considerado como un "manejador de comandos", que son colocados en una jerarquía, separados en grupos de comandos, lo cual da una enorme flexibilidad al usuario, aparte de ser altamente interactivo, facilitando el ingreso de datos y comandos por parte del usuario.

Esta estructura del CC se ilustra en la figura A.10

Al visualizar la figura A.10 se puede apreciar que las más altas jerarquías de comandos del CAD CONTROL se tiene al escoger trabajar en modo discreto o continuo, así, en el modo continuo se está trabajando con la función de transferencia, es decir, utilizando la transformada de Laplace, en el dominio "s". Al hacerlo en modo digital se

estará trabajando con una función de transferencia en el dominio "z".



MODULOS DEL PROGRAMA CC
FIGURA A.10

Un breve resumen de los comandos más importantes del CC se da a continuación:

ANALOG	convierte a modo digital
BODE	grafica la magnitud y la fase versus la frecuencia
BUILD	crea y cambia funciones de transferencia
CONTROLLABILITY	determina si el sistema es controlable y observable
CONVERT	discretiza G(s) para diferentes métodos de integración. Solo es permitido en modo análogo

DIGITAL	cambia a modo digital
DISPLAY	muestra la función de transferencia
DTIME	grafica la respuesta en el tiempo de un sistema digital a diferentes entradas
ENTER	ingreso de la función de transferencia (análogo o discreta)
FREQUENCY	crea un archivo de frecuencia, debe ser llamado antes de cualquier gráfico
ILT	calcula y muestra la transformada inversa de Laplace
INVERSE NYQUIST	grafica el inverso de Nyquist, que es Im{1/F(jW)} vs. Re{1/F(jw), parametrizado en w
IZT	calcula y muestra la transformada inversa z
MACRO	es usado para crear y editar macros, son archivos indirectos de los comandos
MIMO	trabaja con matriz función de transferencia en sistemas multivariables r x m. Cada elemento es una función de transferencia y puede ser tratada como una SISO de los niveles de comandos CC y BUILD
MR	permite trabajar con submúltiplos del periodo de muestreo
NICHOLS	crea la carta de Nichols que es ¦F(jw)¦

	vs. Arg.|F(jw)|, parametrizado en w
NYQUIST	crea el gráfico de Nyquist, que es Im|F(jw)| vs Re|F(jw)|, parametrizado en w
PFE	calcula fracciones parciales
PFZ	calcula polos y ceros de las funciones de transferencia
ROOT LOCUS	calcula y grafica el lugar geométrico de las raíces
ROUTH	determina la estabilidad del sistema en lazo cerrado
STABILITY	determina la estabilidad del sistema en lazo cerrado
STATE	contiene algoritmos para trabajar con sistemas definidos en el espacio de estado
STORE	crea un archivo en un disco y almacena coeficientes de la función de transferencia
TIME	grafica la respuesta en el tiempo de un sistema análogo

Los modelos matemáticos en el CAD CONTROL usan operadores lineales invariantes en el tiempo, así para sistemas análogos las señales de entrada y salida son funciones continuas. Para sistemas digitales la entrada y la

discreto.

Al hablar de Macros se pueden crear archivos, según las necesidades, siendo su principal utilidad el colocar sentencias que serían ejecutas una tras otra, evitando un trabajo tedioso de digitación, cada vez que sea invocada. Se debe aclarar que un macro es ingresado desde los mismos niveles del programa.

A.5. PC-MATLAB.-

Es un programa interactivo que es de gran ayuda para estudiantes de ingeniería tanto para cálculos numéricos, como para simulación. Se debe mencionar que este programa proporciona al usuario la facilidad de crear y/o utilizar los archivos de extensión .m, que permiten añadir nuevas rutinas a las ya existentes, según las necesidades del usuario.

Es un paquete completamente escrito en lenguaje C, lo que le da una gran versatilidad tanto para la realización de gráficos como para funciones aritméticas, así como para funciones definidas por el usuario, haciéndole un programa flexible, poderoso y científico.

Los requerimientos mínimos tanto de hardware como de software son:

- IBM PC, PC/XT, PC/AT ó compatible
- Mínimo 256 Kb de memoria
- Una versión mínima de DOS 2.0
- Mínimo un driver.

Sin embargo se recomienda

- Más memoria (sobre los 512 Kb)
- Una impresora compatible tipo EPSON por ejemplo
- Un segundo driver o disco duro

El PC-MATLAB es un programa que si bien es un paquete capaz de resolver matrices, es decir ser un simple ejecutor de cálculos numéricos, también tiene la gran cualidad de resolver problemas de propósito especial que surgen en disciplinas como por ejemplo la teoría de control automático, entre otras.

El paquete contiene rutinas básicas para cálculo matricial y gráficos. Además posee otro grupo de rutinas básicas para control clásico, identificación de sistemas, sistemas multivariables, procesamiento digital de señales.

Para su mejor entendimiento en el uso de este paquete, se analizarán sus dos posibilidades, es decir la ayuda matemática que puede prestar, resolviendo matrices,

inclusive complejas, y su utilidad para el caso particular en la simulación y análisis en control clásico.

Así, en la resolución de problemas matemáticos, de configuración compleja se trabaja esencialmente con una clase de objetos, matrices rectangulares con la posibilidad de incluir elementos complejos, pudiéndose trabajar con matrices de 1 por 1, que vendrían a ser escalares, matrices con una sola fila o una sola columna, que serían vectores propiamente dichos.

Se debe anotar que si bien no se encontrará ninguna dificultad numérica, obteniéndose del PC-MATLAB respuestas precisas sin considerar el modelo o conversión que se haya escogido, sin embargo, para modelos de mayor orden y modelo MIMO (múltiples entradas / múltiples salidas), la precisión aritmética de un computador personal puede no ser tan adecuada por lo que se debe tener la precaución, para lo cual se debe considerar:

1.- Un buen condicionamiento del problema
2.- Un algoritmo que sea numéricamente estable y de precisión aritmética
3.- Una buena implementación de un algoritmo de software.

Así pues se dice que un problema está "bien

condicionado" cuando a pequeños cambios en los datos causa un correspondiente cambio en la solución.

Un algoritmo es "numéricamente estable" si no introduce ninguna sensibilidad a perturbaciones que sean inherentes al problema.

Se debe aclarar que de un algoritmo estable se puede esperar datos garantizados con una cierta exactitud de problemas mal condicionados, pero un algoritmo inestable produce soluciones pobres aún si el problema está bien condicionado.

A continuación se hace una síntesis de los principales comandos, cuando se lo utiliza como una herramienta matemática:

Operaciones Fundamentales.-

- Ingreso de matrices simples
- Ingreso de los elementos de matrices mediante espresiones
- Ingreso de sentencias y variables
- Ingreso de notación numérica y expresiones aritméticas
- Ingreso de números y matrices complejas

Operaciones básicas entre matrices.-

- Transpuesta

- Suma y resta
- Multiplicación
- División
- Potenciación
- Funciones trascendentes: matriz exponencial, matriz logarítmica, raíces de una matriz cuadrada

Arreglos de operaciones básicas.-

- Arreglos de sumas y restas
- Arreglos de multiplicación y división
- Arreglos de potenciación
- Arreglos de funciones
- Operaciones lógicas

Funciones básicas.-

- Funciones matemáticas elementales: raíz cuadrada, valor absoluto o magnitud compleja, compleja conjugada, parte imaginaria, parte real, seno, coseno, tangente, arco coseno, arco seno, arco tangente, exponencial, logaritmo natural, logaritmo en base 10.
- Funciones estructuradas matriciales
- Manipulación de las funciones de datos: valor máximo, valor mínimo, suma de elementos, producto acumulativo de elementos, valor medio, mediana, valor estándar.

Entrada/Salida.-

- Edición de entrada
- Formato de salida
- Manejadores: exit, help, demo, listado de variables corrientes, limpieza de la pantalla, remover variables, etc.
- Archivos: dir, type, delete, chdir.

Vectores y submatrices.-

- Notación con dos puntos
- Notación de matrices vacías
- Submatrices

Flujo de control.-

- Lazos FOR
- Lazos WHILE
- Lazos IF y END

Funciones de matrices.-

- Factorización triangular
- Factorización ortogonal
- Descomposición del valor singular
- Valores propios
- Rango

Otras funciones para análisis.-

- Polinomios
- Procesamiento de señales: transformada inversa, transformada de Fourier, convolución, implementación de filtros.

Gráficos.-

- Gráficos X-Y
- Gráficos en tres dimensiones
- Escala manual de los ejes
- Hardcopy
- Gráficos logarítmicos y polares, semilogarítmicos

Comandos y funciones de archivo.-

- Ejecución de comandos ingresados dentro de los archivos .m
- Ejecución de funciones ingresadas dentro de los archivos .m
- Comandos INPUT, ECHO y PAUSE
- Ingreso a texto y macros.

Respecto a la ayuda que presta en la resolución de problemas de control el PC-MATLAB tiene una rica colección de funciones muy útiles para sistemas de control. El álgebra lineal, el cálculo del matrices y análisis numérico proveen de un sistema adecuado para la resolución de sistemas de problemas en ingeniería de control, así como en otras

disciplinas.

Los sitemas de control pueden ser modelados como funciones de transferencia o en el espacio de estado, permitiendo ambas técnicas la clásica y la moderna para el análisis mediante el paquete. Ambos sistemas, en el tiempo continuo y tiempo discreto son manejables. Las conversiones entre estos modelos están provistas.

Los modelos pueden ser presentados y analizados en varias representaciones, asi:

CONVERSION DE MODELOS:

ss2tf Conversión de espacio de estado a función de transferencia

[num,den] = ss2tf(a,b,c,d,lu)

ss2zp Conversión de espacio de estado a polos y ceros

[z,p,k] = ss2zp(a,b,c,d,lu)

tf2ss Conversión de función de transferencia a espacio estado

[a,b,c,d] = tf2ss(num,den)

tf2zp Conversión de función de transferencia a polos y ceros

[z,p,k] = tf2zp(num,den)

zp2tf Conversión de polos y ceros a función de transferencia

[num,den] = zp2tf(z,p,k)

zp2ss Conversión de polos y ceros a espacio de estado

[a,b,c,d] = zp2ss(z,p,k)

c2d Conversión de continuo a discreto

[ab,bd] = c2d(a,b,Ts)

d2c Conversión de discreto a continuo

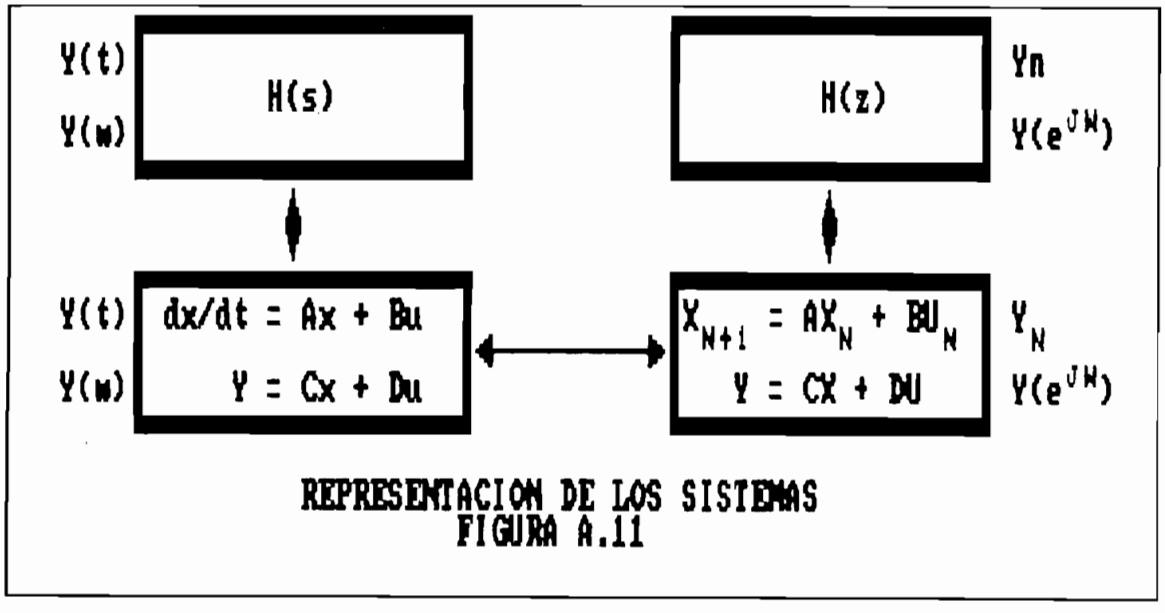
[a,b] = d2c(ad,bd,Ts)

residue Conversión a fracciones parciales

[r,p,k] = residue(b,a)

Cabe anotar que para el uso de estos comandos se hace imprescindible el uso del coprocesador matemático.

Lo dicho se puede esquematizar mediante la figura A.11



REPRESENTACION DE LOS SISTEMAS
FIGURA A.11

RESPUESTA EN EL TIEMPO:

impulse Respuesta impulso, tiempo continuo

[y,x] = impulse(a,b,c,d,lu,t)

[y,x] = impulse(num,den,t)

step Respuesta paso, tiempo continuo

[y,x] = step(a,b,c,d,lu,t)

[y,x] = step(num,den,t)

lsim Simulación continua a entradas arbitrarias

[y,x] = lsim(a,b,c,d,u,t)

[y,x] = lsim(num,den,u,t)

dimpulse Respuesta en tiempo discreto a una entrada impulso

[y,x] = dimpulse(a,b,c,d,lu,n)

[y,x] = dimpulse(num,den,n)

dstep Respuesta en tiempo discreto a una entrada paso

[y,x] = dstep(a,b,c,d,lu,n)

[y,x] = dstep(num,den,n)

dlsim Simulación discreta a entrada arbitrarias

[y,x] = dlsim(a,b,c,d,u)

[y,x] = dlsim(num,den,u)

RESPUESTA DE FRECUENCIA:

bode Gráfico de bode

[mag,phase] = bode(a,b,c,d,lu,w)

[mag,phase] = bode(num,den,w)

nyquist Gráfico de nyquist

```
            [re,im] = nyquist(a,b,c,d,lu,w)
            [re,im] = nyquist(num,den,w)
dbode       Gráfico de bode discreto
            [mag,phase] = dbode(a,b,c,d,lu,w)
            [mag,phase] = dbode(num,den,w)
freqs       Respuesta de frecuencia de filtros análogos
            h = freqs(b,a,w)
freqz       Respuesta de frecuencia de filtros digitales
            h = freqz(b,a,w)
```

CON SELECCION DE GANANCIAS:

```
place       Ubicación de polos
            k = place(a,b,p)
rlocus      Lugar de las raíces
            r = rlocus(num,den,k)
margin      Margen de ganancia y fase
            [Gm,Pm,Wcg,Wcp] = margin(mag,phase,w)
```

UTILITARIOS:

```
damp        Factor de amortiguamiento y frecuencia natural
            [Wn,Z] = damp(A)
ctrb        Matriz de controlbilidad
            co = ctrb(a,b)
obvs        Matriz de observabilidad
            ob = obsv(a,c)
ord2        Generación de un sistema de segundo orden
```

Coma ya se dijo el paquete es una colección de algoritmos expresados como archivos .m, ó conocidos en otros paquetes como macros, que son archivos o funciones que se almacenan conjuntamente, contienen una secuencia de sentencias del paquete, incluyendo referencias de otros archivos del mismo tipo, es decir el PC-MATLAB tiene la habilidad de crear nuevos comandos que resuelven problemas especificos que llegan a ser incómodos al hacerse iterativamente, por lo que son de gran ayuda para la solución de problemas de análisis y diseño.

Como en el caso del paquete CC, en el capítulo II se ilustra la utilización del PC-MATLAB, tanto para el análisis como para el diseño.

2.1.6 **QUICK 500**.-

El Quick 500 es un paquete de software diseñado para la adquisición de datos y control, el mismo que está escrito para ser utilizado por la estación KEITHLEY 500A. Además ha sido diseñado para combinar la velocidad y la programación de un lenguaje estructurado y compilable, con las posibilidades y la capacidad de un paquete de adquisición de datos no compilable.

Los requerimientos básicos de software y hardware son:

- MS-DOS, con una versión mínima de 3.0
- COMPILADOR, Quick Basic con una versión de 4.0 ó mayor
- COMPUTADOR, IBM PS/2, modelo 60 ó compatible, se recomienda trabajar con disco duro, y tener un mínimo de memoria de 640 Kb.
- TARJETA INTERFAZ, las necesarias.

Las principales tareas para la configuración del sistema son:

- la creación de una tabla de configuración CONFIG.TBL, se hace necesario debido a la capacidad que tiene de combinar módulos, rangos, ganancias y configuraciones de entrada y salida para así dar la información de los módulos presentes en el Quick 500.
- la asignación de módulos a slots en forma manual, siempre que no se se utilice la opción SELF-ID que identifica en forma automática los módulos
- la programación de los IONAMES, que no son más que nombres que se ponen para identificar los diferentes canales y puertos de entrada y salida del sistema de adquisición de datos, como parte de la tabla de configuración, así:

  - para canales análogos de entrada: ANLG0, ANLG1, ..., ANLG7

  - para canales análogos de salida: ANOUT0, ANOUT1, ..., ANOUT4.

Los comandos del Quick 500 deben respetar convenciones, de modo que se puedan correr dentro del Quick Basic, siendo el más importante:

CALL, se debe anotar que todos los comandos deben comenzar con esta palabra, ejemplo: CALL ANIN.

Sin embargo existen comandos que además de seguir este formato, permite incluir parámetros de información adicional, los mismos que se deben encerrar entre paréntesis, ejemplo:

```
CALL INTON(rate%, "mil")
```

Se debe tener muy en cuenta que parámetros mal ubicados o escritos causan resultados impredescibles y erróneos.

Además existen palabras reservadas, tales como: MAGNIFY, PAGEO, BT, WBT, WGO, NT, etc., que no pueden ser utilizadas con otras finalidades, sino solo para las que fueron creadas.

Para inicializar el sistema de adquisición y salida de datos, el computador y el área de memoria se tiene:

SOFTINIT, debe utilizarse antes de cualquier otro comando, inicializa parámetros y variables internas.

INIT, es opcional, inicializa algunos elementos del sistema e inicializa el hardware (canales análogos y digitales, enciende interrupciones, remueve ionames, borra áreas de memoria). Las características más relevantes son:

- provee grandes áreas de datos
- provee una estructura de datos con características especiales
- libera espacio de memoria suficiente para el Quick Basic
- implementa dos sistemas de manejo de memoria para el Quick Basic y para el Quick 500.

Los comandos de entrada de datos ANIN, DIGIN, ANINQ, etc.) crean áreas automáticamente al ser ejecutados, no así los de salida, en que las áreas deben ser creadas con anterioridad mediante el uso del comando ARMAKE.

Se tiene básicamente cuatro tipos de áreas creadas:

- BIT: &.- área empaquetada por bits, 1 ó 0. Utilizada para entradas y salidas digitales
- BYTE: @.- cada elemento tiene 8 bits, (0 - 255). Utilizada para entradas y salidas digitales
- WORD: %.- elementos de 2 bytes, (0 - 65535). Utilizada en entradas y salidas análogas.
- REAL: !.- elementos que consumen 4 bytes, almacenados en formato de punto flotante.

Los comandos que ayudan al manejo de las áreas creadas son:

ARSAVE: salvar

ARDEL: borrar

ARLOAD: cargar a memoria

ARSTATUS: pedir información para reportarla

ARLASTP: verifica la ubicación del último elemento del arreglo al que se accesó.

Las tarjeta de interfaz entre KEITHLEY 500A y el computador contienen un generador de interrupciones, que trabaja con las interrupciones no mascarables (NMI) del computador. El comando INTON (Interrupt ON) permite especificar la frecuencia de estas interrupciones.

En cada interrupción el procesador salta de foreground (donde se ejecuta la tarea principal) a background (donde realiza la tarea de fondo), al ejecutarse, regresa al modo de control en foreground hasta la siguiente interrupción.

La interrupción NMI (no mascarable) es utilizada por el sistema de adquisición de datos, siendo la que tiene mayor prioridad. Los comandos que no son de background y los comandos en general de Basic, se llaman comandos de foreground, y se ejecutan en el tiempo en que no se atiende

a las interrupciones.

Background es habilitado por el comando INTON, el cual puede ser utilizado antes o después de que los comandos de background hayan sido inicializados, esto permite arrancar un número de tareas simultáneas ya que estas son sincronizadas.

El comando INTOFF es utilizado cuando se desea suspender las tareas de background.

Los comandos BACKCLEAR, HALT son usados para limpiar tareas de background.

Cada vez que ocurre una interrupción, todos los timers corren y se actualizan, estos timers requieren que el background esté operando, el momento en que éste es suspendido los timers se paran y retienen el último valor hasta que sean reinicializados. Una nueva habilitación de background resetea los timers.

Debido a que se puede trabajar con ciertos niveles de voltaje, se hace referencia a los tipos de relación con el mundo exterior:

1.- Valores Binarios (EUF% = 1).- debe siempre incluirse en

la lista de parámetros. Los valores binarios son positivos y enteros.

2.- Entradas y Salidas de Voltajes (EUF% = 0, 1, 2).-

EUF% = 0 señala voltios

EUF% = 1 señala mv.

EUF% = 2 señala uv.

Es necesario aclarar que el modo de foreground no da la posibilidad de fijar períodos para la adquisición o salida de datos, requerimiento fundamental de un sistema de instrumentación y control, dando como resultado que es impráctica la aplicación de la adquisición y salida tanto unicanal, como multicanal de datos en foreground.

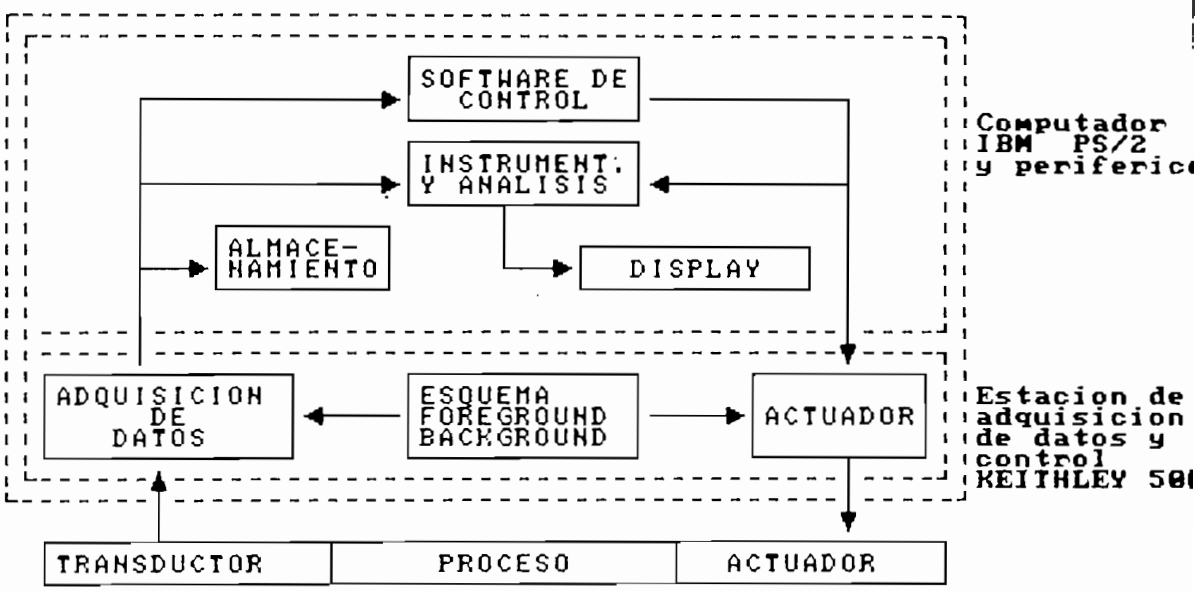
La adquisición de datos se puede realizar utilizando los diferentes esquemas, es decir ya sea en FOREGROUND o BACKGROUND, pudiendo hacerse adquisición de datos unicanal o multicanal.

Un esquema muy general del control en tiempo real es, como se muestra en la figura A.12

Se puede decir que las tareas principales de aplicación o interés de este trabajo son:

UNIDAD DE ADQUISICION DE DATOS Y

CONTROL EN TIEMPO REAL



ADQUISICION, PROCESAMIENTO DE DATOS Y
CONTROL EN TIEMPO REAL

FIGURA A.12

- Adquisición unicanal de datos análogos en background
- Salida unicanal de datos análogos en background
- Identificación de sistemas
- Control digital directo

Actualmente se han desarrollado técnicas que permiten la utilización del computador en línea (ON LINE), siendo un elemento del lazo del control, dejando atrás el control fuera de línea (OFF LINE), en que el computador tomaba un papel auxiliar remoto. Por lo que se pone énfasis en el esquema de control digital directo, que implica el hecho de que el computador está dentro del lazo del control e instrumentación. Este esquema se ve en la figura A.13, y se le conoce como un sistema de datos muestreados donde se manejan datos tanto continuos como discretos.

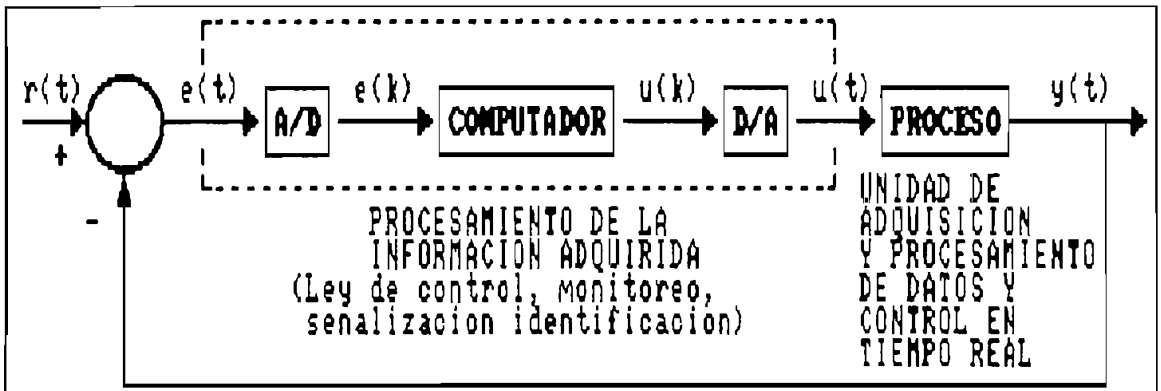


DIAGRAMA DE BLOQUES DE UN CONTROL DIGITAL DIRECTO
FIGURA A.13

A continuación se hace una breve referencia sobre adquisición y salida de datos y sus comandos principales.

Adquisición Unicanal de datos Análogos en Background:

Debido a que se trata de una rutina que realiza el trabajo de fondo mediante interrupciones, hace que permita en un mismo tiempo realizar la adquisición de datos y cualquier otra actividad en foreground, como son impresiones, gráficos, análisis de datos y aplicarlos al control en tiempo real.

- **CALL ANIN:** realiza el muestreo de hasta 64 canales de entrada análogos de datos.

- **CALL ARGETVALF:** permite accesar a un simple valor guardado en un arreglo Quick 500, especificando su ubicación en profundidad y anchura, y expresarlo como un arreglo en Basic. Además guarda el valor en una variable real y puede expresarse en el tipo de unidades más convenientes.

- **CALL INTON:** habilita las interrupciones, a la vez que se inicializa la frecuencia de las mismas de acuerdo a un valor especificado por el usuario.

- **CALL INTOFF:** deshabilita las interrupciones que fueron

habilitadas por el comando INTON.

Adquisición multicanal de datos análogos en background:

Tiene como objetivo realizar la adquisición de datos en tiempo real de varios canales análogos. Los comandos utilizados son prácticamente los mismos que para el caso unicanal, sin embargo se aprovecha para explicar el comando:

- **CALL ARGETF:** permite transferir bloques de datos de un arreglo Quick500 a un arreglo Basic. Maneja cualquier unidad permitida por el sistema.

Salida Unicanal de datos análogos en background:

El objetivo de esta rutina es sacar datos en tiempo real a través de un solo canal análogo, en background, en un esquema que utiliza interrupciones no mascarables.

- **CALL ARMAKE:** crea un arreglo de memoria con el formato Quick500.

- **CALL ANOUT:** rutina que toma valores de un arreglo previamente creado y lo saca a través de canales de salida análogos.

- **CALL ARPUTVALF:** toma valores de una variable Basic y los guarda en un arreglo Quick500. Guarda este valor en una variable real y puede expresarse en el tipo de unidades más conveniente.

Salida multicanal de datos análogos en background:

Esta rutina envía datos en tiempo real a través de varios canales análogos mediante el modo de trabajo de background.

- **CALL ARPUTF:** permite transferir bloques de datos de Basic al formato Quick500, realizando el trabajo 15 veces más rápido que el Basic. Es para todo tipo de unidades de ingeniería.

Comandos Especiales:

Existen comandos que han sido utilizados en tareas referentes a la inicialización de los canales de adquisición y salida de datos. Además se tiene:

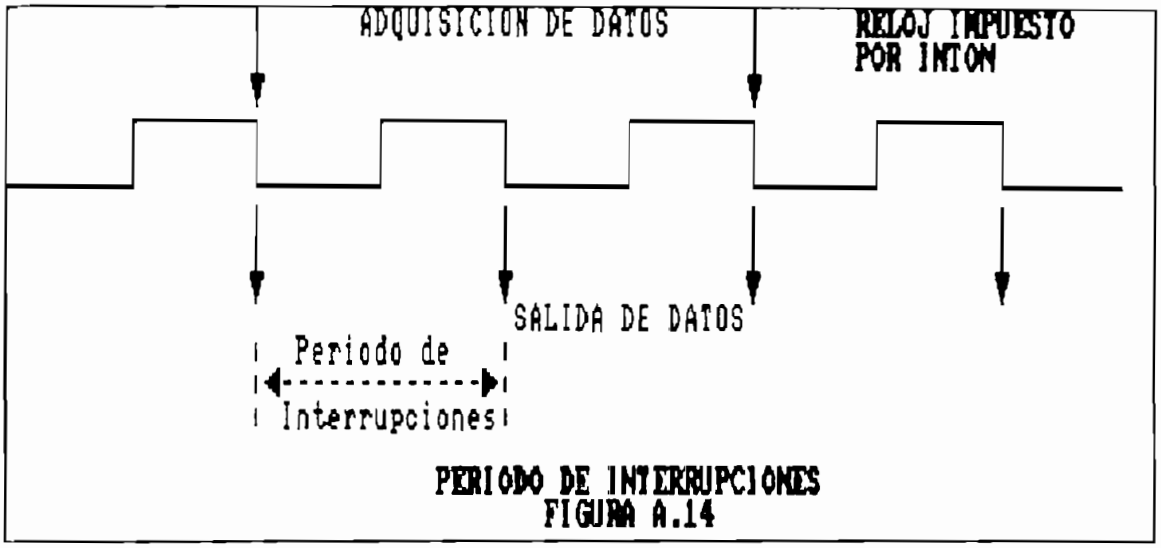
- **CALL IONMANA:** asocia una cadena de caracteres con un canal de entrada o salida de la estación KEITHLEY 500A, identificando el slot de ubicación del módulo, canal conversor análogo a digital y su resolución.

- **CALL IONMDIG:** asocia una cadena de caracteres con un canal de entrada o salida de la estación KEITHLEY 500A, identificando el slot de ubicación del módulo o puerto.

Se debe aclarar que las interrupciones tienen su propio período, el que es impuesto únicamente por el comando CALL INTON y sus parámetros, mientras que el período de muestreo tiene como base el período de las interrupciones. Este período de muestreo está definido por la variable **bintv%** (intervalo en background) en los comandos de entrada y salida de datos en background, como en ANIN, ANOUT, DIGOUT.

El comando INTON inicializa el período, creando un reloj base el que entrega el mando a uno de los dos esquemas, como se muestra en la figura A.14, las tareas de background al terminar, se deshabilitan con el comando INTOFF.

El muestreo de los diferentes canales de adquisición o salida de datos se realiza en cada cierto número de interrupciones por **bintv%**, presente en las tareas de background, y deja la opción de determinar un período diferente para cada tarea de background, con la base de un único período de interrupción.



PERIODO DE INTERRUPCIONES
FIGURA A.14

El período de muestreo suele ser superior al tiempo que se demora entre la adquisición del dato hasta la obtención del resultado que debe ser devuelto a la planta.

Se debe hacer un estimativo de tiempo mínimo que se requiere para obtener el dato de salida tomándose en cuenta los tiempos de Adquisición + conversión A/D + Algoritmo de control + conversión D/A.

Comandos Específicos:

Los comandos que manejan temporizadores del Quick 500 y el comando STATUS ayudan a conocer el estado actual de los timers. Estos son:

- **CALL TIMERSTART:** inicializa los 8 timers, los mismos que incrementan su cuenta en cada interrupción, por lo que su resolución es determinada por la frecuencia de las interrupciones.

- **CALL TIMERREAD:** lee uno de los 8 timers, y permite transferir el valor del tiempo transcurrido a una variable Basic.

- **CALL STATUS:** indica el estado actual de una tarea de background especificada. Informa si la tarea está inactiva o ausente, ejecutándose o esperando algún disparo.

Gráficos en Tiempo real:

Esta técnica representa una gran ventaja, permite planificar en un solo comando muchos parámetros en torno al gráfico, tales como canal o canales a graficarse, arreglos de memoria donde se encuentran los datos, colores del gráfico, tipo de pantalla, valores máximos y mínimos, escalas, etc.

Sin embargo, existe la desventaja de no poder utilizar con eficacia los comandos de graficación ya que toma demasiado tiempo graficar los datos, un tiempo mayor al período de muestreo lo que provoca que se pierdan datos,

haciéndose necesario el uso de un RTMDS de software y una tarjeta aceleradora de gráficos RTM que se instala en el monitor conjuntamente con un adaptador de gráficos. Sin embargo, el equipo con el que se cuenta carece de estos adicionales, por lo que para ciertas frecuencias resulta impráctico graficar en tiempo real.

Dentro de los comandos de gráficos se tiene:

- **CALL GRAPH:** rutina de foreground, grafica horizontalmente los datos almacenados en un arreglo de memoria sobre una base lineal de tiempo.

- **CALL GRAPHRT:** permite graficar en tiempo real, uno o más canales accesados por las siguientes rutinas de background ANIN, ANOUT, DIGOUT.

- **CALL GRLABEL:** imprime el texto que se tiene en pantalla de gráficos de alta resolución, permitiendo tener textos en gráficos creados por HGRAPHRT.

- **CALL HGRAPRT:** rutina en foreground, que realiza gráficos de alta resolución en tiempo real de uno o varios canales de entrada o salida accesados por la rutinas ANIN, ANOUT, DIGOUT de background. Para utilizar este comando antes se debe ubicar la sentencia SCREEN 2 al final se debe tener

SCREEN 0 para abandonar el modo gráfico.

Los arreglos de memoria para datos son creados por los comandos ANIN o ARMAKE. Sirven para almacenar datos que han sido adquiridos a través de canales análogos de entrada o que van a ser enviados a la salida.

Cuando se adquiere datos es importante ubicar estos datos en arreglos de memoria mediante los comandos ARPUTF, ARPUTI, ARPUTVALF o ARPUTVALI; así como también son importantes los arreglos para alamcenamiento para sacar por los canales de salida. En estos casos se tiene los comandos ARGETVALF, ARGETVALI, ARGETF, o ARGETI.

Se debe recalcar que es importante guardar estos datos en disco para su posterior análisis u obtención de resultados gráficos mediante la utilización de hojas electrónicas. Entre los comandos principales se tienen:

- **CALL ARLASTP:** reporta la ubicación del último dato de un arreglo accesado por una rutina de background.

- **CALL ARSTATUS:** reporta información variada acerca de un arreglo de memoria Quick 500 para datos incluyendo profundidad, anchura, último dato accesado e incluso una descripción del arreglo.

- **CALL ARLABEL:** permite hacer una descripción de un arreglo de memoria de datos del Quick 500.

- **CALL ARSAVE:** permite guardar la información en disco. Se le puede ejecutar incluso cuando las interrupciones están habilitadas.

- **CALL ARWRITE:** salva la información de un arreglo de datos en disco en diferentes formatos. Es fácil de exportar datos a hojas electrónicas, procesador de palabras, base de datos y programa de análisis.

- **CALL ARLOAD:** recupera un arreglo especificado desde disco, puede ejecutarse incluso cuando las interrupciones están habilitadas.

- **CALL ARDEL:** borra los arreglos de memoria de datos del Quick 500, puede utilizarse para liberar bloques de memoria previamente ocupados.

Cuando se han recopilado datos y estos son almacenados en arreglos de memoria para datos en formato Quick 500, se los puede procesar para obtener información adicional, pudiendo obtenerse la derivada, integral, media aritmética, desviación estándar de un grupo de datos, etc..

ANEXO B

REALIZACION DE ARCHIVOS DE AYUDA

B.1. MACROS EN CAD CONTROL

B.2. ARCHIVOS .M EN PC-MATLAB

ANEXO B

REALIZACION DE ARCHIVOS DE AYUDA.

B.1. MACROS HEHOS EN EL PROGRAMA CAD CONTROL

El programa CAD CONTROL presenta la facilidad de crear macros, para lo cual se tiene el comando llamado MACRO, logrando ingresar en un medio, en el cual se puede crear, aumentar, listar, añadir, almacenar, etc.. Pueden ser ejecutados desde cualquiera de sus otros comandos.

Los principales archivos del comando MACRO son:

ADD,n,m	permite añadir líneas a un macro
LIST,n,m	permite listar, indicando la primera y la última línea que se desa listar
RECALL,nombre	permite llamar a un macro ya realizado
STORE,nombre	permite almacenar el macro realizado
DELETE,n,m	permite borrar las líneas indicadas
NEW	permite borrar y comenzar un nuevo
QUIT	permite salir del comando macro y regresar a nivel CC>

Para llamar al macro desde cualquier comando del CC

se deberá digitar **@nombre,parámetros**. Estos parámetros están definidos en el macro al encontrar el símbolo &, pudiendo ser &1, &2, etc..

B.1.1. PID

```
1:BUILD
2:G20=(&1*S*S+&2*S+&3)/S
3:G21=G20*&4
4:Q
5:DISP,G21
6:TIME,G21,1,AUTO
```

Para llamar a este macro se deberá digitar:

```
@PID,&1,&2,&3,&4
```

donde:

&1 = constante derivativa

&2 = constante propocional

&3 = constante integral

&4 = función a ser analizada G(s)

B.1.2. REALIMENTACION DE ESTADO

```
1:BUILD
2:G21=&1
3:Q
```

```
 4:STATE
 5:CCF,G21,P1
 6:PDISP,P1
 7:POLE PLACEMENT,P1,P2,1
 8:2
 9:&2,&3
10:&4,&5
11:&6
12:PDISP,P1
13:PDISP,P2
14:FEEDBACK,4,P1,P2,P3 & POLES,P3
15:PDISP,P3
16:SIMULATION
17:P3
18:1,1,1
19:1
20:&7,.1,1
21:PLOT,P3
22:A
23:A
24:SISTEMA CON REALIMENTACION DE ESTADO
25:B
```

Para la utilización de este macro se deberá digitar:

@REALI,&1,&2,&3,&4,&5,&6,&7

donde:

&1 = función a la que se va a aumentar un realimentador de estado

&2,&3,&4,&5,&6 = ingreso de la ubicación de los polos, mismos que deben ser ingresados de la forma:

- parte real, parte imaginaria

&7 = escala del tiempo del gráfico de salida.

B.2. ARCHIVOS .M HECHOS EN EL PROGRAMA PC-MATLAB

Para la creación de archivos .m, que son macros en el paquete PC-MATLAB, es necesario el uso de un EDITOR, la forma más sencilla es utilizando el editor, que puede ser invocado desde el interior del programa escribiendo **EDIT**, una vez ingresado en este ambiente, se procede a digitar el contenido del archivo, luego debe ser grabado con la extensión .m, y con el nombre que consta al invocar la sentencia **function**.

Su ejecución a nivel del programa es sencillo, se digitará su nombre.

B.2.1. LAZO CERRADO

```
%Permite obtener la matriz en lazo cerrado en función
%de la de lazo abierto
```

```
%num = numerador de la función de lazo abierto
%den = denominador de la función de lazo abierto
%nc = numerador  de la función de transferencia en lazo
%cerrado
%dc  = denominador  de la  función de  transferencia en
%lazo cerrado
function[nc,dc]=lazoc(num,den)
nc = num
dc = num + den
end
```

Cabe señalar que el grado del numerador debe ser igual que el denominador en el ingreso de los parámetros, así por ejemplo:

$$G(s) = \frac{35}{s^3+7s^2+14s+8}$$

el ingreso será:

```
num = [0,0,0,35]
den = [1,7,14,8]
```

Cuando se lo necesite se deberá digitar:

```
[nc,dc] = lazoc[num,den]
```

B.2.2. LAZO CERRADO DE UN SISTEMA REALIMENTADO

```
%permite obtener la matriz en lazo cerrado luego de
%haber calculado el realimentador de estado
%a,b son matrices de un sistema
%k es el realimentador de estado
function[lc] = lcsr[a,b,k]
lc = a - b*k
end
```

Para su ejecución se deberá digitar:

```
lc = lcsr[a,b,k]
```

ANEXO C

LISTADO DE PROGRAMAS

C.1. IDENTIFICACION EN SIMULACION (IDENSIMU)

C.2. IDENTIFICACION EN TIEMPO REAL (IDENREAL)

C.3. INSTRUMENTACION EN TIEMPO REAL (INSREAL)

- C.3.1. ADQUISICION DE DATOS
- C.3.2. SALIDA DE DATOS
- C.3.3. ADQUISICION + ALGORITMO + SALIDA DE DATOS

C.4. CONTROL EN TIEMPO REAL

- C.4.1. PID
- C.4.2. REDES

ANEXO C

LISTADO DE PROGRAMAS

Como primer paso para los programas en tiempo real se debe determina el período de muestreo, el mismo que es ajustado, para obtener un período de muestreo y el retardo necesario para la salida del dato, en base a un valor de tiempo necesario para la ejecución del programa, en el que se consideró el tiempo mínimo tanto para la adquisición y salida de datos en background (.1 ms.), así como según la amplitud del programa (número de instrucciones) el tiempo requerido para la ejecución del algoritmo.

Dentro de las subrutinas más importantes que se invocarán más continuamente están:

GRAF.- Realiza el cálculo de la posición del punto en la escala del gráfico, para lo cual ha sido necesario un pequeño sistemas de ecuaciones de transformación de escala, después de lo cual, se tiene las coordenadas del punto en la pantalla y se grafica.

DEFGRAF.- Genera los gráficos iniciales, comienza con la generación de dos bloques con fondo azul y ejes, unos del tiempo y otro de U (salida del control), Y(salida de la

planta), U(entrada de la planta), según el algoritmo que se esté usando.

Se ha añadido un contador del número de iteraciones para determinar el tiempo.

Estas subrutinas mencionadas son usadas en todos los programas, por lo que se vio en la necesidad de comentar acerca de estas. [Ing. Germánico Pinto, Identificación de sistemas en tiempo real, EPN, 1991].

Para la ejecución de cualquiera de los programs es necesario solo digitar su nombre.

C.1. IDENTIFICACION EN SIMULACION (Mínimos Cuadrados Recursivos) (IDENSIMU).-

El programa desarrollado realiza la identificación de parámetros en simulación, es necesario predefinir una planta en base a sus parámetros conocidos; así como el orden del modelo para proceder a la identificación.

El software está desarrollado en el paquete Quick Basic 4.5 de la Microsoft, se trató de respetar la forma estructurada del lenguaje para facilitar la programación y

la utilización por parte del usuario.

Para poder correr el programa se necesita de un computador con tarjeta de gráfico VGA a colores, el programa está diseñado para poder ejecutarlo desde el disco duro de la máquina o desde un diskette, solo es necesario digitar su nombre: IDENSIMU.

Está conformado por las siguientes subrutinas:

CALCULO.- Calcula el valor de "y" al instante "k"

CAMBIO.- Esta subrutina es llamada por la subrutina DATOS. Su función es cambiar los parámetros:

- B: factor de olvido
- TE(): vector de parámetros desconocidos
- TE1(): vector transpuesto de TE()
- α: valor de la varianza

DATAIN.- Ingreso del orden de A(z), B(z), y sus respectivos coeficientes de la planta.

DATAOT.- Ingreso del orden de A(z), B(z) del modelo.

DATOS.- Visualización de los parámetros iniciales de:

- B: factor de olvido = .985
- TE(): inicialmente ceros
- α = 10.000

DEFGRAF.- Realiza la graficación de los ejes: y vs. tiempo y u vs. tiempo de la planta.

DEFGRAF1.- Realiza la graficación de los ejes: y vs. tiempo,

y1 vs. tiempo, de la planta y modelo, en una misma ventana. La otra ventana está **error vs. tiempo**, el error es la diferencia entre el valor de la planta y el modelo.

ENTRAD.- Permite el ingreso del valor de entrada de la planta a ser simulada, la misma que consta de un escalón y de ruido, siendo necesario el ingreso del valor de las dos amplitudes.

GRAF.- Grafica los puntos calculados de la entrada y salida de la planta.

GRAF1.- Grafica los puntos calculados de la salida de la planta, salida del modelo, el error.

INICIO.- Crea la matriz identidad y la matriz de covarianza con el valor α.

MULTFAC.- Permite la multiplicación de una matriz o vector por un escalar.

MULTI.- Permite la multiplicación de matrices

PRINTER.- Permite obtener los valores de los parámetros tanto de la planta como del modelo; así como el número de iteraciones en papel a través de la impresora.

SALIDA.- Hace lo mismo que la subrutina PRINTER, con la única diferencia que los resultados se obtienen solo a nivel de pantalla.

VECTOR.- Permite el cálculo del vector de datos o mediciones x() y su transpuesto.

```
DECLARE SUB defgraf ()
DECLARE SUB defgraf1 ()
DECLARE SUB GRAF (I)
DECLARE SUB GRAF1 (I)
DECLARE SUB DATAIN (N1I, N2I, MI, A(), B())
DECLARE SUB DATAOT (N1O, N2O, MO, N)
DECLARE SUB DATOS (B, TE(), TE1(), MO)
DECLARE SUB CAMBIO (B, TE(), TE1(), ALF, MO)
DECLARE SUB INICIO (P(), AID(), ALF, MO)
DECLARE SUB ENTRAD (NA, POR)
DECLARE SUB CALCULO (U(), Y(), NA, POR, A(), B(), N1I, N2I, JK)
DECLARE SUB VECTOR (JK, X(), X1(), Y(), U(), N1O, N2O)
DECLARE SUB multi (FG(), D(), AD(), MO, MO, MO)
DECLARE SUB MULTFAC (AX(), AZ, AW(), MO, MO)
DECLARE SUB SALIDA (N1I, A(), N2I, B(), N1O, N2O, TE())
DECLARE SUB PRINTER (N1I, A(), N2I, B(), N1O, N2O, TE())


COMMON SHARED N1I, N2I, MI, A(), B(), N1O, N2O, MO, N, TE(), TE1(), B, ALF
COMMON SHARED AI(), P(), AID(), NA, POR, U(), Y(), W1, W2, W3, X(), X1(), JK
COMMON SHARED AL1(), AL2(), AL(), E1(), P1(), P2(), PP(), AL3, PP(), PPT(), TE2()
COMMON SHARED A1(), B1(), Y1(), II, EE()

CLEAR

DIM P(5, 5), TE(10, 10), TE1(1, 10), X(1, 10), X1(10, 1), AID(10, 10)
DIM AL(10, 1), AL1(10, 1), AL2(1, 10), AL3(1, 1), E1(1, 1), TE2(10, 1)
DIM P2(10, 10), A(10), B(10), TEE(10, 1), PP(10, 10), PPT(10, 10), P1(10, 10)
DIM Y(501), U(501), AI(10), Y1(520), A1(501), B1(501), EE(501)


SCREEN 9

COLOR 12, 4
CLS

LOCATE 4, 24
PRINT "MINIMOS CUADRADOS RECURSIVOS"
LOCATE 5, 24
PRINT "--------------------------"
COLOR 14, 1
LOCATE 7, 24
PRINT "Ingrese datos de la planta de la forma"
LOCATE 9, 27
PRINT "-1"
LOCATE 9, 38
PRINT "-k"
LOCATE 9, 45
PRINT "-1"
LOCATE 10, 24
PRINT "A(z  ) Y(t) =(z  ) B(z  ) U(t)"


'                          INGRESO DE DATOS

' Realiza el ingreso de datos de la planta para simulacion,
' modelo a identificar y el numero de iteraciones
```

```
LOCATE 24, 47
INPUT "Presione ENTER para continuar", AD$
CALL DATAIN(N1I, N2I, MI, A(), B())
SALI33:
CALL DATAOT(N1O, N2O, MO, N)
CLS

MI = N1I + N2I
MO = N1O + N2O

'                       INICIALIZACION DE PARAMETROS

' Setea los parametros iniciales tales como la matriz de covarianza P,
' matriz de parametros estimados TE, etc.

' INICIALIZACION DE DATOS
CALL DATOS(B, TE(), TE1(), MO)


U(0) = 5
Y(0) = 0

CALL INICIO(P(), AID(), ALF, MO)


'           SE COLOCA LA ENTRADA CON RUIDO

'Escoje el tipo de entrada u(t) y sus respectivos parametros.
'La senal se scoje escalon mas ruido (random)

CALL ENTRAD(NA, POR)


' CALCULO DE LOS n PRIMEROS VALORES
FOR JK = 1 TO N1I
CALL CALCULO(U(), Y(), NA, POR, A(), B(), N1I, N2I, JK)
NEXT JK


' CALCULO DEL VALOR Y(n) PARA UNA ENTRADA U(n)


FOR JK = N1I TO N1I + N - 1

CALL CALCULO(U(), Y(), NA, POR, A(), B(), N1I, N2I, JK)

'               PROCESO DE IDENTIFICACION

'          COMIENZO ALGORITMO DE MINIMOS CUADRADOS RECURSIVOS

'          ------------------------------------------


COLOR 14, 1
LOCATE 14, 30
PRINT "P R O C E S A N D O"
```

```
'GENERACION DEL VECTOR DE DATOS

CALL VECTOR(JK, X(), X1(), Y(), U(), N1O, N2O)

'CALCULO DEL PROCESO GENERAL

'CALCULO DE LA MATRIZ "L"

CALL multi(P(), X1(), AL1(), MO, 1, MO)


CALL multi(X(), AL1(), AL2(), 1, 1, MO)

AL3 = 1 / (AL2(1, 1) + 1)


CALL MULTFAC(AL1(), AL3, AL(), MO, 1)

CALL multi(X(), TE(), E1(), 1, 1, MO)

'CALCULO DEL ERROR

E = Y(JK) - E1(1, 1)

'CALCULO DE LA MATRIZ "THETA"

CALL MULTFAC(AL(), E, TE2(), MO, 1)

' SUMA DE MATRICES TEE()= TE()+ TE2()

FOR II = 1 TO MO
   FOR JJ = 1 TO 1
      TEE(II, JJ) = TE(II, JJ) + TE2(II, JJ)
   NEXT JJ
NEXT II

FOR II = 1 TO MO
  TE(II, 1) = TEE(II, 1)
NEXT II

' ALMACENAMIENTO DE LOS PARAMETROS DEL MODELO
FOR II = 1 TO N1O
     A1(II) = TE(II, 1)
NEXT II
FOR II = N1O + 1 TO MO
     B1(II - N1O) = TE(II, 1)
NEXT II

' CALCULO DE LA SALIDA DEL MODELO
CALL CALCULO(U(), Y1(), NA, POR, A1(), B1(), N1O, N2O, JK)

EE(JK) = Y(JK) - Y1(JK)


'CALCULO DE LA MATRIZ "P"

CALL multi(AL(), X(), P1(), MO, MO, 1)
```

```
' RESTA DE MATRICES P2() = AID() - P1()
FOR II = 1 TO MO
   FOR JJ = 1 TO MO
      P2(II, JJ) = AID(II, JJ) - P1(II, JJ)
   NEXT JJ
NEXT II

CALL multi(P2(), P(), PP(), MO, MO, MO)

CALL MULTFAC(PP(), B, PPT(), MO, MO)


FOR II = 1 TO MO
     FOR JM = 1 TO MO
       P(II, JM) = PPT(II, JM)
     NEXT JM
NEXT II
NEXT JK




COLOR 14, 1

' SALIDA DE RESULTADOS

CALL SALIDA(N1I, A(), N2I, B(), N1O, N2O, TE())
COLOR 10, 1
LOCATE 25, 35
INPUT "Presione ENTER para continuar"; AC2$

' MENU DE PREGUNTAS

repit:
CLS
LOCATE 3, 1
PRINT SPACE$(28); "1) Desea gráficos"
PRINT SPACE$(28); "2) Desea Imprimir"
PRINT SPACE$(28); "3) Desea otra simulación"
PRINT SPACE$(28); "4) Desea terminar"
LOCATE 11, 29
INPUT opcion
IF opcion < 1 OR opcion > 4 THEN
GOTO repit
END IF
SELECT CASE opcion


CASE 1
   CLS
   COLOR 14, 1
   GOTO repita


CASE 2
  CALL PRINTER(N1I, A(), N2I, B(), N1O, N2O, TE())
```

```
CASE 3
   CLS
   SCREEN 9
   COLOR 14, 1
   GOTO SALI33

CASE 4
    CLS
    SCREEN 9
    COLOR 14, 1
    GOTO finish
    END SELECT

repita:
     COLOR 14, 1
     CLS
     LOCATE 4, 40
     PRINT "GRAFICOS"
     PRINT SPACE$(28); "1) ENTRADA   -  SALIDA de la planta"
     PRINT SPACE$(28); "2) SALIDA de la planta - SALIDA del modelo"
     PRINT SPACE$(28); "3) Salir opción de gráficos"
     LOCATE 11, 29
     INPUT opcion
     IF opcion < 1 OR opcion > 3 THEN
        GOTO repita
    END IF
    SELECT CASE opcion


CASE 1
    CLS
    I = 0
    defgraf
    FOR JK = 0 TO N
       COLOR 13
       LOCATE 11, 40
       PRINT USING "Salida de la planta: ##.#### v."; Y(JK)
       COLOR 10
       LOCATE 13, 40
       PRINT USING "Entrada de la planta: ##.#### v."; U(JK)
    GRAF (I)
    I = I + 1
    NEXT JK
    LOCATE 25, 1
    INPUT "Presione ENTER para continuar"; AC2$
    SCREEN 9
    COLOR 14, 1
    GOTO repita


CASE 2
    CLS
    I = 0
defgraf1
   FOR JK = 0 TO N
      LOCATE 11, 40
      COLOR 13
      PRINT USING "Salida del modelo y: ##.#### v."; Y(JK)
```

```
      LOCATE 12, 40
      COLOR 12
      PRINT USING "Salida de la planta y1: ##.#### v."; Y1(JK + N10)
      LOCATE 13, 40
      COLOR 10
      PRINT USING "Error e: ##.###### v."; EE(JK)
      GRAF1 (I)
      I = I + 1
   NEXT JK
      LOCATE 25, 1
      INPUT "Presione ENTER para continuar"; AC2$
      SCREEN 9
      COLOR 14, 1
   GOTO repita


CASE 3
   CLS
   SCREEN 9
   COLOR 14, 1
   GOTO repit

END SELECT
finish:
END
```

```
SUB CALCULO (U(), Y(), NA, POR, A(), B(), N1I, N2I, JK)
CLS
W1 = 0
W2 = 0
W3 = 0

U(JK) = NA + POR * RND
FOR I = 1 TO N2I
  IF (JK - I) < 0 THEN
    W1 = W1
  ELSE
    W1 = W1 + B(I) * U(JK - I)
  END IF
NEXT I

FOR I = 1 TO N1I
   IF (JK - I) < 0 THEN
    W2 = W2
   ELSE
    W2 = W2 + A(I) * Y(JK - I)
   END IF
NEXT I
W3 = W1 + W2
Y(JK) = W3
```

```
END SUB

SUB CAMBIO (B, TE(), TE1(), ALF, MO)
CLS
OUT2:
SCREEN 9
COLOR 14, 1
SIGA:
LOCATE 3, 20

COLOR 4
PRINT "CAMBIO DE INICIALIZACION DE PARAMETROS"
COLOR 14
     LOCATE 5, 21
     INPUT "Factor de olvido = B = "; B
     IF B > 1 THEN
     GOTO SIGA
     ELSE
     GOTO SIGA1
     END IF
SIGA1:
LOCATE 7, 21
PRINT "Matriz de parámetros iniciales THETA"
LOCATE 8, 1
FOR I = 1 TO MO
     PRINT SPACE$(22); "TE("; I; ") = ";
     INPUT AI(I)
     TE(I, 1) = AI(I)
     TE1(1, I) = AI(I)
NEXT I
PRINT
PRINT SPACE$(22); "Valor de ALFA = ";
INPUT ALF

PRINT
PRINT
COLOR 6
PRINT SPACE$(30); "Desea corregir datos ? (s/n)"
ovt1:
acb$ = INPUT$(1)
   IF acb$ = "s" OR acb$ = "S" THEN
   GOTO OUT2
   ELSE
   IF acb$ = "n" OR acb$ = "N" THEN
   GOTO out13
   ELSE
   BEEP
   GOTO ovt1
   END IF
   END IF


out13:

END SUB

SUB DATAIN (N1I, N2I, MI, A(), B())
```

```
SALT1:
CLS
COLOR 10, 1
LOCATE 4, 25
PRINT "INGRESO DE DATOS DE LA PLANTA"
COLOR 14, 1
LOCATE 7, 31
INPUT "Orden de A(z) ="; N1I
LOCATE 8, 31
INPUT "Orden de B(z) ="; N2I
PRINT
PRINT
FOR I = 1 TO N1I
PRINT SPACE$(31); "Coeficientes de a("; I; ")=";
INPUT A(I)
NEXT I

     FOR I = 1 TO N2I
     PRINT SPACE$(31); "Coeficientes de b("; I; ")=";
     INPUT B(I)
     NEXT I

PRINT
PRINT
COLOR 6
PRINT SPACE$(30); "Desea corregir datos? (s/n)"
sal1:
     acb$ = INPUT$(1)
     IF acb$ = "s" OR acb$ = "S" THEN
     GOTO SALT1
     ELSE
       IF acb$ = "n" OR acb$ = "N" THEN
       GOTO SAL3
       ELSE
       BEEP
       GOTO sal1
       END IF
     END IF
SAL3:

END SUB

SUB DATAOT (N1O, N2O, MO, N)
SAL33:
     CLS
     COLOR 10, 1
     LOCATE 4, 22
     PRINT "INGRESO DEL MODELO A IDENTIFICARSE"
     COLOR 14, 1
     LOCATE 7, 22
     INPUT "Orden de A(z) ="; N1O
     LOCATE 8, 22
     INPUT "Orden de B(z) ="; N2O
VAY1:
LOCATE 11, 22
INPUT "Número de interaciones = N "; N

PRINT
```

```
PRINT
COLOR 6
PRINT SPACE$(30); "Desea corregir datos ? (s/n)"
SALJ:
acb$ = INPUT$(1)
     IF acb$ = "s" OR acb$ = "S" THEN
     GOTO SAL33
     ELSE
        IF acb$ = "n" OR acb$ = "N" THEN
        GOTO SAL3J
        ELSE
        BEEP
        GOTO SALJ
        END IF
     END IF
SAL3J:

END SUB

SUB DATOS (B, TE(), TE1(), MO)

SCREEN 9

COLOR 14, 1
CLS
B = .985
formato$ = "#.###"
COLOR 4
LOCATE 2, 25
PRINT "INICIALIZACION DE PARAMETROS"
LOCATE 4, 1
COLOR 14
PRINT SPACE$(20); "Factor de olvido = B = "; USING formato$; B
FOR I = 1 TO MO
       TE(I, 1) = 0
       TE1(1, I) = 0
NEXT I
LOCATE 6, 21
PRINT "Matriz de parámetros THETA"
LOCATE 7, 1
FOR I = 1 TO MO
PRINT SPACE$(22); "TE("; I; ") = "; TE(I, 1)
NEXT I
ALF = 100000
LOCATE 18, 22
PRINT "Valor de ALFA = "; ALF
PRINT
PRINT
COLOR 6
PRINT SPACE$(30); "Desea corregir datos? (s/n)"
out1:
acb$ = INPUT$(1)
  IF acb$ = "s" OR acb$ = "S" THEN
     CALL CAMBIO(B, TE(), TE1(), ALF, MO)
     ELSE
       IF acb$ = "n" OR acb$ = "N" THEN
       GOTO out3
        ELSE
```

```
        BEEP
        GOTO out1
       END IF
   END IF
out3:
CLS

END SUB

DEFINT I-N
DEFDBL A-H, O-Z
SUB defgraf
SCREEN 9
CLS
LINE (0, 0)-(639, 130), 1, BF
LINE (0, 0)-(639, 130), 15, B
LINE (40, 15)-(629, 15), 15, , &HF0F0
LINE (30, 65)-(629, 65), 17
LINE (40, 115)-(629, 115), 15, , &HF0F0
LOCATE 6, 74
COLOR 15
PRINT "Tiempo"
LINE (40, 6)-(40, 124), 7
LOCATE 2, 4
COLOR 13
PRINT "y"

'Para el canal de entrada
LINE (0, 192)-(639, 310), 1, BF
LINE (0, 192)-(639, 310), 15, B
LINE (40, 200)-(629, 200), 15, , &HF0F0
LINE (30, 250)-(629, 250), 7
LINE (40, 300)-(629, 300), 15, , &HF0F0
LOCATE 19, 74
COLOR 15
PRINT "Tiempo"
LINE (40, 195)-(40, 305), 7
LOCATE 15, 4
COLOR 10
PRINT "u"


END SUB

DEFINT I-N
DEFDBL A-H, O-Z
SUB defgraf1
SCREEN 9
CLS
LINE (0, 0)-(639, 130), 1, BF
LINE (0, 0)-(639, 130), 15, B
LINE (40, 15)-(629, 15), 15, , &HF0F0
LINE (30, 65)-(629, 65), 17
LINE (40, 115)-(629, 115), 15, , &HF0F0
LOCATE 6, 74
COLOR 15
PRINT "Tiempo"
LINE (40, 6)-(40, 124), 7
```

```
LOCATE 2, 4
COLOR 13
PRINT "y"
LOCATE 3, 4
COLOR 12
PRINT "y1"


'Para el canal de entrada
LINE (0, 192)-(639, 310), 1, BF
LINE (0, 192)-(639, 310), 15, B
LINE (40, 200)-(629, 200), 15, , &HF0F0
LINE (30, 250)-(629, 250), 7
LINE (40, 300)-(629, 300), 15, , &HF0F0
LOCATE 19, 74
COLOR 15
PRINT "Tiempo"
LINE (40, 195)-(40, 305), 7
LOCATE 15, 4
COLOR 10
PRINT "e"

END SUB

SUB ENTRAD (NA, POR)
CLS
SCREEN 9
COLOR 10, 1
LOCATE 4, 22
PRINT "Señal de entrada: ESCALON +  RUIDO RANDOM"
COLOR 14, 1
LOCATE 6, 22
PRINT "Amplitud del escalón =";
INPUT NA
LOCATE 7, 22
PRINT "Amplitud del ruido =";
INPUT POR


END SUB

DEFSNG A-Z
SUB GRAF (I)
SCREEN 9
G = Y(JK)
IF G > 0 THEN
  IF G > 9.99 THEN
        PSET (I + 40, 15), 4
  ELSEIF G < .04 THEN
        PSET (I + 40, 65), 4
  ELSE
        Y = -5 * G + 65
        PSET (I + 40, Y), 13
  END IF
ELSE
    IF G < -9.99 THEN
        PSET (I + 40, 115), 4
     ELSEIF G > -.04 THEN
```

```
      PSET (I + 40, 65), 4
      ELSE
      Y = -(5 * G - 65)
      PSET (I + 40, Y), 13
END IF
END IF
L = U(JK)
IF L > 0 THEN
  IF L > 9.99 THEN
        PSET (I + 40, 200), 4
  ELSEIF L < .04 THEN
        PSET (I + 40, 250), 4
  ELSE
        U = -5 * L + 250
        PSET (I + 40, U), 10
 END IF
ELSE
    IF L < -9.99 THEN
        PSET (I + 40, 300), 4
      ELSEIF L > -.04 THEN
      PSET (I + 40, 250), 4
      ELSE
      U = -(5 * L - 250)
      PSET (I + 40, U), 10
      END IF
END IF


END SUB

DEFSNG A-Z
SUB GRAF1 (I)
SCREEN 9
G = Y(JK)
IF G > 0 THEN
  IF G > 9.99 THEN
        PSET (I + 40, 15), 4
  ELSEIF G < .04 THEN
        PSET (I + 40, 65), 4
  ELSE
        Y = -5 * G + 65
        PSET (I + 40, Y), 13
  END IF
ELSE
    IF G < -9.99 THEN
        PSET (I + 40, 115), 4
      ELSEIF G > -.04 THEN
      PSET (I + 40, 65), 4
      ELSE
      Y = -(5 * G - 65)
      PSET (I + 40, Y), 13
END IF
END IF
D = Y1(JK)
IF D > 0 THEN
  IF D > 9.99 THEN
        PSET (I + 40, 15), 4
  ELSEIF D < .04 THEN
```

```
        PSET (I + 40, 65), 4
  ELSE
        Y1 = -5 * D + 65
        PSET (I + 40, Y1), 12
  END IF
ELSE
    IF D < -9.99 THEN
        PSET (I + 40, 115), 4
      ELSEIF D > -.04 THEN
      PSET (I + 40, 65), 4
      ELSE
      Y1 = -(5 * D - 65)
      PSET (I + 40, Y1), 12
END IF
END IF
L = EE(JK)
IF L > 0 THEN
  IF L > .01 THEN
        PSET (I + 40, 200), 4
  ELSEIF L < .000004 THEN
        PSET (I + 40, 250), 4
  ELSE
        U = -5000 * L + 250
        PSET (I + 40, U), 10
 END IF
ELSE
    IF L < -.01 THEN
        PSET (I + 40, 300), 4
      ELSEIF L > -.000004 THEN
      PSET (I + 40, 250), 4
      ELSE
      U = -(5000 * L - 250)
      PSET (I + 40, U), 10
      END IF
END IF
END SUB

SUB INICID (P(), AID(), ALF, MO)
FOR I = 1 TO MO
   FOR J = 1 TO MO
          IF I = J THEN
          P(I, J) = ALF
          AID(I, J) = 1
          ELSE
          AID(I, J) = 0
          P(I, J) = 0
          END IF
   NEXT J
NEXT I

END SUB

SUB MULTFAC (AX(), AZ, AW(), M2, M3)
FOR II = 1 TO M2
   FOR JJ = 1 TO M3
      AW(II, JJ) = AX(II, JJ) * AZ
   NEXT JJ
NEXT II
```

```
END SUB

SUB multi (FG(), D(), AD(), M4, M5, M6)
FOR II = 1 TO M4
    FOR JJ = 1 TO M5
       AD(II, JJ) = 0
         FOR kk = 1 TO M6
            AD(II, JJ) = AD(II, JJ) + FG(II, kk) * D(kk, JJ)
         NEXT kk
    NEXT JJ
NEXT II

END SUB

SUB PRINTER (N1I, A(), N2I, B(), N1O, N2O, TE())
CLS
LPRINT ""
LPRINT SPACE$(35); "RESULTADOS DE LA SIMULACION"
LPRINT ""
LPRINT ""
LPRINT ""
LPRINT SPACE$(42); "P L A N T A"
LPRINT ""
LPRINT ""
FOR I = 1 TO N1I
LPRINT SPACE$(40); "a("; I; ") = "; A(I)
NEXT I
FOR I = 1 TO N2I
LPRINT SPACE$(40); "b("; I; ") = "; B(I)
NEXT I
LPRINT
LPRINT
LPRINT SPACE$(42); "M O D E L O"
LPRINT
formato$ = "###.####"
FOR I = 1 TO N1O
LPRINT SPACE$(40); "a("; I; ") = "; USING formato$; TE(I, 1)
NEXT I
j2 = N1O + 1
FOR I = 1 TO N2O
LPRINT SPACE$(40); "b("; I; ") = "; USING formato$; TE(j2, 1)
j2 = j2 + 1
NEXT I
LPRINT ""
LPRINT ""
LPRINT SPACE$(35); "Número de iteraciones"; N


END SUB

SUB SALIDA (N1I, A(), N2I, B(), N1O, N2O, TE())
CLS
COLOR 12, 1
LOCATE 5, 35
PRINT "P L A N T A"
PRINT
COLOR 14, 1
```

```
FOR I = 1 TO N1I
PRINT SPACE$(35); "a("; I; ") = "; A(I)
NEXT I
FOR I = 1 TO N2I
PRINT SPACE$(35); "b("; I; ") = "; B(I)
NEXT I
COLOR 12, 1
LOCATE 15, 35
PRINT "M O D E L O"
PRINT
formato$ = "###.####"
COLOR 14, 1
FOR I = 1 TO N1O
PRINT SPACE$(35); "a("; I; ") = "; USING formato$; TE(I, 1)
NEXT I
j2 = N1O + 1
FOR I = 1 TO N2O
PRINT SPACE$(35); "b("; I; ") = "; USING formato$; TE(j2, 1)
j2 = j2 + 1
NEXT I


END SUB

SUB VECTOR (JK, X(), X1(), Y(), U(), N1O, N2O)

FOR I = 1 TO N1O
X(1, I) = Y(JK - I)
X1(I, 1) = Y(JK - I)
NEXT I
FOR I = 1 TO N2O
X(1, N1O + I) = U(JK - I)
X1(N1O + I, 1) = U(JK - I)
NEXT I


END SUB
```

C.2. IDENTIFICACION EN TIEMPO REAL (Mínimos Cradrados Recursivos) (IDENREAL).-

El proceso de identificación en tiempo real consiste básicamente en la aplicación del algoritmo antes desarrollado para la identificación en simulación, conjuntamente con la subrutinas de entrada y salida de datos del Keithley 500A. Es decir, que se utilizará los datos medidos para la determinación de los parámetros de la planta mediante el método de mínimos cuadrados.

Las subrutinas ocupadas son:

ALMACE.- Se almacena los parámetros que se van obteniendo.

DATOS.- Pide el ingreso del orden del modelo a ser identificado.

DEFGRAF.- Permite el grafico de los ejes y vs. tiempo y u vs. tiempo.

ENTRADA.- Se ingresa el valor de entrada, el mismo que será el que alimenta a la planta, luego de pasar por el conversor D/A, siendo necesario el ingreso de las amplitudes del escalón y el ruido

GRAF.- Grafica los puntos que se obtienen tanto de "y" como de "u"

IGUA.- Permite trasladar el contenido de una matriz en otra, ejemplo: TEP() = TE()

```
DECLARE SUB defgraf ()
DECLARE SUB graf (I)
DECLARE SUB multi (pt(), xt(), alt(), MI, m1, m2)
DECLARE SUB pmate (m11(), m12, m13(), m5, m6)
DECLARE SUB IGUA (TEI(), TEF(), MP, MU)
DECLARE SUB NUEVO (MI, B, TE(), TE1(), ALF)
DECLARE SUB INIC (P(), AID(), MI)
DECLARE SUB DATOS (N1I, N2I)
DECLARE SUB PARAIN (B, TE(), TE1(), MI, ALF)
DECLARE SUB ENTRADA (NA, POR)
DECLARE SUB ALMACE (DAT(), TEE(), Y, U, N1I, N2I, DK, Maxdat)
DECLARE SUB PROM (MI, DK, DAT(), TEP())
DECLARE SUB LOTUS (DK, DAT(), MI)
DECLARE SUB printer (TEP(), N1I, N2I, nn!, bintv%)
COMMON SHARED sal!, va!, P(), X1(), AL1(), MI, AL2(), X(), TEE(), PPT(), AID()
COMMON SHARED AL3, E1, TE(), TE2(), e, P1(), P2(), PP(), B, TE1(), ALF, AI()
COMMON SHARED N1I, N2I, NA, POR, DAT(), DK, Y, U, Maxdat, TEP(), nn!

CLEAR

DIM P(4, 4), TE(4, 1), TE1(1, 4), X(1, 4), X1(4, 1), AID(4, 4), TEP(4, 1)
DIM AL(4, 1), AL1(4, 1), AL2(1, 1), E1(1, 1), AL3(1, 1), TEE(4, 4)
DIM TE2(4, 1), AI(4), DAT(601, 6), P1(4, 4), P2(4, 4), PP(4, 4), PPT(4, 4)

repita:

CALL SOFTINIT
CALL INIT
CLS
SCREEN 9
LINE (0, 0)-(639, 350), 2, BF
LINE (4, 4)-(635, 345), 0, BF
COLOR 2, 1
COLOR 15
tmin% = 100
LOCATE 3, 17
PRINT "MINIMOS CUADRADOS RECURSIVOS EN TIEMPO REAL"
COLOR 2
LOCATE 10, 10
PRINT "Tiempo de duración del algoritmo (ms) ="; tmin%
LOCATE 11, 10
PRINT "máximo estimado"
LINE (0, 57)-(639, 60), 2, BF
muestra:
LOCATE 12, 10
PRINT "Período de muestreo deseado (ms)     =";
INPUT bintv%
per% = INT(bintv% / tmin%)
IF per% = 0 THEN
LOCATE 14, 10
PRINT "Período de muestreo menor al tiempo de duración"
LOCATE 15, 10
PRINT "del algoritmo"
GOTO muestra
END IF
boutv% = bintv% / per%
bintv% = per% * boutv%
LOCATE 16, 10
```

```
PRINT "Período de muestreo ajustado (ms)     ="; bintv%
LOCATE 15, 10
PRINT "Retardo del dato de salida (ms)       ="; boutv%
bintv% = bintv%
boutv% = boutv%
T1 = bintv%

LOCATE 23, 25
COLOR 4
INPUT "Presione ENTER para continuar"; ad$
CLS


'          INGRESO DE DATOS NECESARIOS PARA EL ALGORITMO
CALL DATOS(N1I, N2I)
CLS

' INICIALIZACION DE VARIABLES
 MI = N1I + N2I
 Maxdat = 601
 FOR I = 1 TO MI
 TEP(I, 1) = 0
 NEXT I

'INICIALIZACION DE CONTADORES

 nn! = 0
 k% = 0
 DK = 1
 nm = 1

' INICIALIZACION DE PARAMETROS

CALL PARAIN(B, TE(), TE1(), MI, ALF)

' Inicialización de matrices identidad y theta

CALL INIC(P(), AID(), MI)


CLS

'SE CREA UNA ENTRADA CON RUIDO QUE SERIVIRA COMO REFERENCIA
CALL ENTRADA(NA, POR)

COLOR 15
LOCATE 18, 10
PRINT "Presione cualquier tecla para iniciar la identificación"

' INICIALIZACION DE VECTORES

FOR I = 2 TO 5
s(I) = 0
NEXT

FOR I = 2 TO 5
z(I) = 0
NEXT
```

```
'                SETEO DE LAS TAREAS DE BACKGROUND

dep! = 1!
CALL ANIN("entrada%", dep!, "ANLG0", bintv%, -1, "nt", "tarea1")
CALL ANIN("dataio%", 600!, "ANLG0", bintv%, 1, "nt", "")
CALL ARMAKE("salida%", dep!, -1, "ANOUT0")
CALL ANOUT("salida%", "ANOUT0", boutv%, -1, "nt", "tarea2")
FOR I! = 1 TO dep!
     CALL ARPUTVALF("salida%", I!, -1, "ANOUT0", 0!, 0)
NEXT I!
DO
   an$ = INKEY$
LOOP WHILE an$ = ""

WIDTH 80, 43
CLS
defgraf
dti = 1
'Inicialización Timer
REDIM dt(0 TO 7) AS DOUBLE, tim(0 TO 7) AS INTEGER
tim(0) = 1
CALL INTON(1, "mil")
CALL TIMERSTART(tim(), "nt", "timer0")

'Inicio del lazo de identificación
an$ = ""
WHILE an$ = ""
  arn$ = "entrada%"
CALL ARLASTP("entrada%", lp!)
CALL ARLASTP("salida%", lp!)


CALL ARGETVALF("entrada%", 1!, -1, "ANLG0", va!, 0)
Y = va!

U = NA + RND * POR
z(1) = Y
s(1) = U
sal! = U
COLOR 12
LOCATE 20, 10
PRINT "Identificación en Proceso"
LOCATE 22, 10
COLOR 6
PRINT "Número de iteraciones:"; nn!
LOCATE 40, 10
PRINT "Presione cualquier tecla para terminar el proceso y mostrar resultados"
COLOR 15

COLOR 14
LOCATE 20, 40
PRINT USING "Salida de la planta:   ##.### v."; va!
COLOR 9

FOR kk = 6 TO 2 STEP -1
s(kk) = s(kk - 1)
```

```
z(kk) = z(kk - 1)
NEXT kk


FOR kk = 1 TO N1I
X(1, kk) = z(kk + 2)
X1(kk, 1) = z(kk + 2)
NEXT kk

1 = 3
FOR kk = N1I + 1 TO MI
X(1, kk) = s(1 + 1)
X1(kk, 1) = s(1 + 1)
1 = 1 + 1
NEXT kk


'            COMIENZO DEL ALGORITMO

'CALCULO DE LA MATRIZ "L"
'------------------------

' MULTIPLICACION DE LA MATRIZ  P * X1 = AL1
CALL multi(P(), X1(), AL1(), MI, 1, MI)


' CALCULO DE LA MATRIZ AL2 = X * AL1
CALL multi(X(), AL1(), AL2(), 1, 1, MI)


AL3 = 1 / (AL2(1, 1) + 1)

' CALCULO DE LA MATRIZ AL = AL1 * AL3
CALL pmate(AL1(), AL3, AL(), MI, 1)

' CALCULO DE LA MATRIZ E1 = X * TE
CALL multi(X(), TE(), E1(), 1, 1, MI)

'CALCULO DEL ERROR
e = z(1) - E1(1, 1)


'CALCULO DE LA MATRIZ TE2 TE2 = e * AL()
CALL pmate(AL(), e, TE2(), MI, 1)

'CALCULO DE LA MATRIZ TEE
FOR II = 1 TO MI
   FOR JJ = 1 TO 1
      TEE(II, JJ) = TE(II, JJ) + TE2(II, JJ)
   NEXT JJ
NEXT II

' ALMACENAMIENTO DE LOS VALORES DE y, u, A(i), B(i) EN LA MATRIZ DAT()
CALL ALMACE(DAT(), TEE(), Y, U, N1I, N2I, DK, Maxdat)

' CALCULO DE LA NUEVA MATRIZ THETA  TE() = TEE()
CALL IGUA(TEE(), TE(), MI, 1)
```

```
'CALCULO DE LA MATRIZ P1 P1() = AL() * X()
CALL multi(AL(), X(), P1(), MI, MI, 1)

' CALCULO DE MATRIZ P2
FOR II = 1 TO MI
   FOR JJ = 1 TO MI
      P2(II, JJ) = AID(II, JJ) - P1(II, JJ)
   NEXT JJ
NEXT II

' CALCULO DE LA MATRIZ PP  PP() = P2() * P()
CALL multi(P2(), P(), PP(), MI, MI, MI)

' CALCULO DE LA MATRIZ PPT PPT() = B * PP()
CALL pmate(PP(), B, PPT(), MI, MI)

' CALCULO DE LA MATRIZ P PPT() = P()
CALL IGUA(PPT(), P(), MI, MI)


CALL ARPUTVALF("salida%", 1!, -1, "ANOUT0", sal!, 0)

COLOR 10
LOCATE 22, 40
PRINT USING "Entrada de la planta:  ##.### v."; sal!
COLOR 9

IF k% > 500 THEN
defgraf
k% = 0
END IF
graf (k%)
k% = k% + 1

nn! = nn! + 1

an$ = INKEY$
DO
CALL TIMERREAD(dt())
LOOP UNTIL (dt(0) / bintv%) > dt1
dt1 = INT(dt(0) / bintv% + 1)
WEND


CALL INTOFF
CALL ARWRITE("dataio%", "dataio.prn", 0, 4, bintv%, 1)
CALL ARDEL("entrada%")
CALL ARDEL("salida%")

' CALCULO DEL PROMEDIO DE LOS VALORES A(i), B(i)
CALL PROM(MI, DK, DAT(), TEP())

CLS
COLOR 10
LOCATE 5, 15
PRINT "RESULTADOS DE LA IDENTIFICACION EN TIEMPO REAL"
COLOR 4
```

```
LOCATE 8, 30
PRINT "M O D E L O"
PRINT
PRINT

COLOR 15
formato$ = "###.######"
FOR I = 1 TO N1I
PRINT SPACE$(20); "a("; I; ")="; USING formato$; -TEP(I, 1)
NEXT I
J2 = N1I + 1
FOR I = 1 TO N2I
PRINT SPACE$(20); "b("; I; ")="; USING formato$; TEP(J2, 1)
J2 = J2 + 1
NEXT I
COLOR 2
LOCATE 25, 1
PRINT SPACE$(20); "Número de iteraciones = "; nn!

' CREACION DE UN ARCHIVO .PRN, PARA PODER VER GRAFICOS DE Y, U, A(i), B(i)
' EN LOTUS

CALL LOTUS(DK, DAT(), MI)

' SE PREGUNTA SI SE REQUIERE IMPRIMIR RESULTADOS NUMERICOS
integ:
LOCATE 39, 5
COLOR 3
PRINT "Desea imprimir resultados numéricos (s/n)"
LOCATE 39, 50
INPUT P$
COLOR 9
IF P$ = "s" OR P$ = "S" THEN
CLS
SCREEN 9
WIDTH 80, 25
CALL printer(TEP(), N1I, N2I, nn!, bintv%)
ELSEIF P$ = "n" OR P$ = "N" THEN
GOTO preg
ELSE
GOTO integ
END IF


' SE PREGUNTA SI REQUIERE HACER OTRA SIMULACION

preg:
CLS
LOCATE 20, 5
COLOR 10
PRINT "Desea realizar una nueva identificación (s/n)"
LOCATE 20, 50
INPUT P$
COLOR 9
IF P$ = "s" OR P$ = "S" THEN
SCREEN 9
WIDTH 80, 25
GOTO repita
```

```
ELSEIF P$ = "n" OR P$ = "N" THEN
GOTO endmr
ELSE
GOTO preg
END IF

endmr:
WIDTH 80, 25

END
```

```
SUB ALMACE (DAT(), TEE(), Y, U, N1I, N2I, DK, Maxdat)
IF DK < Maxdat THEN
  DAT(DK, 1) = Y
  DAT(DK, 2) = U
     FOR I = 1 TO N1I
         DAT(DK, I + 2) = -TEE(I, 1)
     NEXT I
     FOR I = N1I + 1 TO MI
          DAT(DK, I + 2) = TEE(I, 1)
     NEXT I
DK = DK + 1
GOTO VOLV1
END IF
VOLV1:
END SUB

SUB DATOS (N1I, N2I)
vaya:
CLS
LINE (0, 0)-(639, 350), 2, BF
LINE (4, 4)-(635, 345), 0, BF
COLOR 15
COLOR 14
LOCATE 5, 20
PRINT "INGRESO DE PARAMETROS PARA IDENTIFICACION"
COLOR 15
LOCATE 10, 15
PRINT "ORDEN DEL SISTEMA"
LOCATE 11, 15
INPUT "Orden A(z) ="; N1I
LOCATE 12, 15
INPUT "Orden B(z) ="; N2I
PRINT
PRINT
COLOR 6
PRINT SPACE$(32); "Desea corregir datos ? (s/n)"
vaya2:
acb$ = INPUT$(1)
     IF acb$ = "S" OR acb$ = "s" THEN
     GOTO vaya
```

```
    ELSE
      IF acb$ = "N" OR acb$ = "n" THEN
       GOTO vaya1
           ELSE
           BEEP
           GOTO vaya2
      END IF
    END IF
vaya1:

END SUB

DEFINT I-N
DEFDBL A-H, O-Z
SUB defgraf
SCREEN 9
CLS
LINE (0, 0)-(639, 130), 1, BF
LINE (0, 0)-(639, 130), 15, B
LINE (43, 15)-(629, 15), 15, , &HF0F0
LINE (30, 65)-(629, 65), 17
LINE (43, 115)-(629, 115), 15, , &HF0F0

'Impresión nombre de los ejes

LOCATE 10, 72
COLOR 15
PRINT "Tiempo"
LINE (43, 6)-(43, 124), 7
LOCATE 2, 4
COLOR 14
PRINT "Y"

'Construcción gráfico U

LINE (0, 192)-(639, 310), 1, BF
LINE (0, 192)-(639, 310), 15, B
LINE (43, 200)-(629, 200), 15, , &HF0F0
LINE (30, 250)-(629, 250), 7
LINE (43, 300)-(629, 300), 15, , &HF0F0

'Impresión nombre de los ejes
LOCATE 33, 72
COLOR 15
PRINT "Tiempo"
LINE (43, 195)-(43, 305), 7
LOCATE 26, 4
COLOR 10
PRINT "U"

END SUB

SUB ENTRADA (NA, POR)
SCREEN 9
LOCATE 4, 30
PRINT "SEÑAL DE REFERENCIA"
COLOR 14, 1
LOCATE 8, 22
```

```
PRINT "Señal de entrada: ESCALON + RANDOM"
LOCATE 10, 22
PRINT "Amplitud del escalón =";
INPUT NA
LOCATE 12, 22
PRINT "Amplitud de ruido    =";
INPUT POR


END SUB

DEFSNG A-Z
SUB graf (I)
   IF va! > 0 THEN
   IF va! > 9.99 THEN
     PSET (I + 43, 15), 4
   ELSEIF va! < .04 THEN
     PSET (I + 43, 65), 4
   ELSE
      Y = -5 * va! + 65
      PSET (I + 43, Y), 14
   END IF
ELSE
   IF va! < -9.99 THEN
        PSET (I + 43, 115), 4
   ELSEIF va! > -.04 THEN
        PSET (I + 43, 65), 4
   ELSE
   Y = -(5 * va! - 65)
        PSET (I + 43, Y), 14
   END IF
END IF
IF sal! > 0 THEN
   IF sal! > 9.99 THEN
      PSET (I + 43, 200), 4
   ELSEIF sal! < .04 THEN
      PSET (I + 43, 250), 4
ELSE
  U = -5 * sal! + 250
      PSET (I + 43, U), 10
  END IF
ELSE
  IF sal! < -9.99 THEN
     PSET (I + 43, 300), 4
  ELSEIF sal! > -.04 THEN
    PSET (I + 43, 250), 4
  ELSE
     U = -(5 * sal! - 250)
     PSET (I + 43, sal!), 10
 END IF
END IF

END SUB

SUB IGUA (TEI(), TEF(), MP, MU)
FOR II = 1 TO MP
 FOR JJ = 1 TO MU
   TEF(II, JJ) = TEI(II, JJ)
```

```
 NEXT JJ
NEXT II
END SUB

SUB INIC (P(), AID(), MI)
FOR I = 1 TO MI
     FOR J = 1 TO MI
       IF I = J THEN
       P(I, J) = ALF
       AID(I, J) = 1
        ELSE
         P(I, J) = 0
         AID(I, J) = 0
       END IF
     NEXT J
NEXT I

END SUB

SUB LOTUS (DK, DAT(), MI)
OPEN "A:\TREAL.PRN" FOR OUTPUT AS #1
  FOR J = 1 TO DK - 1
  PRINT #1, J;
     FOR I = 1 TO MI + 2
     PRINT #1, USING "##.#####"; DAT(J, I);
     NEXT I
  PRINT #1, ""
  NEXT J
CLOSE #1
END SUB

SUB multi (pt(), xt(), alt(), MI, m1, m2)
FOR II = 1 TO MI
    FOR JJ = 1 TO m1
    alt(II, JJ) = 0
        FOR kk = 1 TO m2
           alt(II, JJ) = alt(II, JJ) + pt(II, kk) * xt(kk, JJ)
        NEXT kk
    NEXT JJ
NEXT II

END SUB

SUB NUEVO (MI, B, TE(), TE1(), ALF)
OUT2I:
CLS
SCREEN 9
COLOR 14, 1
COLOR 4
LOCATE 2, 20
PRINT "INICIALIZACION DE PARAMETROS"
COLOR 14
siga:
LOCATE 4, 21
INPUT "Factor de olvido = B"; B
     IF B > 1 THEN
     GOTO siga
     ELSE
```

```
    GOTO sigai
    END IF
sigai:

LOCATE 6, 21
PRINT "Matriz de Parámetros inciales THETA"
LOCATE 7, 1
FOR I = 1 TO MI
    PRINT SPACE$(22); "TE("; I; ")=";
    INPUT AI(I)
    TE(I, 1) = AI(I)
    TE1(1, I) = AI(I)
NEXT I
PRINT
PRINT
PRINT SPACE$(22); "Valor de ALFA =";
INPUT ALF
PRINT
PRINT
COLOR 6
PRINT SPACE$(30); "Desea corregir datos? (s/n)";
ovti:
acb$ = INPUT$(1)
IF acb$ = "s" OR acb$ = "S" THEN
GOTO OUT2I
ELSE
IF acb$ = "N" OR acb$ = "n" THEN
GOTO OUT3F
ELSE
BEEP
GOTO ovti
END IF
END IF
OUT3F:

END SUB

SUB PARAIN (B, TE(), TE1(), MI, ALF)
SCREEN 9
 COLOR 14, 1
 CLS
 B = .985
 formato$ = "#.###"
 LOCATE 3, 21
 COLOR 4
 PRINT "INICIALIZACION DE PARAMETROS"
 COLOR 14
 LOCATE 5, 1
 PRINT SPACE$(20); "Factor de olvido = B = "; USING formato$; B
FOR I = 1 TO MI
TE(I, 1) = 0
TE1(1, I) = 0
NEXT I
LOCATE 7, 21
PRINT "Matriz de Parámetros iniciales THETA"
LOCATE 8, 1
FOR I = 1 TO MI
    PRINT SPACE$(22); "TE("; I; ")="; TE(I, 1)
```

```
NEXT I
ALF = 10000
LOCATE 18, 21
PRINT "Valor de ALFA = "; ALF
COLOR 6
PRINT
PRINT
PRINT SPACE$(30); "Desea corregir datos ? (s/n)"
out1:
acb$ = INPUT$(1)
     IF acb$ = "S" OR acb$ = "s" THEN
     CALL NUEVO(MI, B, TE(), TE1(), ALF)
     ELSE
        IF acb$ = "N" OR acb$ = "n" THEN
        GOTO OUT2
        ELSE
        BEEP
        GOTO out1
        END IF
     END IF
OUT2:

END SUB

SUB pmate (m11(), m12, m13(), m5, m6)
FOR II = 1 TO m5
   FOR JJ = 1 TO m6
      m13(II, JJ) = m11(II, JJ) * m12
   NEXT JJ
NEXT II

END SUB

SUB printer (TEP(), N1I, N2I, nn!, bintv%)
CLS
LPRINT ""
LPRINT SPACE$(25); "RESULTADOS DE LA IDENTIFICACION EN TIEMPO REAL"
LPRINT ""
LPRINT ""
LPRINT ""
LPRINT SPACE$(40); "M O D E L O"
LPRINT
LPRINT
formato$ = "###.######"
FOR I = 1 TO N1I
LPRINT SPACE$(38); "a("; I; ")="; USING formato$; -TEP(I, 1)
NEXT I
J2 = N1I + 1
FOR I = 1 TO N2I
LPRINT SPACE$(38); "b("; I; ")="; USING formato$; TEP(J2, 1)
J2 = J2 + 1
NEXT I
LPRINT ""
LPRINT ""
LPRINT SPACE$(38); "Número de iteraciones = "; nn!
LPRINT ""
LPRINT ""
LPRINT SPACE$(38); "Período de muestreo = "; bintv%; " ms."
```

```
LPRINT ""
LPRINT ""
LPRINT SPACE$(38); "SEÑAL DE ENTRADA:"
LPRINT SPACE$(56); "Escalón = "; NA
LPRINT SPACE$(56); "Ruido   = "; POR

END SUB

SUB PROM (MI, DK, DAT(), TEP())
IF DK < 50 THEN
FOR II = 1 TO N1I
   TEP(II, 1) = -TEE(II, 1)
NEXT II
FOR II = N1I + 1 TO MI
   TEP(II, 1) = TEE(II, 1)
NEXT II
ELSE
FOR I = 1 TO MI
   FOR J = DK - 50 TO DK
       TEP(I, 1) = TEP(I, 1) + DAT(J, I + 2)
     NEXT J
   TEP(I, 1) = TEP(I, 1) / 50
NEXT I
END IF

END SUB
```

C.3. INSTRUMENTACION EN TIEMPO REAL (INSREAL).-

Este programa consta de tres módulos principales que son:

- Adquisición de datos
- Salida de datos
- Adquisición de datos + algoritmo + salida de datos

Adquisición de datos.- Realiza la adquisición de datos a través del canal ANALGØ, se lo hace en background, lo que permite adicionalmente trabajos en foreground como son gráficos, salida de datos a pantalla. La subrutina se llama **ENTRADA.**

Salida de datos.- Calcula una onda sinusoidal con un período que se ingresa manualmente. Luego se procede a sacar por el canal análogo ANLGØ, pudiendo setearse el período de salida de datos. Para la verficación de esta subrutina es necesario el uso de un osciloscopio. La subrutina se llama **SALIDA.**

Entrada + Algoritmo + Salida de datos.- Este módulo es la unión de los dos anteriores, es decir luego de adquirir el dato, se procede a multiplicarlo por dos, este es el algoritmo, subrutina **ALGORITMO.** Para proceder a sacarlo a través del canal análogo. Esta es la subrutina llamada **ALGO.**

Las principales subrutinas son:

DEFGRAF.- Permite realizar los ejes "y vs. tiempo", es utilizada para la subrutina de entrada de datos.

DEFGRAF1.- Permite realizar los ejes "y vs. tiempo" y "u vs. tiempo", es utilizada en la subrutina ALGO.

GRAF.- Permite el graficar punto a punto el valor de la sinusoide obtenido con la subrutina salida de datos, SALIDA.

GRAF1.- Permite graficar cada punto de y que es obtenida a través de la subrutina de entrada de datos, ENTRADA.

GRAF2.- Permite graficar punto a punto los valores obtenidos tanto de la entrada u como de la salida y de la subrutina ALGO.

```
finish:
END


SUB ALGO (i)
  'ENTRADA SALIDA DE DATOS CON ALGORITMO DE CONTROL
  DEFINT I-N
  DEFDBL A-H, O-Z
  LINE (0, 0)-(639, 350), 2, BF
  LINE (4, 4)-(635, 345), 0, BF
  COLOR 2, 1
  COLOR 15
  CALL SOFTINIT
  CALL INIT
  tmin% = 90
  LOCATE 3, 15
  PRINT "ADQUISICION DE DATOS + ALGORITMO + SALIDA DE DATOS"
  COLOR 2
  LOCATE 8, 10
  PRINT "Tiempo de duracion del algoritmo (ms) = "; tmin%
  LOCATE 9, 10
  PRINT "maximo estimado"
  LINE (0, 80)-(639, 83), 2, BF
muestra:
  LOCATE 11, 10
  PRINT "Periodo de muestreo deseado (ms)= ";
  INPUT bintv%
  per% = INT(bintv% / tmin%)
  IF per% = 0 THEN
  LOCATE 13, 10
  PRINT "Periodo de muestreo menor al tiempo de duración"
  LOCATE 14, 10
  PRINT "del algoritmo"
  GOTO muestra
  END IF
  boutv% = bintv% / per%    'optimización del tiempo de salida'
  bintv% = per% * boutv%    'ajuste del tiempo de muestreo'
  LOCATE 16, 10
  PRINT "Periodo de muestreo ajustado (ms)="; bintv%
  LOCATE 18, 10
  PRINT "Retardo de dato de salida    (ms)="; boutv%
  bintv% = bintv%
  boutv% = boutv%

  '           SETEO DE LAS TAREAS DE BACKGROUND'
  dep! = 1
  CALL ANIN("entrada%", dep!, "ANLG0", bintv%, -1, "nt", "tarea1")
  CALL ANIN("dataio%", 1000!, "ANLG0", bintv%, -1, "nt", "")

  CALL ARMAKE("salida%", dep!, -1, "ANOUT0")

  CALL ANOUT("salida%", "ANOUT0", boutv%, -1, "nt", "tarea2")

  FOR i! = 1 TO dep!
       CALL ARPUTVALF("salida%", i!, -1, "ANOUT0", 0!, 0)
  NEXT i!

  'Lazo creado con el afan de tener todo el arreglo referenciado a cero
```

```
'              CALCULOS PRELIMINARES AL ALGORITMO
'Se debe ingresar los datos, parametros, etc., y realizar todos los calculos
'que no van dentro del lazo de control
LOCATE 21, 10
PRINT "Presione cualquier tecla para iniciar el control"
k% = 0
COLOR 2
DO
          an$ = INKEY$
 LOOP WHILE an$ = ""
SCREEN 9
WIDTH 80, 43
CLS
defgraf1

dt1 = 1
DIM dt(0 TO 7) AS DOUBLE, tim(0 TO 7) AS INTEGER
tim(0) = 1
CALL INTON(1, "mil")
CALL TIMERSTART(tim(), "nt", "timer0")

'Comando que resetea el timer 0, en este caso y lo habilita para iniciar
'su cuenta, no espera disparo

valor! = 0
an$ = ""
WHILE an$ = ""
      arn$ = "entrada%"

      CALL ARLASTP("entrada%", lp!)
      CALL ARLASTP("salida%", lp!)
      CALL ARGETVALF("entrada%", 1!, -1, "ANLG0", valor!, 0)

COLOR 2
LOCATE 20, 10
COLOR 12
PRINT "Tiempo Real"
LOCATE 22, 10
COLOR 11
PRINT "Control en Proceso"
LOCATE 41, 10
COLOR 15
PRINT "Presione cualquier tecla para terminar el proceso"
             LOCATE 20, 36
             COLOR 14
             PRINT USING "Entrada:##.###v."; valor!

'!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
'               ALGORITMO DE CONTROL COMO SUBRUTINA
'!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
ALGORITMO


CALL ARPUTVALF("salida%", 1!, -1, "ANOUT0", sal!, 0)
LOCATE 22, 36
COLOR 10
PRINT USING "Control:##.###v."; sal!
IF k% < 500 THEN      'Reseteo de pantalla grafica
```

```
  defgraf1
  k% = 0
  END IF
  graf2 (k%)
  k% = k% + 1
  an$ = INKEY$
     DO

     CALL TIMERREAD(dt())
     LOOP UNTIL dt(0) / bintv% > dt1
     dt1 = INT(dt(0) / bintv% + 1)

'Sirve para asegurar que se ocupara todo el tiempo restante de la interrupcion
'recibir otro dato
  WEND

  CALL INTOFF
  CALL ARWRITE("dataio%", "algor.prn", 0, 4, bintv%, 1)
  CALL ARDEL("entrada%")
  CALL ARDEL("salida%")
  WIDTH 80, 25

END SUB

DEFINT I-N
DEFDBL A-H, O-Z
SUB ALGORITMO
sal! = valor! * 2
IF sal! > 10 THEN
sal! = 10
ELSEIF valor! > 10 THEN
valor! = 10
ELSEIF sal! < 0 THEN
sal! = 0
END IF
END SUB

SUB defgraf
LINE (0, 0)-(639, 130), 1, BF
LINE (0, 0)-(639, 130), 15, B
LINE (43, 15)-(629, 15), 15, , &HF0F0
LINE (30, 65)-(629, 65), 17
LINE (43, 115)-(629, 115), 15, , &HF0F0
LOCATE 6, 72
COLOR 15
PRINT "Tiempo"
LINE (43, 6)-(43, 124), 7
LOCATE 2, 4
COLOR 14
PRINT "y"

END SUB

DEFINT I-N
DEFDBL A-H, O-Z
  SUB defgraf1
  CLS
  'PARA EL CANAL DE ENTRADA
```

```
COLOR 3
pregu:
LOCATE 7, 3
PRINT "indicaciones:"
LOCATE 8, 3
PRINT "              ciento de microsegundos: hmic"
LOCATE 9, 3
PRINT "              milisegundos:            mil"
LOCATE 10, 3
PRINT "              segundos:                sec"
LOCATE 11, 3
PRINT "              minutos:                 min"
LOCATE 5, 3
INPUT "Periodo de muestreo (UNIDADES DE TIEMPO) = "; tu$
IF tu$ = "hmic" THEN
GOTO siga
ELSEIF tu$ = "mil" THEN
GOTO siga
ELSEIF tu$ = "sec" THEN
GOTO siga
ELSEIF tu$ = "min" THEN
GOTO siga
END IF
GOTO pregu
siga:
LOCATE 13, 3
PRINT "Periodo de muestreo ("; tu$;
INPUT ") = "; binty%
pregui:
LOCATE 15, 3
INPUT "Adquisicion Definida (d) o Indefinida (i)"; a$
IF a$ = "d" THEN
LOCATE 17, 3
INPUT "Cuántos datos desea adquirir"; d!
GOTO sigai
ELSEIF a$ = "i" THEN
GOTO sigai
END IF
GOTO pregui
sigai:
CLS
LOCATE 25, 2
COLOR 10
PRINT "Presione cualquier tecla para terminar la adquisición"
LOCATE 12, 15
COLOR 4
PRINT "MEDIDAS REALIZADAS POR EL CANAL ANALOGO 0 (ANL60)"
defgraf


'     ================================================


CALL SOFTINIT
CALL INIT
ion$ = "ANL60"
deo! = 10
```

```
l! = -dep!
IF a$ = "i" THEN
CALL ANIN("entrada%", dep!, "ANL60", bintv%, -1, "nt", "tarea1")
d! = 10000

ELSEIF a$ = "d" THEN
CALL ANIN("entrada%", dep!, "ANL60", bintv%, -1, "nt", "tarea1")
END IF

CALL ANIN("aninba%", 1000!, "ANL60", bintv%, 1, "nt", "tarea1")
CALL INTON(1, tu$)
an$ = ""
WHILE an$ = ""
CALL ARLASTP("entrada%", lp!)
    IF lp! <> plp! THEN
     CALL ARGETVALF("entrada%", lp!, -1, "ANL60", va!, 0)
        IF lp! = 1 AND lip = 1 THEN
        l! = l! + dep!
        COLOR 14
        lip = 0
        ELSEIF lp! = 1 AND lip = 0 THEN
        l! = l! + dep!
        COLOR 15
        lip = 1
        END IF
            IF lp! + l! > d! THEN GOTO finau
            LOCATE 13 + lp!, 25
            PRINT "medida#"; lp! + l!; "=   ";
            LOCATE 13 + lp!, 40
            PRINT USING "##.###  "; va!

IF k% > 580 THEN
defgraf
k% = 0
END IF
grafi (k%)
k% = k% + 1

plp! = lp!
END IF

an$ = INKEY$

finau:
     WEND
     CALL INTOFF
     CALL ARWRITE("aninba%", "inback.PRN", 0, 4, bintv%, 1)

END SUB


DEFSNG A-Z
SUB graf (i)
IF d! > 0 THEN
     IF d! > 9.99 THEN
          PSET (i + 180, 150), 4
     ELSEIF d! < .04 THEN
          PSET (i + 200, 200), 4
```

```
     ELSE
          y = -5 * d! + 200
          PSET (i + 180, y), 14
     END IF
ELSE
     IF d! < -9.99 THEN
     PSET (i + 180, 250), 4
          ELSEIF d! > -.04 THEN
          PSET (i + 180, 200), 4
     ELSE
          y = -(5 * d! - 200)
          PSET (i + 180, y), 14
     END IF

END IF
END SUB

SUB graf1 (i)
DEFSNG A-Z
     IF va! > 0 THEN
          IF va! > 9.99 THEN
             PSET (i + 43, 15), 4
          ELSEIF va! < .04 THEN
          PSET (i + 43, 65], 4
     ELSE
     y = -5 * va! + 65
     PSET (i + 43, y), 14
     END IF
     ELSE
          IF va! < -9.99 THEN
          PSET (i + 43, 115), 4
          ELSEIF va! > -.04 THEN
          PSET (i + 43, 65), 4
          ELSE
          y = -(5 * va! - 65)
          PSET (i + 43, y), 14
     END IF
     END IF



END SUB

DEFSNG A-Z
SUB graf2 (i)

IF valor! > 0 THEN
   IF valor! > 9.99 THEN
     PSET (i + 40, 15), 4
     ELSEIF valor! < .04 THEN
       PSET (i + 40, 65), 4
     ELSE
       y = -5 * valor! + 65
       PSET (i + 40, y), 14
   END IF

ELSE
```

```
    IF valor! < -9.99 THEN
       PSET (i + 40, 115), 4
     ELSEIF valor! > -.04 THEN
     PSET (i + 40, 65), 4
       ELSE
       y = -(5 * valor! - 65)
       PSET (i + 40, y), 14
     END IF
END IF

     IF sal! > 0 THEN
         IF sal! > 9.99 THEN
         PSET (i + 40, 200), 4
           ELSEIF sal! < .04 THEN
           PSET (i + 40, 250), 4
         ELSE
         u = -5 * sal! + 250
         PSET (i + 40, u), 10
         END IF
     ELSE
         IF sal! < -9.99 THEN
         PSET (i + 40, 300), 4
            ELSEIF sal! > -.04 THEN
            PSET (i + 40, 250), 4
         ELSE
       . u = -(5 * sal! - 250)
         PSET (i + 40, u), 10
     END IF
END IF

END SUB

SUB SALIDA (d!, i, k%)
'SALIDA DE DATOS CON GRAFICOS
CLS
k% = 0
COLOR 12
LOCATE 2, 20
PRINT "SALIDA DE DATOS EN MODO BACKGROUND"
CALL SOFTINIT
CALL INIT
COLOR 2
LOCATE 4, 5
INPUT "Frecuencia de la sinusoide"; f%
LOCATE 6, 5
INPUT "Periodo de muestreo (ms)="; boutv%
COLOR 15
LOCATE 22, 16
PRINT "Salida de datos de una sinuoside de"
LOCATE 23, 23
PRINT "frecuencia ="; f%; " Hz"
COLOR 2
LOCATE 25, 11
PRINT "Presione cualquier tecla para terminar la salida"
CALL ARMAKE("seno%", 100!, -1, "ANOUT0")

FOR dep! = 1! TO 100!
d! = 10! * (SIN((dep! - 1) * 6.28 / 100! * f%))
```

```
DECLARE SUB pid (r, valor!, sal!)
DECLARE SUB redes (r, in!, sal!)
DECLARE SUB defgraf ()
DECLARE SUB algoritmo ()
DECLARE SUB graf (i)
DECLARE SUB grafi (i)
DIM y(1500), u(1500), e(1500), A(10), B(10), x(1, 50), c(4), v(4), n(1, 20)



COMMON SHARED valor!, sal!, c(), v(), r, bintv%, i, kp, ki, kd, in!, A(), B()
COMMON SHARED n(), y(), e(), u(), x()

CALL SOFTINIT
CALL INIT


' PROGRAMA PRINCIPAL DE CONTROL EN TIEMPO REAL
SCREEN 9
LINE (0, 0)-(660, 660), 1, BF
LOCATE 10, 20
COLOR 12
PRINT "CONTROL EN TIEMPO REAL MEDIANTE EL SISTEMA"
LOCATE 12, 22
PRINT "DE ADQUISICION DE DATOS KEITHLEY 500 A"
LOCATE 24, 47
COLOR 15
INPUT "Presione ENTER para continuar", ad$

' MENU DE PREGUNTAS
repit:
CLS
LOCATE 10, 1
COLOR 15
PRINT SPACE$(25), "1) PID"
PRINT SPACE$(25), "2) REDES"
PRINT SPACE$(25), "3) TERMINAR"
LOCATE 14, 29
INPUT opcion
   IF opcion < 1 OR opcion > 3 THEN
     GOTO repit
   END IF

SELECT CASE opcion

CASE 1
CLS
CALL pid(r, valor!, sal!)
GOTO repit

CASE 2
CLS
CALL redes(r, in!, sal!)
GOTO repit

CASE 3
GOTO finish
END SELECT
```

```
finish:
END
```

```
DEFSNG A-Z
SUB algoritmo
y = valor!
e = r - y
v(1) = sal!
FOR L = 4 TO 3 STEP -1
v(L) = v(L - 1)
NEXT
v(2) = e
u = 0
FOR L = 1 TO 4
u = u + c(L) * v(L)
NEXT
sal! = u
IF sal! > 10 THEN
sal! = 10
LOCATE 41, 55
COLOR 15
PRINT "Canal salida sat."
LOCATE 41, 55
COLOR 4
PRINT "Canal salida sat."
 ELSEIF sal! < -10 THEN
LOCATE 41, 55
COLOR 4
PRINT "Canal salida sat."
ELSE
LOCATE 41, 55
PRINT "                    "
END IF
COLOR 9

END SUB

DEFINT I-N
DEFDBL A-H, O-Z
SUB defgraf
```

```
CLS
' PARA EL CANAL DE ENTRADA
LINE (0, 0)-(639, 130), 1, BF
LINE (0, 0)-(639, 130), 15, B
LINE (40, 15)-(629, 15), 15, , &HF0F0
LINE (30, 65)-(629, 65), 17
LINE (40, 115)-(629, 115), 15, , &HF0F0

LOCATE 10, 74
COLOR 15
PRINT "Tiempo"
LINE (40, 6)-(40, 124), 7
COLOR 14
LOCATE 2, 4
PRINT "y"
COLOR 2

      'PARA EL CANAL DE SALIDA
LINE (0, 192)-(639, 310), 1, BF
LINE (0, 192)-(639, 310), 15, B
LINE (40, 200)-(629, 200), 15, , &HF0F0
LINE (30, 250)-(629, 250), 7
LINE (40, 300)-(620, 300), 15, , &HF0F0
LOCATE 33, 74
COLOR 15
PRINT "Tiempo"
LINE (40, 195)-(40, 305), 7
COLOR 10
LOCATE 26, 4
PRINT "u"
COLOR 13
LOCATE 4, 4
PRINT "r"




END SUB

DEFSNG A-Z
SUB graf (i)

IF valor! > 0 THEN
       IF valor! > 9.99 THEN
         PSET (i + 40, 15), 4

       ELSEIF valor! < .04 THEN
         PSET (i + 40, 65), 4

       ELSE
         y = -5 * valor! + 65
         PSET (i + 40, y), 14
       END IF
ELSE
       IF valor! < -9.99 THEN
```

```
            PSET (i + 40, 115), 4
        ELSEIF in! > -.04 THEN
            PSET (i + 40, 65), 4
        ELSE
        y = -(5 * in! - 65)
            PSET (i + 40, y), 14
        END IF
END IF

        IF r > .04 THEN
        w = -5 * r + 65
        PSET (i + 40, w), 13
        ELSE
        w = -(5 * r - 65)
        PSET (i + 40, w), 13
        END IF

IF sal! > 0 THEN
        IF sal! > 9.99 THEN
            PSET (i + 40, 200), 4
        ELSEIF sal! < .04 THEN
            PSET (i + 40, 250), 4
        ELSE
        u = -5 * sal! + 250
        PSET (i + 40, u), 10
        END IF
ELSE
        IF sal! < -9.99 THEN
            PSET (i + 40, 300), 4
        ELSEIF sal! > -.04 THEN
            PSET (i + 40, 250), 4
        ELSE
        u = -(5 * sal! - 250)
            PSET (i + 40, u), 10
        END IF
END IF



END SUB



SUB pid (r, valor!, sal!)
' P. I. D

REPETIR:
CLS
SCREEN 9
LINE (0, 0)-(639, 350), 2, BF
LINE (4, 4)-(635, 345), 0, BF
COLOR 2, 1
COLOR 15
CALL SOFTINIT
CALL INIT
tmin% = 90
LOCATE 3, 16
PRINT "CONTROLADOR P.I.D UNIVARIABLE EN TIEMPO REAL"
```

```
COLOR 2
LOCATE 10, 10
PRINT "Tiempo de duración del algoritmo (ms)="; tmin%
LOCATE 11, 10
PRINT "máximo estimado"
LINE (0, 80)-(639, 83), 2, BF
muestra:
LOCATE 12, 10
PRINT "Período de muestreo deseado (ms)    =";
INPUT bintv%
per% = INT(bintv% / tmin%)
IF per% = 0 THEN
LOCATE 9, 10
PRINT "Período de muestreo menor al tiempo de duración"
LOCATE 15, 10
PRINT "del algoritmo"
GOTO muestra
END IF
boutv% = bintv% / per%
bintv% = per% * boutv%
LOCATE 16, 10
PRINT "Período de muestreo ajustado (ms)    ="; bintv%
LOCATE 17, 10
PRINT "Retardo del dato de salida (ms)      ="; boutv%
bintv% = bintv%
boutv% = boutv%
k% = 0
LOCATE 23, 25
COLOR 4
INPUT "Presione ENTER para continuar"; ad$
CLS

'INGRESO DE DATOS NECESARIOS PARA EL ALGORITMO
iria1:
CLS
LINE (0, 0)-(639, 350), 2, BF
LINE (4, 4)-(635, 345), 0, BF
COLOR 12
LOCATE 2, 10
PRINT "Ingreso de parametros del controlador P.I.D univariable"
COLOR 15
LOCATE 5, 15
INPUT "Kp         ="; tkp
LOCATE 6, 15
INPUT "Ki         ="; tki
LOCATE 7, 15
INPUT "Kd         ="; tkd
LOCATE 8, 15
INPUT "Referencia ="; r
COLOR 7
LOCATE 10, 1
PRINT SPACE$(25); "Desea corregir datos? (s/n)";
iria:
ACB$ = INPUT$(1)
     IF ACB$ = "s" OR ACB$ = "S" THEN
     GOTO iria1
     ELSE
          IF ACB$ = "N" OR ACB$ = "n" THEN
```

```
            GOTO iria2
            ELSE
            BEEP
            GOTO iria
            END IF
        END IF
iria2:

t = bintv% / 1000
c(1) = 1
c(2) = tkp + tki * t / 2 + tkd / t
c(3) = tki * t / 2 - tkp - 2 * tkd / t
c(4) = tkd / t
FOR L = 1 TO 4
v(L) = 0
NEXT


' SETEO DE LAS TAREAS DE BACKGROUND

dep! = 1
CALL ANIN("entrada%", dep!, "ANLG0", bintv%, -1, "nt", "tarea1")
CALL ANIN("acumule%", 500!, "ANLG0", bintv%, 1, "nt", "")
CALL ARMAKE("salida%", dep!, -1, "ANOUT0")
CALL ARMAKE("pid%", 500!, -1, "ANOUT0")
CALL ANOUT("salida%", "ANOUT0", boutv%, -1, "nt", "tarea2")
FOR i! = 1 TO dep!
CALL ARPUTVALF("salida%", i!, -1, "ANOUT0", 0!, 0)
NEXT i!

COLOR 13
LOCATE 20, 15
PRINT "Presione cualquier tecla para iniciar el control"
COLOR 2
DO
        an$ = INKEY$
  LOOP WHILE an$ = ""

SCREEN 9
WIDTH 80, 43
CLS
defgraf
COLOR 9
LOCATE 20, 10
COLOR 12
PRINT "Control en Proceso"
LOCATE 40, 10
PRINT "Presione cualquier tecla para terminar el proceso"
COLOR 7
dt1 = 1
REDIM dt2(0 TO 7) AS DOUBLE, tim1(0 TO 7) AS INTEGER
tim1(0) = 1
CALL INTON(1, "mil")
CALL TIMERSTART(tim1(), "nt", "timer0")
valor! = 0
n! = 0
an$ = ""
WHILE an$ = ""
```

```
        arn$ = "entrada%"
        CALL ARLASTP("entrada%", lp!)
        CALL ARLASTP("salida%", lp!)
        CALL ARGETVALF("entrada%", 1!, -1, "ANL60", valor!, 0)
COLOR 14
LOCATE 20, 40
PRINT USING "Salida de la planta: ##.### v."; valor!
COLOR 9
n! = n! + 1
algoritmo

        LOCATE 22, 10
        COLOR 15
        PRINT "Número de iteraciones:"; n!
        CALL ARPUTVALF("salida%", 1!, -1, "ANOUT0", sal!, 0)
        COLOR 10
        LOCATE 22, 40
        PRINT USING "Ley de control : ##.### v."; sal!
        COLOR 9
        IF k% > 500 THEN
        defgraf
        k% = 0
        END IF
        graf (k%)
        k% = k% + i
        an$ = INKEY$

DO
        CALL TIMERREAD(dt2())
        LOOP UNTIL dt2(0) / bintv% > dt1
        dt1 = INT(dt2(0) / bintv% + 1)
WEND
        CALL INTOFF
        CALL ARWRITE("acumule%", "pid1.prn", 0, 4, bintv%, 1)
        CALL ARWRITE("pid%", "pid2.prn", 0, 4, boutv%, 1)

        CALL ARDEL("entrada%")
        CALL ARDEL("salida%")
pregp:
        LOCATE 41, 10
        COLOR 2
        PRINT "Desea realizar un nuevo control (s/n) ="
        LOCATE 41, 47
        INPUT p$
        COLOR 9
        IF p$ = "s" OR p$ = "S" THEN
        SCREEN 0
        WIDTH 80, 25
        GOTO REPETIR
        ELSEIF p$ = "n" OR p$ = "N" THEN
        GOTO endpid
        ELSE
        GOTO pregp
        END IF
endpid:
WIDTH 80, 25

END SUB
```

```
SUB redes (r, in!, sal!)

REPITAS:
SCREEN 9
CLS
LINE (0, 0)-(639, 359), 2, BF
LINE (4, 4)-(635, 345), 0, BF
COLOR 2, 1
COLOR 15
tmin% = 90
n! = 0
k% = 0
LOCATE 3, 35
PRINT "REDES"
COLOR 2
LOCATE 10, 10
PRINT "Tiempo de duración del algoritmo (ms) ="; tmin%
LOCATE 11, 10
PRINT "máximo estimado"
LINE (0, 80)-(639, 83), 2, BF
muestras:
LOCATE 12, 10
PRINT "Período de muestreo deseado (ms)     =";
INPUT bintv%
per% = INT(bintv% / tmin%)
IF per% = 0 THEN
LOCATE 14, 10
PRINT "Período de muestreo menor al tiempo de duración"
LOCATE 15, 10
PRINT "del algoritmo"
GOTO muestras
END IF
boutv% = bintv% / per%
bintv% = per% * boutv%
LOCATE 16, 10
PRINT "Período de muestreo ajustado (ms)     ="; bintv%
LOCATE 17, 10
PRINT "Retardo del dato de salida (ms)       ="; boutv%
boutv% = boutv%
bintv% = bintv%
LOCATE 23, 25
COLOR 4
INPUT "Presione ENTER para continuar"; ad$
CLS

CALL SOFTINIT
CALL INIT


salti1:
'           INGRESO DE DATOS NECESARIOS PARA EL ALGORITMO
vaya:
CLS
LINE (0, 0)-(639, 350), 2, BF
LINE (4, 4)-(635, 345), 0, BF
COLOR 14
```

```
LOCATE 2, 18
PRINT "INGRESO DE PARAMETROS PARA EL CONTROLADOR"
LOCATE 4, 25
PRINT "MEDIANTE LA TECNICA DE REDES"
LOCATE 8, 15
COLOR 15
INPUT "Orden de la ecuación de diferencias = M= "; M
LOCATE 10, 15
INPUT "Ingrese la Referencia = r= "; r
PRINT
PRINT
PRINT
PRINT
PRINT SPACE$(30); "Desea corregir datos ? (s/n)"
sal11:
ACB$ = INPUT$(1)
     IF ACB$ = "S" OR ACB$ = "s" THEN
     GOTO salt11
     ELSE
          IF ACB$ = "N" OR ACB$ = "n" THEN
          GOTO sal31
          ELSE
          BEEP
          GOTO sal11
          END IF
     END IF
sal31:
SALT1:
CLS
COLOR 2
LOCATE 4, 25
PRINT "INGRESO DE PARAMETROS DE LA RED"
COLOR 15
LOCATE 7, 1
FOR i = 1 TO M
PRINT SPACE$(20); "Coeficiente a("; i; ")=";
INPUT A(i)
NEXT i
FOR i = 0 TO M
PRINT SPACE$(20); "Coeficiente b("; i; ")=";
INPUT B(i)
NEXT i
PRINT
PRINT
PRINT SPACE$(30); "Desea corregir datos ? (s/n)"
sal1:
ACB$ = INPUT$(1)
     IF ACB$ = "S" OR ACB$ = "s" THEN
     GOTO SALT1
     ELSE
          IF ACB$ = "N" OR ACB$ = "n" THEN
          GOTO sal3
          ELSE
          BEEP
          GOTO sal1
          END IF
     END IF
sal3:
```

```
M1 = 2 * M
M2 = M + 1
FOR i = 1 TO M
n(1, i) = -A(i)
NEXT i
FOR i = 0 TO M
n(1, i + M2) = B(i)
NEXT i

j = 0
in! = 0

'    SETEO DE LAS TAREAS DE BACKGROUND
dep! = 1!
CALL ANIN("entrada%", dep!, "ANLG0", bintv%, -1, "nt", "tarea1")
CALL ANIN("datas%", 500!, "ANLG0", bintv%, 1, "nt", "")
CALL ARMAKE("salida%", dep!, -1, "ANOUT0")
CALL ARMAKE("red%", 500!, -1, "ANOUT0")
CALL ANOUT("salida%", "ANOUT0", boutv%, -1, "nt", "tarea2")
FOR i! = 1 TO dep!
CALL ARPUTVALF("salida%", i!, -1, "ANOUT0", 0!, 0)
NEXT i!

COLOR 2
LOCATE 22, 10
PRINT "Presione cualquier tecla para iniciar el control"
COLOR 6
DO
an$ = INKEY$
LOOP WHILE an$ = ""
SCREEN 9
WIDTH 80, 43
CLS
defgraf

COLOR 7
dt1 = 1
REDIM dt(0 TO 7) AS DOUBLE, tim(0 TO 7) AS INTEGER
tim(0) = 1
CALL INTON(1, "mil")
CALL TIMERSTART(tim(), "nt", "timer0")
an$ = ""
WHILE an$ = ""
CALL ARLASTP("entrada%", lp!)
CALL ARLASTP("salida%", lp!)
CALL ARPUTVALF("salida%", 1!, -1, "ANOUT0", sal!, 0)
CALL ARGETVALF("entrada%", 1!, -1, "ANLG0", in!, 0)

y(j) = in!
COLOR 12
LOCATE 20, 10
PRINT "Control en Proceso"
LOCATE 22, 10
COLOR 6
PRINT "Número de iteraciones:"; n!

LOCATE 40, 10
PRINT "Presione cualquier tecla para terminar el proceso"
```

```
COLOR 14
LOCATE 20, 40
PRINT USING "Salida de la planta: ##.### v."; in!


'==================================
'      ALGORITMO DE REDES
'==================================

w1 = 0
w2 = 0
w3 = 0
e(j) = r - y(j)

FOR i = 0 TO M
IF (j - i) < 0 THEN
w1 = w1
ELSE
w1 = w1 + B(i) * e(j - i)
END IF
NEXT i
FOR i = 1 TO M
IF (j - i) < 0 THEN
w2 = w2
ELSE
w2 = w2 + A(i) * u(j - i)
END IF
NEXT i
w3 = w1 - w2
u(j) = w3
jk = j
FOR i = 1 TO M
IF jk - 1 < 0 THEN
x(1, i) = 0
ELSE
x(1, i) = u(jk - 1)
END IF
jk = jk - 1
NEXT i
ii = M + 1
kj = j
FOR i = 0 TO M
IF kj < 0 THEN
x(1, ii) = 0
ELSE
x(1, ii) = e(kj)
END IF
kj = kj - 1
ii = ii + 1
NEXT i
sal! = u(j)
COLOR 10
LOCATE 22, 40
PRINT USING "Ley de control: ##.### v."; sal!
COLOR 9
IF k% > 500 THEN
deforaf
```

```
k% = 0
END IF
grafi (k%)
k% = k% + 1

j = j + 1
n! = n! + 1


an$ = INKEY$

DO
CALL TIMERREAD(dt())
LOOP UNTIL (dt(0) / bintv%) > dt1
dt1 = INT(dt(0) / bintv% + 1)
WEND
CALL INTOFF
CALL ARWRITE("datas%", "red1.prn", 0, 4, bintv%, 1)
CALL ARWRITE("red%", "red2.prn", 0, 4, boutv%, 1)
CALL ARDEL("entrada%")
CALL ARDEL("salida%")

finau:
LOCATE 41, 10
COLOR 2
PRINT "Desea realizar un nuevo control (s/n) = "
LOCATE 41, 47
INPUT p$
COLOR 9
IF p$ = "s" OR p$ = "S" THEN
SCREEN 0
WIDTH 80, 25
GOTO REPITAS
ELSEIF p$ = "n" OR p$ = "N" THEN
GOTO endred
ELSE
GOTO finau
END IF
endred:
WIDTH 80, 25
END SUB
```

ANEXO D

MANUAL DE USO

ANEXO D

MANUAL DE USO

Los programas desarrollados básicamente se encuentran en cuatro módulos principales:

1.- **IDENSIMU.EXE** ejecutable, realiza la identificación en simulación.

2.- **IDENREAL.EXE** ejecutable, realiza la identificación en tiempo real.

Los dos programas mencionados son realizados utilizando el algoritmo de mínimos cuadrados recursivos.

3.- **INSREAL.EXE** ejecutable, realiza instrumentación en tiempo real. Tiene un menú:

1) ADQUISICION DE DATOS
2) SALIDA DE DATOS
3) ADQUISICION + ALGORITMO + SALIDA
4) TERMINAR

4.- **CONREAL.EXE** ejecutable, realiza control en tiempo real. Presenta el menú:

1) PID

2) REDES

3) TERMINAR

- Conjuntamente se tiene los archivos .BAS
- Además todos los comandos que se necesitan del Quick 500 (tiempo real) presentan archivos .BAT, para que la ejecución de los programas sean ejecutados solo digitando su nombre.
- Todos los programas en tiempo real (utilización estación KEITHLEY 500A), tiene la opción de crear archivos .PRN.

Requerimientos:

Para poder ejecutarlos, compilarlos o editarlos se requiere:

- Microsoft Quick Basic 4.5
- Tarjeta de gráficos VGA colores
- Librerías y rutinas del Quick 500 software del Keithley 500A
- Equipo de adquisición de datos Keithley 500A
- Todos los archivos que conforman el sistema deberán estar en un diskette, para este caso en particular se los ha guardado dentro del directorio A:\EXE>. El mismo debe ser autoarrancable, y tener cargados los comandos necesarios para que el computador reconozca al sistema de adquisición y

ANEXO E

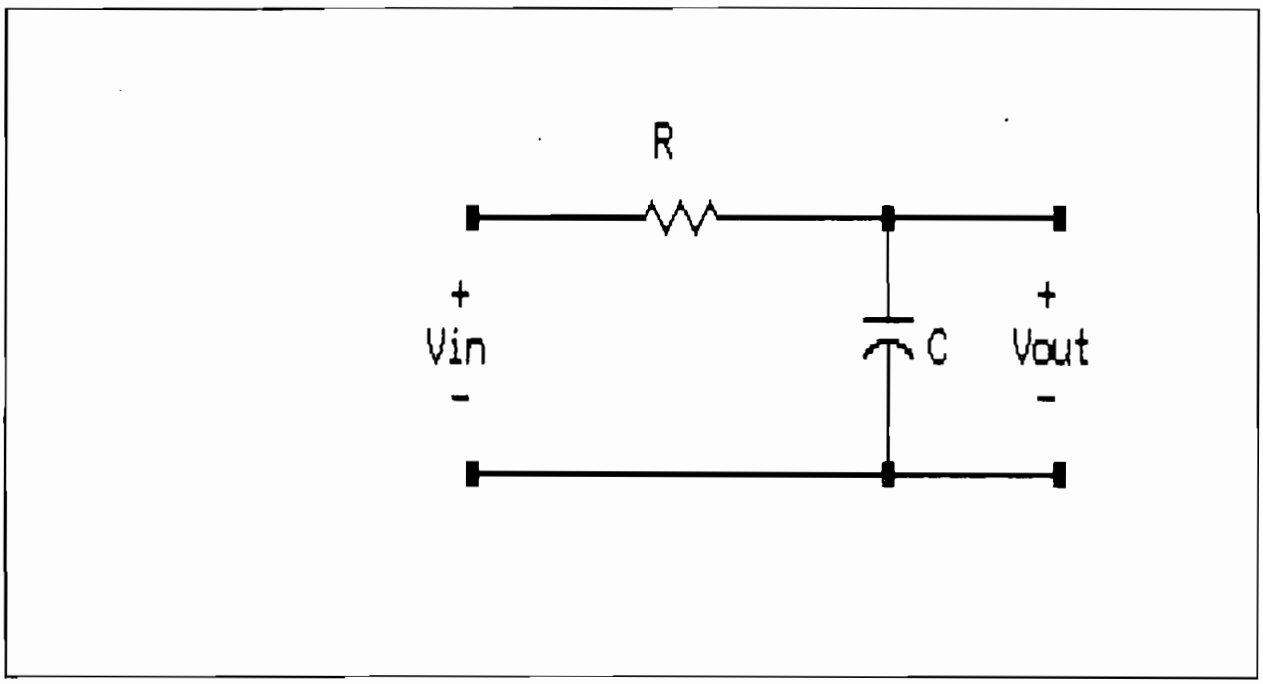
EXPLICACION DE LOS CIRCUITOS IMPRESOS

ANEXO E

El objetivo de este anexo es dar una breve explicación acerca de los circuitos impresos para la identificación y control en tiempo real.

– Circuito primer orden pasivo (RC).–

Es un circuito RC elemental, como se ve en la figura E.1:



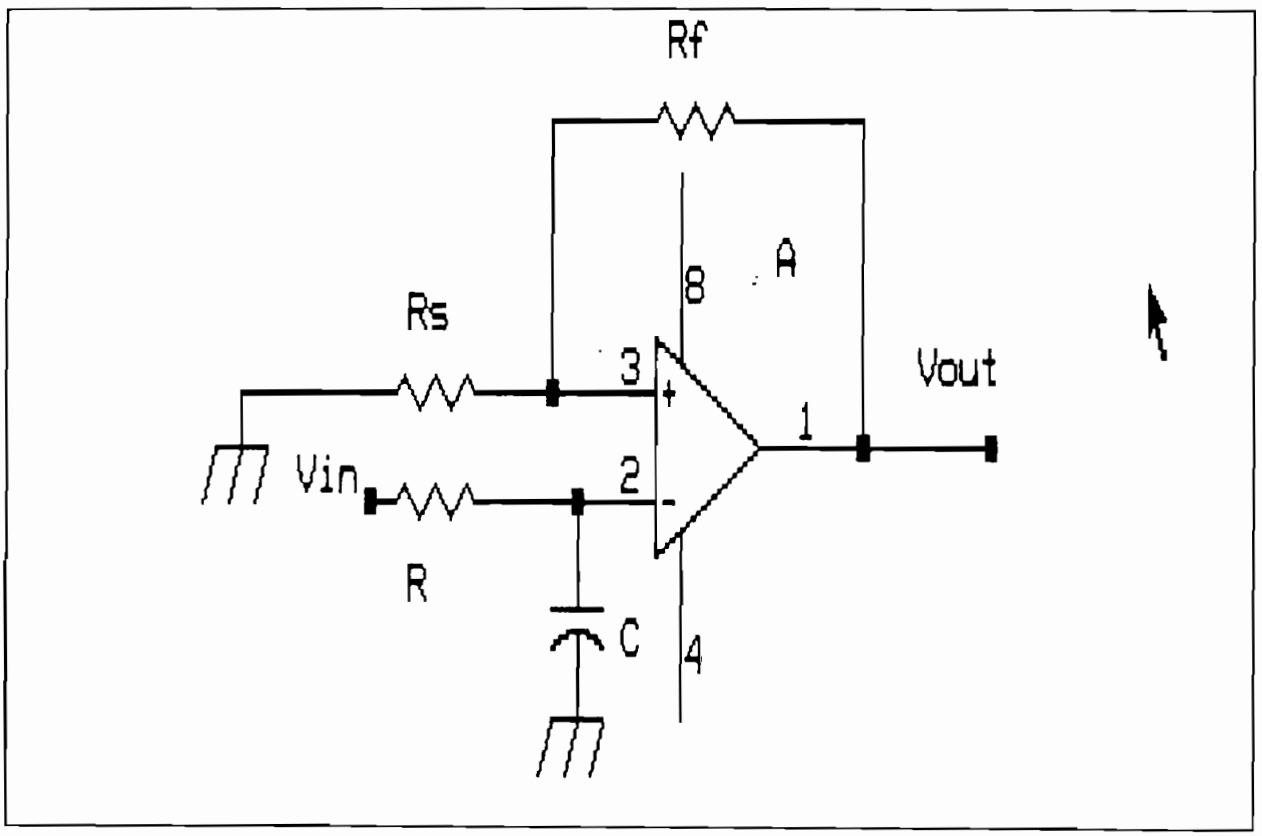
CIRCUITO RC PASIVO
FIGURA E.1

Este circuito no necesita alimentación externa. Sus únicas

conexiones externas son la entrada del canal análogo cero y la del canal de salida análogo cero del sistema de adquisición de datos. Su diagrama en smart se ve en la figura E.2, con el que se hizo el circuito impreso.

- Circuito filtro activo pasa bajos.-

Su circuito se muestra en la figura E.3:



FILTRO PASA BAJOS
FIGURA E.3

1X checkplot 10 Sep 1993 23:06:19
circ2
v1.2 r3 holes: 7 solder side
approximate size: 2.95 by 1.55 inches

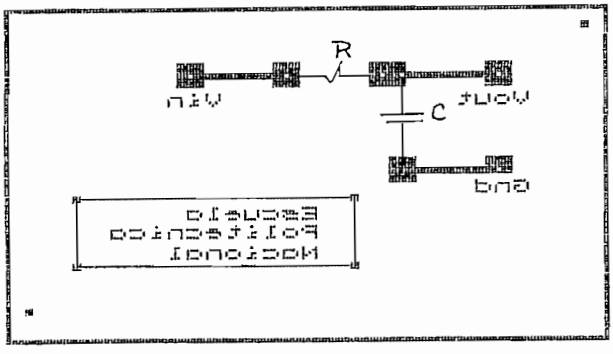


FIGURA E.2

1X checkplot 10 Sep 1993 23:07:23
circ1
v1.2 r3 holes: 22 solder side
approximate size: 3.50 by 2.35 inches

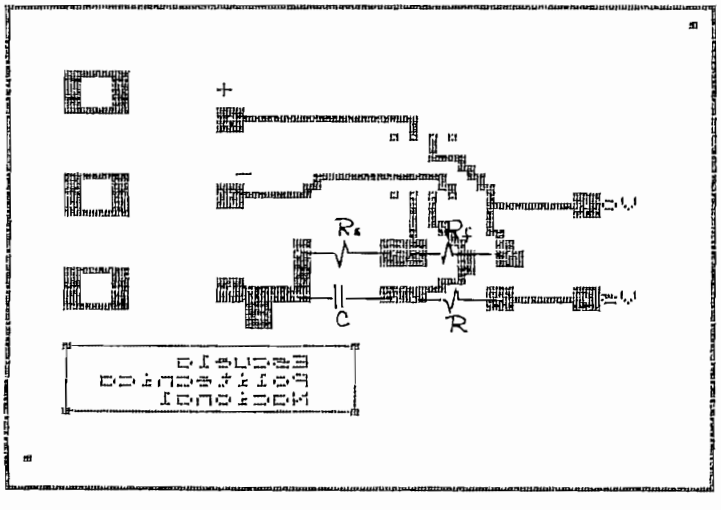


FIGURA E.4

Necesita alimentación externa, el ingreso de dos valores +15 y −15 voltios, y la tierra común de las dos fuentes, con el fin de polarizar al amplificador operacional utilizado.

Adicionalmente tiene el ingreso de los canales de entrada y salida (canal Ø, para los dos casos) del sistema de adquisición de datos.

Su diagrama en smart se presenta en la figura E.4.

− Circuito activo de segundo orden.−

La ecuación a ser simulada es:

$$y'' = (+/-)\,ay' (+/-)\,by (+/-)\,c$$

Para cumplir estos requerimientos se presenta el siguiente diagrama de la figura E.5 y su correspondiente tabla para el uso de los switches presentes en dicho diagrama, donde:

Ø => switch cerrado

1 => switch abierto

sw2	sw1	sw0	a	b	c
Ø	Ø	Ø	+	−	−
Ø	Ø	1	+	−	+
Ø	1	Ø	+	+	−
Ø	1	1	+	+	+
1	Ø	Ø	−	+	−
1	Ø	1	−	+	−
1	1	Ø	−	−	−
1	1	1	−	−	+

La existencia de los switches sw2', sw1' se deben para abrir el otro camino no necesario, por lo que su condición debe ser inversa a los switches sw1, sw2. Mientras que cuando sw0 esté cerrado, R1 no debe estar presente en el circuito.

Es un circuito que necesita polarización externa +15, -15 voltios y tierra para polarizar a los amplificadores operacionales.

Adicionalmente tiene el ingreso de los canales de entrada y salida cero del sistema de adquisición de datos.

Su diagrama en smart se presenta en la figura E.6.

1X checkplot 15 Jul 1993 16:19:15
circ4
v1.2 r3 holes= 99 solder side
approximate size= 7.95 by 3.55 inches

FIGURA E.6