ESCUELA POLITECNICA NACIONAL

FACULTAD DE INGENIERIA ELECTRICA

TESIS DE GRADO

"IDENTIFICACION DE SISTEMAS EN TIEMPO REAL"

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

GERMANICO ALFREDO PINTO TROYA

MARZO, 1991

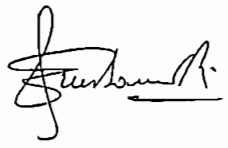
Dedicatoria

A mis padres

Agradecimiento

Un agradeciendo sincero al Ing. Patricio Burbano sin cuyo apoyo e interés no hubiera sido posible este trabajo.

Certifico que el presente trabajo ha sido elaborado en su totalidad por el Sr. Germánico Alfredo Pinto Troya.



Ing. Patricio Burbano R.

Director

CONTENIDO

1. INTRODUCCION GENERAL

1.1. INTRODUCCION

1.2. MODELACION DE SISTEMAS

1.3. CONTROL DISCRETO

1. INTRODUCCION GENERAL

1.1. INTRODUCCION

Se entiende por sistema a una combinación de componentes que actúan conjuntamente y cumplen determinado objetivo (OGATA 1974: 3). Los sistemas pueden ser no solamente físicos (eléctricos, mecánicos, químicos...) sino también económicos, biológicos, sociales.

El estudio de los sistemas no sólo busca lograr una comprensión de su funcionamiento, dinámica, respuesta ante ciertas excitaciones sino también el ajuste de estos para que cumplan con características preestablecidas. Los sistemas de control son justamente aquellos que permiten la regulación del sistema estudiado introduciendo dentro del sistema elementos de control, elementos de medida y actuadores.

La selección del tipo de control a utilizarse y de los elementos que se requieren introducir así como la determinación de las señales de entrada al sistema se basa principalmente en el grado de conocimiento que se tenga del mismo. La forma más útil para el estudio de un sistema está en la representación de éste en base de fórmulas y ecuaciones matemáticas que describan de la mejor manera posible su funcionamiento. Por tanto el estudio de un sistema pasa por

su modelación (descripción matemática) y su control (ajuste de su comportamiento).

1.2. MODELACION DE SISTEMAS

Un modelo es una representación matemática de los aspectos más importantes y relevantes de un sistema físico existente o propuesto (LEIGH 1983: xi). En sistemas bastante sencillos el modelo puede conseguir explicar el comportamiento global del sistema. En sistemas complejos, el modelo busca encontrar una formulación matemática de sus aspectos más relevantes para la aplicación específica que se busca estudiar. La generalidad de los sistemas que se encuentran en la vida real son complejos, por lo que el modelo es tan sólo una aproximación de esta realidad, aproximación que se basa en un compromiso entre la complejidad matemática y la exactitud deseadas (OGATA 1974: 76).

Los modelos matemáticos son una expresión de la dinámica del sistema. De esta manera es posible, con el sólo conocimiento del modelo, preveer el comportamiento del sistema ante excitaciones de diversa índole. No siempre es posible encontrar una solución del modelo que pueda expresarse matemáticamente mediante funciones o series. En la mayoría de los casos la resolución se da usando métodos computacionales. Esto es lo que se conoce como la simulación, cuando el modelo encontrado se implementa en un computador (analógico o

digital), obteniéndose así un "análogo computacional" del sistema físico. (LEIGH 1983: xii).

Al ser un sistema dinámico, significa que el parámetro tiempo es determinante para su comportamiento. El modelo que se obtenga por tanto tendrá características que serán una función del tiempo. Si los datos obtenidos son contínuos en el tiempo, se tendrá un modelo contínuo que se expresará matemáticamente por ecuaciones diferenciales (en una aproximación lineal del sistema). Si los datos están muestreados en el tiempo (datos discretos) el modelo será discreto y su espresión será en términos de ecuaciones de diferencias (también como una aproximación lineal). Teniendo un modelo contínuo es posible transformarlo en uno discreto a través del proceso llamado de discretización que permite la obtención de un modelo equivalente "discreto".

El proceso de elaboración de un modelo matemático para un sistema dado, parte generalmente del estudio de los componentes físicos de la planta. Esto significa la búsqueda de las ecuaciones matemáticas correspondientes a los diferentes fenómenos físicos que se consideren más importantes y relevantes del proceso: se utilizan ecuaciones basadas en los principios de invarianza (balance de masa, balance de energía, balance térmico) y de equilibrio (Leyes de Newton, Leyes de Kirchhoff, etc.) (LEIGH 1983: 16, 17). Es por tanto necesario tener un profundo conocimiento de los diferentes procesos existentes en el sistema, de manera que sea posible

la formulación matemática de los fenómenos que allí se presenten. Los sistemas reales presentan una integración de subsistemas mecánicos, eléctricos, químicos interrelacionados entre sí. Las ecuaciones que describan un sistema dado por tanto deberán no sólo expresar los procesos individuales de ciertas partes del sistema sino también su interrelación. Así, resultará bastante sencillo escribir las ecuaciones que representen a componentes individuales de la planta, pero la mayor dificultad residirá en agrupar estas ecuaciones para obtener una expresión coherente del sistema a través de un solo modelo.

La modelación por leyes físicas puede llevar a resultados muy exactos para sistemas muy simples. Basta con ubicar los fenómenos físicos con una buena aproximación y determinar adecuadamente las variables del sistema. Así, es posible modelar con una buena precisión circuitos eléctricos pasivos, sistemas de nivel de líquido con flujo laminar, sistemas mecánicos elementales de translación y de rotación, sistemas neumáticos, sistemas térmicos con temperaturas uniformes (D'AZZO 1977: 51-94; OGATA 1974: 101-130). Esto se consigue usando análogos eléctricos o mecánicos de los diferentes componentes de los sistemas (Ver Tabla 1.1). Los modelos así obtenidos son lineales e invariantes en el tiempo, es decir que sus características intrínsecas no cambian con el transcurso del tiempo.

Tabla 1.1

Elemento Mecánico de Traslación		Elemento Eléctrico	
Símbolo	Cantidad	Símbolo	Cantidad
F	Fuerza	i	Corriente
v	Velocidad	v	Voltaje
M	Masa	C	Capacidad
K	Coef. rigidez	1/L	Inductancia^{-1}
B	Coef. Amortiguamiento	G	Conductancia

Todo sistema que presente nolinealidades o formas que no puedan ser expresadas en ecuaciones físicas no podrán ser modelados directamente mediante este proceso. Por tanto se requerirá establecer ciertas limitaciones al sistema estudiado para que pueda ser modelado. Un sistema nolineal será "linealizado" en torno a un punto normal de operación. Un sistema cuyos componentes físicos sean muy complejos, requerirá una aproximación a un sistema más simple para que pueda ser expresado en forma de modelo. En ambos casos, se estaría comprometiendo la exactitud del modelo en aras a encontrar una forma de expresión más simple y se estaría circunscribiendo el rango de aplicación del modelo. No obstante esto, sigue siendo posible, pese a lo largo y laborioso, llegar a un modelo del sistema.

Un primer problema aparece cuando el sistema es muy complejo y no es posible alcanzar a individualizar sus componentes: es la primera gran limitación de la modelación por leyes físicas. Un segundo problema y posiblemente más serio aparece cuando el sistema no es invariante con el tiempo. Los parámetros del

modelo dejan de ser constantes y a su vez se tornan dependientes del tiempo: las ecuaciones físicas se vuelven extremadamente complejas y en la mayoría de los casos no es posible expresarlas matemáticamente.

Adicionalmente puede estar presente ruido en el sistema, el cual al no ser considerado en la modelación, no permitirá ninguna aplicación práctica del modelo para análisis de señales.

La solución a estos problemas pasa por una reformulación completa del proceso de modelación. Ahora se va a trabajar con un sistema que se asume muy complejo y del cual se desea encontrar un modelo que cumpla con sus características esenciales. Ya no se busca la información sobre el sistema por sus componentes físicos, sino la información sobre su dinámica a través de una segunda fuente: los datos experimentales de entrada y salida de la planta (FRANKLIN 1980: 207). El proceso consiste ahora en la estimación estadística de sus parámetros a partir de la información de entrada y salida. La estimación estadística significa la obtención de valores de parámetros que se ajusten de la mejor manera posible a la dinámica real del sistema. Este proceso es conocido como la identificación de sistemas por estimación de parámetros.

Las ventajas principales de este proceso de modelación son:

1) La información que se requiere es fácil de obtener y no es necesario ningún estudio físico previo de la planta. Basta medir adecuadamente los datos de entrada y salida.

2) Se hace posible la modelación de sistemas muy complejos que contengan incluso nolinealidades.

3) El proceso de modelación se puede realizar fácilmente mediante un computador. Allí será necesario tener la información en forma discreta y el modelo encontrado será discreto. Para esto se requerirán algoritmos que aseguren la convergencia y permitan la obtención de los "mejores" parámetros.

4) En control digital, la obtención directa de un modelo discreto evita el trabajo de discretizar el modelo.

Los inconvenientes principales se basan en el proceso mismo de modelación:

1) Antes de iniciar la identificación es necesario determinar el orden del modelo que se desea construir. Allí es importante el conocimiento de la dinámica de la planta. Esto lleva obligatoriamente a realizar una aproximación que será mejor en tanto mayor sea el orden del sistema con las consiguientes dificultades en los cómputos.

2) En las plantas industriales reales no es posible realizar las pruebas del sistema en base a los métodos acostumbrados (pruebas de impulso o escalón a la entrada) para encontrar la información de entrada y salida. No se puede parar un proceso industrial: es necesario encontrar entonces formas para obtener información de entrada y salida que abarque lo suficientemente la dinámica de la planta sin alterar su funcionamiento habitual. Esto se consigue mediante pequeñas perturbaciones a la entrada.

Resolviendo estos dos problemas, el método de modelación por identificación de sistemas resulta más fácilmente aplicable a los diferentes tipos de sistemas. En este trabajo, éste será el método de modelación utilizado ya que el objetivo final es realizar control digital.

1.3. CONTROL DISCRETO

Cuando se desea que un sistema cumpla con ciertos objetivos predeterminados, se hace necesario la construcción de un sistema de control realimentado que actúe sobre las entradas de la planta para conseguir estos objetivos. La configuración básica de un sistema de control realimentado consta de la planta (su modelo), del controlador, del lazo de realimentación, y de la referencia externa, según se muestra en la figura 1.1.

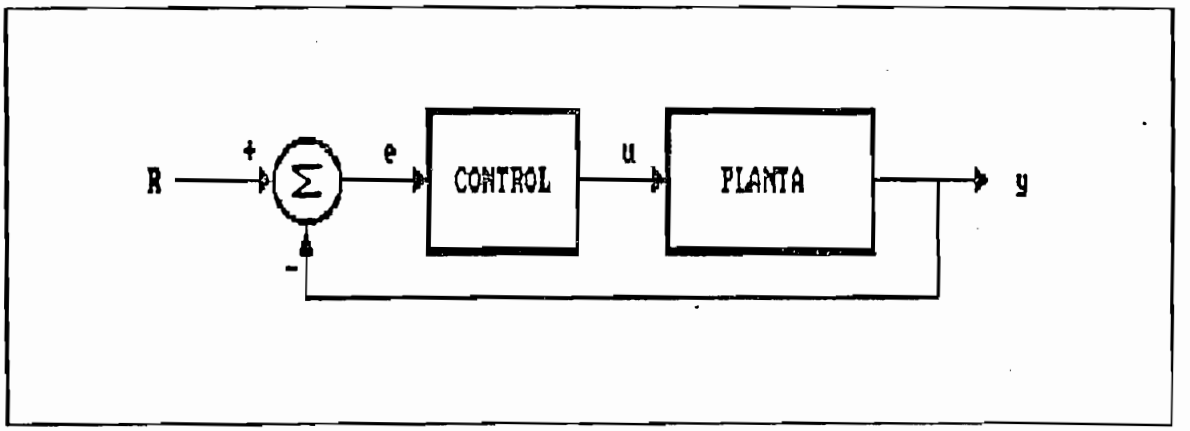


Figura 1.1 – CONTROL REALIMENTADO

Conociendo por un lado la referencia externa y por otro la planta a través de su modelo, el problema radica en el diseño del controlador de manera que se cumplan ciertas características del sistema. Seleccionando una ley de control que se considere adecuada, se deberá estudiar la estabilidad del sistema, su respuesta transitoria, su regulación de salida, sus restricciones y su robustez (GOODWIN 1984: 118-119) y con esta información llegar al diseño final del controlador.

Si el controlador trabaja con señales continuas en el tiempo, se tratará de un controlador analógico. En este caso, su salida corresponderá directamente a las entradas requeridas por la planta. Por el contrario, si el controlador trabaja con señales discretas en el tiempo, el controlador será digital (se basará en las técnicas del control discreto), y su salida necesitará un conversor digital-analógico para generar las señales que requiera la planta. Este será el tipo de controlador del cual se tratará en este trabajo por acoplarse

más facilmente al modelo proveniente de la identificación de sistemas ya mencionada y por poder ser utilizado mediante un computador digital para control en tiempo real.

El uso de los computadores digitales en el control está generalizándose cada vez más por su facilidad de incorporación al sistema de control y por la flexibilidad que permite en el diseño mismo del controlador. Existen dos formas de introducir a un computador en un sistema de control:

1) Como simple herramienta de cálculo, es decir que recibe datos del sistema, pero entrega sus respuestas del diseño al usuario para que éste a su vez implemente un control para que ejecute los correctivos a realizar en el sistema. Es el control FUERA DE LINEA (OFF-LINE) como se puede apreciar en la figura 1.2.

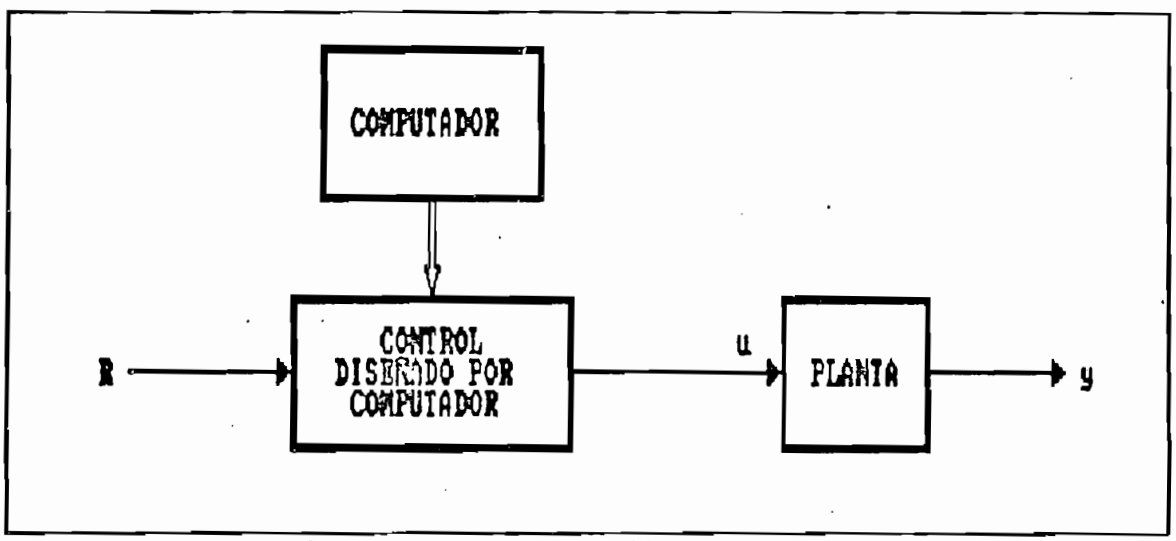


Figura 1.2 - CONTROL OFF-LINE

2) Como parte integrante del lazo de realimentación, en cuyo caso el computador actúa como controlador, es decir que la ley de control aparece como un algoritmo que procesa la información de error y entrega los datos de salida directamente a la planta, sin la participación del usuario en este proceso. El papel del usuario se limitará a realizar las inicializaciones que sean necesarias sobre la ley de control. Esta es la forma conocida como control EN LINEA (ON-LINE). El Control EN LINEA permite, por tanto, el uso de un computador como parte integrante del sistema realimentado para realizar las tareas de control. Este tipo de control se conoce también como CONTROL EN TIEMPO REAL. La figura 1.3 ilustra este tipo de control.

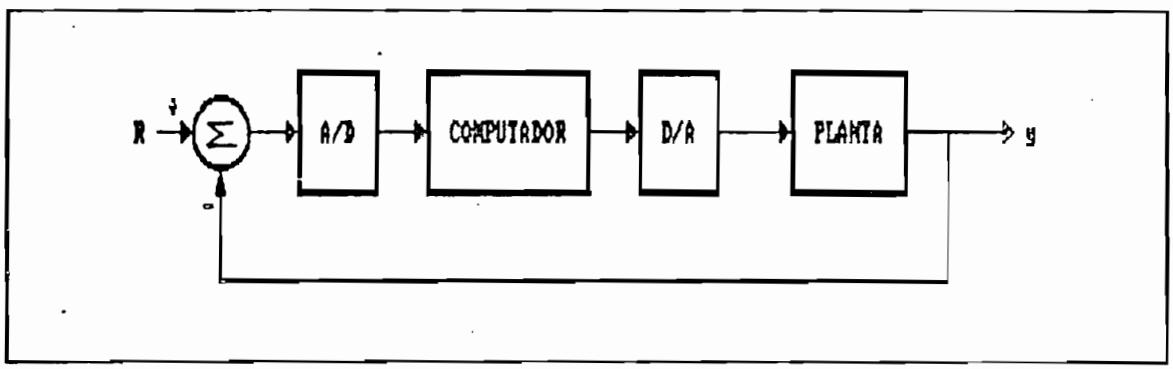


Figura 1.3 - CONTROL ON-LINE

Existen diversas formas de realizar un control digital cada una con aplicaciones particulares, la mayoría de ellas basadas en la forma general de un control realimentado EN LINEA o en TIEMPO REAL, sea mediante el uso de computadores o microcontroladores.

a) Control Para Instrumentación.

Los controladores de instrumentación se utilizan generalmente como medios de interfase entre los diferentes componentes del sistema: estación de asquisición de datos y control, monitoreo de señales (instrumentación inteligente) para supervisión de la planta, y, en general, acondicionamiento de señales desde los sensores y hacia los actuadores.

b) Control Supervisor

En este tipo de control, el computador es utilizado como una herramienta de cálculo o de supervisión de las referencias de control de la misma manera que lo haría un operador manual, es decir en operación OFF-LINE. (BENNETT 1982: 19). En sistemas más complejos, el control supervisor se encarga de calcular los puntos de referencia en base a los cuales actúa el control con el fin de optimizar el funcionamiento de la planta.

c) Control Digital Directo

El control digital directo (Digital Direct Control - DDC) es el resultado esencial de la inclusión del computador dentro del lazo de control. Aparecen entonces los convertidores Análogo-Digitales para los datos medidos y Digital-Análogos

para los datos que van a los actuadores. La ley de control que anteriormente se construía en términos analógicos, en este caso se transforma en un algoritmo de computación. Este tipo de control actualmente es el más utilizado, especialmente desde el advenimiento de los microcontroladores que permiten abaratar costos y mejorar el desempeño del control. Este control corresponde a la estructura básica del control ON-LINE (ver figura 1.3)

d) Control Distribuído

En un sistema de control distribuído existen múltiples procesadores que se encargan de realizar el control de un planta. Las acciones de control no se encuentran centralizadas en un solo procesador sino que se encuentran distribuídas en la planta. Así se tiene procesadores dedicados exclusivamente al control de sensores y actuadores, a la implementación de algoritmos de control, a la interrelación con el operador, a la comunicación entre los diferentes componentes del sistema (BENNETT 1982: 112).

e) Control Adaptivo

Un regulador adaptivo es aquel que puede modificar su comportamiento como respuesta a cambios en la dinámica de un proceso o a la existencia de perturbaciones (ASTROM 1989: p. 1). En

este caso, además del lazo de control realimentado, aparece un segundo lazo que ajusta los nuevos parámetros del regulador. En sistemas de control comunes, la calibración de los parámetros del control se hace una sola vez, al inicio de su operación. En los sistemas de control adaptivos, esta calibración se realiza constantemente durante el funcionamiento del sistema, permitiendo de esta manera que el control pueda responder adecuadamente a nuevas condiciones de la planta, no previamente definidas. El control adaptivo puede definirse entonces como la automatización del proceso de diseño de sistemas de control, diseño que se realiza íntegramente por el computador. (ASTROM 1988: p. 8)

f) Control Secuencial

Muchos de los procesos industriales requieren pasar por algunas etapas predefinidas. El control secuencial se encarga de llevar al sistema a estas diferentes etapas en forma automática, generalmente variando el valor de la referencia del sistema. Esto se realiza con el uso de un computador que, en base al estado del sistema, determina el paso a la siguiente etapa. Todo el proceso de toma de decisiones y temporalización se realiza mediante algoritmos en software. (BENNETT 1982: 7-8).

2. MINIMOS CUADRADOS ESTOCASTICO

2.1. CRITERIOS PARA LA IDENTIFICACION PARAMETRICA DISCRETA

2.2. MINIMOS CUADRADOS ESTOCASTICO

2.3. CRITERIOS ESTADISTICOS PARA LA IDENTIFICACION PARAMETRICA

2.4. MINIMOS CUADRADOS ESTOCASTICO

2.5. EXCITACION PERSISTENTE

2. MINIMOS CUADRADOS ESTOCASTICO

2.1. CRITERIOS PARA LA IDENTIFICACION PARAMETRICA DISCRETA

Todo modelo lineal se puede describir matemáticamente en base de su función de transferencia o de sus variables de estado. A su vez es posible aproximar a los sistemas no lineales a un modelo lineal en base a procesos de linealización. Por tanto, la identificación paramética se basará en la estimación estadística de los parámetros sea de una función de transferencia, sea de una descripción en variables de estado del sistema.

En términos discretos, el modelo a identificar será por tanto de la siguiente forma:

a) Descripción a variables de estado

$$x_{k+1} = \Phi x_k + \Gamma u_k$$
$$y_k = H x_k + J u_k$$

Si el modelo es de orden n, existirán n variables de estado, por lo tanto la matriz Φ y H tendrán n^2 parámetros, Γ y J tendrán n lo que dá un total de $(2n^2 + 2n)$ parámetros a identificar.

b) Descripción con función de transferencia

$$G(z) = \frac{b_1 z^{-1} + b_2 z^{-2} + \ldots + b_n z^{-n}}{1 + a_1 z^{-1} + a_2 z^{-2} + \ldots + a_n z^{-n}}$$

Si esta función de transferencia es equivalente a la descripción en variables de estado antes mencionada, se tendrá que identificar n parámetros a y n parámetros b, es decir un total de (2n) parámetros.

Esto significa que es posible describir el funcionamiento del sistema con tan sólo (2n) parámetros, por lo que la descripción en variables de estado es susceptible de ser simplificada a un modelo con (2n) parámetros.

Efectivamente, la teoría de las variables de estado establece que la definición de estado no es única, por lo que existen múltiples posibles expresiones en variables de estado de un mismo sistema. No obstante esto, es posible expresar el modelo de una forma simplificada mediante las conocidas Formas Canónicas.

Para la identificación paramétrica discreta, la información que se tiene de la planta son las señales de entrada y salida a diferentes intervalos de tiempo. El modelo buscado por tanto debe ajustarse a esta información y además debe requerir del mínimo número de parámetros a ser identificados. Esto se consigue definiendo a los estados como estas señales de

entrada y salida y a partir de ahí construyendo una forma canónica apropiada.

Un modelo de orden 3, por ejemplo, requerirá de 6 parámetros para su expresión como función de transferencia. La forma canónica buscada deberá contemplar estos 6 parámetros. La solución a este problema es la utilización de un modelo ARMA que cumple con estos dos requisitos fundamentales para la identificación.

El modelo ARMA (AUTOREGRESSIVE MOVING AVERAGE - Autoregresivo y de Media Móvil) se caracteriza por lo siguiente: el último valor de un cierto proceso es una función lineal tanto de los valores anteriores del propio proceso (Autoregresivo) como de los valores anteriores de la señal de entrada (Media Móvil) (BOX 1976).

Matemáticamente:

$$y(t) + a_1 y(t-1) + a_2 y(t-2) + \ldots + a_n y(y-n) = b_1 u(t-1) + b_2 y(t-2) + \ldots + b_n u(t-n)$$

o en el plano z:

$$[1 + A(z)]y(t) = B(z)u(t)$$

donde:

$$A(z) = a_1 z^{-1} + a_2 z^{-2} + \ldots + a_n z^{-n}$$
$$B(z) = b_1 z^{-1} + b_2 z^{-2} + \ldots + b_n z^{-n}$$

Entonces, se definen como variables de estado a los valores anteriores de las señales de entrada y salida. Por tanto, la expresión matricial será:

$$x(t) = \Phi x(t-1) + \Gamma u(t-1)$$
$$y(t) = Hx(t-1)$$

donde:

$$x^T(t) = [-y(t-1)\ -y(t-2)\ \dots\ -y(t-n)\ u(t-1)\ \dots\ u(t-n)]$$

$$\Gamma^T = [0\ 0\ \dots\ 0\ 1\ \dots\ 0]$$

$$\Phi = \left(\begin{array}{ccccc|ccccc} a_1 & a_2 & \dots & a_{n-1} & a_n & b_1 & b_2 & \dots & b_{n-1} & b_n \\ 1 & 0 & \dots & 0 & 0 & 0 & 0 & \dots & 0 & 0 \\ \vdots & \vdots & & \vdots & \vdots & \vdots & \vdots & & \vdots & \vdots \\ 0 & 0 & \dots & 0 & 0 & 0 & 0 & \dots & 0 & 0 \\ 0 & 0 & \dots & 1 & 0 & 0 & 0 & \dots & 0 & 0 \\ - & - & - & - & - & + & - & - & - & - \\ 0 & 0 & \dots & 0 & 0 & 0 & 0 & \dots & 0 & 0 \\ 0 & 0 & \dots & 0 & 0 & 1 & 0 & \dots & 0 & 0 \\ \vdots & \vdots & & \vdots & \vdots & \vdots & \vdots & & \vdots & \vdots \\ 0 & 0 & \dots & 0 & 0 & 0 & 0 & \dots & 0 & 0 \\ 0 & 0 & \dots & 0 & 0 & 0 & 0 & \dots & 1 & 0 \end{array}\right)$$

$$H^T = [a_1\ a_2\ \dots\ a_n\ b_1\ \dots\ b_n]$$

Si bien esta descripción a variables de estado no es mínima (es un sistema de orden n pero hay 2n variables de estado), es de mucho interés por la facilidad con que se presenta la ecuación de Hxlida, que no es otra cosa que la expresión matemática del modelo ARMA, en donde se ha despejado y(t).

$$
\begin{aligned}
y(t) &= Hx(t-1) \\
&= [a_1\ a_2\ \dots\ a_n\ b_1\ \dots\ b_n] \cdot \\
&\qquad \cdot [-y(t-1)\ -y(t-2)\ \dots\ -y(t-n)\ u(t-1)\ \dots\ u(t-n)]^T \\
&= -a_1 y(t-1) - a_2 y(t-2) - \dots - a_n y(t-n) + b_1 u(t-1) + \\
&\quad + b_2 u(t-2) + \dots + b_n u(t-n)
\end{aligned}
$$

La expresión es por tanto bastante sencilla. Será con ella que se realizará el proceso de identificación paramétrica.

Entonces, la función de transferencia es evidente:

$$G(z) = \frac{y(z)}{u(z)} = \frac{B(z)}{1 + A(z)}$$

Habiendo definido el modelo para la identificación como un modelo ARMA, es necesario definir ahora el método general mediante el cual se va a proceder a la estimación estadística de parámetros.

El modelo puede ser ahora expresado como el producto del vector de estados $\mathbf{x}(t)$ y de un vector de parámetros $\Theta(t)$ que no es sino el H anterior.

$$y(t) = \mathbf{x}^T(t)\Theta(t)$$

donde:

$$\mathbf{x}^T(t) = [-y(t-1)\ -y(t-2)\ \dots\ -y(t-n)\ u(t-n)\ \dots\ u(t-n)]$$
$$\Theta(t) = [a_1\ a_2\ \dots\ a_n\ b_1\ \dots\ b_n]^T$$

El resultado de la estimación será un vector $\bar{\Theta}(t)$ de parámetros estimados que deberá ser muy cercano sino igual al

vector $\theta(t)$ de parámetros verdaderos. El reemplazo de $\bar{\theta}(t)$ por $\theta(t)$ conlleva un error que es necesario determinar y minimizar. La forma más elemental de medir este error es encontrando la diferencia entre ambos: $\bar{\theta}(t) - \theta(t)$; pero esto no es posible al no conocerse $\theta(t)$, es decir los parámetros verdaderos. Se hace necesario encontrar un método que permita obtener el error a partir de los datos conocidos.

Se han desarrollado varias formas para medir el error: el error de ecuación, el error de salida y el error de predicción de salida. En este trabajo se utilizará el error de ecuación que se explicará con detalle; los otros errores se dan como referencia.

a) error de ecuación.

Para determinar el error de ecuación se parte de la ecuación que describe el sistema, es decir de la expresión mediante variables de estado. Se supone además que es posible medir todos los estados, sus derivadas (o sus valores siguientes discretos) y las señales de entrada. Entonces, se calcula el error como la diferencia entre los estados reales $x(t)$ y los estados $\bar{x}(t)$ resultado de la aplicación en el modelo de los parámetros estimados $\bar{\theta}(t)$.

$$\bar{x}(t+1) = f\{x(t+1), \bar{\theta}(t)\}$$

$$e(t; \bar{\theta}) = x(t+1) - \bar{x}(t+1)$$
$$= x(t+1) - \Phi x(t) - \Gamma u(t)$$

donde Φ es una función del vector de parámetros $\bar{\theta}(t)$.

Resolviendo se obtiene:

$$e_1(t; \bar{\theta}) = x_1(t+1) - a_1 x_1(t) - a_2 x_2(t) \ldots - a_n x_n(t) - b_1 x_{n+1}(t) - \ldots - b_n x_{2n}(t)$$
$$= y(t) + a_1 y(t-1) + a_2 y(t-2) + \ldots + a_n y(t-n) - b_1 u(t-1) - \ldots - b_n u(t-n)$$
$$= y(t) - x^T(t)\bar{\theta}(t)$$
$$= y(t) - \bar{y}(t)$$

donde:

$\bar{y}(t)$ es el valor estimado de y(t) con los parámetros $\bar{\theta}(t)$

$$e_2(t; \bar{\theta}) = x_2(t+1) - x_1(t)$$
$$= -y(t-1) + y(t-1)$$
$$= 0$$

Y de la misma forma, para $2 < i \leq n$

$$e_i(t; \bar{\theta}) = 0$$

Por tanto el error de ecuación se reduce a:

$$e_1(t; \bar{\theta}) = y(t) - \bar{y}(t) \qquad \text{(Ec. 2.1)}$$

Si se busca los mejores valores de $\bar{\theta}(t)$, será necesario minimizar la expresión siguiente:

$$J(\overline{\theta}) = \sum_{k=0}^{N} \left(e_1(k;\overline{\theta})\right)^2$$

El error de ecuación es por tanto fácilmente adaptable al modelo ARMA pues los estados del sistema corresponden justamente a los valores de entrada y salida medidos. Esto hace posible utilizar este tipo de error para la estimación de los parámetros de la planta.

La minimización de $J(\overline{\theta})$ se realizará mediante el método de mínimos cuadrados.

b) error de salida

Cuando resulta difícil medir todos los estados de un sistema así como sus derivadas, se suele utilizar el error de salida. Para este método, basta medir el valor de la salida del sistema y compararlo con un valor estimado $y_e(t)$ resultante de la aplicación de los parámetros $\overline{\theta}(t)$ a un modelo. Este modelo, a diferencia del error de ecuación tendrá como variables de estado de salida, los diferentes valores estimados $y_e(t)$.

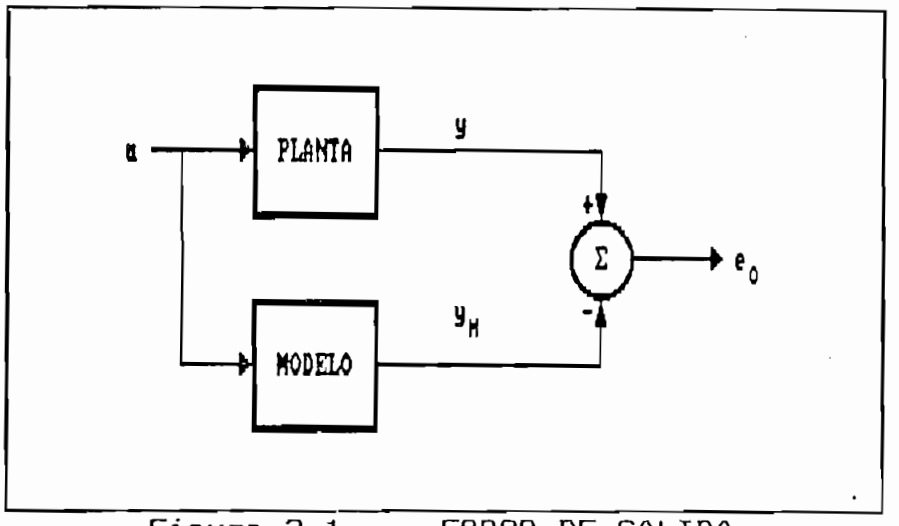


Figura 2.1 – ERROR DE SALIDA

El diagrama de la figura 2.1 muestra la formulación del error de salida, en donde $e_o(t; \bar{\Theta})$ está dado por:

$$e_0(t;\bar{\Theta}) = y(t) - y_m(t)$$

y por consiguiente se define al nuevo $J(\bar{\Theta})$

c) error de predicción de salida

En lugar de utilizar un modelo para obtener el valor de $y_m(t)$, se trabaja con un algoritmo de predicción que trata de conseguir el valor más cercano de $\bar{y}(t)$ en base de las observaciones previas. Con este error se formula igualmente la expresión $J(\bar{\Theta})$.

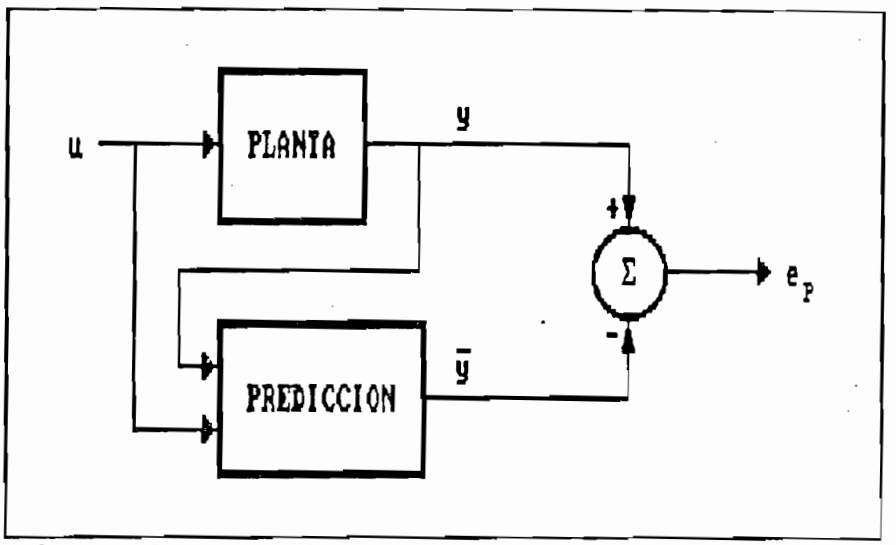


Figura 2.2 - ERROR DE PREDICCION DE SALIDA

El diagrama de la figura 2.2 indica la forma de trabajo del error de predicción de salida.

2.2 MINIMOS CUADRADOS DETERMINISTICO

Una de las formas más utilizadas para conseguir un buen $\bar{\theta}(t)$ estimado, es decir una minimización de $J(\bar{\theta})$ es el método de mínimos cuadrados determinístico. Para la aplicación de este método se parte de un modelo ARMA donde se miden entradas y salidas y una expresión de error dada por el error de ecuación.

Habiendo definido previamente el modelo ARMA a variables de estado, con los vectores $\theta(t)$, $\bar{\theta}(t)$, $x(t)$ y considerando, para mayor simplicidad que $e_i(t; \bar{\theta}) = e(t; \bar{\theta})$ en el error de ecuación, se observarán un conjunto de valores de entrada y

salida, que en cada medición pueden ser resumidos en el vector x(t). Para facilitar la expresión matemática, el vector x(t) estará dado de aquí en adelante por:

$$\boldsymbol{x}(t) = [-y(t-1)\ -y(t-2)\ \ldots\ -y(t-n)\ u(t-1)\ \ldots\ u(t-n)]$$

Con estos valores medidos, se construyen la matriz de datos X(t) y el vector de datos Y(t) dados por:

$$\boldsymbol{X}^T(t) = [\boldsymbol{x}(0)\ \boldsymbol{x}(1)\ \boldsymbol{x}(2)\ \ldots\ \boldsymbol{x}(t)]$$
$$\boldsymbol{Y}^T(t) = [y(0)\ y(1)\ y(2)\ \ldots\ y(t)]$$

Entonces, a partir de (Ec. 2.1), se puede generalizar para los diferentes valores de t, con:

$$\boldsymbol{e}(t) = \boldsymbol{Y}(t) - \boldsymbol{X}(t)\overline{\Theta}(t)$$

donde:

$$\boldsymbol{e}(t) = [e(0)\ e(1)\ e(2)\ \ldots\ e(t)]$$

Con esta expresión general del error es posible entrar a la minimización por medio del método de mínimos cuadrados determinístico:

$$J(\overline{\Theta}) = \sum_{k=0}^{N} \left(e_1(k;\overline{\Theta})\right)^2 = \sum_{k=0}^{N} \left(e(k;\overline{\Theta})\right)^2 = \sum_{k=0}^{N} \left(e(k)\right)^2$$
$$= \boldsymbol{e}^T\boldsymbol{e}$$
$$= \left(\boldsymbol{Y}(t) - \boldsymbol{X}(t)\overline{\Theta}(t)\right)^T\left(\boldsymbol{Y}(t) - \boldsymbol{X}(t)\overline{\Theta}(t)\right)$$

Lo que más sencillamente corresponde a:

$$J = (Y - X\bar{\Theta})^T (Y - \bar{\Theta})$$

La minimización se realiza igualando a cero la derivada de J respecto a $\bar{\Theta}$ y resolviendo la ecuación correspondiente:

$$\frac{dJ}{d\bar{\Theta}} = \frac{d}{d\bar{\Theta}}\left[Y^T Y - Y^T X\bar{\Theta} - \bar{\Theta}^T X^T Y - \bar{\Theta}^T X^T X\bar{\Theta}\right] = 0$$

Usando las reglas de derivación siguientes

$$\frac{d}{dx}\left[y^T x\right] = y$$

$$\frac{d}{dx}\left[x^T y\right] = y$$

se obtiene:

$$\frac{dJ}{d\bar{\Theta}} = -X^T Y - X^T Y - 2X^T X\bar{\Theta} = 0$$

entonces:

$$\bar{\Theta} = (X^T X)^{-1} X^T Y$$

que es la solución general del problema.

Evidentemente esta solución existe si, por un lado, el modelo escogido es del tipo ARMA o de tipo canónico obteniéndose un único valor de $\bar{\Theta}$ es decir que el sistema es "identificable" y si, por otro lado, la matriz $X^T X$ no es singular de manera

que se pueda calcular su inversa. Esta última propiedad se verifica cuando la señal u(k) es de "excitación persistente". Si, por ejemplo, la señal de control es igual a una constante, la matriz X^TX será singular y no existirá una solución única al problema. El estudio de la "excitación persistente" se realizará con más detalle en la sección 2.5.

Hasta el momento se ha podido resolver el problema de identificación en base a recoger un cierto número de datos y aplicar los mínimos cuadrados. Para una aplicación real esto significaría ir obteniendo soluciones por paquetes, lo que dificulta su aplicación. Esto se resuelve encontrando un algoritmo recursivo que vaya encontrando y actualizando los soluciones. Es lo que se conoce como los mínimos cuadrados recursivos.

El objetivo es poder obtener un valor de $\overline{\Theta}$ en base de su valor anterior y de una corrección que se iría aplicando en cada cálculo sucesivo. Así, después de un cierto número de iteraciones $\overline{\Theta}$ convergería al valor buscado y la corrección tendería a cero.

$$\overline{\Theta}(t+1) = \overline{\Theta}(t) + \textit{corrección}$$

Se parte del cálculo de $\overline{\Theta}(t+1)$ y se va a tratar de ponerlo en función de $\overline{\Theta}(t)$.

$$\overline{\theta}(t) = \left(X^T(t)X(t)\right)^{-1} X^T(t)Y(t)$$

$$\overline{\theta}(t+1) = \left(X^T(t+1)X(t+1)\right)^{-1} X^T(t+1)Y(t+1)$$

$$= \left(\left[X^T(t) \;\vdots\; x^T(t+1)\right]\begin{bmatrix} X(t) \\ \cdots\cdots \\ x(t+1) \end{bmatrix}\right)^{-1}\left(\left[X^T(t) \;\vdots\; x^T(t+1)\right]\begin{bmatrix} Y(t) \\ \cdots\cdots \\ y(t+1) \end{bmatrix}\right)$$

$$= \left(X^T(t)X(t) + x^T(t+1)x(t+1)\right)^{-1}\left(X^T(t)Y(t) + x(t+1)y(t+1)\right)$$

Se define ahora:

$$P(t) = \left(X^T(t)X(t)\right)^{-1}$$

Entonces:

$$\overline{\theta}(t) = P(t)X^T(t)Y(t)$$

$$\overline{\theta}(t+1) = P(t+1)\left[X^T(t)Y(t) + x^T(t+1)y(t+1)\right]$$

$$P(t+1) = \left(X^T(t+1)X(t+1)\right)^{-1}$$

$$= \left(X^T(t)X(t) + x^T(t+1)x(t+1)\right)^{-1}$$

$$= \left(P^{-1}(t) + x^T(t+1)x(t+1)\right)^{-1}$$

LEMA DE INVERSION DE MATRICES:

Dentro del álgebra de matrices existe un lema de inversión de matrices que dice:

$$(A + BCD)^{-1} = A^{-1} - A^{-1}B\left(C^{-1} + DA^{-1}B\right)^{-1}DA^{-1}$$

Para este caso se tiene:

$$A = P^{-1}$$
$$B = x^T(t+1)$$
$$C = I$$
$$D = x(t+1)$$

Entonces, aplicando el lema enunciado anteriormente:

$$P(t+1) = P(t) - P(t)x^T(t+1)\left(I + x(t+1)P(t)x^T(t+1)\right)^{-1}x(t+1)P(t)$$
$$= \left(I - \frac{P(t)x^T(T+1)x(T+1)}{1 + x(T+1)P(t)X^T(t+1)}\right)P(t)$$

De donde,

$$\overline{\theta}(t+1) = P(t)X^T(t)Y(t) + P(t)x^T(t+1)y(t+1) -$$
$$\left(\frac{P(t)x^T(t+1)x(t+1)}{1 + x(t+1)P(t)x^T(t+1)}\right).$$
$$.\left(P(t)X^T(t)Y(t) + P(t)x^T(t+1)y(t+1)\right)$$
$$= P(t)X^T(t)Y(t) + \left(1 + x(t+1)P(t)x^T(t+1)\right)^{-1} .$$
$$\left(P(t)x^T(t+1)y(t+1) + x(t+1)P(t)x^T(t+1)P(t) .\right.$$
$$.x^T(t+1)y(t+1) - P(t)x^T(t+1)x(t+1)P(t)X^T(t)Y(t) +$$
$$\left. + P(t)x^T(t+1)x(t+1)P(t)x^T(t+1)y(t+1)\right)$$

Con lo que,

$$\overline{\theta}(t+1) = P(t)X^T(t)Y(t) + \left(1 + x(t+1)P(t)x^T(t+1)\right)^{-1} .$$
$$\left(P(t)x^T(t+1)y(t+1) - P(t)x^T(t+1)x(t+1)P(t)X^T(t)Y(t)\right)$$
$$= \overline{\theta}(t) + \frac{P(t)x^T(t+1)}{1 + x(t+1)P(t)x^T(t+1)}\left(y(t+1) - x(t+1)\overline{\theta}(t)\right)$$

Por tanto,

$$\overline{\Theta}(t+1) = \overline{\Theta}(t) + L(t+1)\epsilon(t+1)$$

donde,

$$L(t+1) = \frac{P(t)x^T(t+1)}{1 + x(t+1)P(t)z^T(t+1)}$$

$$P(t) = \left[I - \frac{P(t-1)x^T(t+1)x(t+1)}{1 + x(t+1)P(t-1)x^T(t+1)}\right]P(t-1)$$

$$\epsilon(t+1) = y(t+1) - x(t+1)\overline{\Theta}(t)$$

De esta manera queda definido el proceso recursivo de cálculo de los parámetros $\overline{\Theta}(t)$. El algoritmo general estaría dado por:

1) Dar valores iniciales para $\overline{\Theta}(0)$, P(0)

2) Obtener y(0); iniciar con t = 0

3) Obtener u(t); y(t+1)

4) Actualizar x(t+1)

5) Calcular L(t+1); ϵ(t+1)

6) Calcular $\overline{\Theta}(t+1)$; P(t+1)

7) Repetir desde 3) con un nuevo valor de t (t <-- t + 1)

El resultado final está dado por los valores de $\overline{\Theta}(t)$ al realizarse el número de iteraciones deseado.

La inicialización del algoritmo se dá con valores de $\overline{\Theta}(0)$ y de $P(0)$. Para iniciar convenientemente el algoritmo, los valores de $\overline{\Theta}(0)$ pueden ser dados según la estimación intuitiva que se haga de los parámetros reales Θ. Por el contrario, los valores iniciales de $P(0)$ deben permitir que el algoritmo trabaje, es decir que la matriz $P(0)$ debe ser no singular y definida positiva. Para una convergencia rápida de los parámetros, $P(0)$ debe ser suficientemente grande, por lo que normalmente se escoge $P(0) = \alpha I$, con un valor grande de α (generalmente entre 1000 y 10000).

Hasta aquí se ha trabajado con parámetros constantes Θ, pero en muchos casos, existen variaciones de los parámetros de la planta. Esto tiene consecuencias especialmente al realizar control en lazo cerrado en base a los parámetros identificados. Si se desea una rápida convergencia en el cálculo de los nuevos parámetros cuando hay una variación brusca pero espaciada de los parámetros, la demora en la convergencia se puede resolver con un "reset" de la matriz de covarianzas a un valor αI, con un valor grande de α, para así acelerar la velocidad de convergencia de los nuevos parámetros.

El caso de parámetros que varían lentamente en el tiempo puede ser resuelto con la incorporación del "factor de olvido" en el algoritmo de identificación. El factor de olvido (γ) es un parámetro que permite ponderar en forma exponencial los datos. El valor más reciente tendrá una ponderación de 1 mientras que el n-ésimo valor estará ponderado con γ^n.

Con esto, la función $J(\bar{\Theta}(t))$ se transforma en:

$$J(\bar{\Theta}) = \sum_{k=0}^{N} \gamma^{N-k} (e(k))^2$$

El algoritmo general será el mismo solo que con las modificaciones siguientes en el cálculo de las matrices P(t) y L(t).

Por tanto, ahora se tendrá:

$$\bar{\theta}(t+1) = \bar{\Theta}(t) - L(t+1)\epsilon(t+1)$$

donde,

$$L(t+1) = \frac{P(t)x^T(t+1)}{\gamma + x(t+1)P(t-1)x^T(t+1)}$$

$$P(t) = \left[I - \frac{P(t-1)x^T(t+1)x(t+1)}{\gamma + x(t+1)P(t-1)x^T(t+1)} \right] \frac{1}{\gamma} P(t-1)$$

$$\epsilon(t+1) = y(t+1) - x(t+1)\bar{\theta}(t)$$

Como se había mencionado anteriormente, este algoritmo funciona bien cuando existe excitación persistente y cuando

el sistema es identificable. Estas características se convierten entonces en las limitaciones fundamentales para la convergencia adecuada del método de mínimos cuadrados determinístico. De todas maneras, para los sistemas de tipo determinístico más utilizados, tomando en cuenta las restricciones sobre la señal de entrada (excitación persistente), este algoritmo permite encontrar el modelo de la planta estudiada.

2.3. CRITERIOS ESTADÍSTICOS PARA LA IDENTIFICACIÓN PARAMETRICA.

Anteriormente se estudió el método de mínimos cuadrados determinístico, para la identificación de parámetros en una planta dada. Allí se suponía que las señales provenientes de la planta no tenían ninguna perturbación: por eso se llamaba "modelo determinístico". En los casos prácticos casi siempre se tienen perturbaciones de diferente tipo, muchas de las cuales son de tipo aleatorio. Para realizar el estudio de la identificación de sistemas en presencia de este tipo de perturbaciones, es necesario previamente hacer una rápida revisión de algunos conceptos estadísticos.

a) Procesos estocásticos.

Se dice que un fenómeno estadístico que evoluciona en el tiempo de acuerdo a las leyes de probabilidad es un "proceso estocástico" (BOX 1976: 25). El proceso estocástico es un modelo probabilístico cuyas realizaciones son similares a los datos observados.

b) Ruido blanco.

El Ruido blanco es una secuencia de variables aleatorias e_i cuya distribución es normal, cuya media es cero y cuya varianza σ^2 está dada. (BOX 1976: 8) El ruido blanco es por tanto un caso particular de proceso estocástico.

c) Estadística matricial.

* Esperanza: Se define al operador $\mathcal{E}$ tal que,

$$\mathcal{E}(Z) = \left[\left(E(Z_{ij}) \right) \right]$$

donde $Z = (Z_{ij})$ es una matriz formada por un conjunto de variables aleatorias Z_{ij}.

* Propiedades del operador Esperanza:

$$\mathcal{E}(AZB + C) = A\mathcal{E}(Z)B + C$$

* Covarianza: Se define al operador C tal que,

$$C(X, Y) = \left[\left(\text{cov}(X_i, Y_i) \right) \right]$$

siendo X y Y vectores de variables aleatorias y

$$\text{cov}(X_i, Y_j) = E\left\{ (X_i - E(X_i))(Y_j - E(Y_j)) \right\}$$

* Dispersión: Se define al operador D tal que,

$$D(X) = C(X, X) = \left[\left(\text{cov}(X_i, X_j) \right) \right]$$

si i=j, $\text{cov}(X_i, X_i) = \text{var}(X_i)$

* Propiedades del operador Covarianza y Dispersión:

$$C(AX, BY) = AC(X, Y)B^T$$
$$D(AX) = C(AX, AX) = AD(X)A^T$$
$$D(X) = C(X, X) = \mathcal{E}\left\{ (X - \mathcal{E}(X))(X - \mathcal{E}(X))^T \right\}$$

d) Estimación estadística.

La estimación estadística busca la determinación de los parámetros de un cierto modelo estocástico luego de que, en base de los datos experimentales, se ha escogido el tipo de modelo que se ajusta al proceso. Existen varias características de los métodos de estimación estadística que requieren ser conocidas al realizarse la estimación de parámetros. Estas características permiten saber cuán "buena" es la estimación paramétrica y cuales son sus limitaciones.

Estas características son:

* Consistencia.

Se dice que un estimador es consistente si el valor estimado converge en probabilidad al valor a estimarse cuando la dimensión de la muestra aumenta indefinidamente.

En forma matricial, se tendrá:

$$\lim_{t \to \infty} D\left(\tilde{\boldsymbol{\theta}}(t)\right) = 0$$

* Desviación.

La desviación es una medida de la separación existente entre el valor estimado y el valor real. Se dirá que un estimador

es no desviado si esta diferencia es cero para todo valor de t, es decir si:

$$\mathcal{E}\left(\hat{\theta}(t) - \theta\right) = 0$$

* Mejor Estimador Lineal No Desviado (BLUE - Best Linear Unbiased Estimator)

No basta con conocer que el estimador es consistente y que es no desviado para asegurarse de que este estimador sea el mejor posible. Justamente, es necesario demostrar que el estimador escogido es realmente el mejor de los posibles estimadores lineales. Para esto se escoge el criterio de minimización del error medio cuadrático de los parámetros.

2.4. MÍNIMOS CUADRADOS ESTOCASTICO.

A diferencia del modelo determinístico, en un modelo ARMA estocástico, aparece una perturbación en la ecuación de salida. El modelo se presenta entonces como:

$$y(t) = x(t)\theta(t) + v(t)$$

siendo $x^T(t)$, $\theta(t)$ los vectores antes definidos para el modelo determinístico y $v(t)$ una perturbación en forma de ruido blanco.

Si se forma la matriz Y(t) antes mencionada, en base a t+1 valores de y(t), la ecuación anterior será:

$$Y(t) = X(t)\Theta(t) + V(t)$$

donde V(t) es un vector dado por:

$$V(t) = [v(0)\ \ v(1)\ \ v(2)\ \ \ldots\ \ v(t)]^T$$

Al ser v(t) un proceso estocástico en forma de ruido blanco, se está asumiendo las siguientes propiedades para estas variables aleatorias:

$$E\{v(t)\} = 0,$$
$$E\{v(i)v(j)\} = 0, \quad si\ i \neq j$$
$$= \sigma^2, \quad si\ i = j$$

es decir la media es cero y que no existe correlación entre los diferentes valores de v(t); σ^2 es la varianza del ruido blanco.

En forma matricial, esto significa:

$$D(VV^T) = \sigma^2 I$$

Estas suposiciones son fundamentales para la estimación de parámetros según el método de mínimos cuadrados estocástico.

Aquí el problema planteado es similar al anterior: se busca la estimación de los parámetros del modelo, ahora con la presencia de perturbaciones en la salida. Utilizando las

fórmulas ya desarrolladas para los mínimos cuadrados determinístico, se tendrá:

$$e(t) = Y(t) - X(t)\overline{\Theta}(t)$$

En este caso, el error e(t) dependerá tanto del valor de $\overline{\Theta}(t)$ cuanto del ruido aleatorio.

Entonces, aplicando el método de mínimos cuadrados, se tendrá:

$$J = (Y - X\overline{\Theta})^T(Y - X\overline{\Theta})$$

de donde, usando los resultados anteriores,

$$\overline{\Theta} = (X^T X)^{-1} X^T Y$$

Aparentemente el problema ha sido resuelto pues basta realizar este cálculo sea directamente con la fórmula dada, sea a través del método recursivo: con esto se habría encontrado $\overline{\Theta}$ como el conjunto de parámetros buscados. En el método determinístico, así sería: $\overline{\Theta}$ converge efectivamente a Θ después de un cierto número de iteraciones. Pero ahora, dada la presencia de perturbaciones, nada asegura que realmente esto suceda. La solución será una variable aleatoria pues la información en la que se basa a su vez es aleatoria. Es por tanto necesario aplicar los conceptos estadísticos enunciados anteriormente para determinar si el estimador $\overline{\Theta}(t)$ realmente es válido para la solución de este problema.

a) Consistencia de $\overline{\Theta}$.

Por la definición dada en 2.3 c) y las propiedades del operador D, se tendrá que $\overline{\Theta}$ es un estimador consistente si:

$$\lim_{t\to\infty} \mathcal{E}(\overline{\Theta}(t) - \Theta)(\overline{\Theta}(t) - \Theta)^T = 0$$

Aquí:

$$\begin{aligned}\overline{\Theta} - \Theta &= (X^T X)^{-1} X^T Y - \Theta \\ &= (X^T X)^{-1} X^T (X\Theta + V) - \Theta \\ &= \Theta + (X^T X)^{-1} X^T V - \Theta \\ &= (X^T X)^{-1} X^T V\end{aligned}$$

De donde,

$$\begin{aligned}\mathcal{E}(\overline{\Theta} - \Theta)(\overline{\Theta} - \Theta)^T &= \mathcal{E}\left\{(X^T X)^{-1} X^T V \left((X^T X)^{-1} X^T V\right)^T\right\} \\ &= \mathcal{E}\left\{(X^T X)^{-1} X^T V V^T X (X^T X)^{-1}\right\} \\ &= (X^T X)^{-1} X^T \mathcal{E}\{VV^T\} X (X^T X)^{-1} \\ &= (X^T X)^{-1} X^T \sigma^2 X (X^T X)^{-1} \\ &= \sigma^2 (X^T X)^{-1}\end{aligned}$$

dado que $\mathcal{E}\{VV^T\} = \sigma^2$

Basta por tanto demostrar que

$$\lim_{t\to\infty} \sigma^2 (X^T X)^{-1} = 0$$

Dado que la excitación es persistente, se demuestra que este límite tiende efectivamente a cero, dado que la matriz $(X^T X)^{-1}$ tiende a hacerse cada vez más cercana a cero (ASTROM 1989). Por tanto es posible decir que el estimador $\overline{\Theta}$ es consistente.

b) Desviación de $\overline{\Theta}$.

La desviación se mide a través de la verificación de la siguiente propiedad (2.3. c)):

$$\mathcal{E}\{\overline{\Theta} - \Theta\} = 0$$

Realizando los cálculos:

$$\mathcal{E}\{\overline{\Theta} - \Theta\} = \mathcal{E}\{(X^T X)^{-1} X^T V\} = (X^T X)^{-1} \mathcal{E}\{V\} = 0$$

c) BLUE.

Para mostrar que $\overline{\Theta}$ es el mejor estimador lineal no desviado es necesario mostrar que: a) el estimador es lineal, es decir una función lineal de Y; b) el estimador es el mejor de todos, es decir que si se escoje otro estimador lineal, $\overline{\Theta}$ resultará mejor que el escogido. En base a estos criterios, se muestra

que efectivamente $\bar{\theta}$ es el mejor estimador lineal no desviado (BLUE)[1].

En base a lo anteriormente demostrado, resulta que el estimador $\bar{\theta}$ escogido es consistente, no desviado y es además el mejor estimador lineal no desviado. Esto permite decir por tanto que $\bar{\theta}$ es un estimador adecuado de θ y que la identificación paramétrica en base al algoritmo de mínimos cuadrados da lugar al valor buscado de los parámetros del modelo.

Existen estudios pormenorizados sobre estas propiedades del estimador $\bar{\theta}$, pero con el algoritmo de mínimos cuadrados recursivo. Se demuestra que también en este caso el estimador es consistente, no desviado y es además es el BLUE (GOODWIN 1984). El algoritmo desarrollado anteriormente, por tanto, permite la identificación de los parámetros de la planta, incluso en el caso de un sistema estocástico, con la presencia de ruido.

El estudio de los mínimos cuadrados estocástico se ha limitado a una señal de ruido con las características de ruido blanco, por tanto de media cero y no correlacionado. Si el ruido o la perturbación aparece ahora correlacionada, es decir que

---

[1]) Para la demostración completa, ver FRANKLIN, pp. 228-231.

$$cov[e(t)e(t-i)] \neq 0 \qquad para\ i = 0,1,2,...$$

se demuestra que ya no hay convergencia en los mínimos cuadrados estocástico por lo que este método es inviable para la identificación. En este caso se utilizan otros algoritmos como el de mínimos cuadrados generalizado o el de "Maximum Likelihood", cuyo estudio va más allá del propósito de este trabajo (ASTROM 1989).

Existe por tanto una seria limitación para la identificación mediante el método de mínimos cuadrados estocástico: la existencia de ruido correlacionado. Para simplificar el problema y suponiendo que efectivamente el ruido es blanco en la mayoría de las aplicaciones, este será el método utilizado.

Como conclusión final es posible decir que el método de mínimos cuadrados estocástico puede ser utilizado siempre y cuando la excitación sea persistente y no exista correlación en la perturbación, lo cual, en la mayoría de los casos reales se cumple con buena aproximación.

2.5. EXCITACIÓN PERSISTENTE.

Una de las condiciones principales para que un sistema sea identificable, es decir que se puedan estimar sus parámetros es que la excitación sea persistente. El método de mínimos

cuadrados no convergerá si la señal de entrada a la planta no es de excitación persistente. Este concepto puede entenderse inicialmente como la señal que requiere la planta para que tenga una respuesta que presente sus características fundamentales. Esto hace pensar que la señal u(t) de entrada no puede ser constante, sino que debe variar en el tiempo para lograr una respuesta adecuada de la planta. A continuación se dará la definición de la excitación persistente y se estudiarán algunos ejemplos (ASTROM 1989).

a) Definición.

Una señal u es llamada de excitación persistente de orden n si la matriz C_n es definida positiva, siendo esta matriz,

$$C_n = \begin{pmatrix} c(0) & c(1) & \dots & c(n-1) \\ c(1) & c(0) & \dots & c(n-2) \\ \vdots & \vdots & \dots & \vdots \\ c(n-1) & c(n-2) & \dots & c(0) \end{pmatrix}$$

donde,

$$c(k) = \lim_{t \to \infty} \frac{1}{t} \sum_{i=1}^{t} u(i)u(i-k)$$

b) Teorema

Una señal u es de excitación persistente de orden n si para todos los polinomios A, de grado n-1 o menos,

$$\lim_{t\to\infty} \frac{1}{t}\left(\sum_{k=1}^{t} A(q)u(k)\right)^2 > 0$$

Con esta información es posible entonces estudiar algunas señales y determinar si son o no de excitación persistente.

Al realizar el estudio respectivo, se obtienen los resultados siguientes[2].

1) Impulso: el impulso no es de excitación persistente para ningún valor de n.

2) Función paso: la función paso es de excitación persistente de orden 1.

3) Función senoidal: la función senoidal es de excitación persistente de orden 2.

4) Función periódica: si u(t) es de periodo n, entonces será de excitación persistente de orden n.

5) Señal aleatoria: si u(t) es una señal aleatoria, es decir un proceso estocástico no predecible, la señal será de excitación persistente de cualquier orden.

---

[2]) Para la demostración completa ver ASTROM 1989: 80-81.

La señal de entrada u(t) puede adoptar muchas formas. Para el caso que aquí interesa, se requiere que u(t) sea de excitación permanente de manera que el algoritmo de mínimos cuadrados sea convergente. Por la informción anterior, esto se consigue, en el mejor de los casos, cuando u(t) es una señal aleatoria. Allí no se tiene una dependencia con el orden de la excitación persistente, por lo que es la mejor opción. Para los casos prácticos, bastará por tanto añadir una señal de este tipo a la entrada del sistema y así asegurar la convergencia de los mínimos cuadrados.

En la práctica, esto será resuelto mediante la generación de una señal aleatoria que se añadirá a la señal u(t) requerida de manera que se tenga la excitación permanente. La señal aleatoria añadida deberá ser equivalente a un ruido blanco, es decir de media cero, para no afectar la respuesta media del sistema, es decir su respuesta final.

3. CONTROL EN TIEMPO REAL

3.1. ESQUEMAS DE CONTROL

3.2. ADQUISICION DE DATOS Y CONTROL

3.3. IDENTIFICACION Y CONTROL EN TIEMPO REAL

3.4. SISTEMA DE CONTROL EN TIEMPO REAL

3. CONTROL EN TIEMPO REAL

3.1. ESQUEMAS DE CONTROL.

Habiendo ya definido la metodología para realizar la identificación de parámetros de una planta dada, es necesario ahora pasar al estudio de los esquemas generales de control de manera que sea posible la realización de la identificación (y posteriormente el control) en tiempo real.

a) Control Digital Directo.

Como fue enunciado en el Capítulo 1, el método de control que se usará en este trabajo es el control digital directo, esto es la inserción de un computador en el lazo, para la generación de la señal de control, fruto de una ley especificada como un algoritmo de control. El objetivo de esto es el control de una planta real.

Como se puede ver en la figura 3.1, además del computador en línea es necesario establecer mecanismos de transformación de la señal analógica en digital, a la entrada y viceversa a la salida. Esto se hace mediante un equipo de adquisición de datos controlado a su vez por el computador. La figura muestra un sistema realimentado basado en una planta real y un sistema

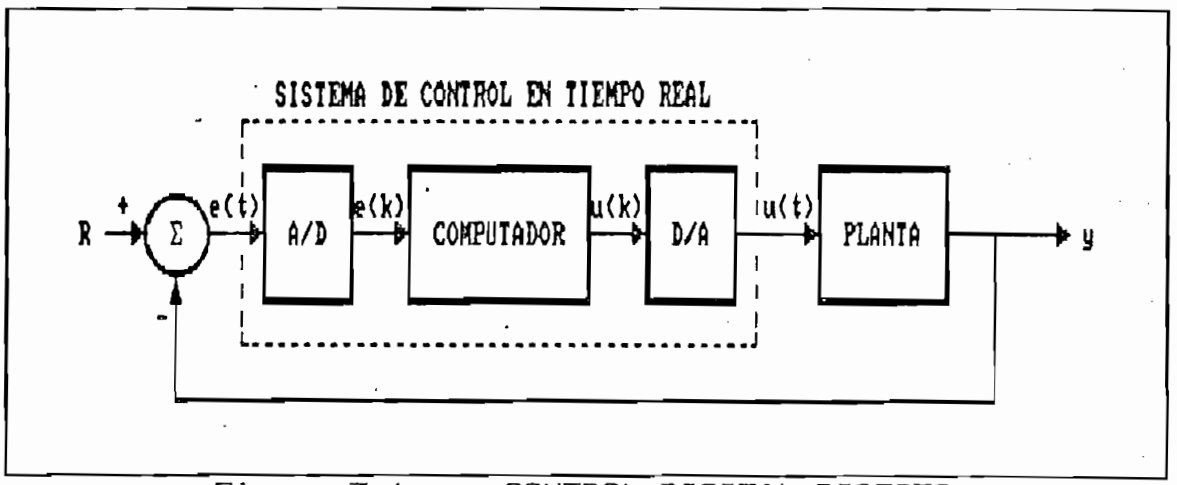


Figura 3.1 - CONTROL DIGITAL DIRECTO

de control en tiempo real.

b) Diseño del Control Digital.

La parte central de un sistema de control es el diseño del controlador. Para el diseño del controlador digital existen varias técnicas disponibles. La figura 3.2 muestra algunos de los caminos posibles, para un control digital en tiempo real.

1. Modelación y diseño del controlador en tiempo continuo de un sistema discreto, discretizando los resultados del control análogo.

2. Modelación en tiempo continuo y obtención de la función de transferencia G(s), la que es discretizada y para la cual se diseña un controlador digital en base a la información del modelo.

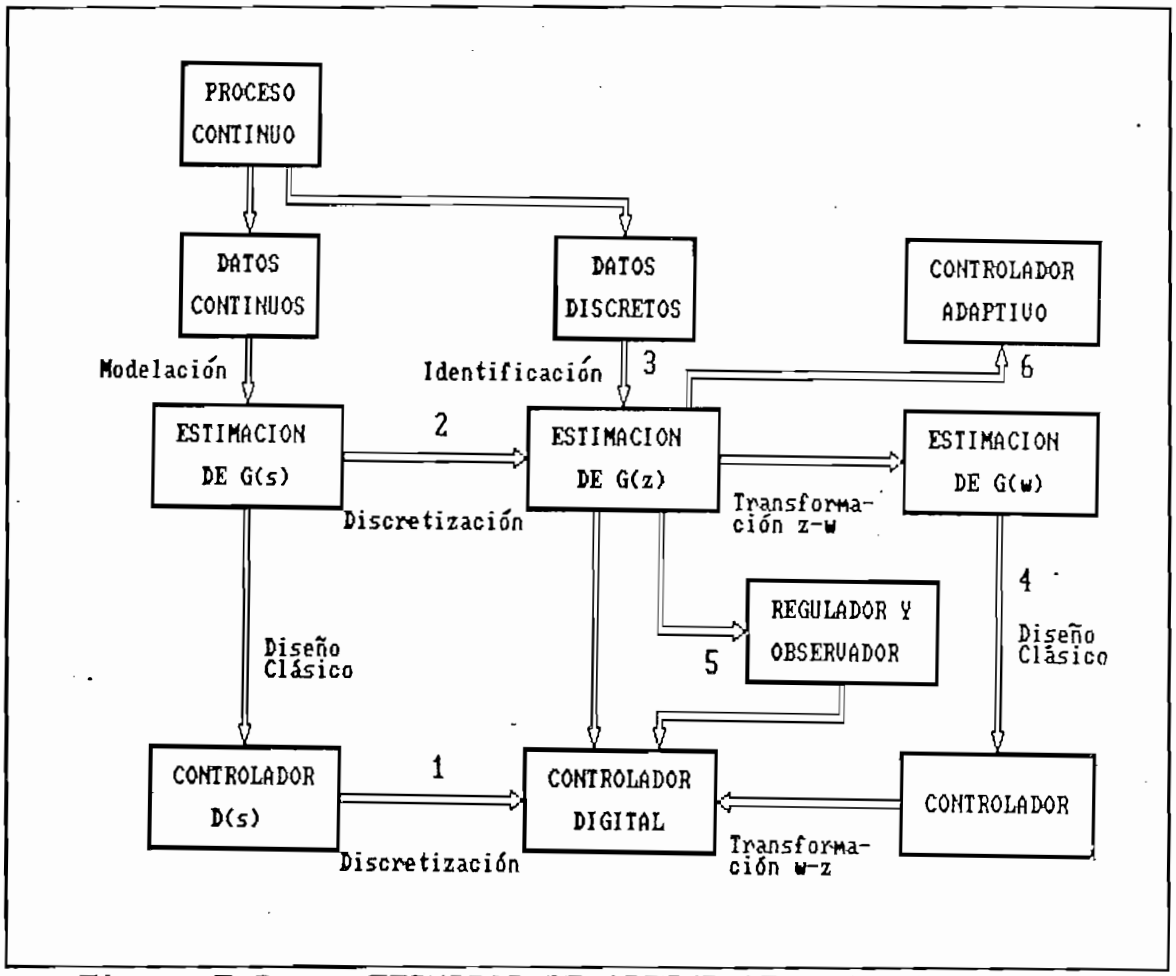


Figura 3.2 - TECNICAS DE DISEÑO DE CONTROL DIGITAL

3. Modelación y diseño del controlador en tiempo discreto. La modelación se realiza mediante el uso de los datos muestrados obtenidos por el sistema.

4. Para el diseño de controlador, no se usa directamente el modelo, sino se lo hace en el plano w' para permitir el uso de los diagramas de Bode, es decir los métodos de diseño clásico.

5. Diseño del observador de estado y el regulador mediante las técnicas de espacio de estado.

6. Diseño del controlador en base a métodos de control adaptivo (control adaptivo directo e indirecto).

La técnica que aquí interesa es aquella que liga la modelación y diseño del controlador en forma discreta, con los métodos de control adaptivo. Este trabajo permite esta ligazón por la capacidad del algoritmo de identificación de estimar los parámetros de la planta en tiempo real y por tanto poder conocer el modelo incluso de una planta cuyos parámetros no sean completamente estables en el tiempo.

c) Esquemas de control adaptivo.

Si bien el objetivo de este trabajo no es la profundización sobre los métodos de control adaptivo, sino la identificación de sistemas en tiempo real, es necesario conocer los métodos básicos de control adaptivo, pues la perspectiva es realizar posteriormente control en tiempo real para varios sistemas. Los esquemas básicos de control adaptivo son los siguientes:

* Control robusto de alta ganancia.
* Sistema adaptivo auto oscilante.
* Gain Scheduling
* Model-Reference Adaptive System.
* Self-Tuning Regulator.

3.2. ADQUISICION DE DATOS Y CONTROL.

Lo que caracteriza a un sistema de control digital en tiempo real es la transformación de la información analógica en digital para su posterior procesamiento en el computador y la reconstrucción de la señal analógica de control en base a la que se obtiene del computador. La primera función es la adquisición de datos, y la segunda es el control. La precisión y versatilidad del equipo de adquisición de datos y control permitirá que el control diseñado sea adecuado o no para la planta en consideración.

En este trabajo se utilizará un equipo de adquisición de datos KEITHLEY Serie 500, conectado a un computador IBM PS/60 con coprocesador matemático.

El equipo Keithley modelo 500A utilizado es un sistema de adquisición de datos y control que contiene un "motherboard" de 10 "slots" para 10 diferentes tarjetas de adquisición de datos y control; una fuente de poder; y un interface lógico para la conexión al computador.

Este modelo está instalado con 3 tarjetas de diferente tipo:

1) El módulo AMM1 es una tarjeta para medir señales analógicas, combinando funciones de acondicionamiento de señal con las de conversión Analógica/Digital. El acondicionamiento de señal se realiza mediante un amplificador de ganancia

programable. La conversión A/D tiene una resolución de 12 bits con rangos de entrada variables y un tiempo máximo de conversión de 25 µs. Este módulo consta de 8 entradas analógicas. Además tiene la característica de que todas las entradas analógicas que se conecten al sistema 500 pasan por este módulo. La frecuencia máxima a la que puede trabajar este módulo es de 32 kHz.

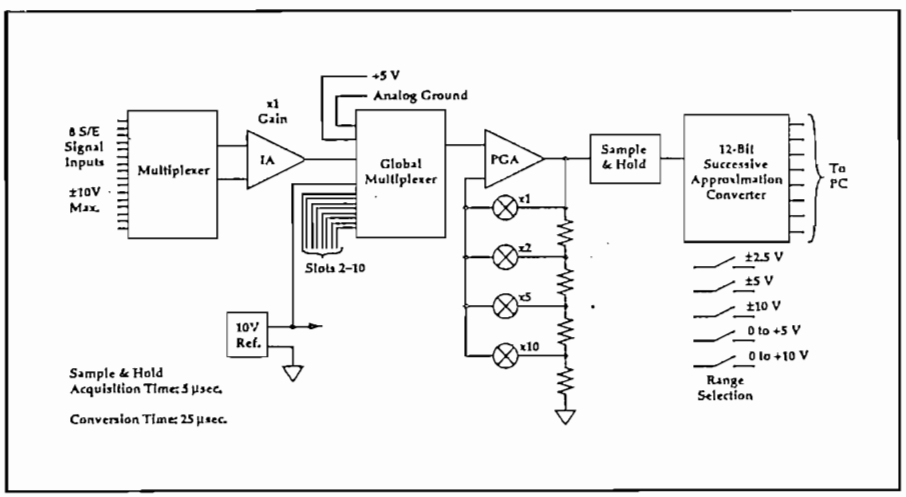


Figura 3.3 - DIAGRAMA DEL MODULO AMM1

2) El módulo AOM1/5 es una tarjeta de 2 salidas analógicas con una resolución de 12 bits. Cada canal tiene un convertidor D/A independiente con rangos variables de salida (±2.5 V, ±5 V, ±10 V, 0 a +5 V , y 0 a + 10 V.

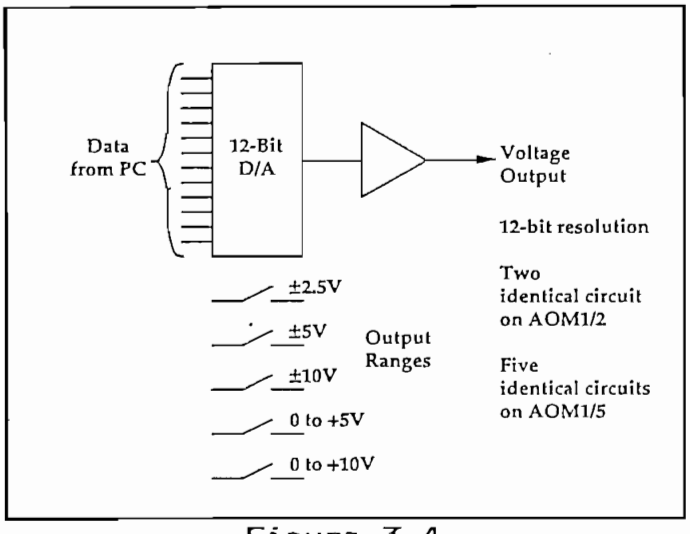


Figura 3.4
DIAGRAMA DEL MODULO AOM1/5

3) El módulo DOM1 contiene 16 canales digitales de salida, aislados ópticamente, cuyo acceso se puede realizar tanto individualmente como a través de dos pórticos de 8 bits cada uno. Las salidas pueden ser configuradas como compatibles a TTL o en un rango de hasta +28 V.

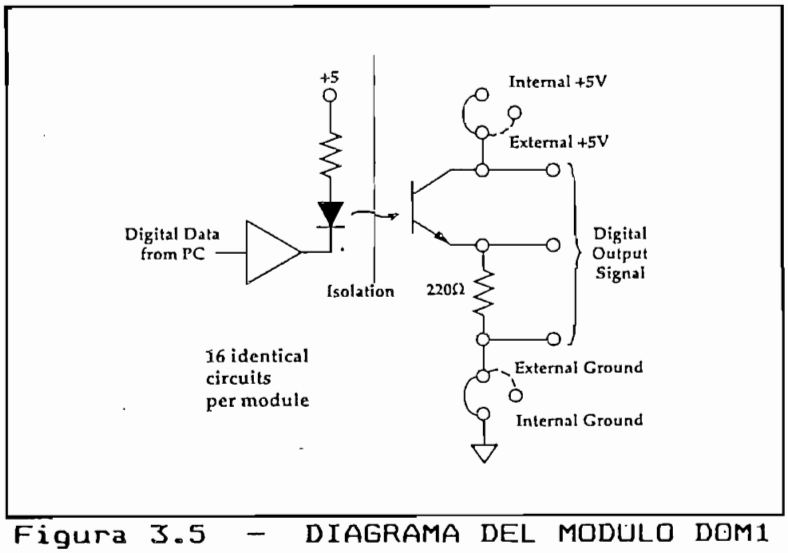


Figura 3.5 - DIAGRAMA DEL MODULO DOM1

Además de estos módulos es posible conectar otros para diversas aplicaciones de control:

a) Entradas analógicas: Transductores de inductancia, Termocuplas, Transductores mecánicos, Sensores de temperatura, etc...

b) Conversión análoga-digital: 16 bits.

c) Salida Analógica: 16 bits, Salida de corriente, Salida de voltage programable.

d) Entrada/Salida Digital: 32 canales Entrada/Salida, 16 canales de salida aislados, 16 canales de entrada aislados.

e) Acelerador de gráficos.

f) Controlador IEEE-488.

g) Contadores de Pulsos: Entrada de Frecuencia, Contador de eventos.

h) Control de Potencia: AC, AC/DC.

i) Probador de prototipos y módulo de extensión.

j) Motores de pasos.

Existe una amplia gama de programas (software) para la adquisición de datos y control. Sus características se pueden medir en base a tres factores: la facilidad de uso, la adaptibilidad y la velocidad de procesamiento. Mientras más fácil de usar será el software (paquetes de programas), éste será más rígido y más lento. Por el contrario, las mayores velocidades y la mayor flexibilidad se consiguen con comandos muy difíciles de usar (lenguage máquina).

Dado que generalmente no se requiere una excesiva flexibilidad pero sí una razonable velocidad de procesamiento y una facilidad de uso adecuada, el equipo viene con una serie de rutinas preestablecidas en BASIC: es el sistema Soft500 de Keithley. Este tipo de BASIC es tan sólo intérprete, por lo que se imposibilita la compilación de los programas y se limita su velocidad de procesamiento.

Este problema ha sido resuelto con el sistema Quick500, que es una ampliación del Soft500 al lenguage de programación Quick BASIC de la Microsoft. La diferencia fundamental entre el Quick BASIC y el BASIC intérprete es que el primero puede ser compilable y además es un lenguage de programación estructurado con la posibiliddad de usar programación modular.

Además de su facilidad de programación, tanto el Soft500 cuanto el Quick500 pueden trabajar en la doble modalidad de foreground/background. El modo foreground es el más fácil de usar: se trata de realizar las operaciones en tiempo real

dentro de los límites impuestos por el compilador y como parte de los comandos en BASIC. El modo background permite la realización de las tareas de adquisición de datos y control como interrupciones de la secuencia que se está ejecutando en BASIC, es decir en el foreground. La utilización simultánea de estos modos permite el aprovechamiento de la capacidad de procesamiento del BASIC junto al control cuidadoso del proceso de muestreo, que se realiza en background. La figura 3.6 muestra cómo mientras se ejecuta una tarea C en foreground (por ejemplo la realización de gráficos en tiempo real), las tareas A y B, de adquisición de datos se realizan en foreground y a intervalos de tiempo predefinidos por el tiempo de muestreo requerido.

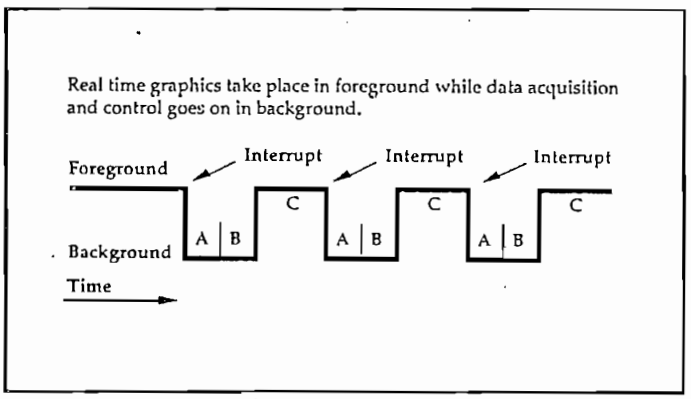


Figura 3.6
FOREGROUND Y BACKGROUND

Para este trabajo, se utilizará el equipo Keithley 500A como equipo de adquisición de datos y control mediante los módulos AMM1 de entrada analógica y AOM1 de salida analógica y las rutinas elaboradas en Quick BASIC, contenidas en el sistema

Quick500, para el manejo de los datos. La programación se realizará en los modos foreground y background para usar al máximo el potencial del equipo.

3.3. IDENTIFICACION Y CONTROL EN TIEMPO REAL.

El proceso de identificación en tiempo real consiste fundamentalmente en la aplicación de los algoritmos de identificación antes desarrollados conjuntamente con las rutinas de entrada de datos del Keithley 500A. De esta manera, los datos medidos son los datos efectivamente utilizados por el algoritmo para permitir la determinación de los parámetros de la planta mediante el método de los mínimos cuadrados. El proceso se considerará aquí de tipo estocástico por las perturbaciones que se presentan en la medición de las señales de entrada y de salida. Estas perturbaciones serán consideradas como un ruido blanco (señal aleatoria no predecible) y de esta forma se podrán aplicar los resultados del capítulo anterior en el caso de un modelo estocástico. La figura 3.7 muestra la estructura básica de la identificación paramétrica en tiempo real. Allí es posible visualizar el muestreo de la entrada y de la salida de la planta, y la utilización de esa información para la determinación de los parámetros de la planta.

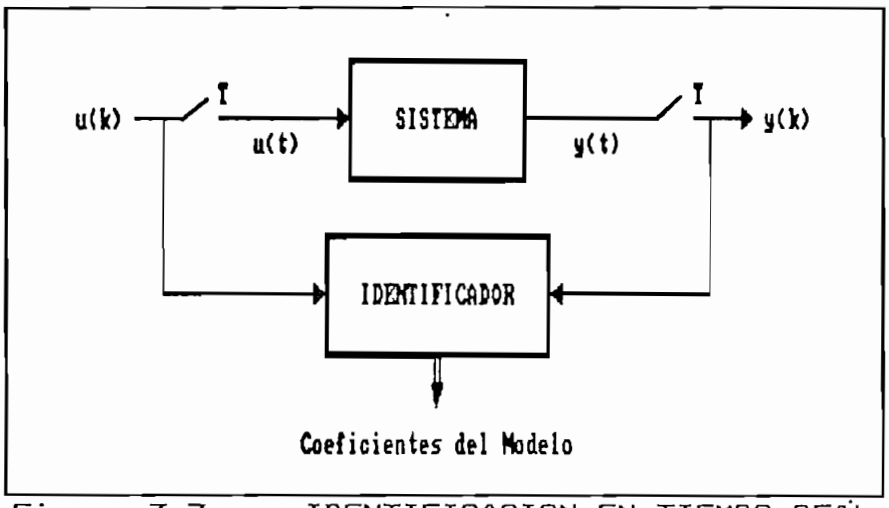


Figura 3.7 - IDENTIFICACION EN TIEMPO REAL

Si se desea cerrar el lazo, es decir además de identificación realizar control realimentado, es necesario desarrollar una ley de control que realice el ajuste permanente de la señal de entrada (o de control) para la obtención de la señal de salida deseada. Las leyes de control dependen generalmente de los parámetros calculados de la planta. En este caso, la ley de control utilizará los parámetros de la planta en ajuste permanente, por lo que tendrá la capacidad de adaptarse a las variaciones de los parámetros de la planta: este es el principio básico del control adaptivo.

La ley de control escogida para este trabajo se basa en el principio de asignación de polos, para la función de transferencia en lazo cerrado. La particularidad de esta ley es que se fijan a priori los polos de esta función de transferencia, con lo cual queda asegurada la estabilidad y una respuesta transitoria adecuada. La ley de control se encarga entonces de fijar la señal de entrada a la planta para

que se cumplan estos requisitos. Este tipo de ley de control tiene la ventaja de poder ser descrita mediante un algoritmo, además de poder ser fácilmente implementada para un control adaptivo.

Sea una planta dada por su función de transferencia:

$$F(z) = \frac{B(z)}{1+A(z)}$$

donde:

$$A(z) = a_1 z^{-1} + a_2 z^{-2} + \ldots + a_{n_a} z^{-n_a}$$
$$B(z) = b_1 z^{-1} + b_2 z^{-2} + \ldots + b_{n_b} z^{-n_b}$$

Entonces la ecuación de salida será:

$$y(t) = \frac{B(z)}{1+A(z)} u(t) + \frac{1}{1+A(z)} \epsilon(t)$$

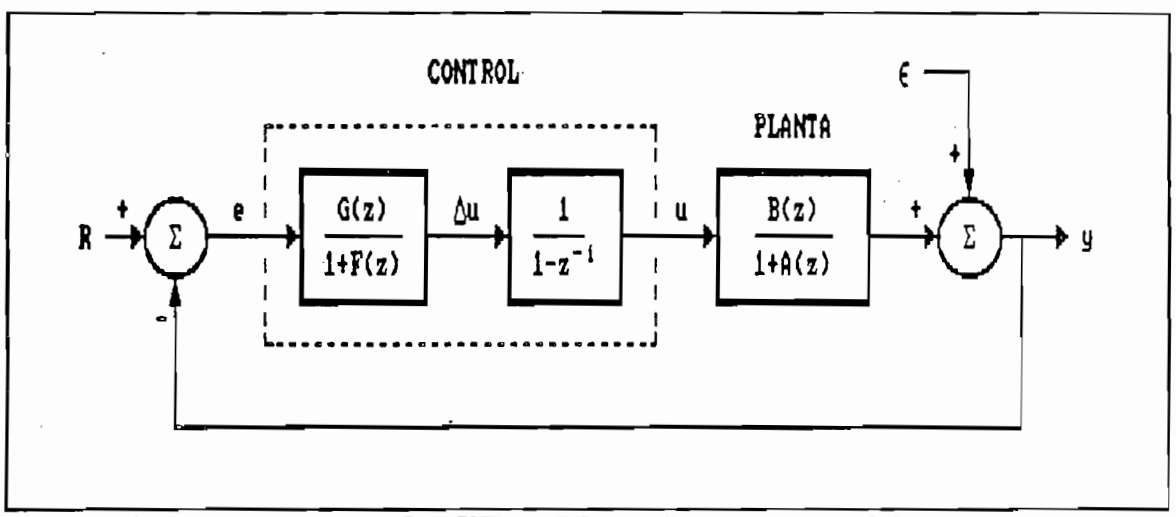


Figura 3.8 – SISTEMA DE CONTROL EN LAZO CERRADO

En un sistema de lazo cerrado, se aplica una referencia r(t) al sistema, con la cual se compara la señal de salida y(t) para dar lugar a e(t), el error sobre el cual actúa el algoritmo de control. Para asegurar que y(t) = r(t) en estado estable, se añade un integrador digital al control de la forma:

$$\Delta u(t) = u(t) - u(t-1)$$

es decir que:

$$u(t) = \frac{1}{1 - z^{-1}} \Delta u(t)$$

La ecuación de salida será entonces:

$$y(t) = \frac{B(z)}{1+A(z)} \cdot \frac{1}{1 - z^{-1}} \Delta u(t) + \frac{1}{1+A(z)} \epsilon(t) \qquad (3.1)$$

Entonces es posible definir la ley de control como:

$$\Delta u(t) = \{r(t) - y(t)\} \frac{G(z)}{1+F(z)} \qquad (3.2)$$

donde:

$$A(z) = g_0 + g_1 z^{-1} + g_2 z^{-2} + \ldots + g_{n_g} z^{-n_g}$$
$$B(z) = f_1 z^{-1} + f_2 z^{-2} + \ldots + f_{n_f} z^{-n_f}$$

Para que exista una solución única en el proceso de diseño del control, es necesario que: $n_g = n_a - 1$ y $n_f = n_b - 1$.

Para simplificar la ecuación (3.1), se calcula,

$$\{1+A(z)\}\{1 - z^{-1}\} = 1 - z^{-1} + A(z) - z^{-1}A(z) = 1+\bar{A}(z)$$

siendo:

$$\bar{A}(z) = \alpha_1 z^{-1} + \alpha_2 z^{-2} + \ldots + \alpha_{n_\alpha} z^{-n_\alpha}$$

donde,

$$\begin{aligned}
\alpha_1 &= a_1 - 1 \\
\alpha_2 &= a_2 - a_1 \\
&\vdots \\
\alpha_{n_\alpha} &= a_{n_a} - a_{n_a-1} \\
\alpha_{n_\alpha+1} &= -a_{n_a}
\end{aligned}$$

Evidentemente, se tendrá: $n_\alpha = n_a + 1$.

La ecuación de salida es entonces:

$$y(t) = \frac{B(z)}{1+\bar{A}(z)}\Delta u(t) + \frac{1}{1+\bar{A}(z)}\varepsilon(t) \qquad (3.3)$$

Si se ingresa en esta ecuación la ley de control dada por la ecuación (3.2), y si se desprecia el ruido, se tendrá:

$$y(t) = \frac{B(z)}{1+\overline{A}(z)} \cdot \frac{G(z)}{1+F(z)}\{r(t) - y(t)\}$$

de donde,

$$\left[1+\overline{A}(z)\right]\left[1+F(z)\right]y(t) = B(z)G(z)r(t) - B(z)G(z)y(t)$$

entonces, simplificando la escritura,

$$\left[(1+\overline{A})(1+F) + BG\right]y(t) = BGr(t)$$

La ley de control basada en la ubicación de polos requiere una definición de los polos de la función de transferencia en lazo cerrado. Estos corresponden a un polinomio definido en el plano z, que aquí se llamará T(z). Por tanto, la ecuación a ser resuelta será la siguiente, donde las incógnitas están dadas por los coeficientes de G(z) y de F(z).

$$(1+\overline{A})(1+F) + BG = 1 + T$$

Si a esta ecuación se la desarrolla, se obtiene:

$$(1+\overline{A})F + BG = 1 + T - \overline{A} - 1.$$
$$(1+\overline{A})F + BG = T - \overline{A}$$

Esta última ecuación se puede representar en forma matricial de la manera siguiente:

$$\left(\begin{array}{ccc|ccc} 1 & 0 & \cdots & b_1 & 0 & \cdots \\ \alpha_1 & 1 & \cdots & b_2 & b_1 & \cdots \\ \alpha_2 & \alpha_1 & \cdots & b_3 & b_2 & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \alpha_{n_\alpha} & \alpha_{n_\alpha-1} & \cdots & b_{n_b} & b_{n_b-1} & \cdots \\ 0 & \alpha_{n_\alpha} & \cdots & 0 & b_{n_b} & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & 0 & 0 & \cdots \end{array}\right)\left(\begin{array}{c} f_1 \\ f_2 \\ \vdots \\ f_{n_f} \\ g_0 \\ g_1 \\ \vdots \\ g_{n_g} \end{array}\right) = \left(\begin{array}{c} t_1 - \alpha_1 \\ t_2 - \alpha_2 \\ \vdots \\ t_{n_t} - \alpha_{n_t} \\ -\alpha_{n_t+1} \\ \vdots \\ 0 \end{array}\right) \quad (3.4)$$

El polinomio T(z) deberá tener un orden: $n_t = n_a + n_b$.

La resolución de este sistema de ecuaciones permite obtener los coeficientes de los polinomios correspondientes a la ley de control. Basta ahora establecer el algoritmo de cálculo mediante el cual será posible programar esta ley de control e incorporarla en el proceso.

La ley de control está dada por la ecuación (3.2). Con los coeficientes de G(z) y F(z) calculados y con los de A(z) y B(z) determinados por la identificación paramétrica, se tiene:

$$u(t) = \frac{1}{1 - z^{-1}} \Delta u(t) = \frac{1}{1 - z^{-1}} \cdot \frac{G(z)}{1+F(z)} \{r(t) - y(t)\}$$

Resolviendo, se obtiene:

$$u(t) = \left\{(1+F)z^{-1} - F\right\}u(t) + G\{r(t) - y(t)\}$$
$$u(t) = u(t-1) + \left(Fz^{-1} - F\right)u(t) + G(t)\{r(t) - y(t)\}$$
$$u(t) = \left(1-f_1\right)u(t-1) + \sum_{i=1}^{n_f}\left(f_{i-1}-f_i\right)u(t-i) +$$
$$+ f_{n_f}u(t-n_f-1) + \sum_{j=0}^{n_g} g_j(r-y(t-j))$$

Es decir en forma matricial,

$$u(t) = CX^T + r.\sum_{i=0}^{n_g} g_i \quad (3.5)$$

donde:

$$C = \left[g_0\ g_1\ \dots\ g_{n_g} \vdots 1-f_1\ f_1-f_2\ f_2-f_3\ \dots\ f_{n_f}\right]$$
$$X^T = \left[-y(t)\ -y(t-1)\ \dots\ -y(t-n_a) \vdots u(t-1)\ u(t-2)\ \dots\ u(t-n_b)\right]$$

Con esta última expresión queda definido el método de cálculo de u(t), señal de entrada para la planta.

El algoritmo será entonces:

1. Calcular los coeficientes g_i y f_i en base a los parámetros a_i y b_i, resolviendo el sistema de ecuaciones dado por la ecuación (3.4).
2. Calcular la señal de control u con la ecuación 3.22.

3.4. SISTEMA DE CONTROL EN TIEMPO REAL

Con el anterior estudio de la ley de control, queda completamente definido el sistema de control en tiempo real. Este se muestra en la figura 3.9.

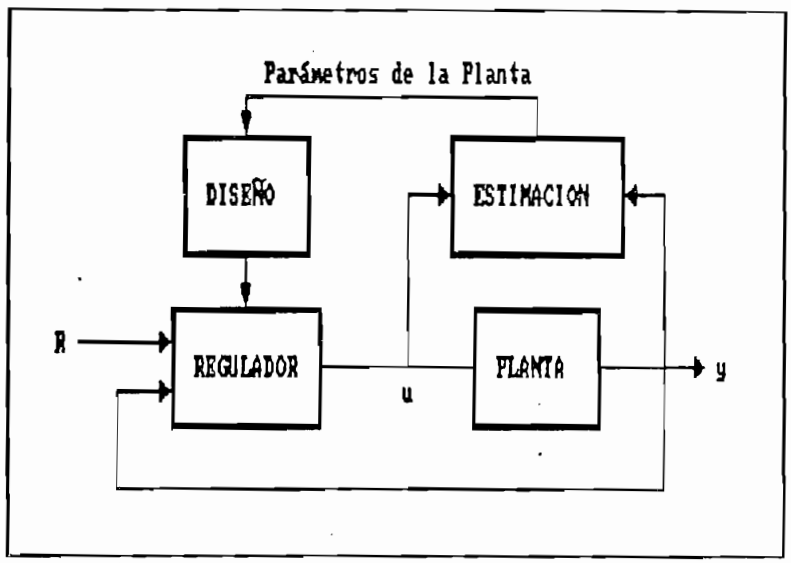


Figura 3.9 - CONTROL ADAPTIVO

El sistema basado en el esquema de control adaptivo tipo self-tuning, consta de:

* Una referencia definida por el usuario;

* Un controlador cuyos parámetros se ajustan a los parámetros de la planta en base a la previa medida de señales de entrada y salida y la posterior estimación de parámetros;

* La planta sobre la cual actúa el control;

* La realimentación que cierra el lazo de control; y los convertidores A/D y D/A para el acondicionamiento de la señal.

El proceso de identificación y cálculo del control en tiempo real se esquematiza entonces con la figura 3.10.

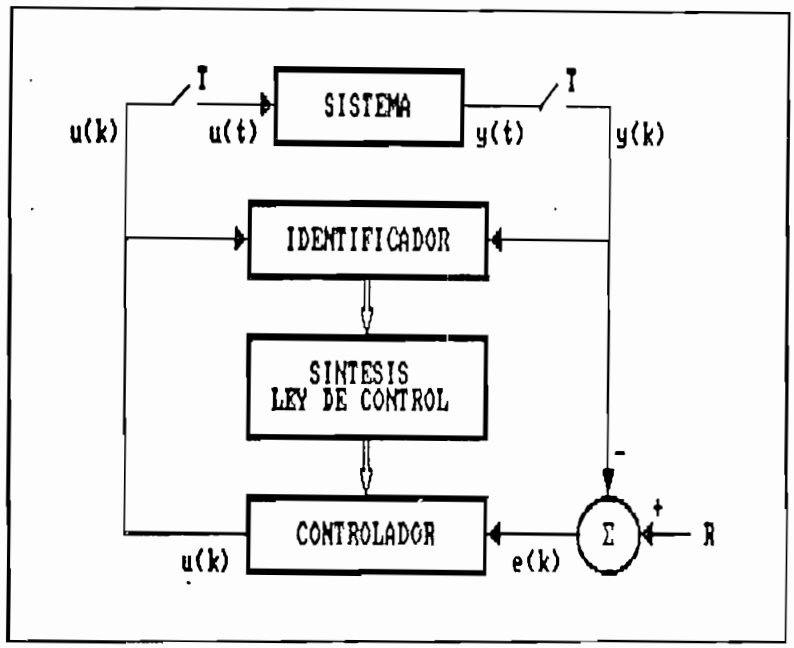


Figura 3.10
SISTEMA DE CONTROL EN TIEMPO REAL

Estando completamente definido el sistema a utilizarse en este trabajo, el capítulo 4 tratará de la elaboración del programa digital propiamente dicho y el capítulo 5 mostrará los resultados obtenidos tanto de la identificación de sistemas cuanto del control en simulación y en tiempo real.

4. IMPLEMENTACION DEL SISTEMA

4.1. ESTRUCTURA DEL SOFWARE DESARROLLADO

4.2. EL MODULO PRINCIPAL: TESISAD.BAS

4.3. SIMULACION

4.4. TIEMPO REAL

4.5. RUTINAS DE GRAFICOS, MENUS Y VENTANAS

4. IMPLEMENTACION DEL SISTEMA

4.1. ESTRUCTURA DEL SOFTWARE DESARROLLADO

El programa elaborado está en capacidad de realizar la identificación de parámetros y el control tanto en simulación cuanto en tiempo real. Incluye las rutinas de mínimos cuadrados recursivo para la identificación así como el diseño y cálculo de la ley de control. Para la simulación es necesario predefinir una planta en base a sus parámetros conocidos y para el tiempo real, la planta física se conecta al computador a través del equipo de adquisición de datos Keithley.

El software está desarrollado enteramente en el QUICK 500, una ampliación del QUICK BASIC 4.5 de la Microsoft para trabajar con las rutinas de adquisición de datos del equipo KEITHLEY 500 A. Se ha buscado respetar en lo posible la forma estructurada del lenguaje QUICK BASIC para facilitar la programación y la utilización por parte del usuario.

El programa consta de un módulo ejecutable TESISAD.EXE, de varios programas pequeños tipo BATCH para el sistema operativo DOS 3.30 (TSGRAF.BAT, COMPIL.BAT), y de una hoja electrónica para LOTUS (TESISAD.WK1). El programa puede ejecutarse sin la presencia de las librerías del QUICK 500 siempre y cuando sólo

se trabaje en simulación. Para trabajar en tiempo real es indispensable la presencia del QUICK 500 así como del equipo de adquisición de datos Keithley 500. Dado que se trabajó en el computador al cual está conectado este equipo, se ha buscado aprovechar las posibilidades gráficas que ofrece ese computador. Se trata de un IBM PS/60 con tarjeta de gráficos VGA a colores, cuya presencia es indispensable para ejecutar el programa sea en simulación o en tiempo real. Además el software debe estar instalado en el directorio D:\ADAPTIVO, es decir que todos sus programas y archivos deben estar ubicados allí.

La estructura básica del programa TESISAD.EXE se presenta en la figura 4.1.

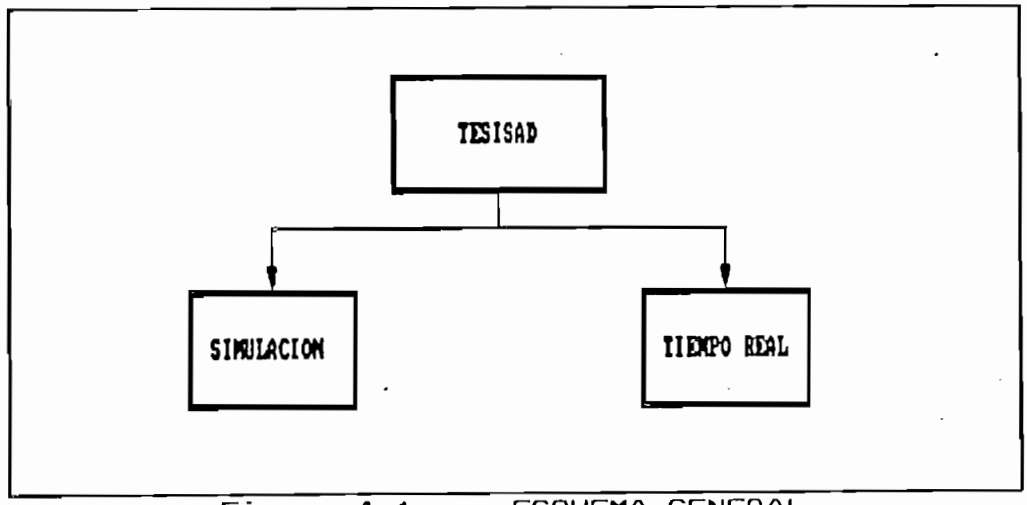


Figura 4.1 – ESQUEMA GENERAL

4.2. EL MODULO PRINCIPAL: TESISAD.BAS

El modulo principal corresponde al tronco del programa TESISAD.EXE. En él se hallan definidas las rutinas y funciones a usarse en el programa, las variables comunes, las constantes, las rutinas de error, y el diseño del MENU PRINCIPAL para la bifurcación a simulación o a tiempo real (ver diagrama 4.1). Además en él se chequea si el equipo de adquisición de datos está encendido previo a permitir el paso a las rutinas de tiempo real.

Son rutinas del programa:

CALCU: Calcula el valor de U en simulación

CALCY: Calcula el valor de Y en simulación

CLERR: Borra la zona de mensajes de error

CONTROL: Realiza la identificación y el control en simulación

CONVERT$: Transforma un dato numérico en arreglo de caracteres

DATIN: Recupera los valores predefinidos de parámetros, valores iniciales y polos

DATOS: Crea un archivo de datos y accede a LOTUS para graficación

ERSIST: Genera mensajes de error del sistema de ecuaciones

GRAF: Grafica el nuevo valor de U y Y en la pantalla

IDENT: Realiza la identificación en simulación

MENU: Genera los Menus Principal, Simulación y Tiempo Real

MENUDA: Genera los menus y cuadros no creados por MENU

MODELO: Define el modelo de la planta para simulación

PARIN: Recupera los parámetros del modelo predefinido

PESC: Producto escalar entre dos vectores

PMAT: Producto de dos matrices

PRINTER: Impresión de Reportes en papel

PVECT: Producto de una matriz por un vector

SALIDAU: Exporta la señal U de Salida en tiempo real

SCDAT: Genera los Menus de identificación y control en simulación

SIDAT: Genera los Menus de identificación en simulación

SIMULA: Ingreso a las rutinas de simulación

SIST: Resuelve un sistema de ecuaciones

TRAERY: Adquiere el último valor de Y del equipo de adquisición

TRCDAT: Genera los Menus de identificación y control en tiempo real

TRCONTROL: Realiza la identificación y el control en tiempo real

TREAL: Ingreso a las rutinas de tiempo real

TRIDAT: Genera los Menus de identificación en tiempo real

TRIDENT: Realiza la identificación en tiempo real

TRPARAM: Define Periodo de muestreo en tiempo real

VALID: Realiza la validación de los datos ingresados

Son variables comunes:

N: (Orden del modelo) * 2

NA: Grado del polinomio A(z) identificado

NB: Grado del polinomio B(z) identificado

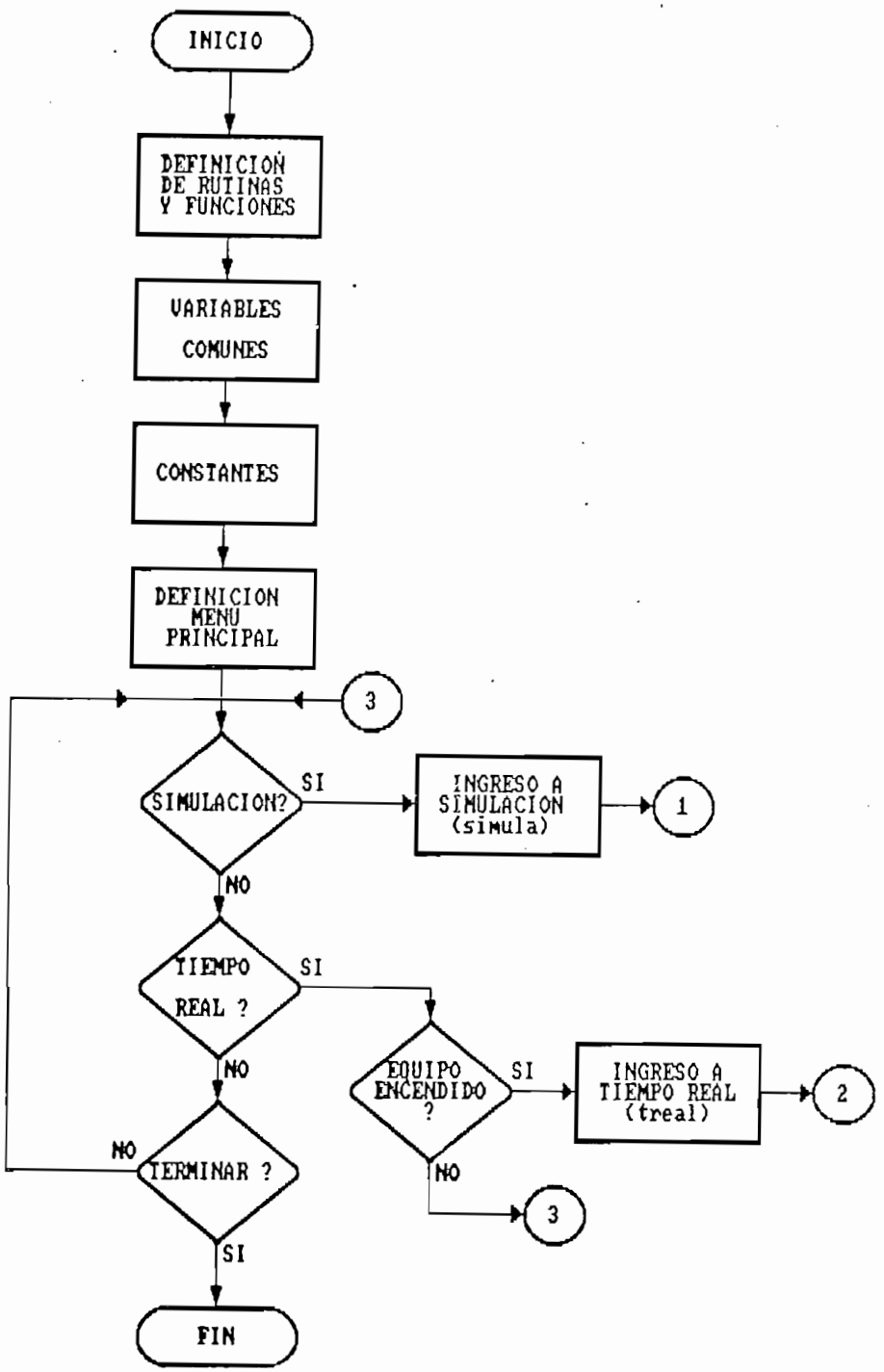


Diagrama 4.1

TESISAD

NS: Orden del Modelo * 2 (Planta en Simulación)

NAS: Grado del Polinomio A(z)

NBS: Grado del Polinomio B(z)

XS(): Datos de Entrada-Salida en simulación

TS(): Parámetros del Modelo de la Planta para simulación

Y: Señal de Salida de la planta

U: Señal de entrada a la planta

R: Referencia

dep!: Longitud del Arreglo de datos para Tiempo Real

bintv%: Intervalo de muestreo

DAT(): Matriz de datos

MaxDat: Número máximo de datos

ERF%: Detección de Error

4.3. SIMULACION

La parte de simulación tiene su propia estructura. Esta se muestra en la figura 4.2. Es posible en simulación: definir el modelo que servirá de planta; realizar la identificación paramétrica de esta planta en el mismo orden o en uno diferente; realizar la identificación y el control (control adaptivo tipo self-tuning); extraer los datos obtenidos en uno de los dos procesos anteriores a un archivo en el disco duro.

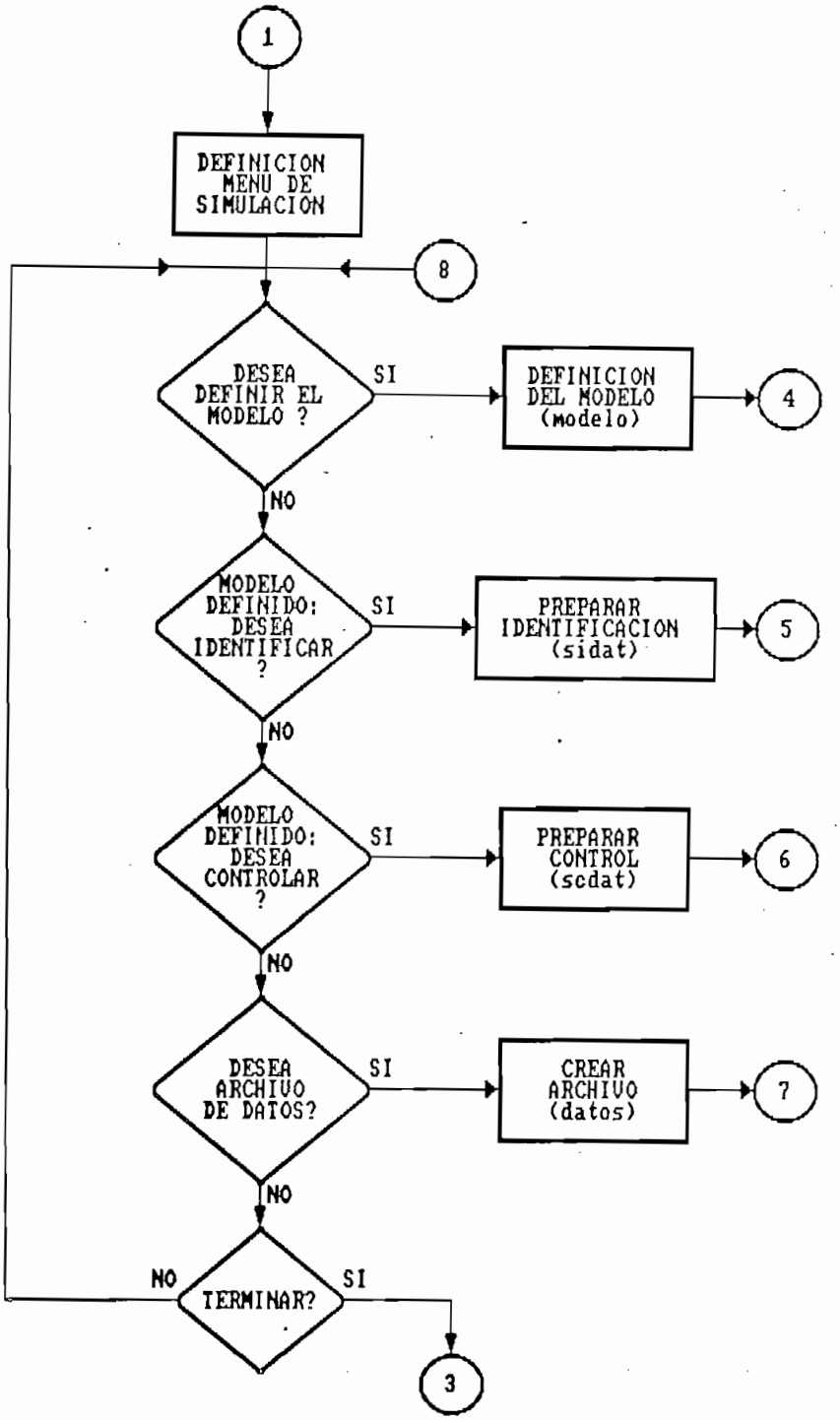


Diagrama 4.32 SIMULA

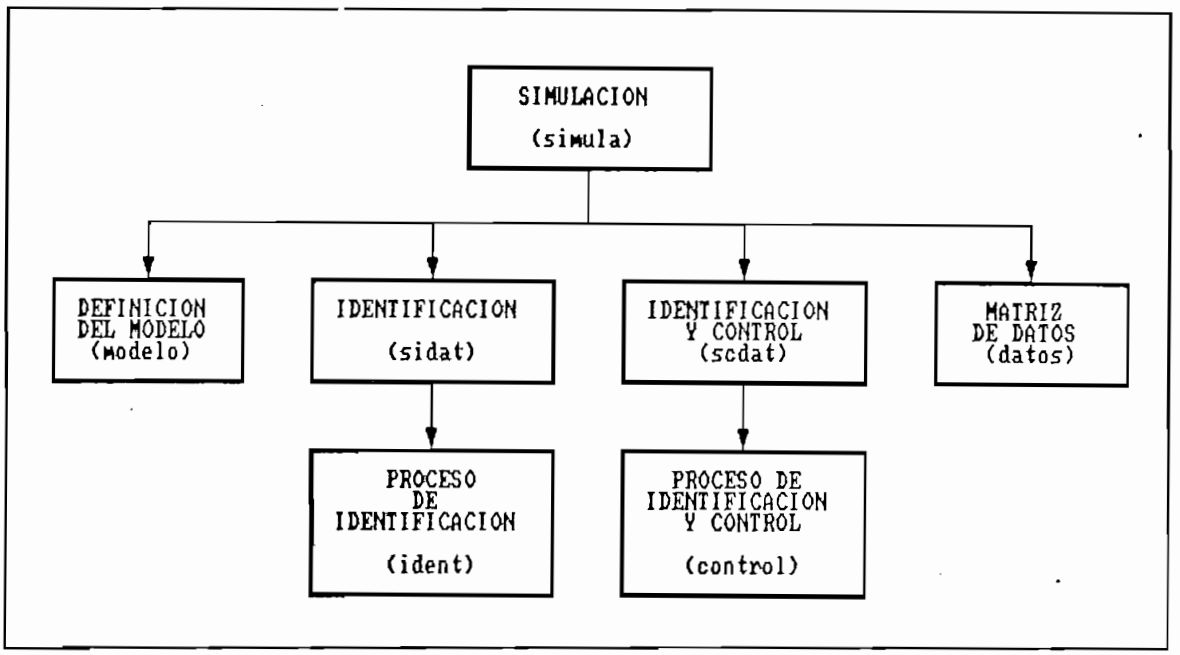


Figura 4.4 - SIMULACION

Al pasar del módulo principal a simulación, se ejecuta la subrutina SIMULA. Esta permite la elaboración de un menú con las diferentes opciones posibles en simulación: Definición del modelo; Identificación; Identificación y control; Creación de un archivo de datos; y Terminar. Esta última opción simplemente devuelve el control del programa al módulo principal. El diagrama de flujo de SIMULA se presenta en el diagrama nº 4.2.

a) Definición del modelo

Esta opción es obligatoria antes de iniciar la simulación. Es necesario que esté definido un modelo de la planta a identificar o controlar. Esto se consigue usando una bandera

(s1) que impide escoger el resto de opciones mientras no esté definido el modelo.

La rutina MODELO crea un Menú de Modelo, uno de Orden y un Cuadro de Parámetros. Como norma para todo este programa, las ventanas que aparezcan en celeste no serán accesibles para el usuario sino tan sólo visibles. El ingreso de los datos se facilita muchísimo con el método de ventanas, por lo que será ampliamente utilizado en este programa. La validación de los datos ingresados se realiza a través de la rutina VALID que provoca una señal auditiva en la máquina cada vez que se detecta un error.

El orden máximo previsto en este programa es el cuarto orden. Para cada orden existe un modelo predefinido en el propio programa. En la diagrama 4.3 se presenta el diagrama de flujo de esta rutina. Los nuevos parámetros y el nuevo orden se almacenan en TS() y NS, respectivamente.

b) Identificación

Antes de realizar la identificación propiamente dicha, es decir la ejecución del algoritmo de mínimos cuadrados, es necesario definir algunos parámetros que permiten que este algoritmo sea muy versátil. La rutina SIDAT permite la definición de los parámetros de identificación y de los valores iniciales.

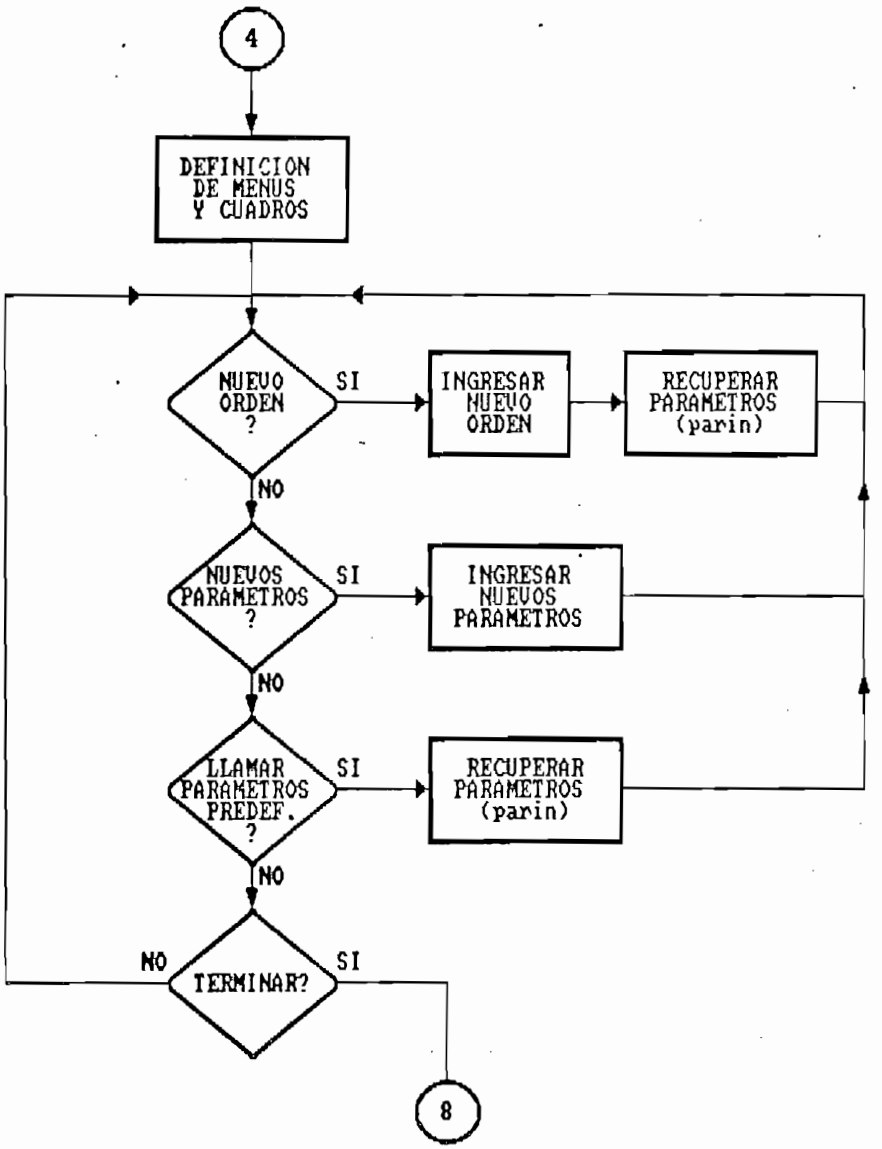

4
DEFINICION DE MENUS Y CUADROS
NUEVO ORDEN ?
SI
INGRESAR NUEVO ORDEN
RECUPERAR PARAMETROS (parin)
NO
NUEVOS PARAMETROS ?
SI
INGRESAR NUEVOS PARAMETROS
NO
LLAMAR PARAMETROS PREDEF. ?
SI
RECUPERAR PARAMETROS (parin)
NO
NO
TERMINAR?
SI
8

Diagrama 4.3

Los parámetros a definirse son:

N: Orden del Modelo a Identificar (N, entero entre 1 y 4)

α: Valor inicial matrix de covarianzas P

R: Referencia ($0 \leq R \leq 10$)

Umin: Valor mínimo permitido de la señal de control

Umax: Valor máximo permitido de la señal de control

($0 \leq Umin \leq Umax \leq 10$)

MaxDat: Número de Datos a guardarse para la Matriz de Datos

Ruido (%): Porcentaje de ruido respecto a Y

($0\% \leq Ruido \leq 100\%$)

Los valores iniciales pueden ser cualquier número salvo el cero pues éste valor provoca problemas especialmente en el control.

En esta rutina, así como en el resto de rutinas similares (SIDAT, SCADAT, TRIDAT, TRCDAT), las variables son de tipo STATIC, es decir que su valor se mantiene aún si se está fuera de la rutina. Esto permite mantener las condiciones para las pruebas que se deseen realizar. Su funcionamiento está descrito en su diagrama de flujo (ver diagrama 4.4). La transferencia de información se dá a través de los arreglos para$() y val$().

Después de estar definidos los parámetros, se pasa a la identificación propiamente dicha mediante la rutina IDENT. En los diagrama 4.5 a) y b), se presenta su diagrama de flujo.

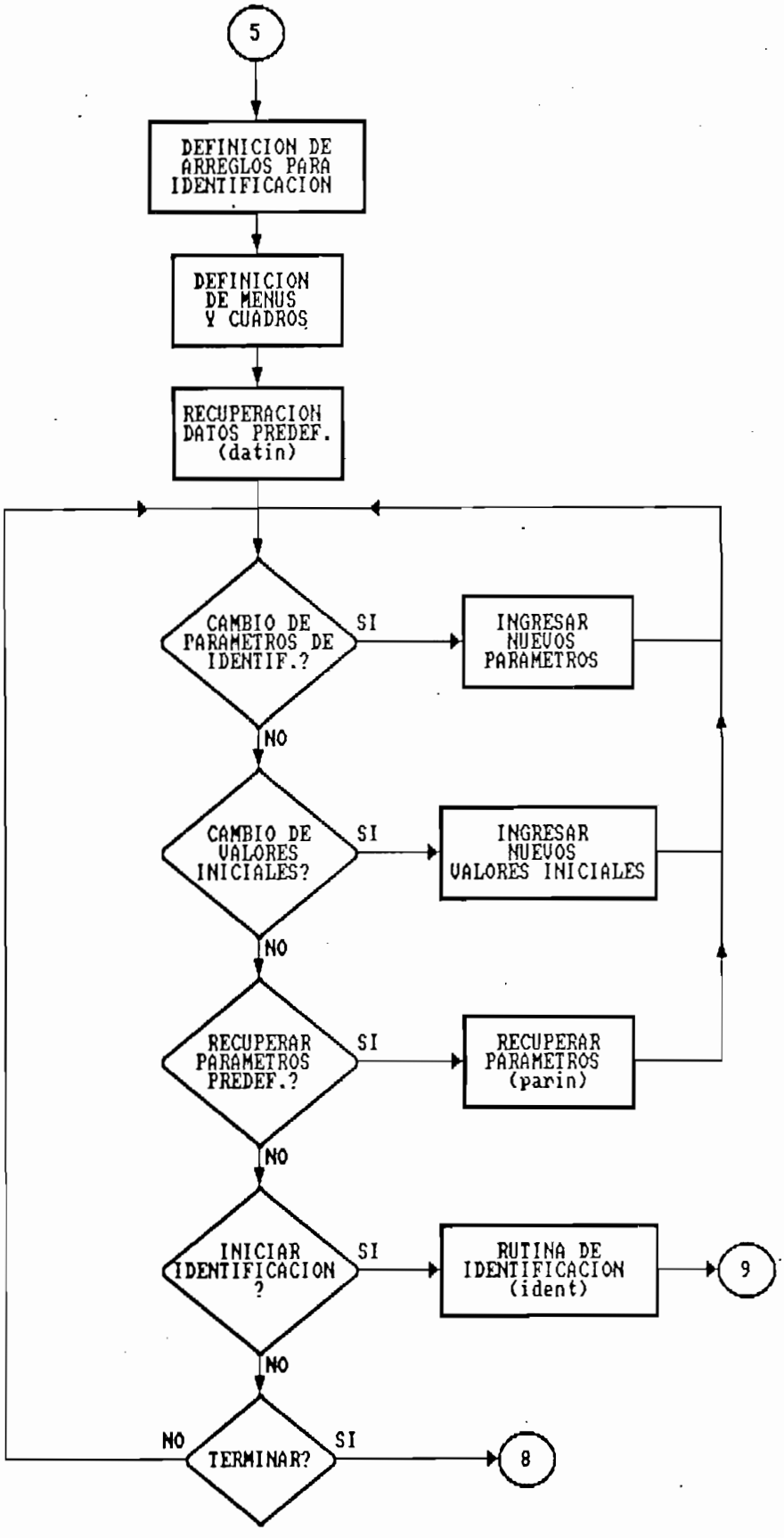


Diagrama 4.4

SIDAT

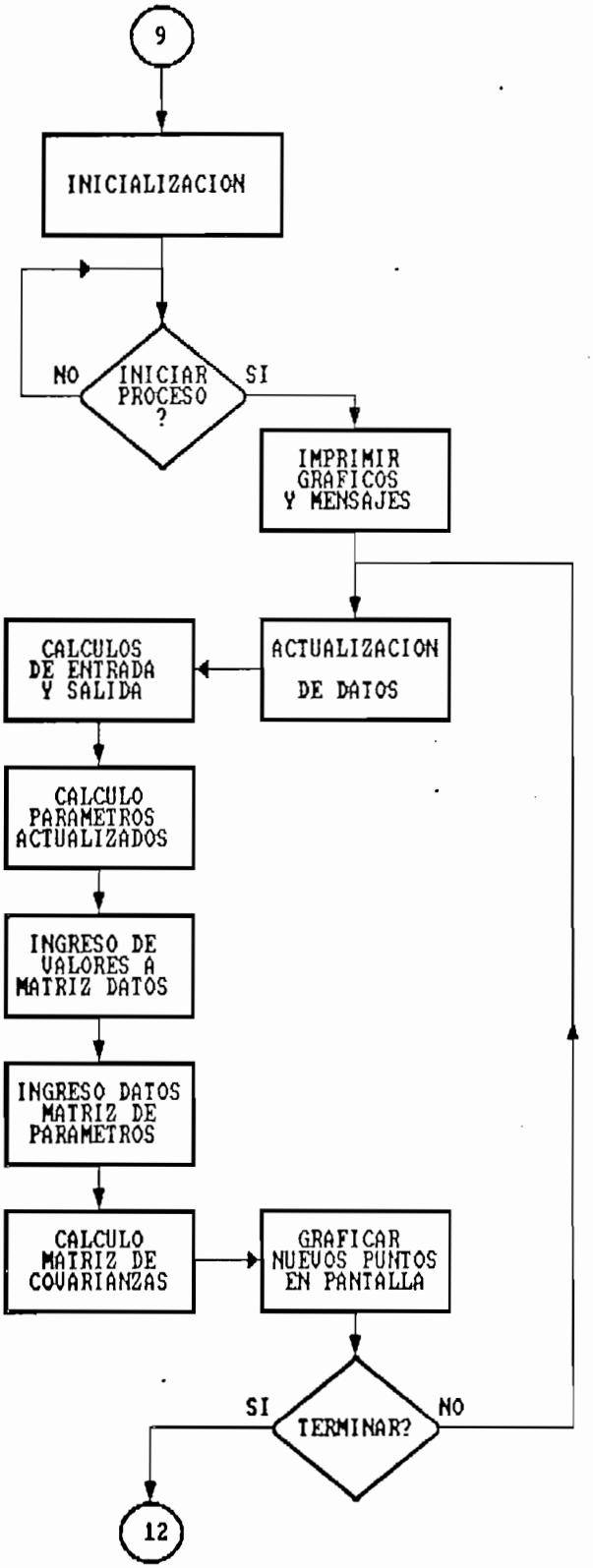


Diagrama 4.5. a)

IDENT

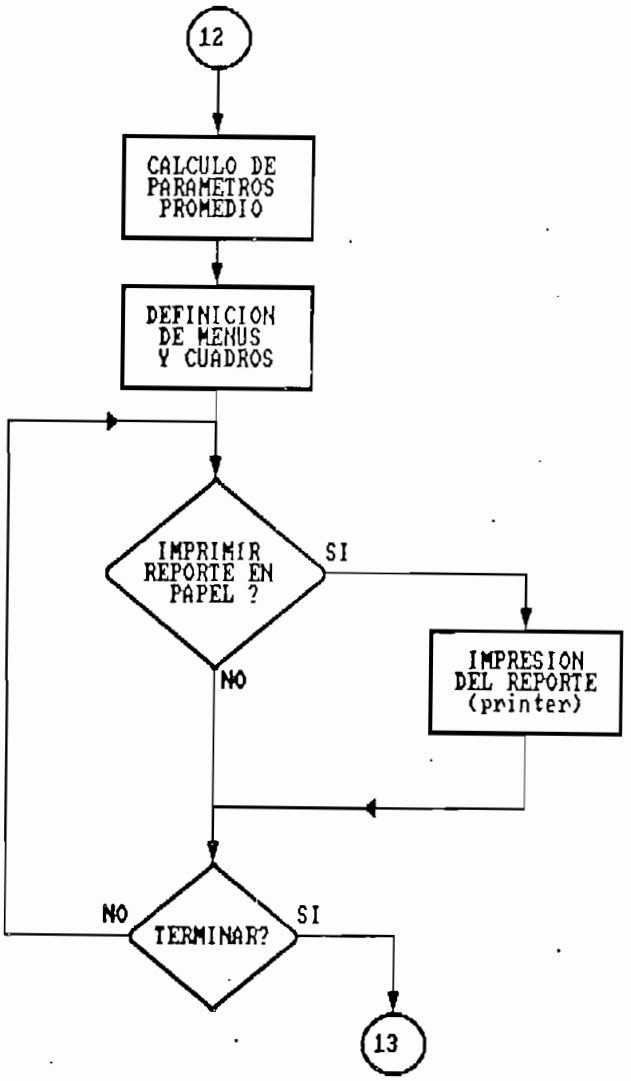


Diagrama 4.5. b)

Son variables específicas de esta rutina:

P(): Matriz de Covarianzas

MA(), MB(): Matrices auxiliares de cálculo

T(): Vector de Parámetros (modelo ARMA)

X(): Vector de Datos Entrada-Salida (modelo ARMA)

V(): Vector auxiliar de cálculo

TM(): Matriz con las últimos 50 cálculos de parámetros

TF(): Vector con los Parámetros Promedio

DAT(): Matriz con los Parámetros y las señales de entrada y salida.

Aquí lo que se ha hecho es implementar en QUICK BASIC el algoritmo de mínimos cuadrados desarrollado en el Capítulo 2. La identificación hace uso de los valores definidos en TS() y NS así como los que se encuentran en para$() y val$().

La actualización de los valores se basa en el cálculo (se trata de simulación) de Y a partir de U y de los valores anteriores, en la planta cuyo modelo ha sido previamente definido.

c) Identificación y control

El control se realiza a través de la rutina CONTROL. Esta contiene el algoritmo de identificación tal como aparece en la sección anterior, pero se le añade el diseño y generación

de la ley de control. Como se vió en el capítulo 3, esto requiere la resolución de un sistema de ecuaciones y el cálculo de la señal de control a partir de los parámetros de control.

Adicionalmente a las variables de la rutina IDENT, se definen aquí:

s(): Matriz de Control
a(): Vector de Parámetros a para Control
pol(): Vector de coeficientes de polos

La rutina SCDAT, permite la definición previa de parámetros. Su funcionamiento es muy similar al de la rutina SIDAT salvo que como se busca hacer control es necesario además definir la ubicación de los polos, los que se transfieren mediante el arreglo pol$(), adicionalmente a para$() y val$() (diagrama 4.6).

La rutina CONTROL (diagrama 4.7) se encarga de construir el vector de parámetros a (polinomio A), así como la matriz s() que contiene el sistema de ecuaciones a resolverse. Su resolución se consigue a través de la rutina SIST que contiene un algoritmo para resolver sistemas ecuaciones lineales (BURDEN 1985: 356-359). Con esto quedan determinados los parámetros f_i y g_i de la ley de control. Basta entonces calcular la señal de control U en base a la fórmula ya desarrollada.

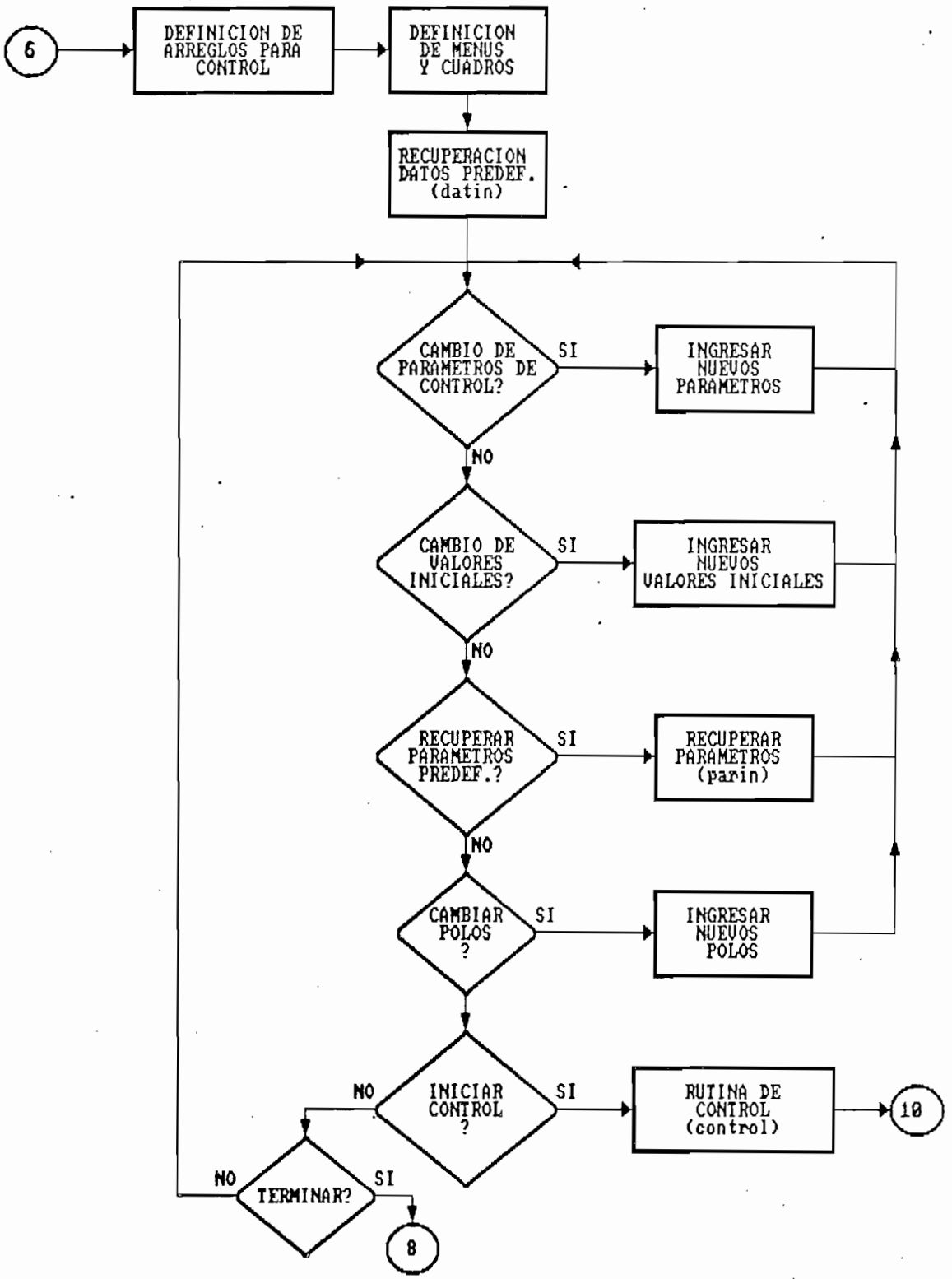


Diagrama 4.6 SCDAT

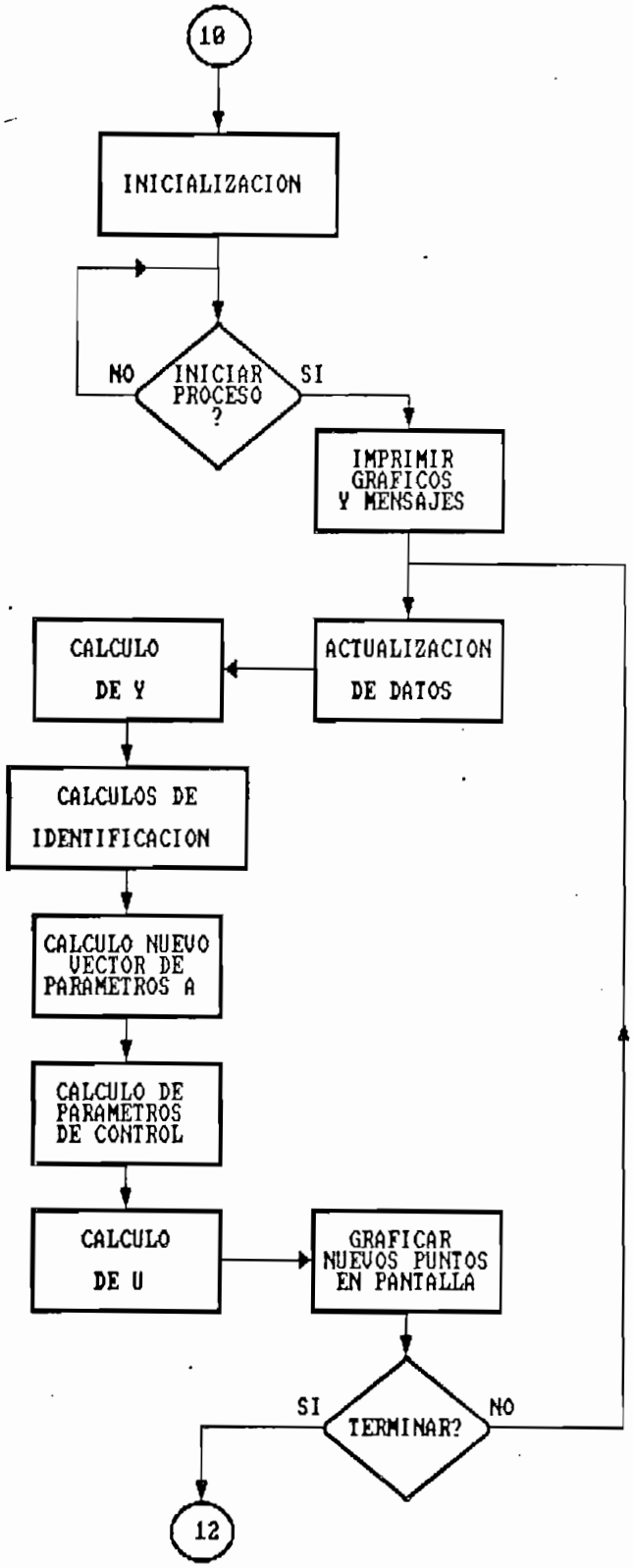


Diagrama 4.7 CONTROL

d) Transferencia de la información al disco

Después de haberse realizado los procesos de identificación o de control, se requiere en muchas ocasiones conocer el tipo de convergencia de los parámetros, y los gráficos de entrada y salida impresos en papel. Para esto, se ha previsto en este programa una opción que permite transferir a un archivo del disco la información recabada. Esto es lo que se consigue con la rutina DATOS, cuyo diagrama de flujo se presenta en el diagrama 4.8. Esta rutina permite un nombrar un archivo, guardar allí la información y pasar al LOTUS 123 para la graficación de los resultados. Esta última parte puede realizarse siempre y cuando el computador tenga suficientememoria disponible como para tener al mismo tiempo el programa TESISAD.EXE, el programa LOTUS.COM y el archivo deseado. No siempre esto es posible, por lo que se recomienda utilizar el programa TSGRAF.BAT después de haber terminado de trabajar con TESISAD.EXE. TSGRAF.DAT es un BATCH que permite el acceso al LOTUS usando la autoinicialización de la hoja electrónica TESISAD.WK1 (ver apéndice B).

La rutina DATOS se encarga de borrar la matriz de datos cuando ésta ya ha sido transferida al disco para así liberar memoria RAM del computador y permitir la transferencia al LOTUS. Los archivos TSGRAF.BAT y TESISAD.WK1 deben estar presentes en el directorio D:\ADAPTIVO, caso contrario una señal de error será emitida. Los archivos se guardan en disco en el directorio D:\ADAPTIVO y con un sufijo .DAT.

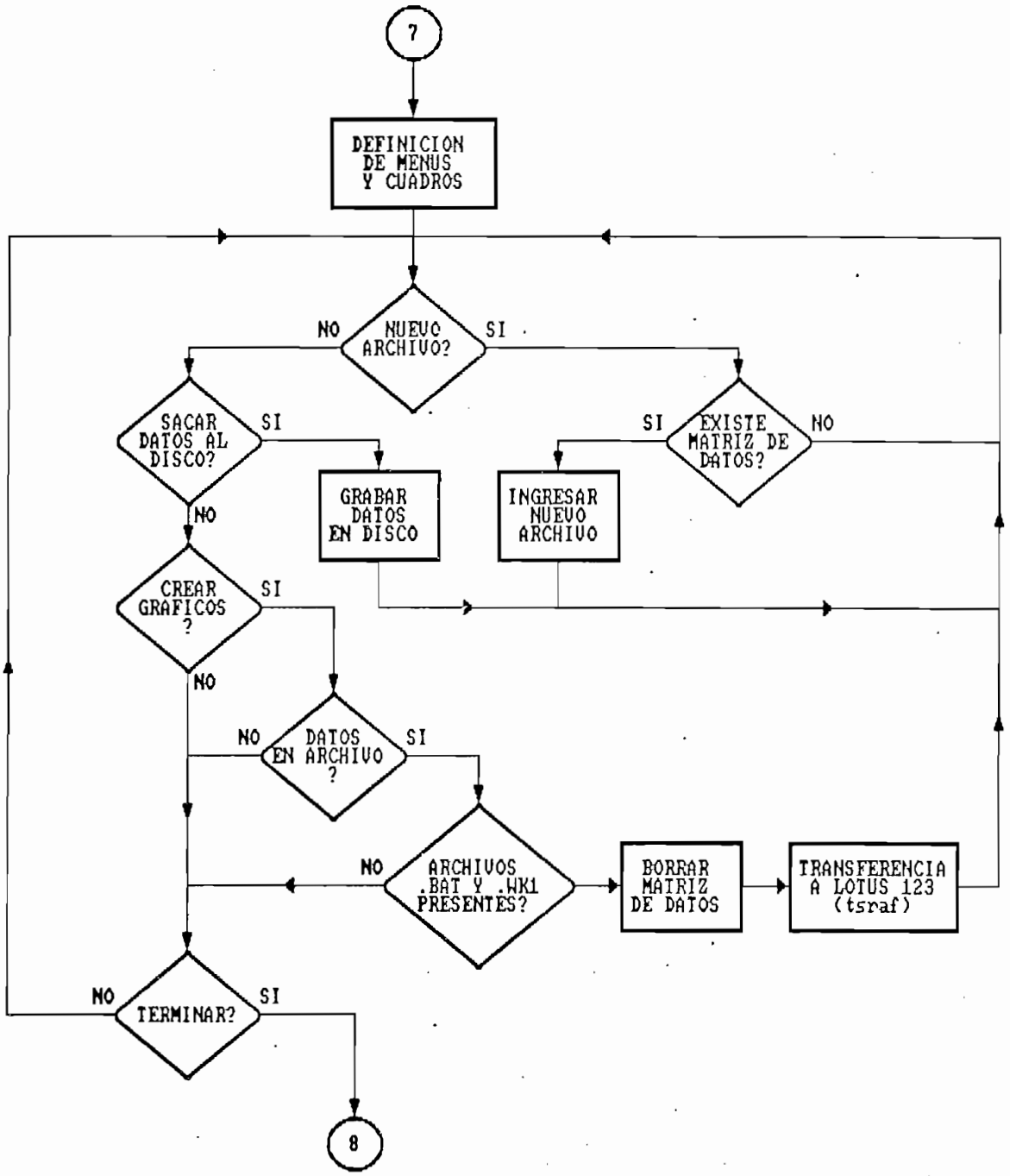


Diagrama 4.8 DATOS

4.4. TIEMPO REAL

La estructura de la parte de tiempo real es similar a la de simulación y se muestra en la figura 4.3. Ya no es necesario definir un modelo sino que se conecta directamente una planta real al equipo de adquisición de datos para que éste discretice las señales analógicas y las acondicione para ser utilizadas por el computador. Para el tiempo real, el modulo principal llama a una rutina TREAL que permite iniciar las rutinas de tiempo real. Allí aparecen algunas opciones: Definir el periodo de muestreo (obligatorio al inicio); Identificación; Identificación y Control; Inicialización; grabar la información obtenida en el disco duro.

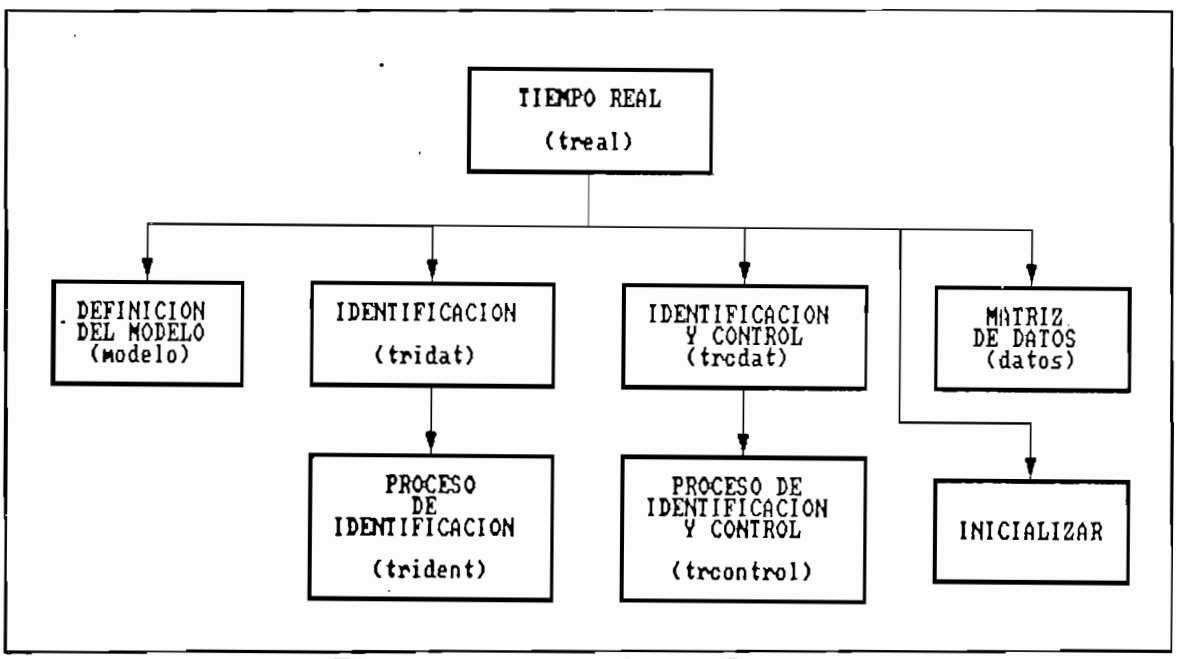


Figura 4.3 - TIEMPO REAL

La rutina TREAL (diagrama 4.9) se encarga de inicializar al equipo de adquisición de datos y prepararlo para un funcionamiento adecuado. Esto se realiza con las rutinas INIT y SOFTINIT del QUICK 500. Además, en base a la variable TR!, se obliga a iniciar el proceso con la definición del periodo de muestreo.

a) Periodo de muestreo

Antes de realizar cualquier prueba de tiempo real es esencial definir el periodo de muestreo. Como se verá posteriormente en las pruebas, el periodo de muestreo puede resultar crítico para conseguir un control adecuado de la planta, pues afecta directamente a la estabilidad de la señal de control. Se recomienda generalmente que su valor esté cercano a la décima parte del tiempo de respuesta de la planta, para obtener unos resultados satisfactorios. La definición del tiempo de muestreo se realiza tomando como base interrupciones de 1 ms para el trabajo en background.

b) Identificación

El paso previo a la identificación es una rutina de definición de parámetros (TRIDAT), tal como en simulación. Su funcionamiento es el mismo que la SIDAT salvo que ahora la identificación se realiza con la rutina TRIDENT,

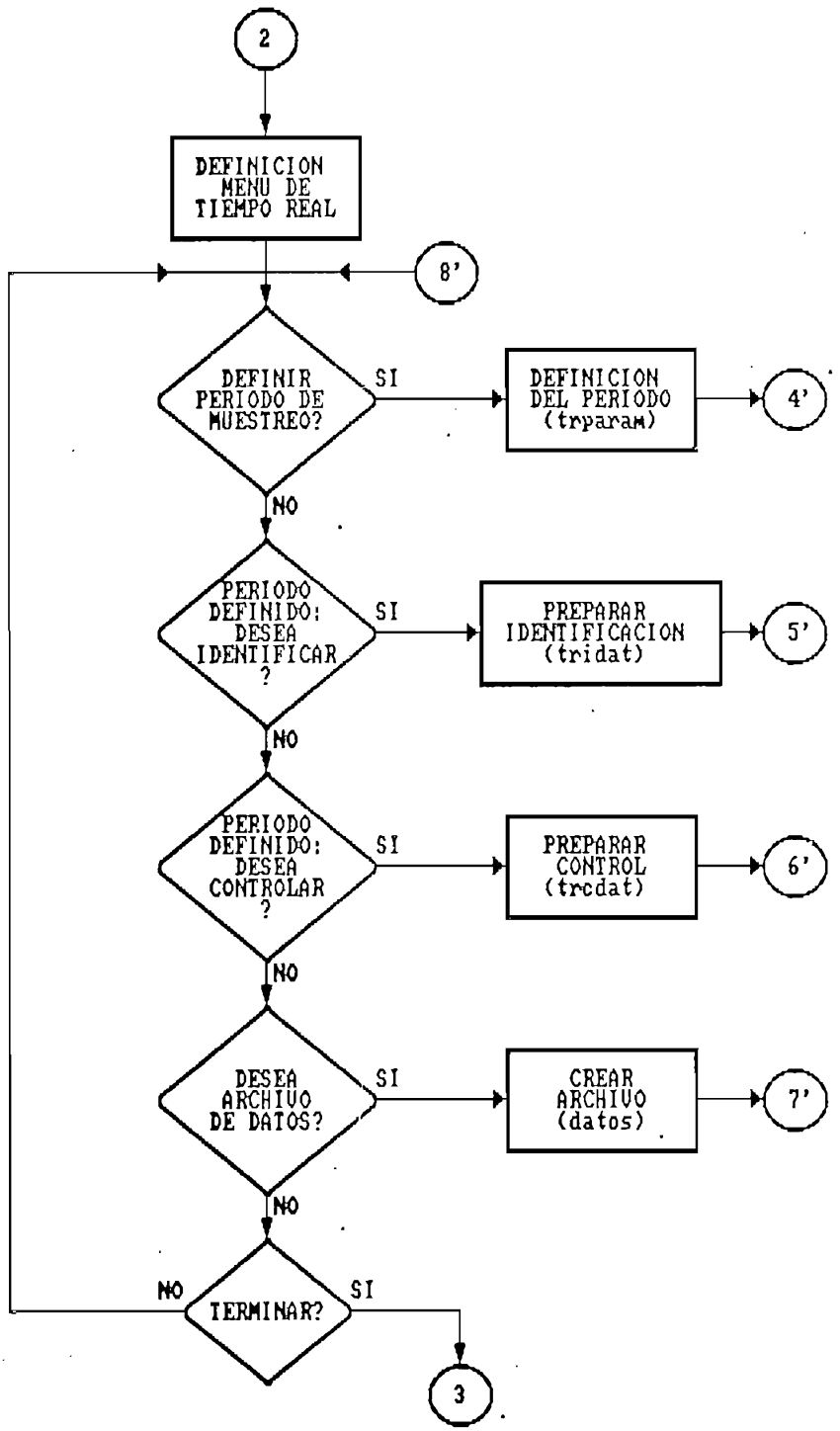


Diagrama 4.9 TREAL

identificación en tiempo real.

Lo que cambia esencialmente entre TRIDENT e IDENT es la captación y entrega de información, como puede verse en el diagrama 4.10. En lugar de cálculos de U y de Y, ahora se calcula U y se la envía a la planta a través del equipo de adquisición y Y es medida (leída) a través del mismo equipo. En cada interrupción correspondiente al periodo de muestreo definido, se obtiene la salida de U y la entrada de Y al computador. Entre dos interrupciones se realiza un solo lazo de identificación.

Las rutinas SALIDAU y TRAERY reemplazan a los cálculos de U y Y de la simulación. Para la salida de U, se utilizan ANOUT, ARMAKE, ARPUTVALF y ARLASTP de la librería QUICK 500. Para el ingreso de Y, se usa ANIN, ARLASTP y ARGETVALF de la misma librería. El uso de estas rutinas requiere mucho cuidado en la definición de sus parámetros, pues el cambio de tipo en uno de los parámetros puede causar un bloqueo en el computador.

Las rutinas del QUICK 500 que aquí se usan tienen las siguientes características:

* ANIN: Es una rutina de background para muestrear los canales de la entrada analógica. Este comando causa que las medidas sean tomadas a un cierto intervalo (periodo de muestreo) de los canales analógicos especificados (aquí se utiliza uno solo) y crea un

arreglo de Quick500 (entraday%) en el que se guarda la información adquirida, fruto de la conversión A/D. Esta rutina permite el ingreso de Y al computador.

* ARLASTP: Este comando recupera el índice (ubicación) del último punto accedido por una rutina de Quick 500 en un arreglo dado. En este programa esta rutina permite detectar si hay valores de Y ingresados y también la ubicación del último valor de U sacado del computador a la planta por el equipo de adquisición de datos.

* ARGETVALF: Esta rutina permite acceder a los valores existentes en un arreglo de Quick 500. El acceso se realiza dato por dato y la información es trasladada a una variable en BASIC. Su utilización en este programa permite obtener el último valor de Y adquirido desde la planta.

* ANOUT: Es una rutina de background que transfiere información, de un arreglo de Quick 500 (entrada%) previamente definido, a los canales de salida analógica del KEITHLEY 500A. El intervalo de salida de datos es el periodo de muestreo. Esta rutina permite la salida de U a la planta.

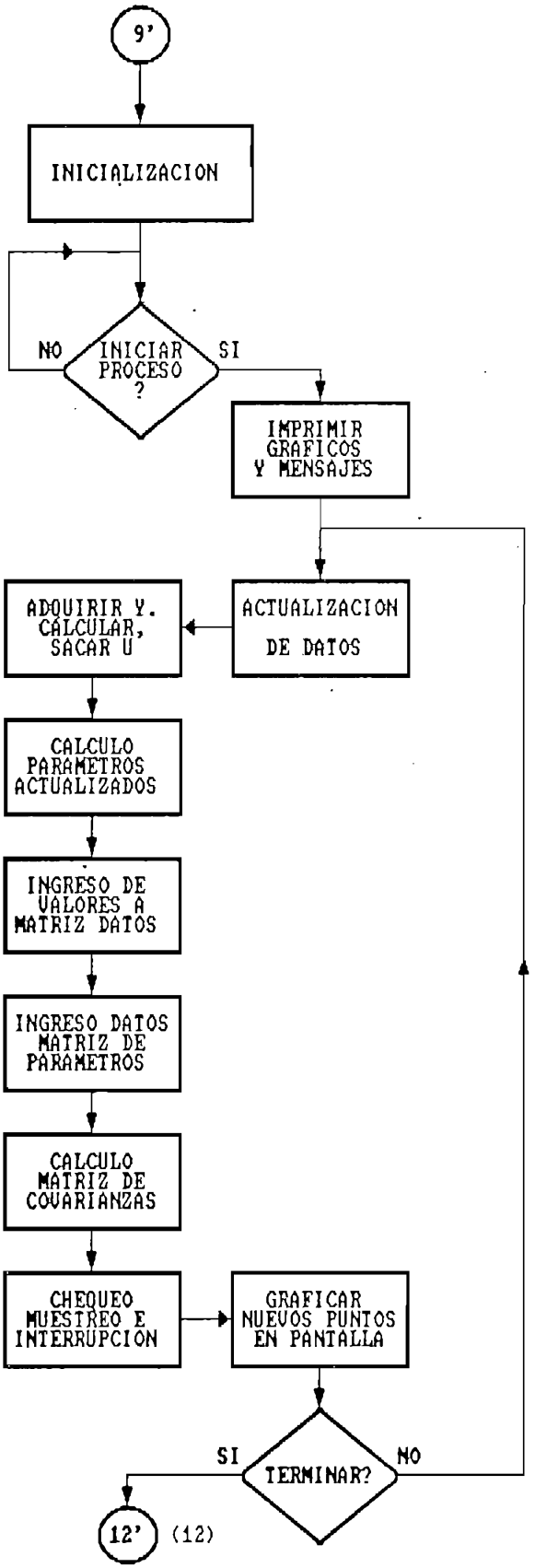


Diagrama 4.10

TRIDENT

* ARMAKE: A través de este comando se crean arreglos de Quick 500. En este programa ARMAKE crea el arreglo "salida%" en el cual se almacenan los valores de U.

* ARPUTVALF: Este comando pasa la información de una variable BASIC a un arreglo de Quick 500, en la ubicación especificada. Permite el almacenamiento de U para su posterior traslado a la planta.

Además de las variables de IDENT, se requieren aquí:

tim(): Timer

DT(): Conteo de Timer

Estas variables se definen para hacer trabajar uno de los 4 timers con los que cuenta el sistema. Este Timer es utilizado aquí para detectar el ritmo de las interrupciones del background, evitar que se realice más de un lazo de identificación dentro del mismo periodo de muestreo, y detectar si el periodo de muestreo es insuficiente para que se ejecute todo el lazo (especialmente cuando el orden es elevado). Dado que el timer tiene un límite de conteo, se ha añadido además una pequeña rutina para evitar que en una prueba que requiera tiempos muy largos, pueda fallar el conteo del timer. Esta corrección consiste en detectar el reset del timer y hacer un reset del contador interno de iteraciones.

La exitación persistente generada por el computador para la identificación es un escalón al que se le añade ruido blanco (en realidad una variable seudo-aleatoria) con media cero. El ruido blanco se genera usando el comando RND que genera un valor randómico entre 0 y 1. El nivel del ruido depende además de un parámetro que debe ser introducido por el usuario (ERP, en porcentaje). La ecuación general es entonces:

$$U = R(1 + \frac{ERP}{100}\text{RND} - \frac{ERP}{200})$$

c) Identificación y control

Las rutinas de control en tiempo real son dos: TRCDAT y TRCONTROL. TRCDAT define los parámetros de control que son los mismos que para simulación salvo el ruido en la salida que ya no es necesario en tiempo real pues la propia planta tiene perturbaciones. TRCONTROL realiza el control propiamente dicho.

Para el control en tiempo real se utilizan las rutinas de entrada y salida de datos señaladas para la identificación. Es necesario el timer y además se debe chequear que el algoritmo no salga de control. Esto puede suceder si aparecen muchos ceros en la señal de control, con lo que se crea un sistema linealmente dependiente incapaz de detectar las variaciones a la salida. Esto se evita generando una señal de

recuperación correspondiente a Umax para evitar la aparente estabilidad que ocurrirá al detectarse una sucesión larga de ceros. Esto tiene directa relación con el criterio de excitación persistente.

En el diagrama 4.11, se presenta el diagrama de flujo de la rutina TRCONTROL. La estructura de la rutina TRCDAT es la misma que para SCDAT (diagrama 4.6).

d) Inicialización

Esta opción permite la inicialización del equipo de adquisición de datos con la ejecución de la rutina INIT del QUICK 500, antes del inicio de un nuevo experimento. Esta inicialización no es automática al entrar en las rutinas de identificación o control.

e) Transferencia al disco

Esta opción es idéntica la de simulación pues al escogerla se ejecuta la rutina DATOS.

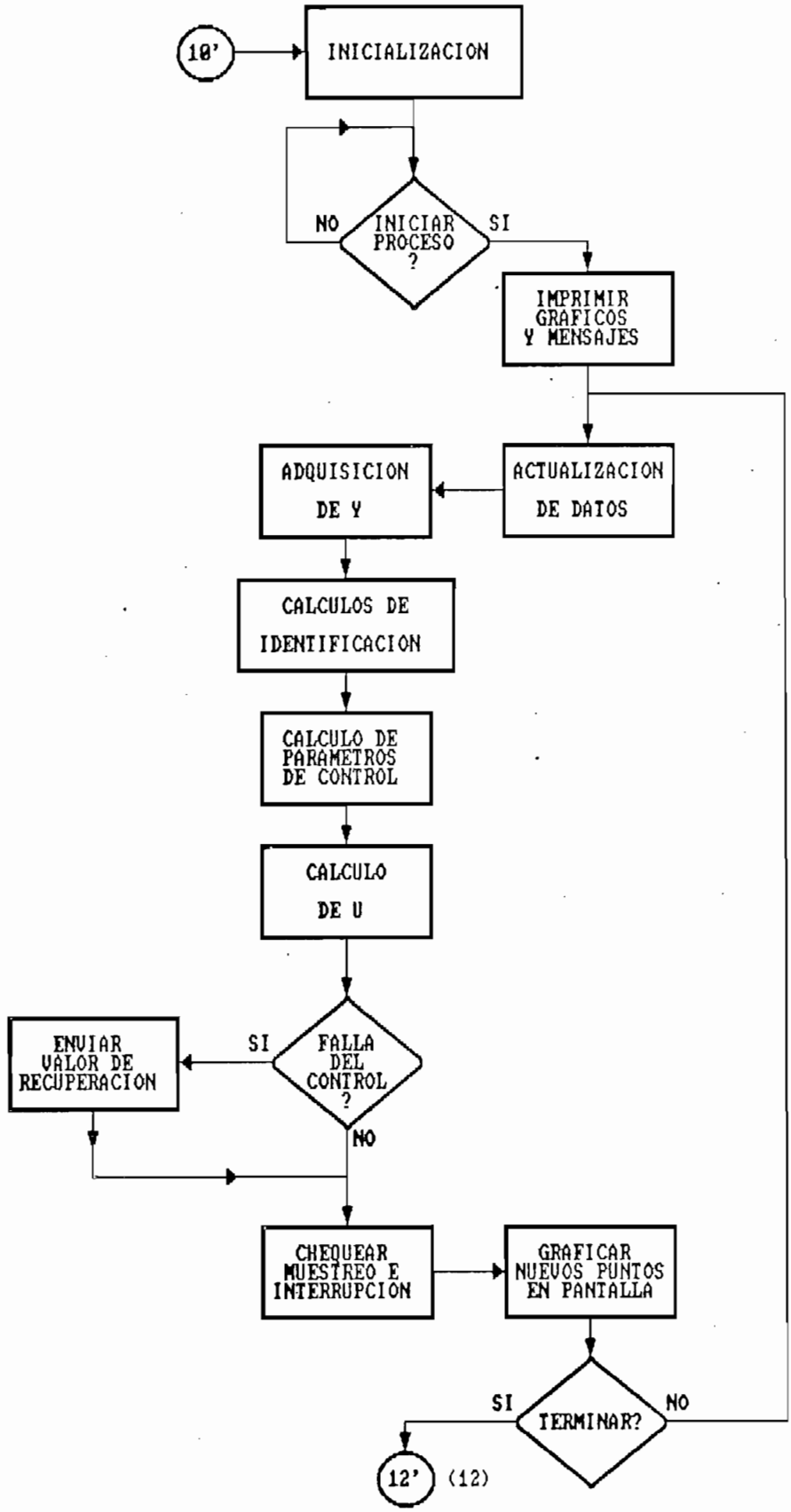


Diagrama 4.13

TRCONTROL

4.5. RUTINAS DE GRAFICOS, MENUS Y VENTANAS

Se ha buscado que este programa no presente ningún tipo de problema al interactuar con el usuario. Por esto se ha escogido una presentación que permita tener una información ágil sobre lo que se está haciendo, así como un fácil manejo de la entrada y salida de datos. Se ha puesto mucho énfasis en la información visual (Menús, ventanas, gráficos de entrada y salida) pero se ha tenido en cuenta también la necesidad de tener respaldos escritos de las pruebas y experiencias que se realicen con este programa.

a) Uso de menus y ventanas

Los menus y ventanas facilitan la utilización de este programa. El recorrido a lo largo del programa se realiza enteramente a través de menús y ventanas para escoger las opciones y el ingreso de datos. Esto se consigue mediante la utilización de dos rutinas MENU y MENUDA que permiten la creación de las ventanas y menus en pantalla. Para que sea posible acceder a la mayor cantidad de información, se presentan en la pantalla ventanas de color celeste no accesibles por el usuario, mientras que aquellas habilitadas para una interacción con el usuario tienen un fondo azul o rojo (para la opción escogida). Se trabaja tan sólo con las flechas y la tecla ENTER, para las opciones y con el teclado de letras o números según el caso, para el ingreso de

información. El programa dará una señal auditiva en caso de detectarse un error en el ingreso de datos, en el uso de teclas no permitidas o ante cualquier error al trabajar con el DOS.

La rutina MENU permite la graficación en pantalla de los tres primeros menus: el Principal, y los de Simulación y Tiempo Real. Para diferenciarlos del resto, el color de fondo para la opción escogida es magenta. El principio de funcionamiento de esta rutina se expresa con la ayuda de su diagrama de flujo (diagrama 4.12). Los parámetros que se entregan a la rutina son el título y las opciones posibles en un arreglo predefinido. Con esta información, se crea el encabezado y se imprime cada vez que se detecta una tecla el resto del menú, cambiando de colores a la posición de la opción a escoger. Las flechas hacia arriba y hacia abajo y el ENTER son las teclas habilitadas. La selección (un ENTER) hace que la rutina entregue un valor que indica la opción detectada. Existe además un parámetro (b!) que impide la selección de una opción cuando hay alguna que es obligatoria (Modelo, en simulación y Parámetros en tiempo real).

La rutina MENUDA permite la creación y manejo de todos los otros menus y ventanas. Crea los Menús habilitados por la interacción con el usuario y presenta las ventanas no accesibles para visualizar la información y resultados. Esta rutina tiene una estructura similar a la rutina MENU salvo que en la detección de teclas tiene la posibilidad de permitir el

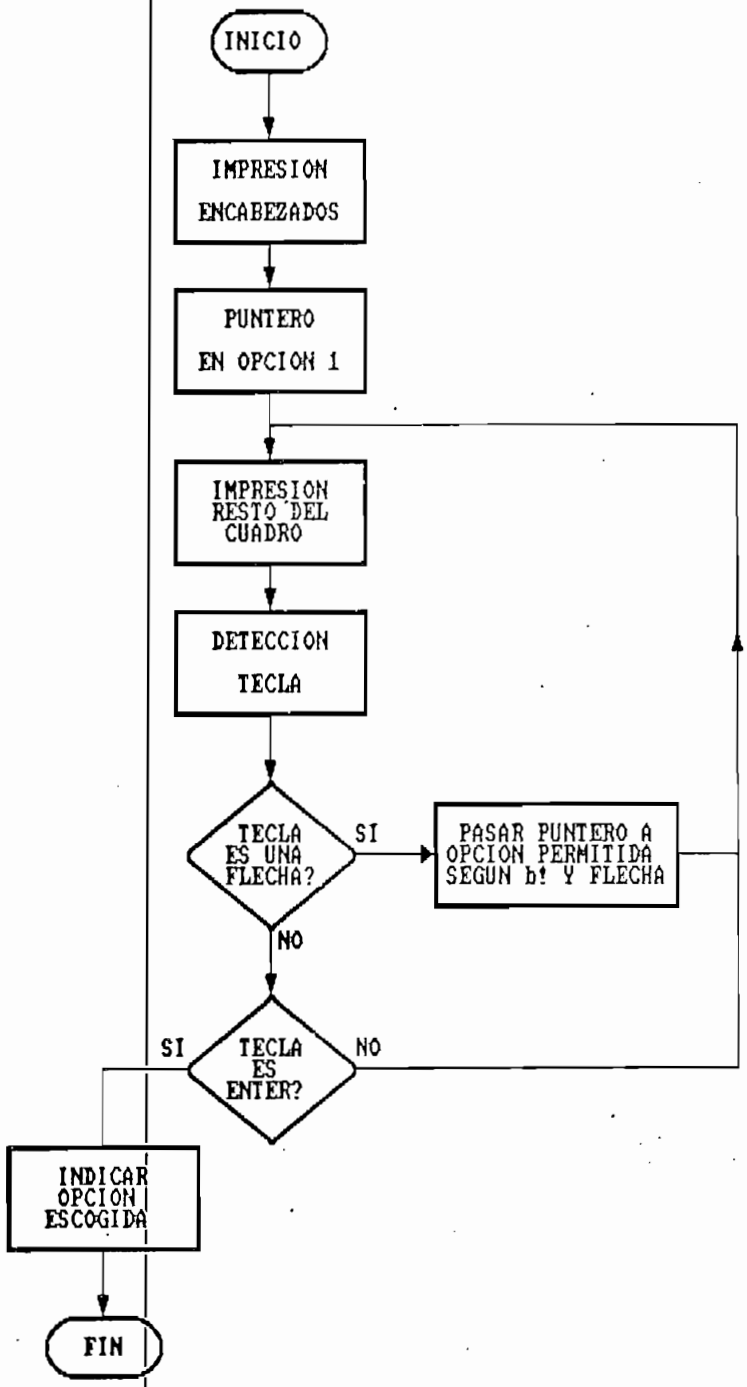


Diagrama 4.12

MENU

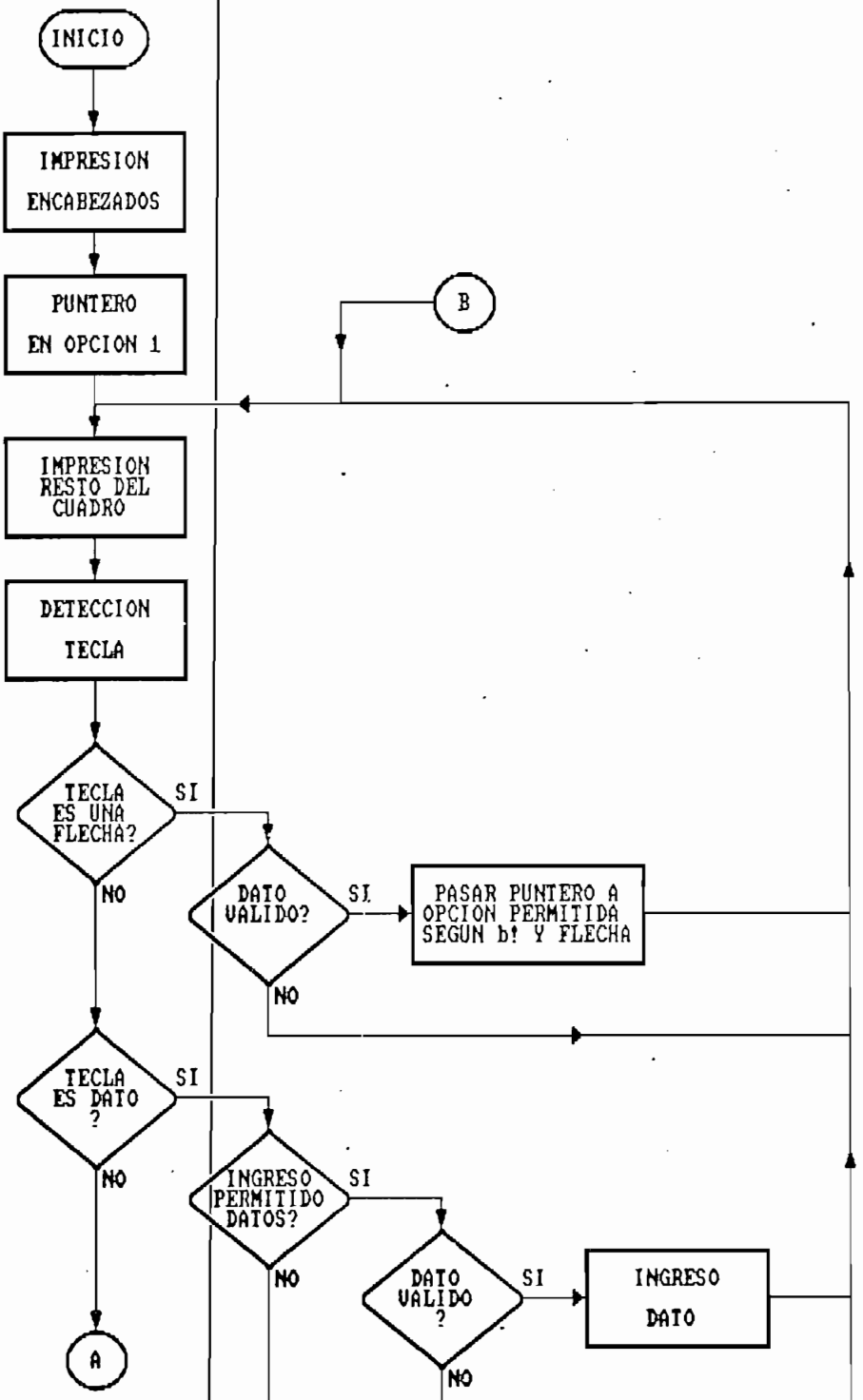


Diagrama 4.13. a) MENUDA

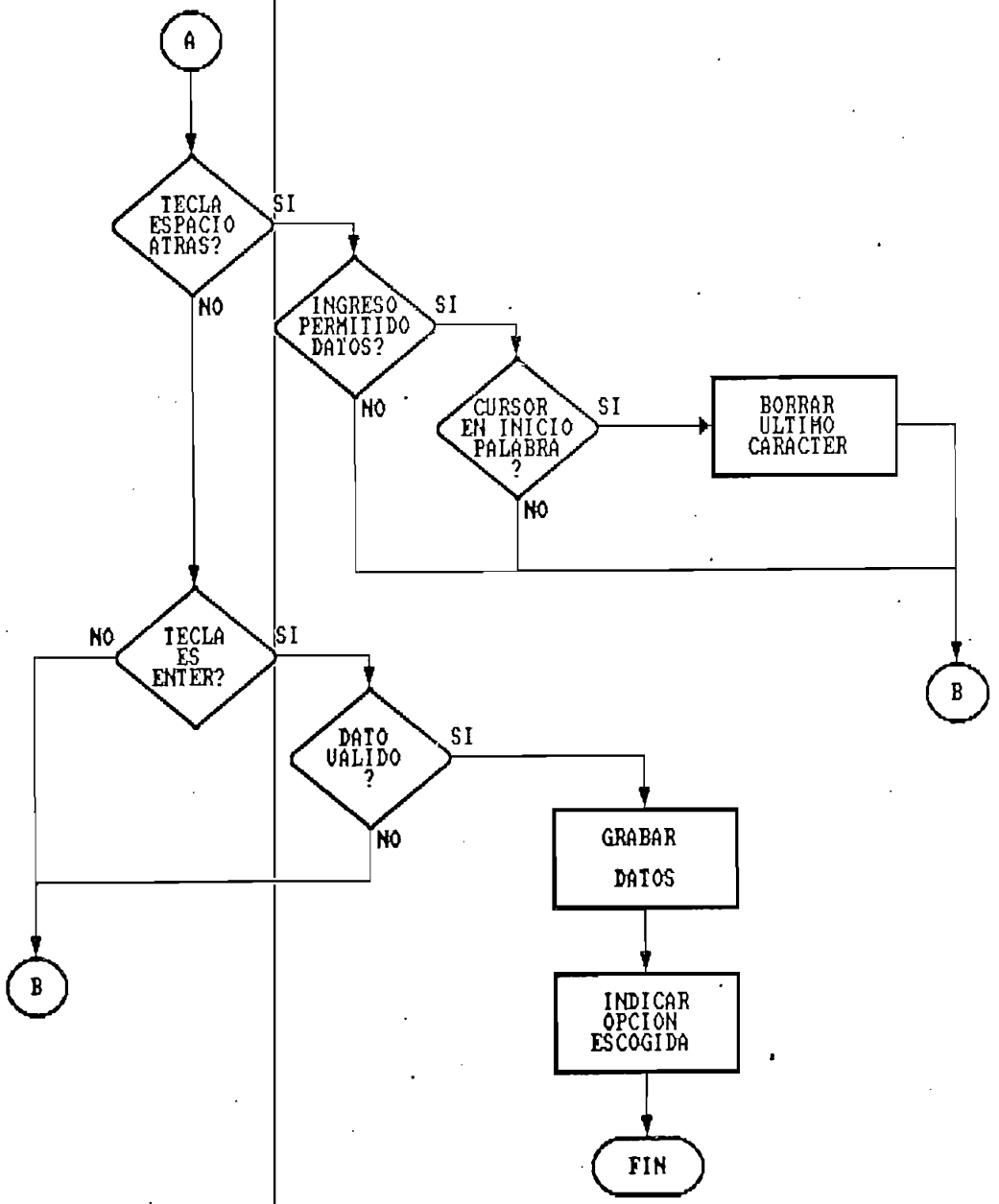


Diagrama 4.13. b)

MENUDA

ingreso de información (diagrama 4.13). Los parámetros que requiere esta rutina son el arreglo con titulo y nombres odatos y una información adicional (b!) sobre el tipo de menú que indicará si está: solo visible; habilitado como menu de opciones; habilitado para el ingreso de datos solo numéricos; habilitado para el ingreso de datos literales y numéricos (nombre del archivo). Según el valor de b! la rutina además realiza la validación de la información ingresada mediante el llamado a la rutina VALID. Al igual que en MENU, MENUDA entrega la información sobre la opción escogida cuando detecta un ENTER. La impresión del menú habilitado se realiza con cada detección de tecla y se cambia de color para la posición de la opción escogida.

b) Gráficos

Durante todo el proceso de identificación o control sea en simulación o en tiempo real, se presentan en pantalla dos gráficos en alta resolución y a colores, utilizando la tarjeta VGA de gráficos del computador. Esto permite tener una información de primera mano e inmediata sobre el proceso que sigue la planta. Para el caso de tiempo real esto es mucho más interesante al tenerse en pantalla gráficos de entrada y salida que reflejan lo que está sucediando en ese momento en la planta.

Los gráficos son generados en un tipo especial de pantalla: el SCREEN 9 definido con fondo negro. Los gráficos tienen un fondo azul para facilitar su visión. Las rutinas encargadas de la graficación son GRAF y DEFGRAF. La primera (GRAF, diagrama 4.14) realiza el cálculo de la posición del punto en la escala del gráfico para lo cual ha sido necesario un pequeño sistema de ecuaciones de transformación de escala, después de lo cual, se tienen las coordenadas del punto en la pantalla y se lo grafica allí. La segunda (DEFGRAF, diagrama 4.15) genera los gráficos iniciales (vacíos) como dos bloques con fondo azul y dos ejes, uno del tiempo y otro de U y de Y según el gráfico.

En las rutinas de identificación y en las de control, se ha añadido un contador de puntos graficados (K) para detectar el final del gráfico y reinicializar la pantalla con un nuevo gráfico vacío y así poder continuar con la graficación del proceso. Esto permite tener una información permanente sobre la dinámica de la planta.

c) Impresión de resultados

Al final de cada una de las rutinas de identificación o control (IDENT, CONTROL, TRIDENT, TRCONTROL), se presentan los resultados de la identificación en pantalla. Si se desea un reporte escrito de estos resultados así como de los parámetros que caracterizan la prueba realizada, es posible escoger la

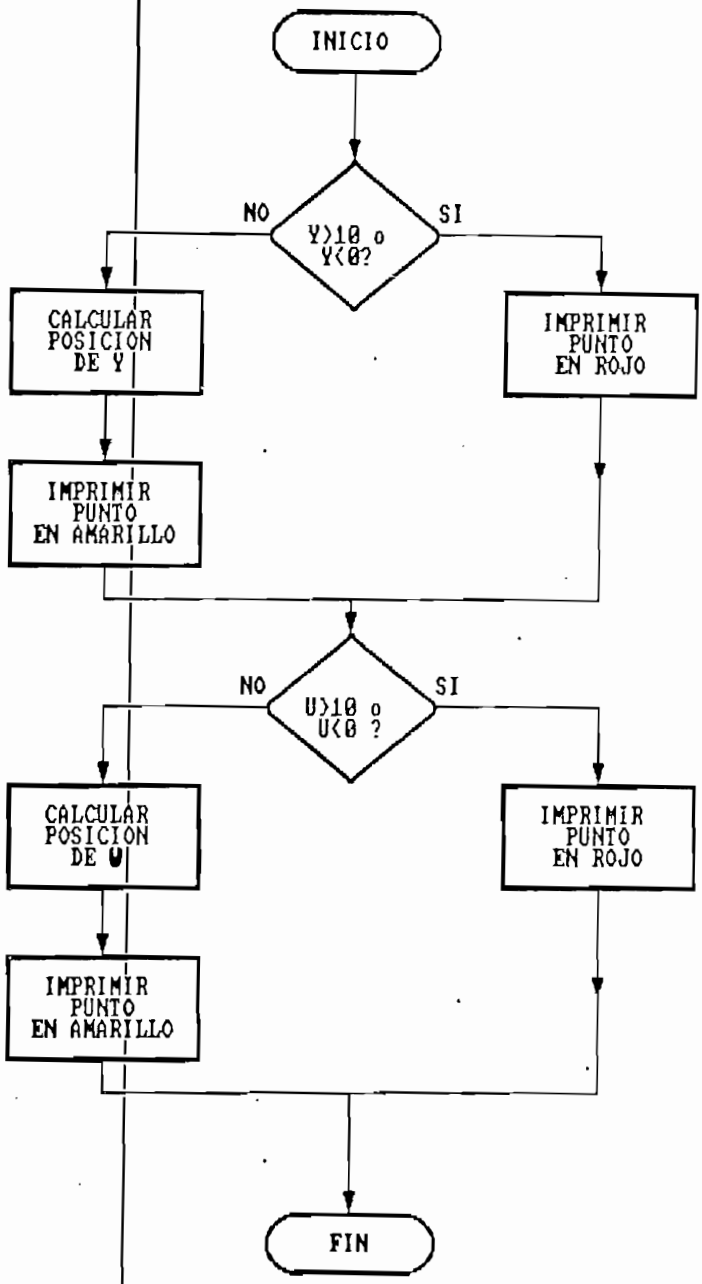


Diagrama 4.13 GRAF

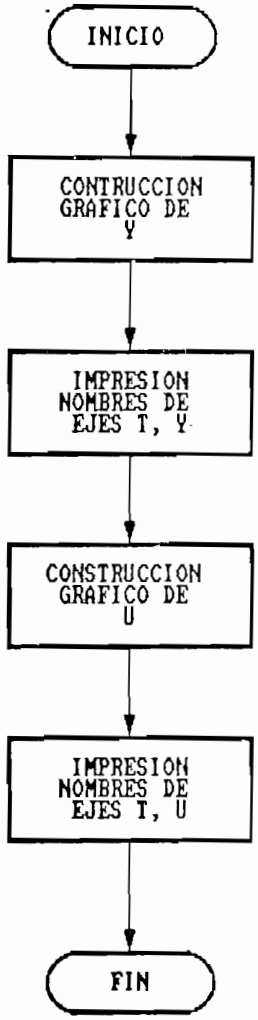


Diagrama 4.15 DEFGRAF

opción imprimir. El resultado es un reporte con la información básica de la experiencia. Si esto es insuficiente y se desea conocer las características en función del tiempo de señales de entrada y salida y parámetros estimados, entonces se debe transferir los datos a un archivo en disco para posteriormente crear los respectivos gráficos en una hoja electrónica. Elnúmero de datos permitidos depende únicamente de la capacidad de memoria de la máquina, tomando en cuenta que se encuentran cargados el QUICK 500 y el programa TESISAD.EXE.

En el apéndice A se encuentra el listado completo del programa TESISAD.BAS. En el Apéndice B se presentan los programas adicionales: TSGRAF.BAT, TESISAD. WK1. El apéndice C consta de un pequeño programa basado en los gráficos desarrollados aquí para realizar el control PID de una planta y finalmente en el apéndice D se presenta el manual de uso del programa.

5. RESULTADOS Y CONCLUSIONES

5.1. PRUEBAS DE SIMULACION

5.2. IDENTIFICACION Y CONTROL EN TIEMPO REAL

5.3. CONCLUSIONES

5.4. RECOMENDACIONES

5. RESULTADOS Y CONCLUSIONES

5.1 PRUEBAS DE SIMULACION

Las pruebas de simulación para este trabajo consisten en la realización de identificación y control usando los algoritmos de simulación anteriormente desarrollados. Se pretende con ésto verificar la validez de las formulaciones teóricas tanto en lo referente a identificación de sistemas por el método de mínimos cuadrados, cuanto a la realización de control adaptivo para una planta predefinida.

Las pruebas realizadas en simulación corresponden a la identificación y al control de cada una de las plantas o modelos previamente escogidos. En simulación, dado que todo el proceso se realiza enteramente en el computador, es necesario definir el modelo de la planta a utilizarse.

a) Primera Prueba

La primera prueba consta de un modelo de segundo orden sobre el cual se hará:

1.- Identificación de parámetros;

2.- Identificación y control sin ruido a la salida.

En las siguientes páginas se muestran las características de cada prueba, obtenidas como un reporte del programa, así como los respectivos gráficos de entrada y salida y de parámetros en función del tiempo.

1.- Identificación de parámetros.

Para la identificación de parámetros se genera una señal de excitación persistente (U) correspondiente a un escalón con un ruido del 20% la que, alimentada a la planta, provoca una señal de salida (Y), como se muestra en la figura 5.1.

Los resultados de la identificación muestran una rápida convergencia a los parámetros de la planta (Figuras 5.2 y 5.3). El algoritmo trabaja con muchísima precisión, tal como se puede ver en los resultados numéricos (ver Reporte nº 1).

2.- Identificación y control.

Aquí se trata ahora de realizar la identificación y el control de la planta. Se ha definido un polinomio de polos cuyas raíces se encuentran dentro del círculo unitario y en la parte de reales positivos, en el plano z (ver Reporte nº2). No existirá ruido inducido en la señal de salida.

Como se muestra en la figura 5.4, el control se realiza adecuadamente, manteniendo la señal en el valor de la referencia definida (R = 3).

ESCUELA POLITECNICA NACIONAL

FACULTAD DE INGENIERIA ELECTRICA

TESIS: IDENTIFICACION DE SISTEMAS EN TIEMPO REAL

SIMULACION

MODELO DE LA FORMA:

$$Y(z) = \frac{\sum b(i) z^{-i}}{1 + \sum a(i) z^{-i}} U(z)$$

1) MODELO DE LA PLANTA

Orden del Modelo = 2
A(1) = 1
A(2) = .4
B(1) = .5
B(2) = .9

2) PARAMETROS DE IDENTIFICACION

Orden del modelo = 2
Parametro Alfa = 10000
Referencia = 3
U minimo = 0
U máximo = 10
Ruido Porcentual = 20 %
Número de Iteraciones = 521

3) RESULTADOS DE IDENTIFICACION

A(1) = .9999999897568216
A(2) = .3999999995515172
B(1) = .5000000011493323
B(2) = .899999993894583

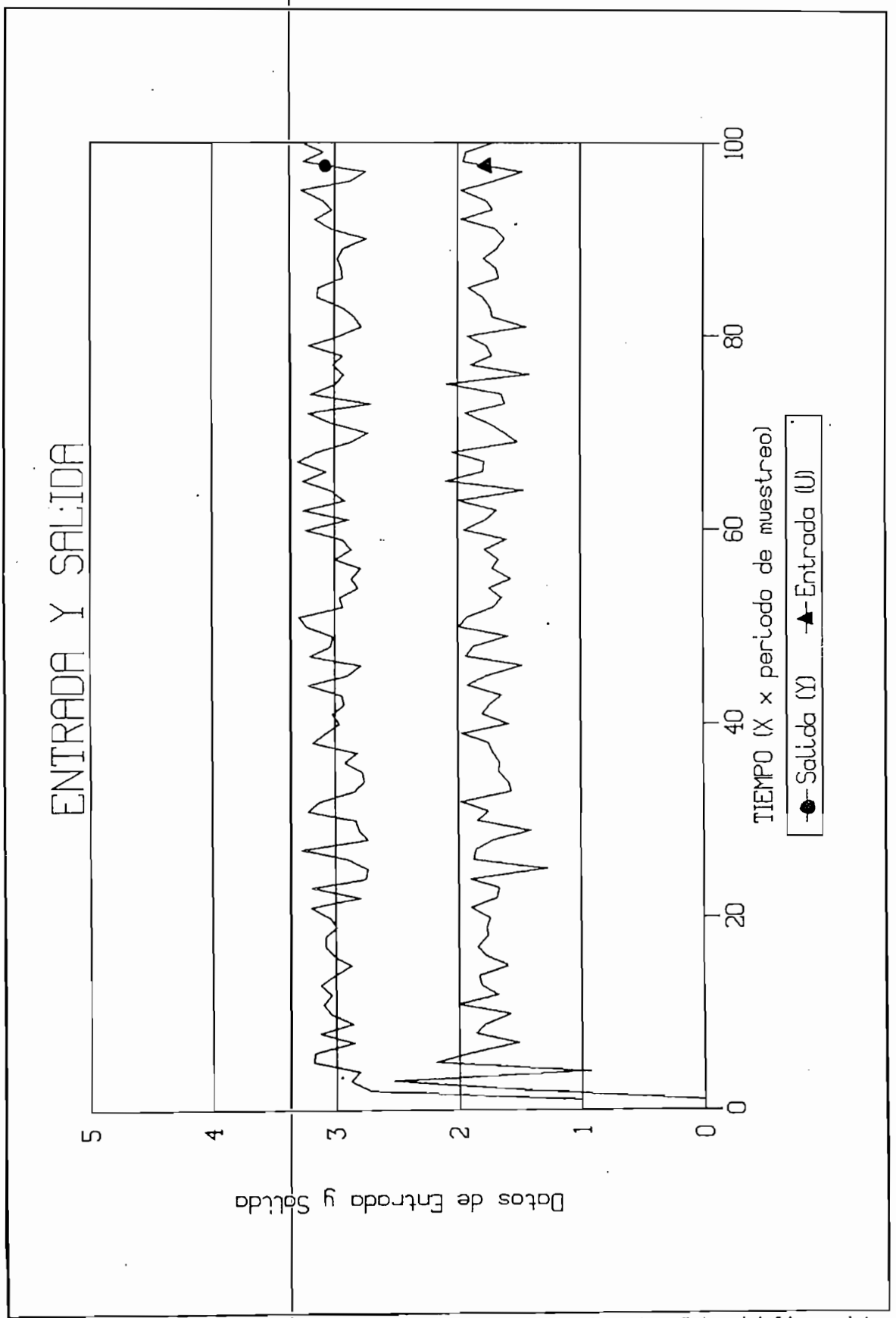


Figura 5.1 Prueba 1: Identificación

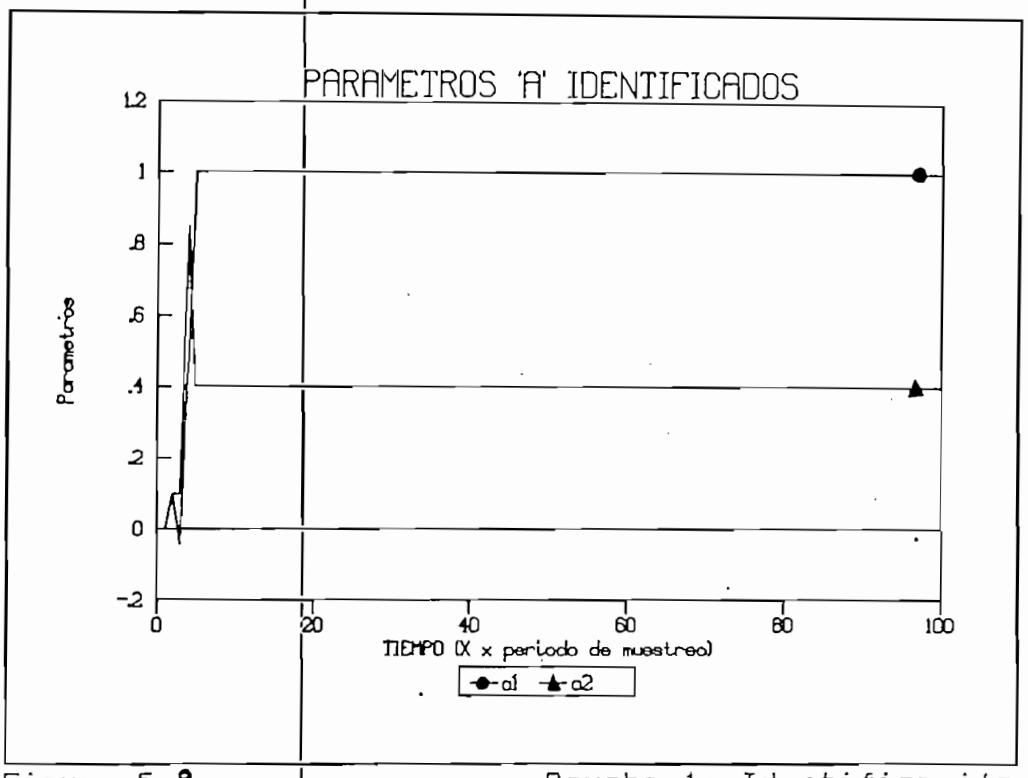


Figura 5.2 Prueba 1: Identificación

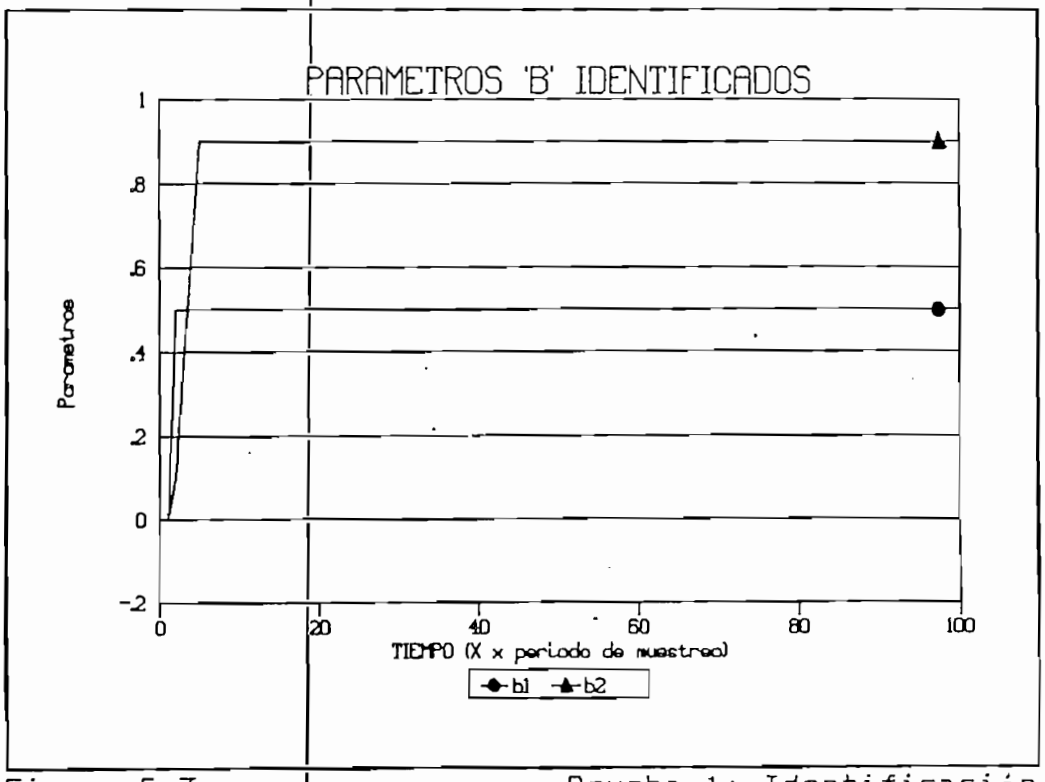


Figura 5.3 Prueba 1: Identificación

ESCUELA POLITECNICA NACIONAL

FACULTAD DE INGENIERIA ELECTRICA

TESIS: IDENTIFICACION DE SISTEMAS EN TIEMPO REAL

SIMULACION

MODELO DE LA FORMA: $$Y(z) = \frac{\Sigma\, b(i)z^{-i}}{1 + \Sigma\, a(i)z^{-i}}\, U(z)$$

1) MODELO DE LA PLANTA

Orden del Modelo = 2
A(1) = 1
A(2) = .4
B(1) = .5
B(2) = .9

2) PARAMETROS DE IDENTIFICACION Y CONTROL

Orden del modelo = 2
Parametro Alfa = 10000
Referencia = 3
U minimo = 0
U máximo = 10
Ruido Porcentual = 0 %
Número de Iteraciones = 246

3) UBICACION DE POLOS

Polinomio de polos: $T(z) = 1 + \Sigma\, t(i)z^{-i}$

t(1) = -1.32
t(2) = .5

4) RESULTADOS DE IDENTIFICACION Y CONTROL

A(1) = .9997163436681317
A(2) = .3999122648694817
B(1) = .5000306930888929
B(2) = .8997526617952615

Figura 5.4 Prueba 1: Control

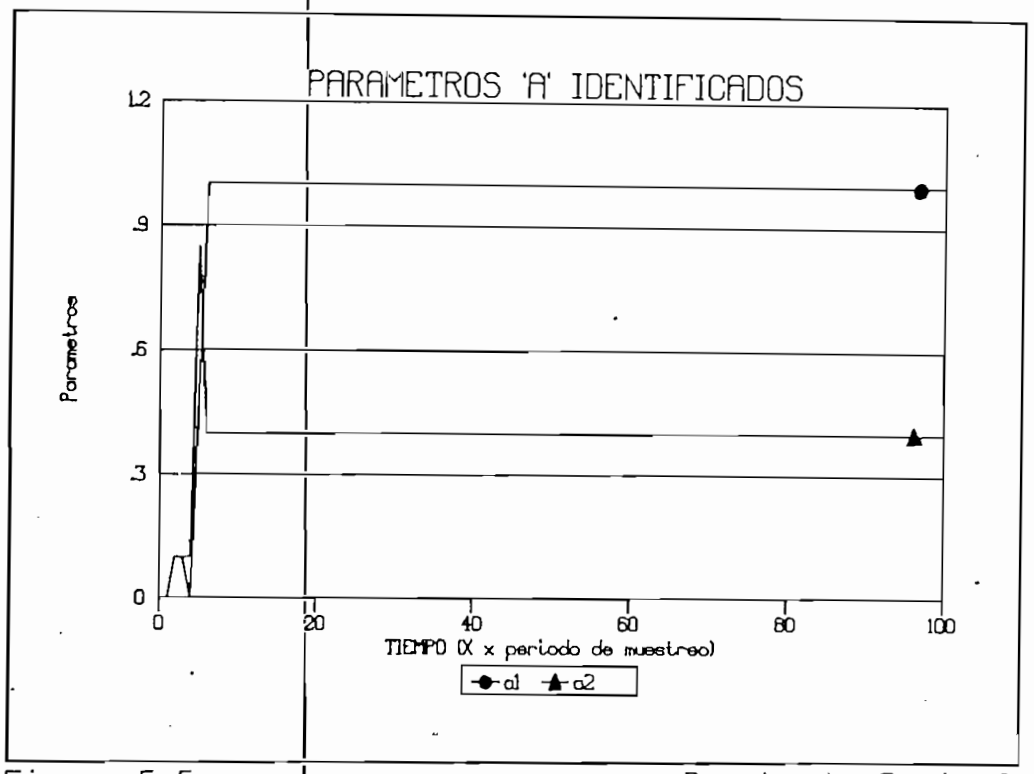


Figura 5.5 Prueba 1: Control

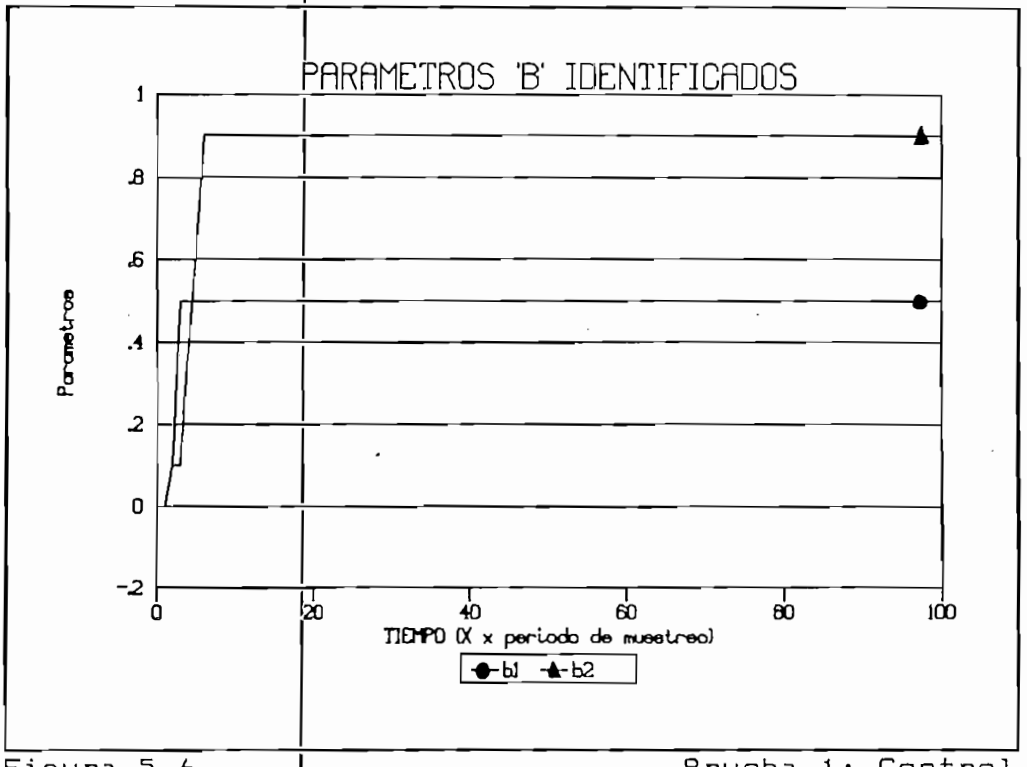


Figura 5.6 Prueba 1: Control

Dado que no existe ruido a la salida, la identificación se realiza adecuadamente, obteniéndose una rápida convergencia a los parámetros deseados (figuras 5.5 y 5.6).

b) Segunda Prueba

En esta prueba se realizan los procesos de identificación y control para una planta de tercer orden en forma similar que en la primera prueba, sólo que ahora se verifica el efecto del ruido en la salida de la planta sobre los parámetros identificados para el control así como el control en base a un modelo de segundo orden. En las páginas siguientes se muestran los reportes y gráficos correspondientes.

1.- Identificación de parámetros.

Los resultados de esta parte de la prueba se muestran en el Reporte nº 3 y los gráficos respectivos son las figuras 5.7 y 5.8.

Como se vió anteriormente, la identificación se realiza rápidamente (ver figuras 5.8 y 5.9) y con mucha precisión en base a la señal de excitación persistente (ver Reporte nº 3). La excitación persistente es de características similares a la anterior prueba.

ESCUELA POLITECNICA NACIONAL

FACULTAD DE INGENIERIA ELECTRICA

TESIS: IDENTIFICACION DE SISTEMAS EN TIEMPO REAL

SIMULACION

MODELO DE LA FORMA:

$$Y(z) = \frac{\Sigma\, b(i) z^{-i}}{1 + \Sigma\, a(i) z^{-i}}\, U(z)$$

1) MODELO DE LA PLANTA

Orden del Modelo = 3
A(1) = .5
A(2) = .4
A(3) = -.5
B(1) = .9
B(2) = -.3
B(3) = .2

2) PARAMETROS DE IDENTIFICACION

Orden del modelo = 3
Parametro Alfa = 10000
Referencia = 3
U minimo = 0
U máximo = 10
Ruido Porcentual = 20 %
Número de Iteraciones = 427

3) RESULTADOS DE IDENTIFICACION

A(1) = .4999999388202624
A(2) = .3999999401197676
A(3) = -.5000000646353036
B(1) = .8999999695906201
B(2) = -.300000056547664
B(3) = .1999999800254703

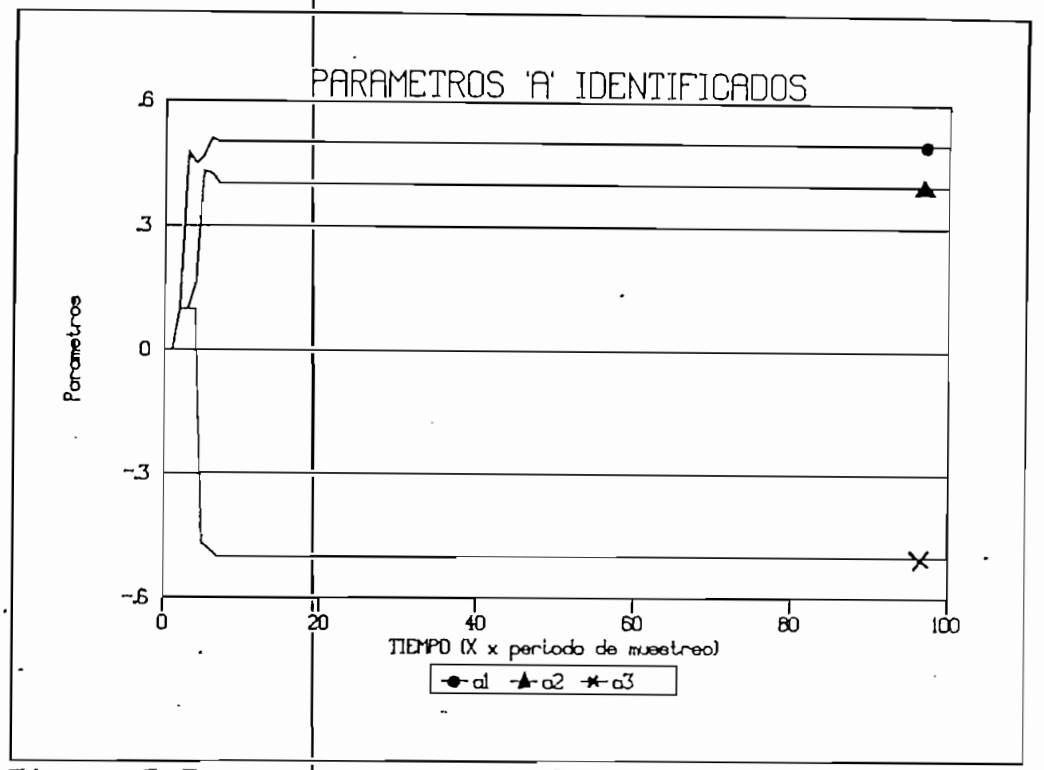


Figura 5.7 Prueba 2: Identificación

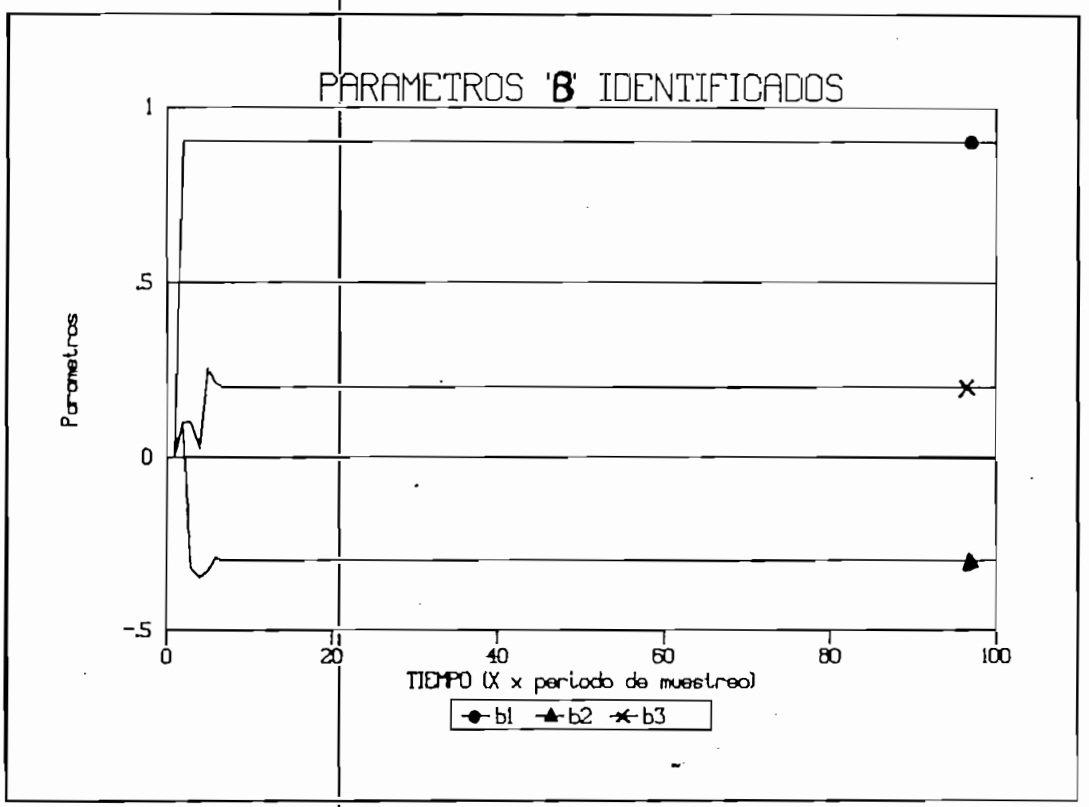


Figura 5.8 Prueba 2: Identificación

2.- Identificación y control (3° orden)

En esta parte se realizará el control de la planta de tercer orden pero con ruido a la salida (Y). El objeto de esta prueba es realizar el control en condiciones similares a las que existen en plantas reales. Se utilizan aquí los mismos polos que en la prueba anterior (ver Reporte nº 4).

Los resultados del control se muestran en la figura 5.9 en la que se aprecia la señal de control y la señal de salida de la planta. El control logra que la señal de salida se ajuste al valor de referencia (R = 5) mientras que los parámetros sufren una variación (no hay convergencia) (ver figuras 5.10 y 5.11). Esto se explica por cuanto al existir una señal de ruido y al estar cerrado el lazo de realimentación, existe también realimentación del ruido con lo que se presenta un ruido correlacionado que impide la convergencia de los mínimos cuadrados.

A diferencia de la prueba anterior, donde no existía ruido a la salida y se daba perfecta convergencia de los parámetros, aquí, con presencia de ruido, no hay una clara convergencia de parámetros. Pese a esto, el control adaptivo sigue cumpliendo su función: esto es lo que se busca al tratarse del control.

SIMULACION

MODELO DE LA FORMA: $$Y(z) = \frac{\Sigma\, b(i)z^{-i}}{1 + \Sigma\, a(i)z^{-i}}\, U(z)$$

1) MODELO DE LA PLANTA

Orden del Modelo = 3
A(1) = .5
A(2) = .4
A(3) = -.5
B(1) = .9
B(2) = -.3
B(3) = .2

2) PARAMETROS DE IDENTIFICACION Y CONTROL

Orden del modelo = 3
Parametro Alfa = 10000
Referencia = 3
U minimo = 0
U máximo = 10
Ruido Porcentual = 5 %
Número de Iteraciones = 323

3) UBICACION DE POLOS

Polinomio de polos: $T(z) = 1 + \Sigma\, t(i)z^{-i}$

t(1) = -1.32
t(2) = .5
t(3) = 0

4) RESULTADOS DE IDENTIFICACION Y CONTROL

A(1) = .6169443784545113
A(2) = .4198441660066312
A(3) = -.5662117407943313
B(1) = 1.057728546077858
B(2) = -.4408111885389456
B(3) = .2223876832591194

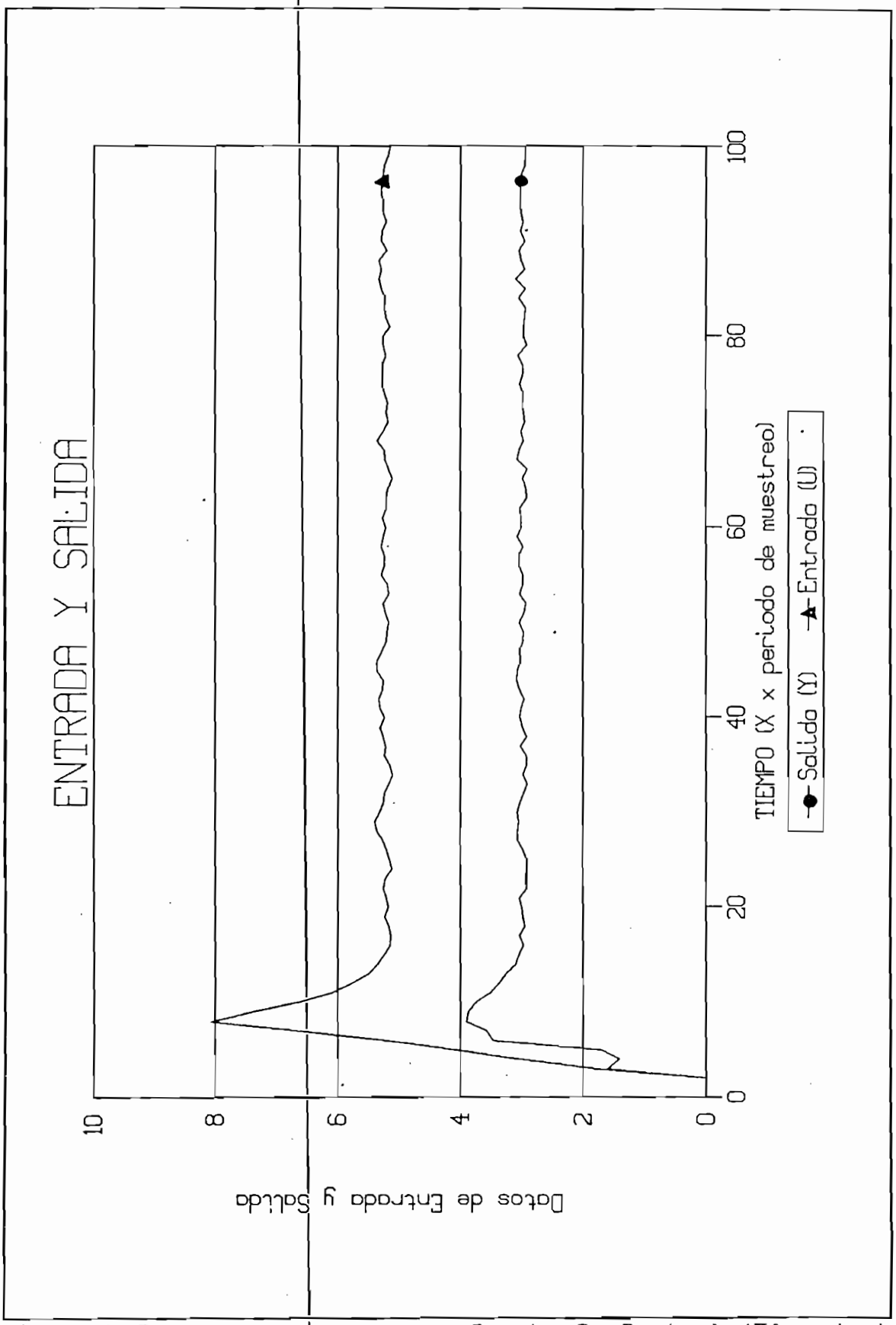


Figura 5.9 Prueba 2: Control (3° orden)

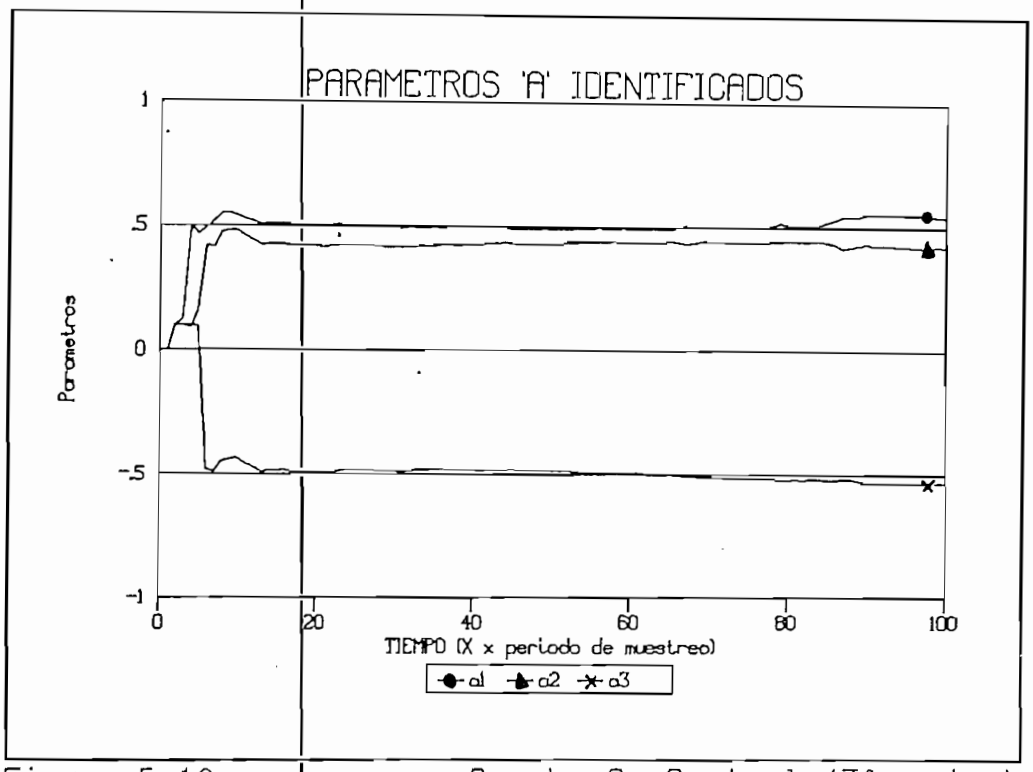


Figura 5.10 Prueba 2: Control (3° orden)

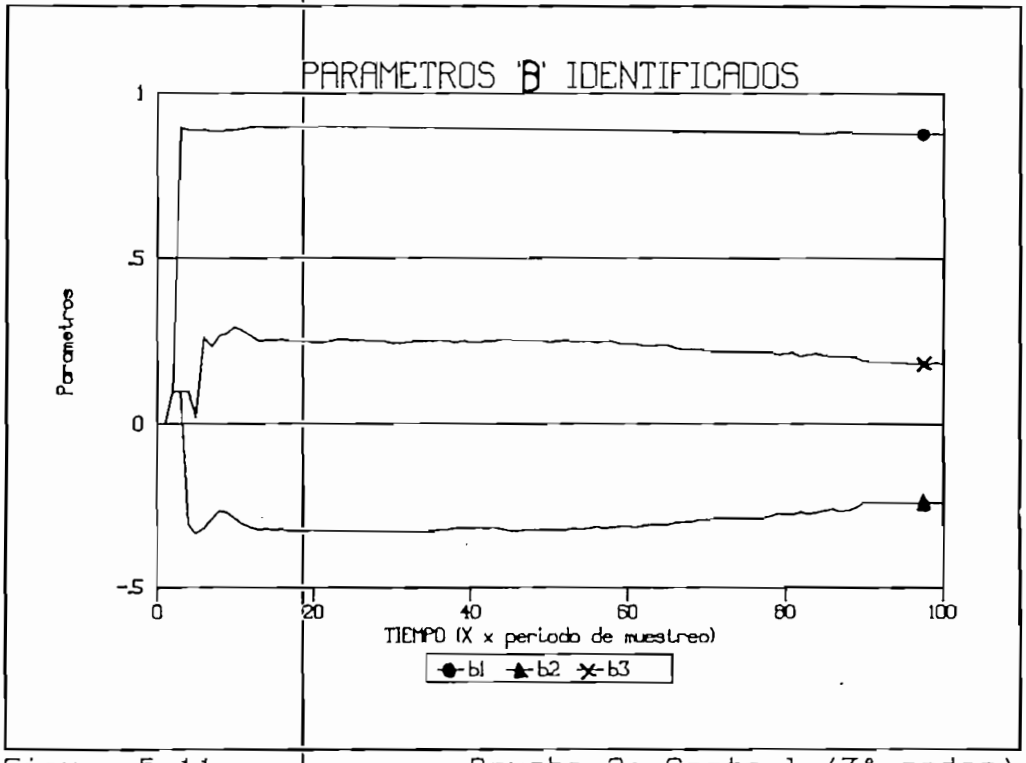


Figura 5.11 Prueba 2: Control (3° orden)

3.- Identificación y control (2° orden)

En esta parte, se busca realizar el control en base a un modelo de identificación de segundo orden para la planta de tercer orden. Los resultados se muestran en el reporte nº 5 y en las figuras 5.12, 5.13 y 5.14.

Como era de esperarse, el control es un poco menos exacto, existiendo una oscilación que se pierde en el tiempo, acampañada de ruido. La pérdida de información sobre la dinámica de la planta crea este tipo de problemas. Esto se puede corregir cambiando los polos de manera que éstos se adecúen a la nueva dinámica del modelo para así obtener un mejor control. Los parámetros, evidentemente, no alcanzan una convergencia debido a la realimentación del ruido.

De todas maneras, es posible realizar una aproximación de una planta de un orden mayor a una de menor orden, siempre y cuando se tenga presente la variación en la dinámica del modelo. Esto se realiza en la práctica para plantas que se encuentran trabajando en estado estable.

SIMULACION

MODELO DE LA FORMA: $$Y(z) = \frac{\Sigma\, b(i)z^{-i}}{1 + \Sigma\, a(i)z^{-i}}\, U(z)$$

1) MODELO DE LA PLANTA

Orden del Modelo = 3
A(1) = .5
A(2) = .4
A(3) = -.5
B(1) = .9
B(2) = -.3
B(3) = .2

2) PARAMETROS DE IDENTIFICACION Y CONTROL

Orden del modelo = 2
Parametro Alfa = 10000
Referencia = 1
U minimo = 0
U máximo = 10
Ruido Porcentual = 5 %
Número de Iteraciones = 310

3) UBICACION DE POLOS

Polinomio de polos: $T(z) = 1 + \Sigma\, t(i)z^{-i}$

t(1) = -1.32
t(2) = .5

4) RESULTADOS DE IDENTIFICACION Y CONTROL

A(1) = .9654697252921632
A(2) = .7324961514429478
B(1) = 1.115915193901 74
B(2) = .427294924993 36

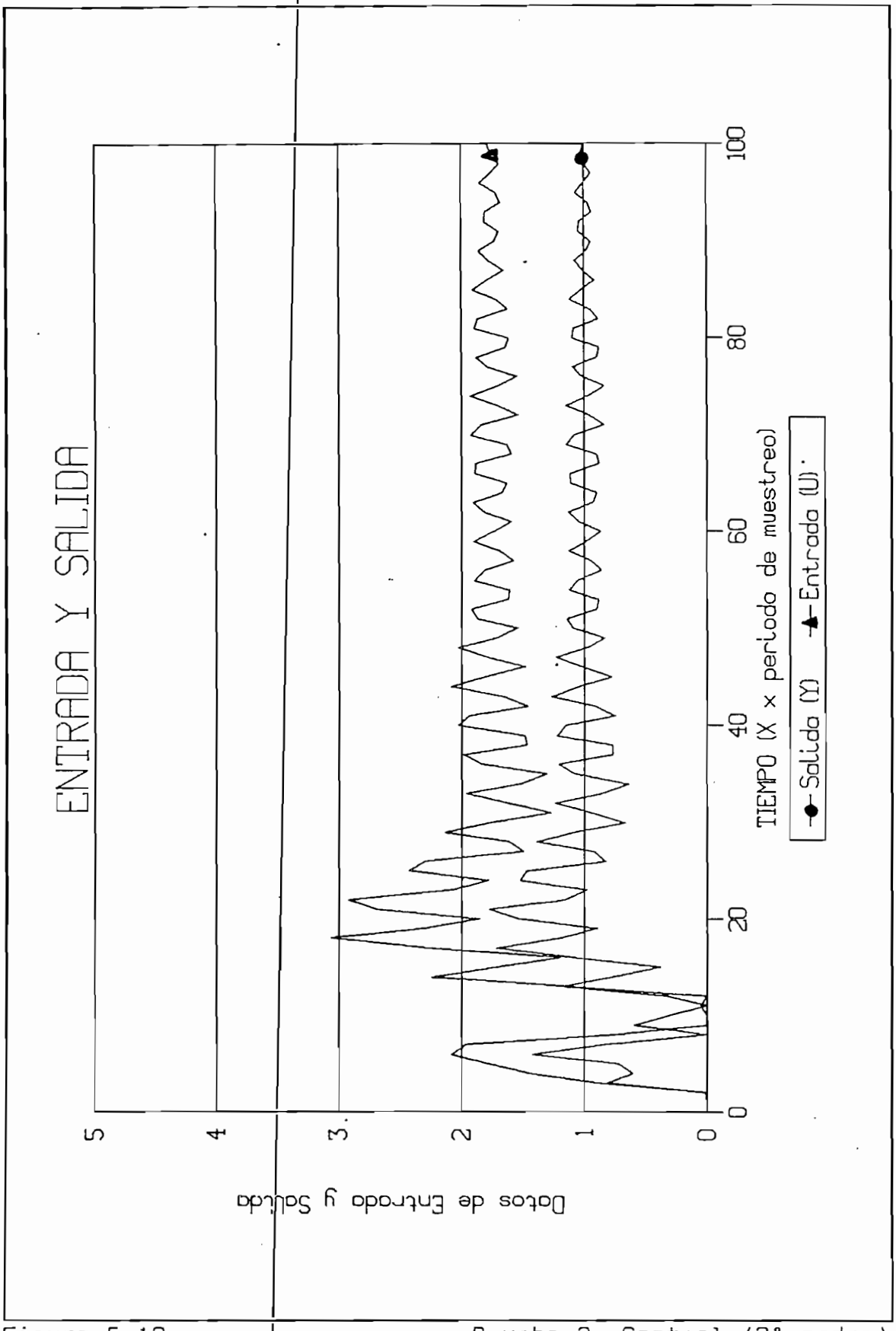


Figura 5.12 Prueba 2: Control (2° orden)

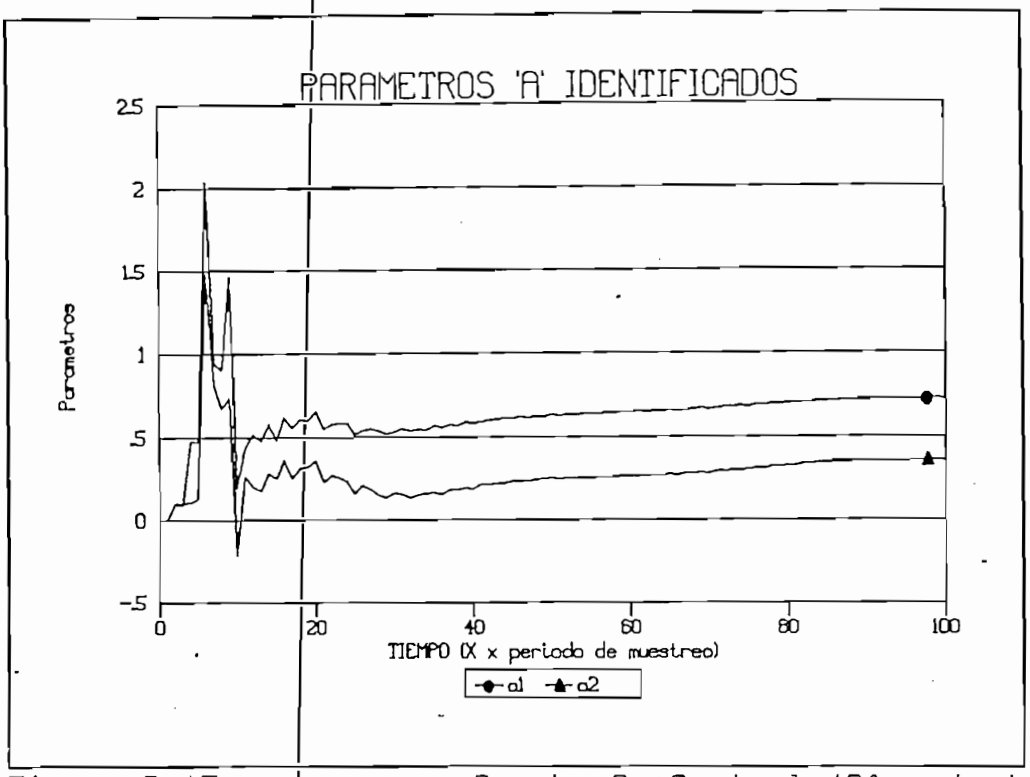


Figura 5.13 Prueba 2: Control (2° orden)

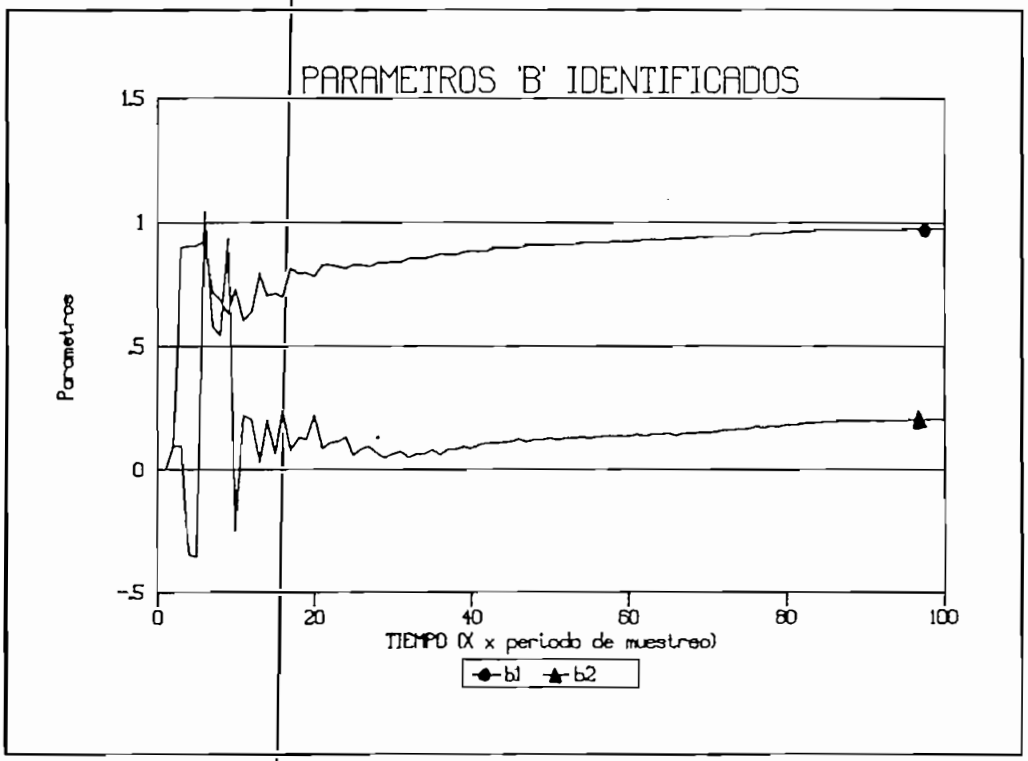


Figura 5.14 Prueba 2: Control (2° orden)

c) Tercera Prueba

La planta con la que se va a trabajar ahora es una planta de tercer orden con un fuerte retardo: los dos primeros parámatros b_i son iguales a cero. Se procederá entonces a realizar la identificación de la planta así como el control.

1.- Identificación de parámetros.

La identificación se realiza mediante la introducción de una señal de excitación persistente que consiste de un escalón con un ruido del 20%.

Los resultados se muestran en el Reporte nº 6 y las figuras 5.15 y 5.16. La identificación se dá en forma rápida, siendo los resultados los esperados. Se aprecia que los valores de a_3, b_1 y b_2 son cero (con una aproximación de 10^{-6}).

2.- Identificación y control.

Para simular un caso real se introduce un ruido pequeño (5%) en la salida al realizar el control (lazo cerrado). Los resultados de entrada y salida así como de los parámetros identificados se muestran en el Reporte nº 7 y las figuras 5.17, 5.18 y 5.19. En ellas se puede apreciar que la señal de salida se ubica en la referencia cuando se ha llegado a valores cercanos de los parámetros de la planta. La señal de control y de salida se estabilizan.

SIMULACION

MODELO DE LA FORMA: $$Y(z) = \frac{\Sigma\, b(i)z^{-i}}{1 + \Sigma\, a(i)z^{-i}}\, U(z)$$

1) MODELO DE LA PLANTA

Orden del Modelo = 3
A(1) = -.8
A(2) = .16
A(3) = 0
B(1) = 0
B(2) = 0
B(3) = 1.2

2) PARAMETROS DE IDENTIFICACION

Orden del modelo = 3
Parametro Alfa = 10000
Referencia = 2
U minimo = 0
U máximo = 10
Ruido Porcentual = 20 %
Número de Iteraciones = 303

3) RESULTADOS DE IDENTIFICACION

A(1) = -.7999994596503681
A(2) = .1599993636491028
A(3) = 8.680983990179415D-08
B(1) = 1.544247339649846D-07
B(2) = 3.009599790000546D-07
B(3) = 1.199999527763713

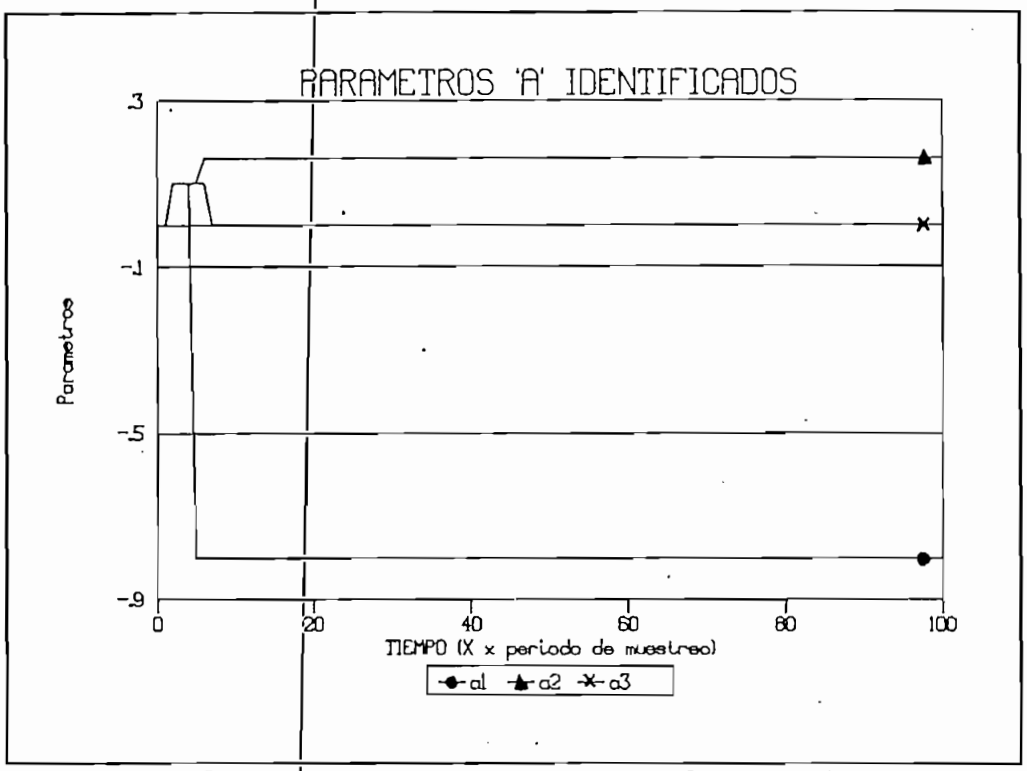


Figura 5.15 Prueba 3: Identificación

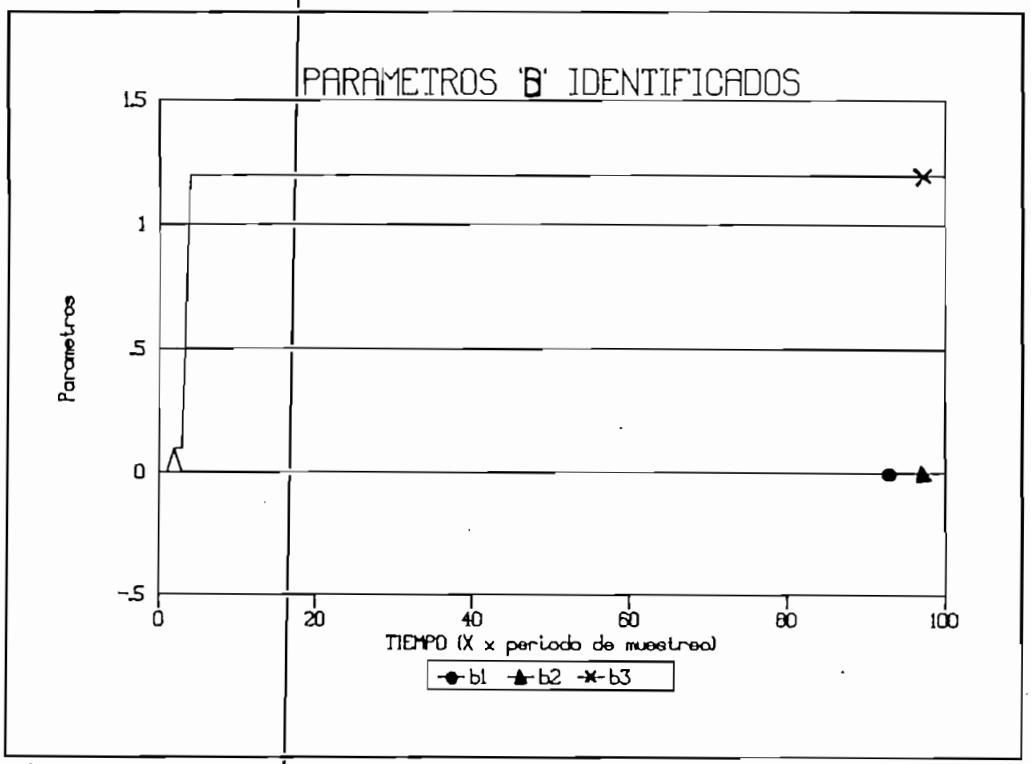


Figura 5.16 Prueba 3: Identificación

SIMULACION

MODELO DE LA FORMA: $$Y(z) = \frac{\Sigma\, b(i)z^{-i}}{1 + \Sigma\, a(i)z^{-i}}\, U(z)$$

1) MODELO DE LA PLANTA

Orden del Modelo = 3
A(1) = -.8
A(2) = .16
A(3) = 0
B(1) = 0
B(2) = 0
B(3) = 1.2

2) PARAMETROS DE IDENTIFICACION Y CONTROL

Orden del modelo = 3
Parametro Alfa = 10000
Referencia = 3
U minimo = 0
U máximo = 10
Ruido Porcentual = 5 %
Número de Iteraciones = 454

3) UBICACION DE POLOS

Polinomio de polos: $T(z) = 1 + \Sigma\, t(i)z^{-i}$

t(1) = -1.32
t(2) = .5
t(3) = 0

4) RESULTADOS DE IDENTIFICACION Y CONTROL

A(1) = -.8003896836354046
A(2) = .1685583621487389
A(3) = -1.098851566245744D-02
B(1) = -5.557733423149758D-02
B(2) = 3.500281189875284D-03
B(3) = 1.242367147944185

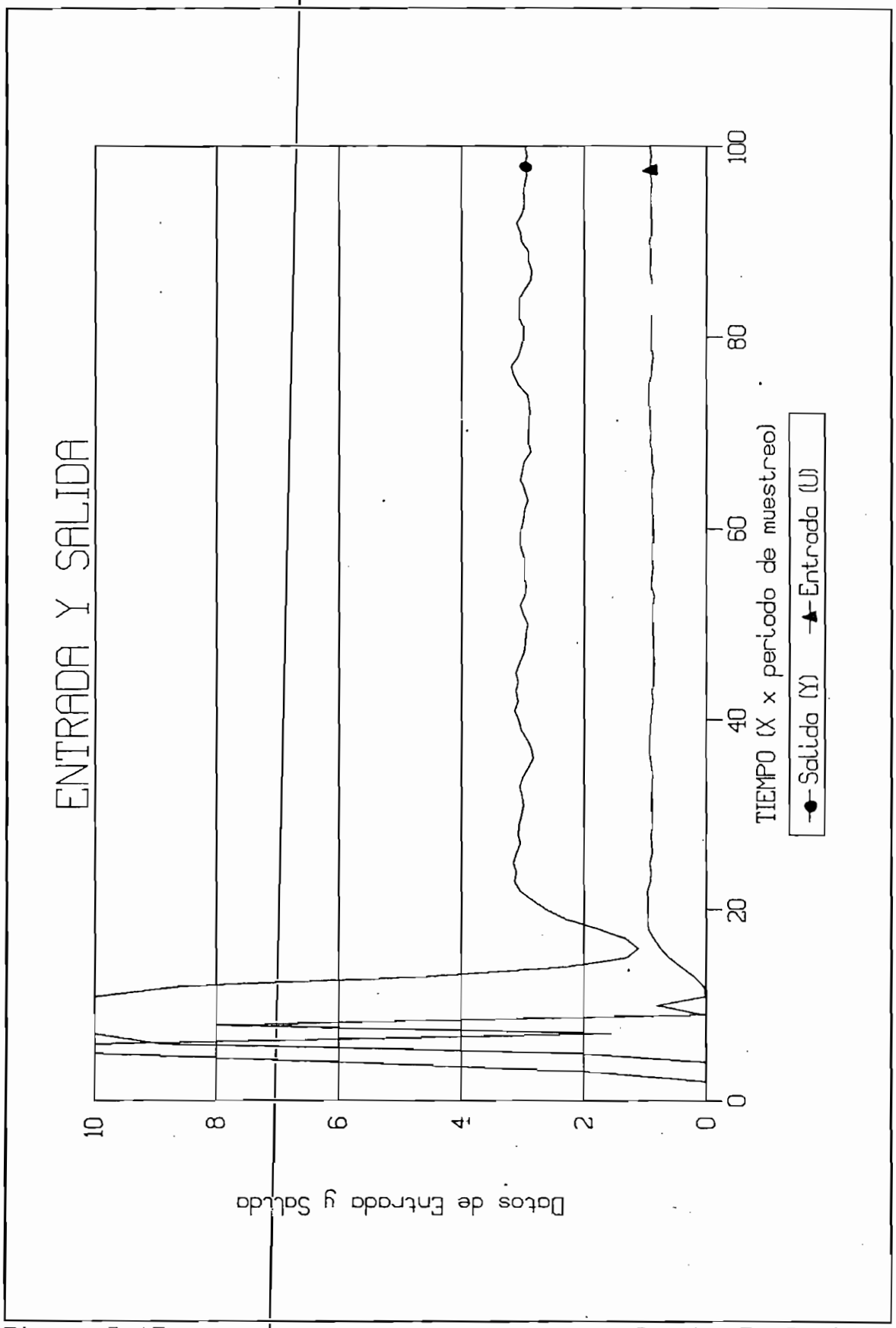


Figura 5.17 Prueba 3: Control

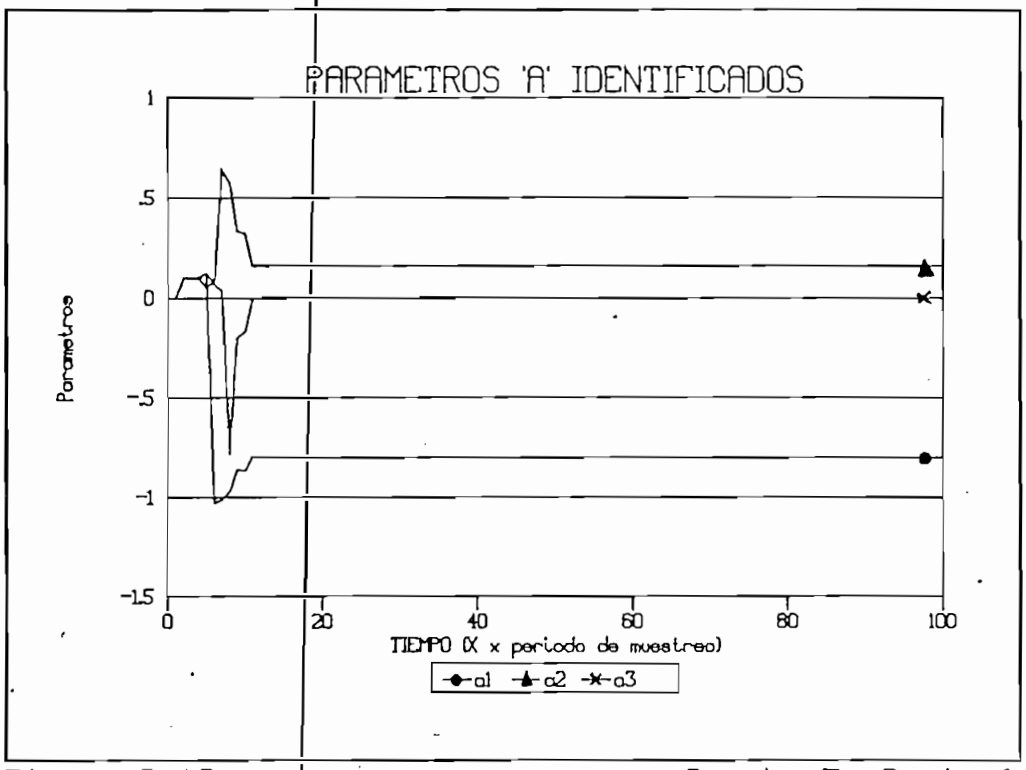


Figura 5.18 Prueba 3: Control

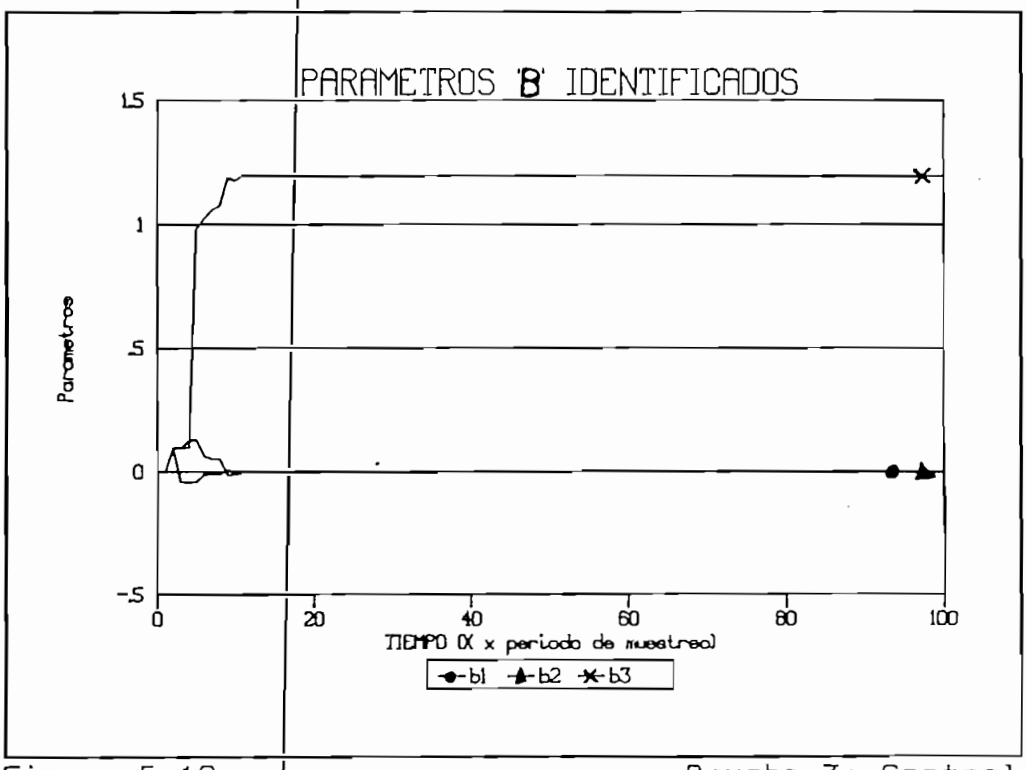


Figura 5.19 Prueba 3: Control

d) Cuarta Prueba

En la cuarta prueba se trabaja con un modelo de cuarto orden, en base al cual se procederá a realizar la identificación de parámetros y el control. Para el control se realizarán dos experiencias: la primera con un modelo de identificación de cuarto orden, y la segunda con un modelo de identificación de segundo orden.

1.- Identificación de parámetros.

La identificación de parámetros se realiza similarmente a los procesos anteriores de identificación. Los resultados (ver Reporte nº 8 y figuras 5.20 y 5.21) son totalmente satisfactorios en relación a la estimación de los parámetros de la planta. La convergencia se dá rápidamente a los valores reales de los parámetros.

2.- Identificación y control (4° orden)

En esta parte se realiza el control de la planta en base a un modelo de cuarto orden. Dado que existe ruido en la salida para simular condiciones reales, la convergencia de los parámetros se ve afectada debido a la realimentación del ruido (ver figuras 5.23 y 5.24). Esto no afecta al control, pues se ve que la señal de salida se ajusta a la referencia, siendo bastante estable la señal de control. Los resultados se muestran en el Reporte nº 9.

SIMULACION

MODELO DE LA FORMA: $$Y(z) = \frac{\Sigma\, b(i)z^{-i}}{1 + \Sigma\, a(i)z^{-i}}\, U(z)$$

1) MODELO DE LA PLANTA

Orden del Modelo = 4
A(1) = .2
A(2) = .4
A(3) = -.5
A(4) = -.3
B(1) = .8
B(2) = .5
B(3) = -.3
B(4) = .4

2) PARAMETROS DE IDENTIFICACION

Orden del modelo = 4
Parametro Alfa = 10000
Referencia = 2
U minimo = 0
U máximo = 10
Ruido Porcentual = 20 %
Número de Iteraciones = 218

3) RESULTADOS DE IDENTIFICACION

A(1) = .1999970462344161
A(2) = .4000005095521382
A(3) = -.5000012606082154
A(4) = -.2999976717014644
B(1) = .7999984398187908
B(2) = .4999971264146461
B(3) = -.2999986087629179
B(4) = .4000006349546449

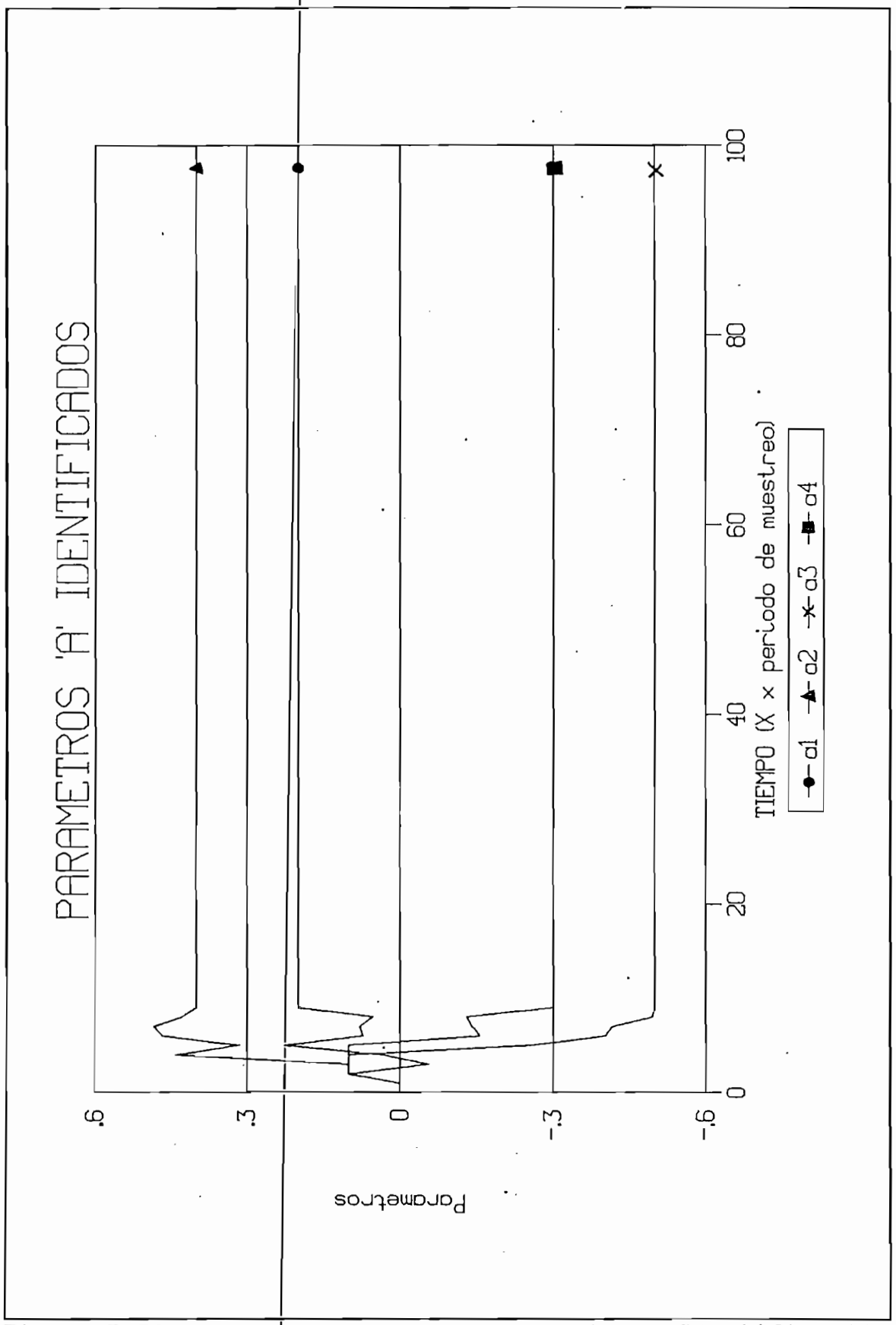


Figura 5.20 Prueba 4: Identificación

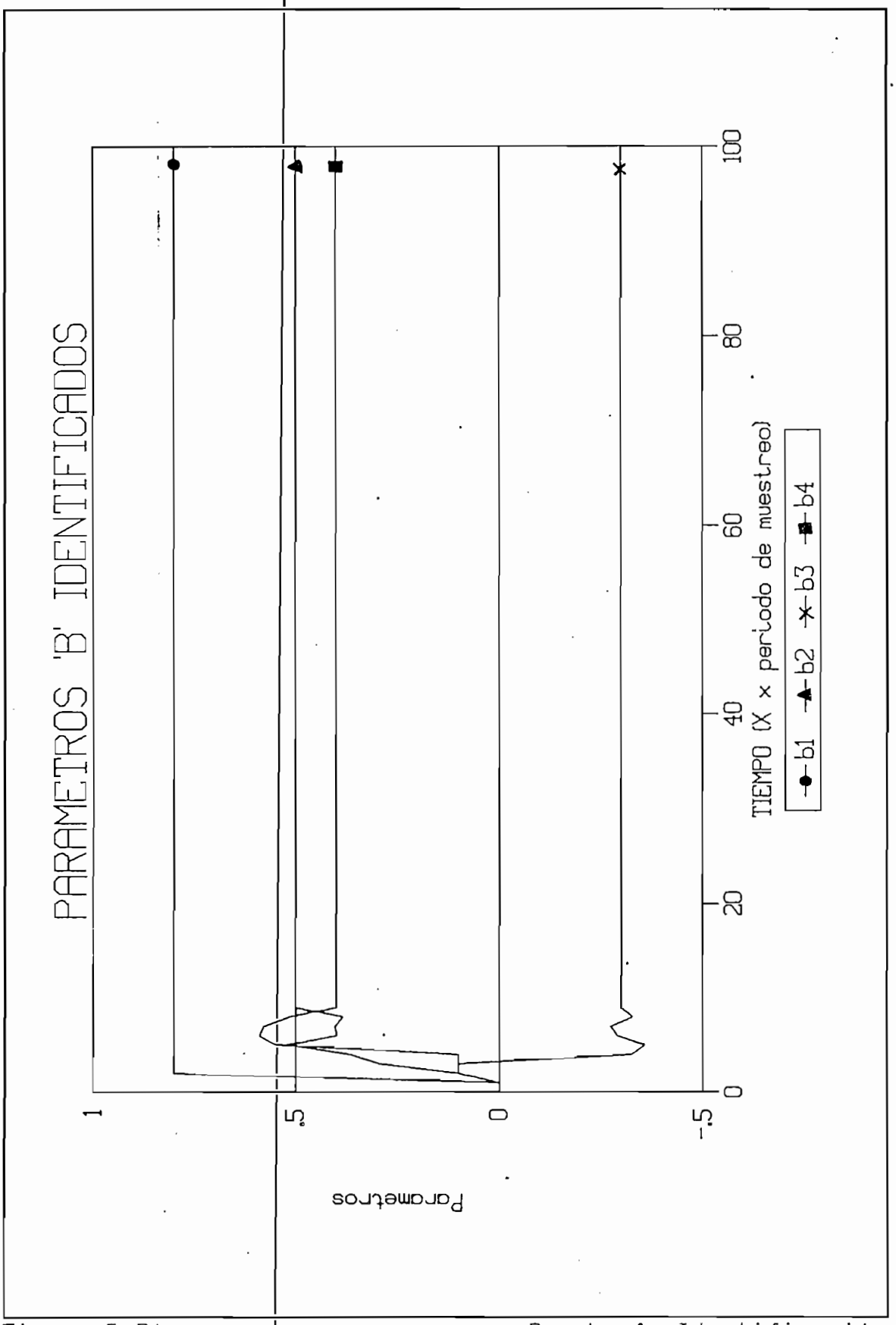


Figura 5.21 Prueba 4: Identificación

SIMULACION

MODELO DE LA FORMA: $$Y(z) = \frac{\Sigma\, b(i)z^{-i}}{1 + \Sigma\, a(i)z^{-i}}\, U(z)$$

1) MODELO DE LA PLANTA

Orden del Modelo = 4
A(1) = .2
A(2) = .4
A(3) = -.5
A(4) = -.3
B(1) = .8
B(2) = .5
B(3) = -.3
B(4) = .4

2) PARAMETROS DE IDENTIFICACION Y CONTROL

Orden del modelo = 4
Parametro Alfa = 10000
Referencia = 3
U minimo = 0
U máximo = 10
Ruido Porcentual = 5 %
Número de Iteraciones = 183

3) UBICACION DE POLOS

Polinomio de polos: $T(z) = 1 + \Sigma\, t(i)z^{-i}$

t(1) = -1.32
t(2) = .5
t(3) = 0

4) RESULTADOS DE IDENTIFICACION Y CONTROL

A(1) = .2288794176428081
A(2) = .4310797767395594
A(3) = -.5570845795992289
A(4) = -.2855597890389361
B(1) = .8061262690703187
B(2) = .5330223609140284
B(3) = -.2788746340145793
B(4) = .3698867381479382

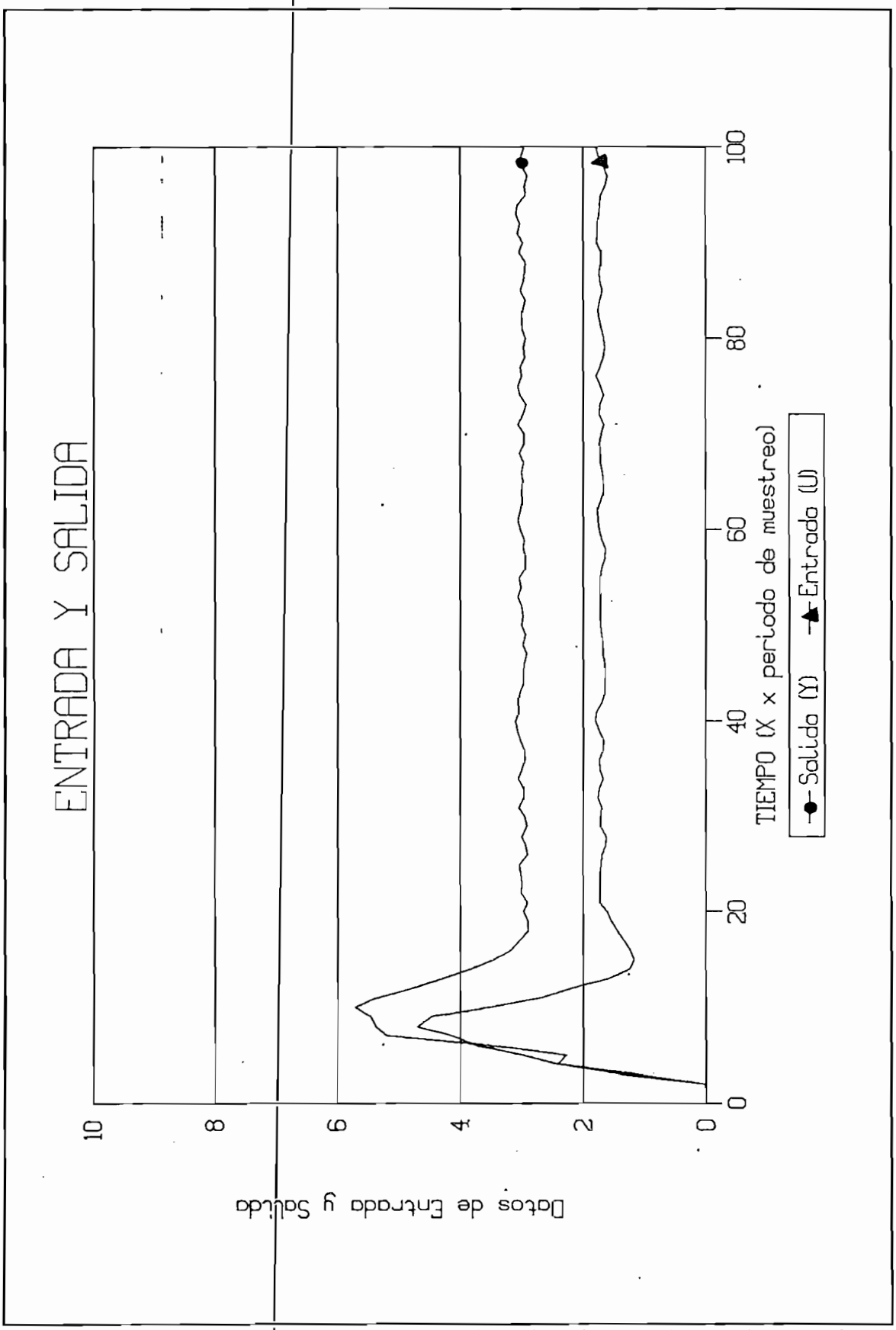


Figura 5.22 Prueba 4: Control (4° orden)

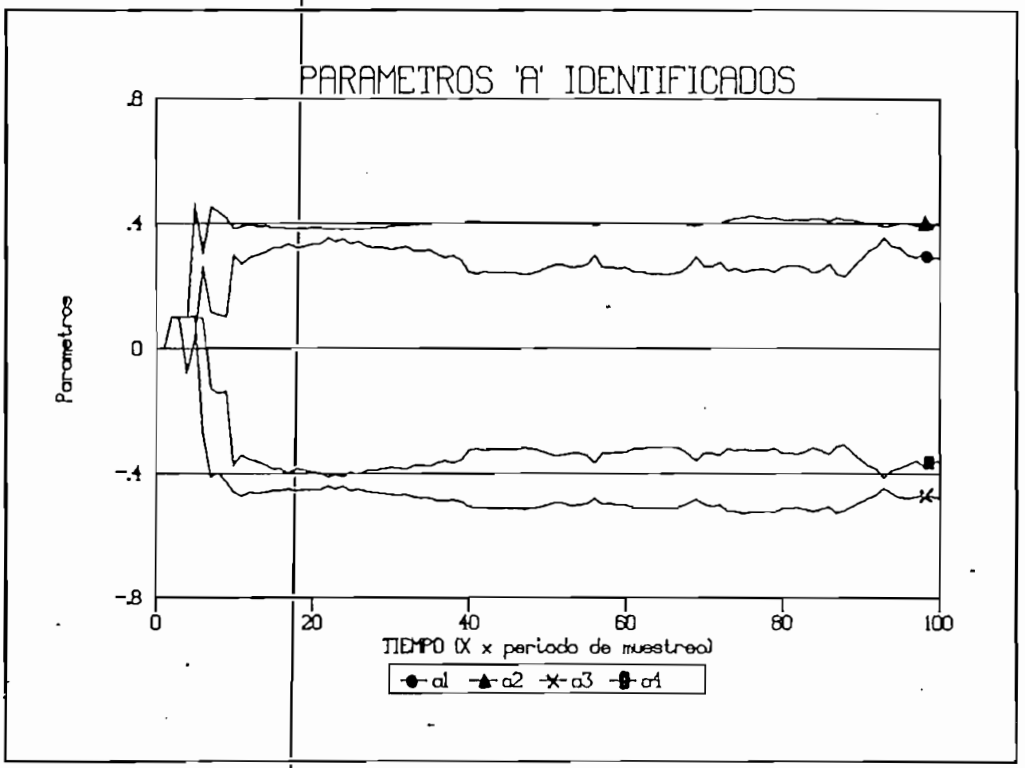


Figura 5.23 Prueba 4: Control (4° orden)

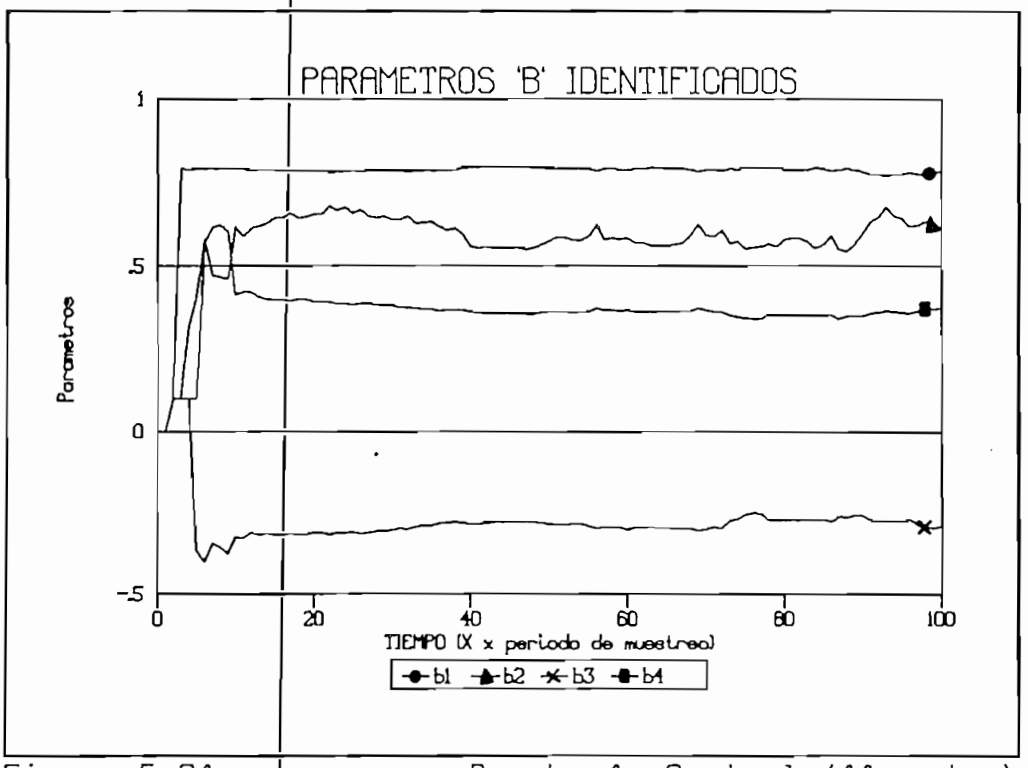


Figura 5.24 Prueba 4: Control (4° orden)

3.- Identificación y control (2° orden)

La última parte de esta prueba consiste en la realización del control de la planta de cuarto orden mediante un modelo de segundo orden identificado cuyos parámetros se utilizan para el diseño de la ley de control. La planta es simplificada por tanto a un modelo de segundo orden.

Los resultados se muestran en el Reporte nº 10 y en las figuras 5.25, 5.26 y 5.27, correspondientes a los valores de entrada y salida, a los parámetros a_i, y a los parámetros b_i. Pese a ser un modelo identificado de un orden menor, se puede apreciar que el control logra su objetivo al estabilizar la señal de salida el el valor de la referencia. A diferencia del caso anterior, aquí la señal de salida tiene un tiempo más largo para su estabilización. Como se podía preveer, los parámetros no tienen una perfecta convergencia.

SIMULACION

MODELO DE LA FORMA: $$Y(z) = \frac{\Sigma\, b(i)z^{-i}}{1 + \Sigma\, a(i)z^{-i}}\, U(z)$$

1) MODELO DE LA PLANTA

Orden del Modelo = 4
A(1) = .2
A(2) = .4
A(3) = -.5
A(4) = -.3
B(1) = .8
B(2) = .5
B(3) = -.3
B(4) = .4

2) PARAMETROS DE IDENTIFICACION Y CONTROL

Orden del modelo = 2
Parametro Alfa = 10000
Referencia = 8
U minimo = 0
U máximo = 10
Ruido Porcentual = 5 %
Número de Iteraciones = 529

3) UBICACION DE POLOS

Polinomio de polos: $T(z) = 1 + \Sigma\, t(i)z^{-i}$

t(1) = -1.32
t(2) = .5

4) RESULTADOS DE IDENTIFICACION Y CONTROL

A(1) = .1514024202872753
A(2) = .1640571445452206
B(1) = 1.807333609840823
B(2) = .496881048154486

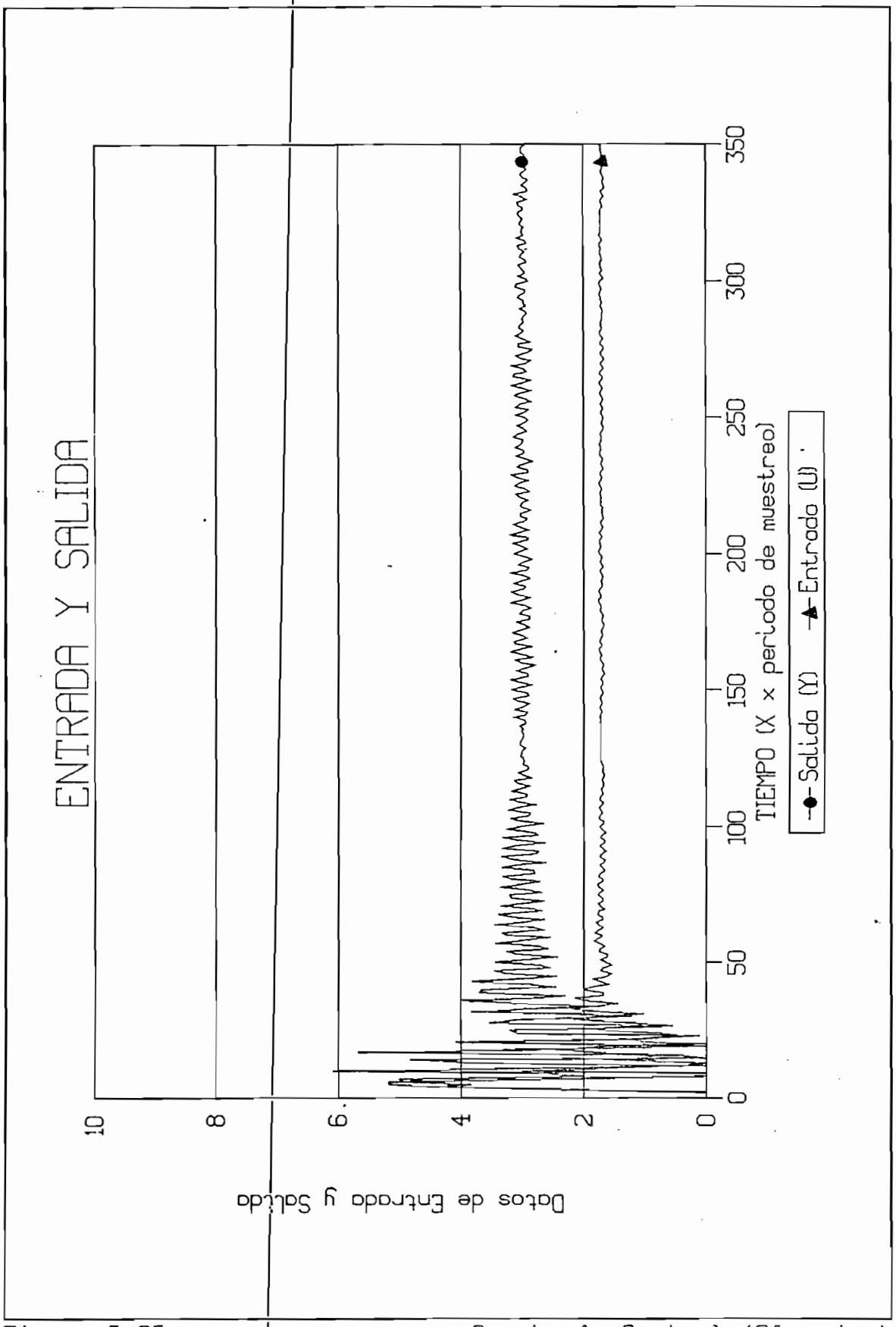


Figura 5.25 Prueba 4: Control (2° orden)

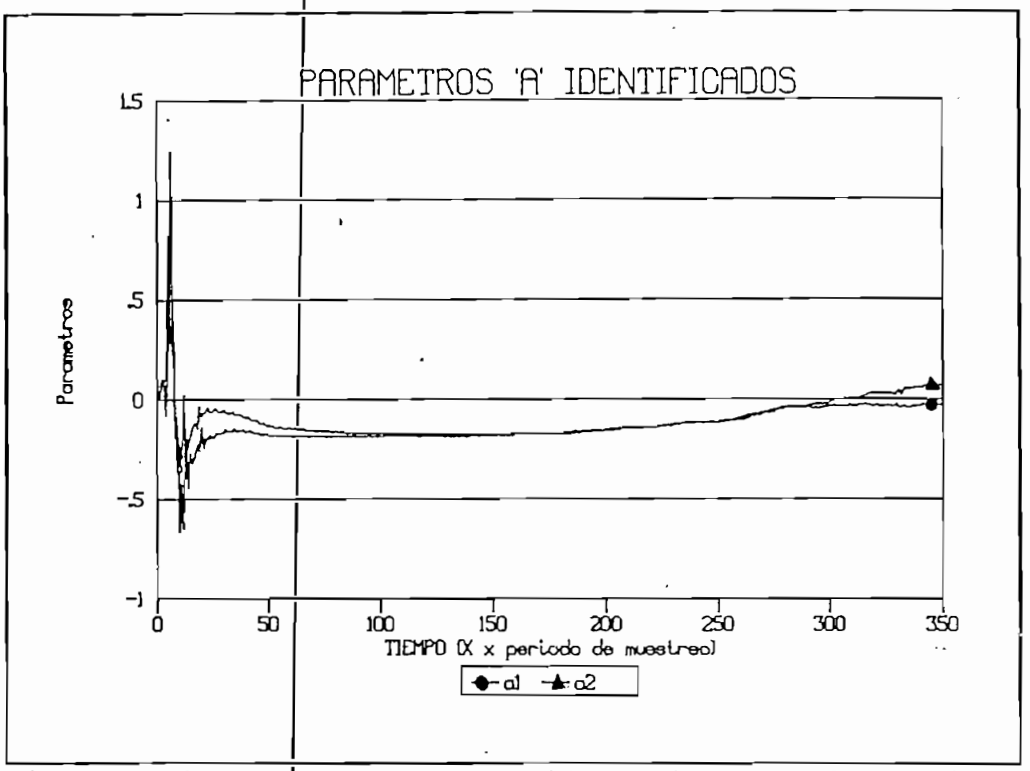


Figura 5.26 Prueba 4: Control (2° orden)

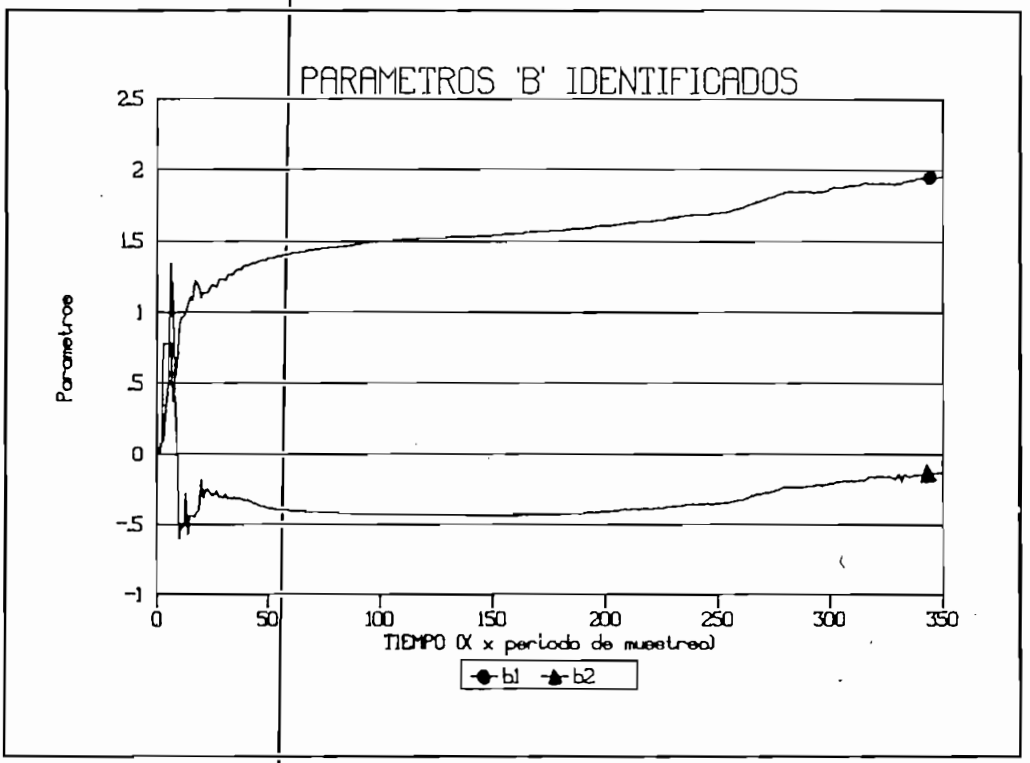


Figura 5.27 Prueba 4: Control (2° orden)

5.2. IDENTIFICACION Y CONTROL EN TIEMPO REAL

A diferencia de la simulación, las pruebas en tiempo real se realizarán usando plantas físicas a identificar y controlar. El computador se encarga de generar la señal de control y de medir la señal de salida, tanto para la identificación de parámetros, cuanto para el control propiamente dicho. La transferencia de la señal se realiza a través del equipo de adquisición de datos Keithley anteriormente descrito.

Para el tiempo real se van a utilizar dos tipos de plantas: una de segundo orden y una de primer orden.

La planta de primer orden consiste en un circuito RC, cuyos valores son: R = 10 kΩ y C = 1000 μF. El circuito se presenta en el diagrama 5.1 que se muestra a continuación.

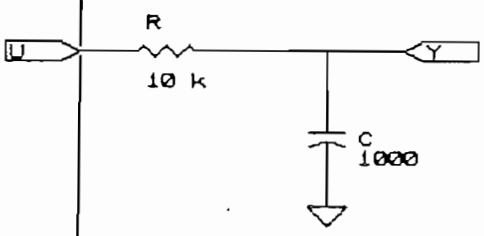


Diagrama 5.1 Planta de primer orden

La planta de segundo orden es un circuito RC en cascada que consta de dos resistencias y de dos condensadores tal como se muestra en el diagrama 5.2. Los valores de los elementos

utilizados son: R = 10 kΩ y C = 100 μF. Para realizar variaciones de la planta se añadirá en algunas pruebas un condensador de 1000 μF en paralelo a la salida.

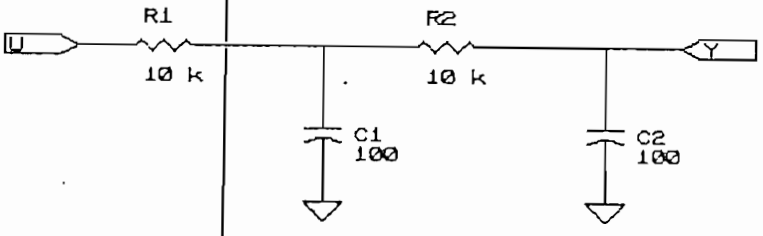


Diagrama 5.2 Planta de segundo orden

a) Quinta Prueba

Esta prueba consiste en realizar la identificación de parámetros para la planta de segundo orden.

En tiempo real es necesario definir el periodo de muestreo para la identificación. Para esta prueba se trabajará con un valor de 400 ms.

1.- Identificación de parámetros (2° orden)

Aquí se buscará estimar los parámetros en base a un modelo de identificación de segundo orden. Para esto, al igual que para la simulación, se alimentará a la planta con una señal de excitación persistente generada por el computador y que

llegará a la planta a través del equipo de adquisición de datos. Para obtener una adecuada excitación, el ruido añadido al escalón es del 50%.

Los resultados se muestran en el Reporte nº 11 y en los gráficos 5.30 y 5.31. Dado que las pruebas se han realizado en tiempo real, se tiene ahora resultados obtenidos en un grafizador para la señal de entrada y la señal de salida. Estos gráficos se muestran en las figuras 5.28 y 5.29.

La señal de entrada es discreta por ser obtenida desde el computador a través del conversor D/A del equipo de adquisición de datos y correponde a la excitación persistente. La señal de salida es la respuesta de la planta a la señal de excitación; esta señal es contínua y es discretizada por el equipo de adquisición de datos para ser procesada por el computador. Con esta información se procede a la identificación.

En las figuras 5.30 y 5.31 se presentan las curvas de convergencia de los parámetros. Como aquí se trata de tiempo real, existen pequeñas variaciones en los parámetros debido a perturbaciones, ruido y errores en la aproximación de los valores contínuos a discretos. No obstante esto existe una clara tendencia en la convergencia de los parámetros, obteniéndose un valor promedio que corresponde al resultado final.

TIEMPO REAL

1) PARAMETROS DE TIEMPO REAL

Tiempo de muestreo = 400 ms

2) PARAMETROS DE IDENTIFICACION

Orden del modelo = 2
Parametro Alfa = 10000
Referencia = 5
U minimo = 0
U máximo = 10
Ruido Porcentual = 50 %
Número de Iteraciones = 1240

3) RESULTADOS DE IDENTIFICACION

A(1) = -1.587376378454333
A(2) = .6307932733873145
B(1) = 8.482750603635582D-04
B(2) = 4.249328883525075D-02

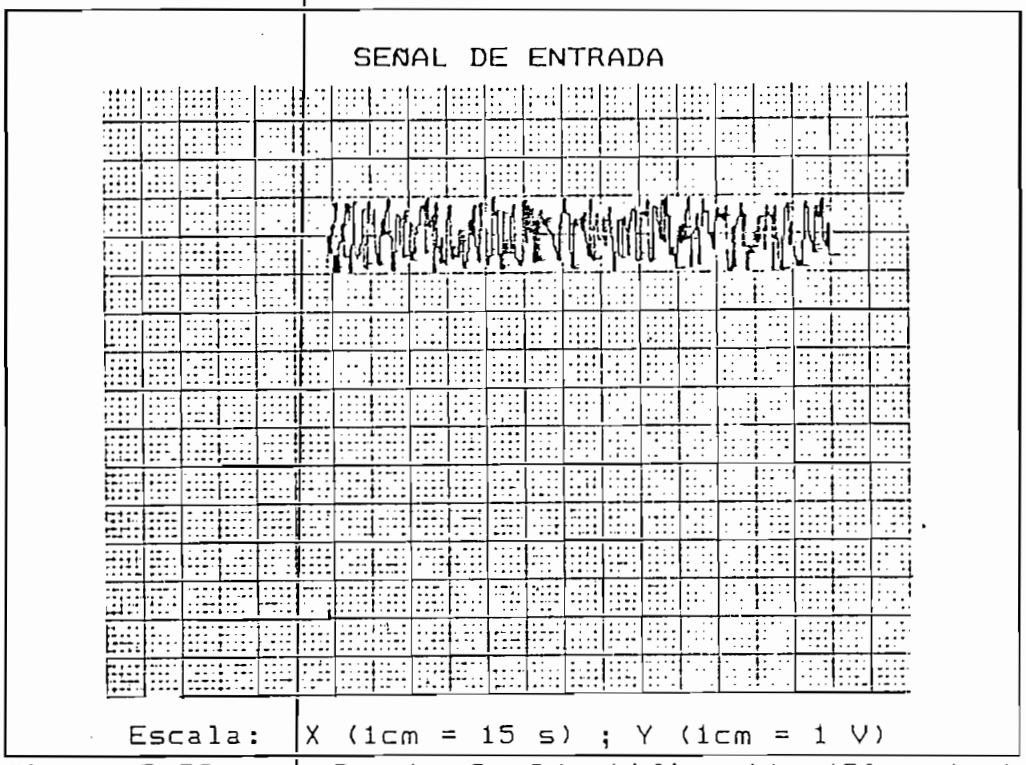


Figura 5.28 Prueba 5: Identificación (2° orden)

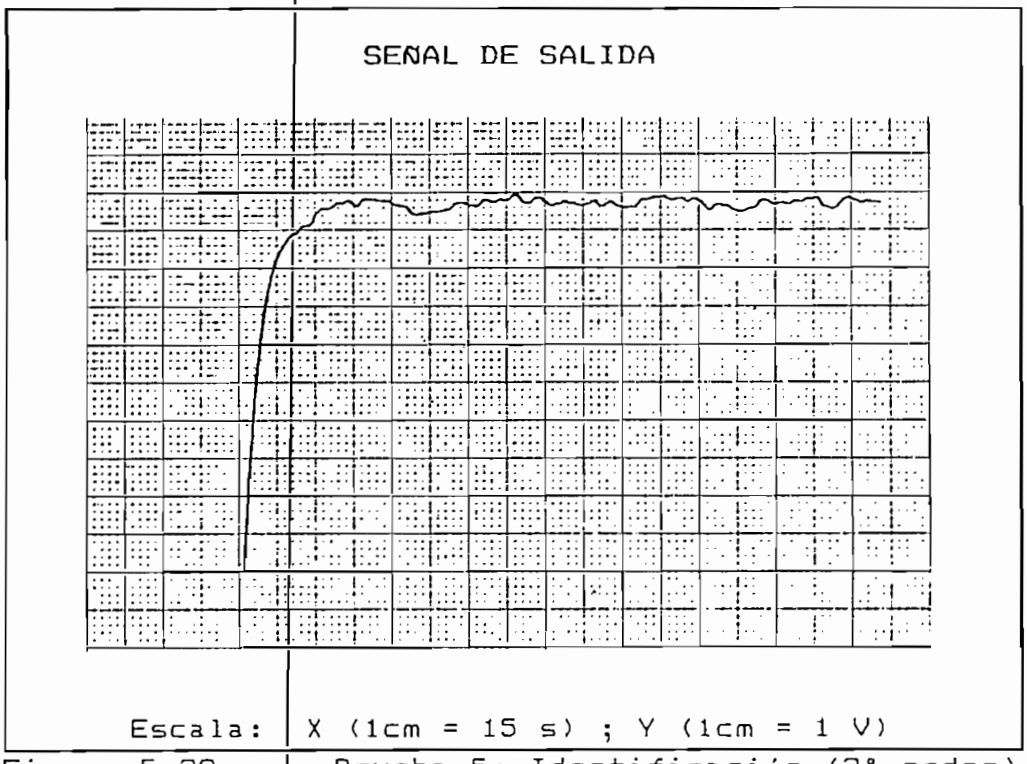


Figura 5.29 Prueba 5: Identificación (2° orden)

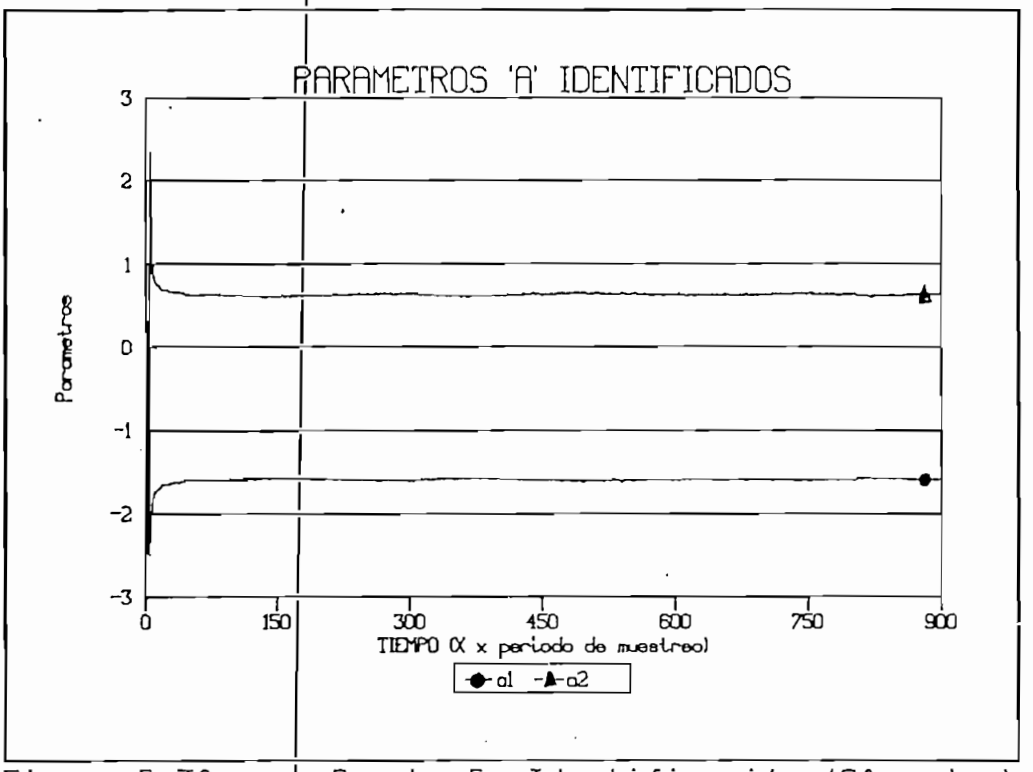


Figura 5.30 Prueba 5: Identificación (2° orden)

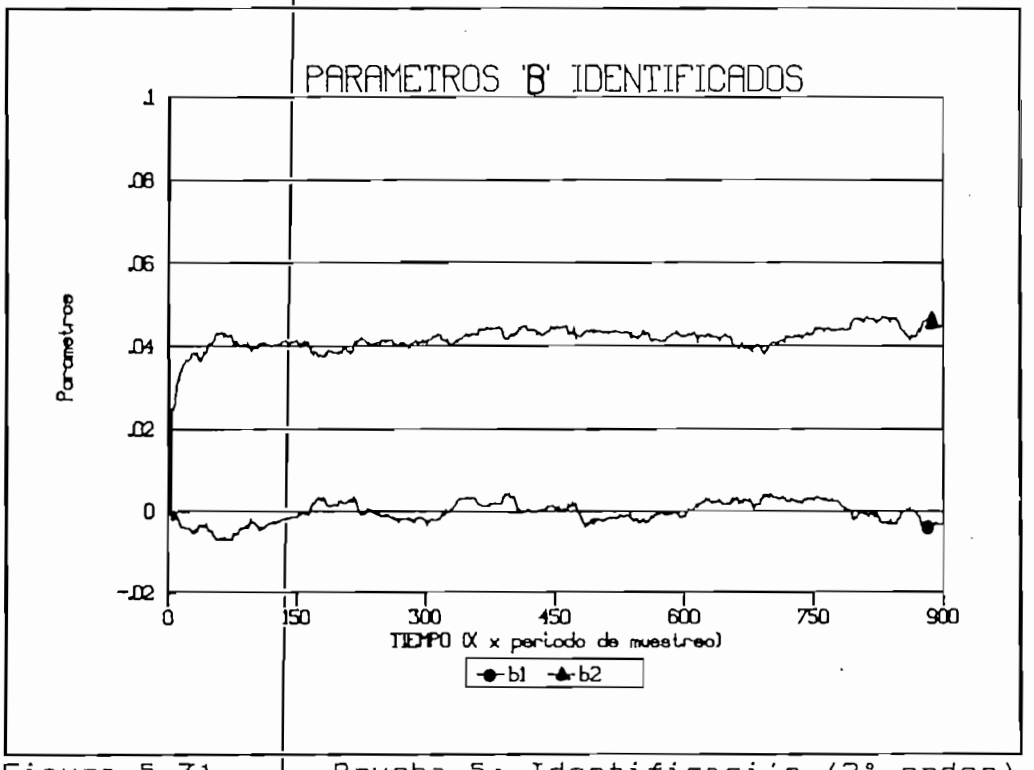


Figura 5.31 Prueba 5: Identificación (2° orden)

2.- Identificación de parámetros (3° orden)

En tiempo real, la presencia de los instrumentos de medida y las perturbaciones y retardos propios del sistema crean condiciones que vuelven más complejos los modelos. Por esta razón se trata ahora de estimar los parámetros de la planta usando un modelo de tercer orden que recoja esas característica adicionalmente a las de la planta como tal.

Usando el mismo tiempo de muestreo y alimentando la planta con la misma señal de excitación persistente, se obtienen los resultados que se muestran en el Reporte nº 12 y las figuras 5.32 y 5.33.

La convergencia de los parámetros es más lenta que para la simulación pero ésta se realiza con un número mayor de iteraciones. Las pequeñas variaciones presentes, menores que para el segundo orden tienen las mismas causas anteriores. Existe no obstante un valor promedio de convergencia, por lo que ha sido posible identificar es decir, explicar la dinámica de la planta en base a los parámetros de un modelo de tercer orden. Para este caso (planta de segundo orden) resulta adecuada la identificación de los parámetros con un modelo de tercer orden para incluir los efectos de medición y traslado de la información.

TIEMPO REAL

1) PARAMETROS DE TIEMPO REAL

Tiempo de muestreo = 400 ms

2) PARAMETROS DE IDENTIFICACION

Orden del modelo = 3
Parametro Alfa = 10000
Referencia = 5
U minimo = 0
U máximo = 10
Ruido Porcentual = 50 %
Número de Iteraciones = 1285

3) RESULTADOS DE IDENTIFICACION

A(1) = -1.232017205191199
A(2) = .2811859396689912
A(3) = 2.181222077332559D-02
B(1) = -3.129648365532331D-04
B(2) = 4.134452133653811D-02
B(3) = 2.985949753247799D-02

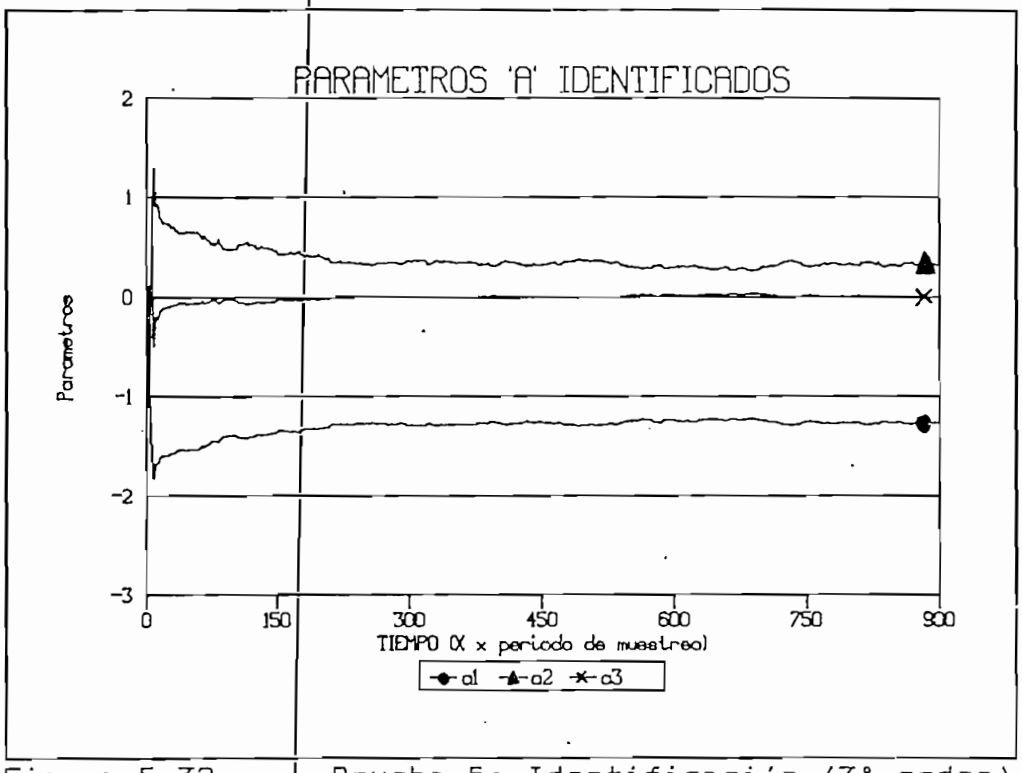


Figura 5.32 Prueba 5: Identificación (3° orden).

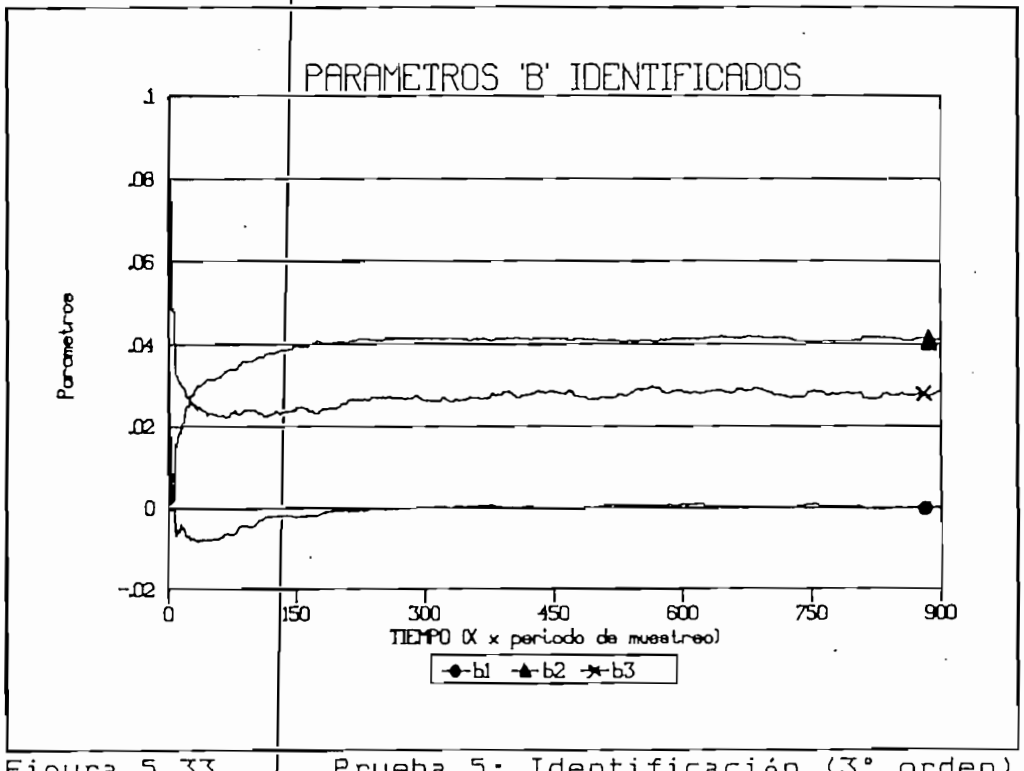


Figura 5.33 Prueba 5: Identificación (3° orden)

3.- Identificación con variación de planta

La planta de segundo orden sufrirá aquí una súbita variación en su dinámica debido al cambio de uno de sus componentes físicos. Esto se realiza añadiendo un condensador de 1000 μF en paralelo con el de la salida, con lo que la planta se vuelve mucho más lenta. Al realizarse la identificación, el algoritmo debe reaccionar a esta variación, identificando los parámetros de la nueva planta. El Reporte nº 13 y las figuras 5.34 y 5.35 muestran los resultados con una identificación paramétrica de tercer orden.

A partir de los resultados se puede encontrar el punto en que cambió la dinámica de la planta y se encuentra también las dos tendencias de convergencia diferentes de los parámetros de la planta. Al final, se habrán obtenido los parámetros de la planta más lenta.

4.- Identificación con doble variación de planta

A diferencia del ejemplo anterior, la identificación se realizará cambiando la planta a su condición más lenta con el condensador añadido y volviendo a retirarlo. El efecto de estos cambios deben reflejarse en la identificación.

Como se puede ver en el Reporte nº 14, se han alcanzado los valores de la planta inicial de segundo orden. En las figuras 5.36 y 5.37 se aprecian los dos cambios.

TIEMPO REAL

1) PARAMETROS DE TIEMPO REAL

Tiempo de muestreo = 400 ms

2) PARAMETROS DE IDENTIFICACION

Orden del modelo = 3
Parametro Alfa = 10000
Referencia = 5
U minimo = 0
U máximo = 10
Ruido Porcentual = 50 %
Número de Iteraciones = 1230

3) RESULTADOS DE IDENTIFICACION

A(1) = -1.0452136100236
A(2) = -.2102799596587042
A(3) = .2654201928728577
B(1) = -9.508560198011648D-04
B(2) = 5.188653220421337D-03
B(3) = 5.714370306029975D-03

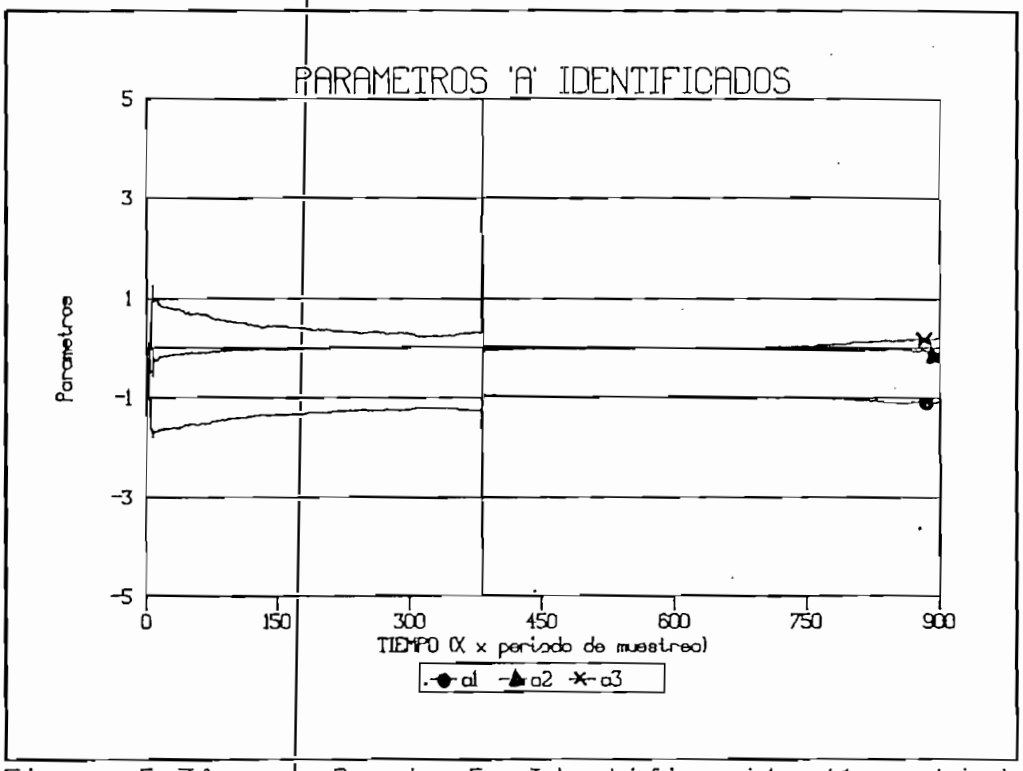


Figura 5.34 Prueba 5: Identificación (1 cambio)

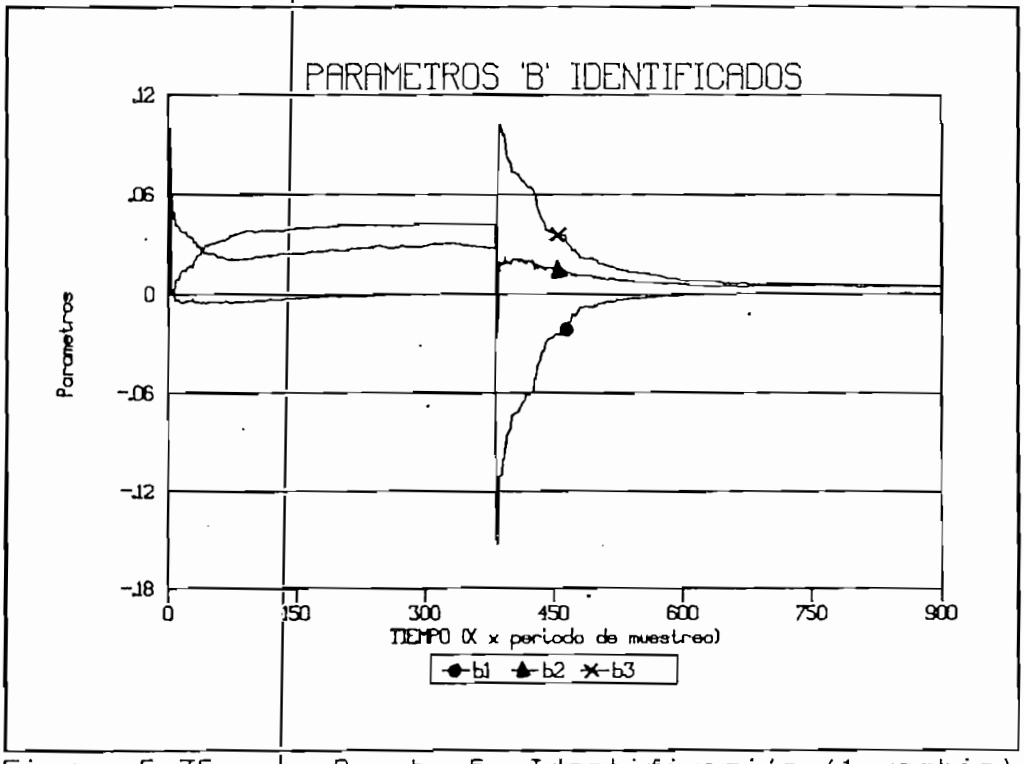


Figura 5.35 Prueba 5: Identificación (1 cambio)

TIEMPO REAL

1) PARAMETROS DE TIEMPO REAL

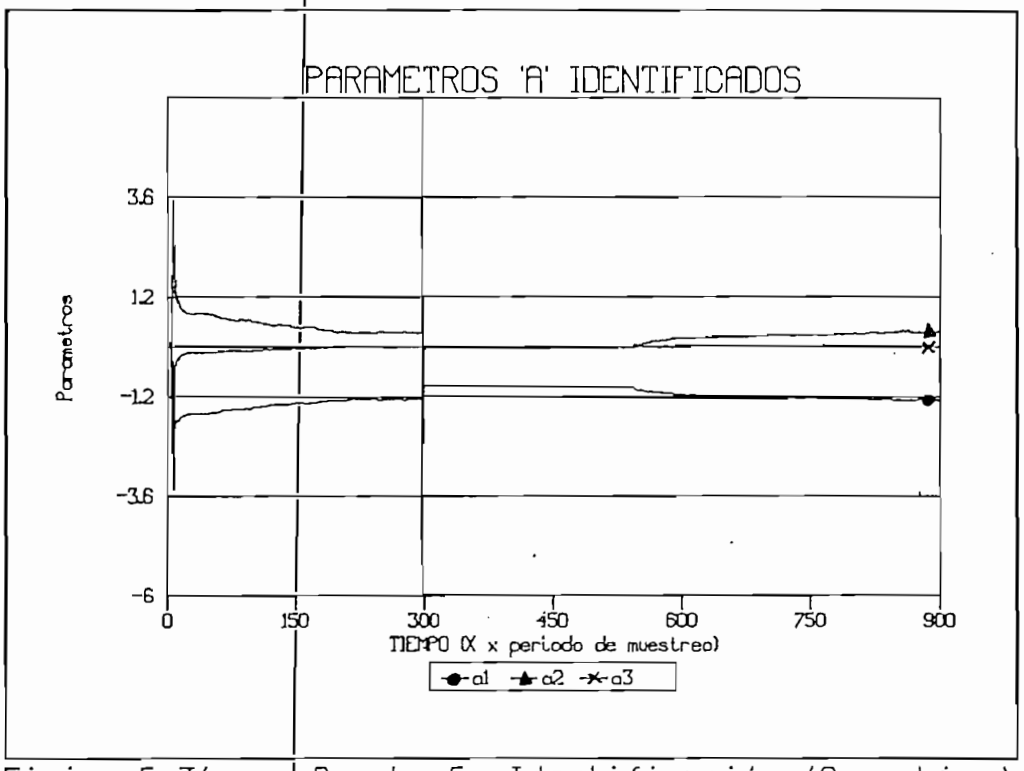
Tiempo de muestreo = 400 ms

2) PARAMETROS DE IDENTIFICACION

Orden del modelo = 3
Parametro Alfa = 10000
Referencia = 5
U minimo = 0
U máximo = 10
Ruido Porcentual = 50 %
Número de Iteraciones = 1219

3) RESULTADOS DE IDENTIFICACION

A(1) = -1.242418789540457
A(2) = .2965284553325764
A(3) = 1.673621765981356D-02
B(1) = 3.825431555882785D-04
B(2) = 4.127278530091934D-02
B(3) = 2.910197117222644D-02



Figira 5.36 Prueba 5: Identificación (2 cambios)

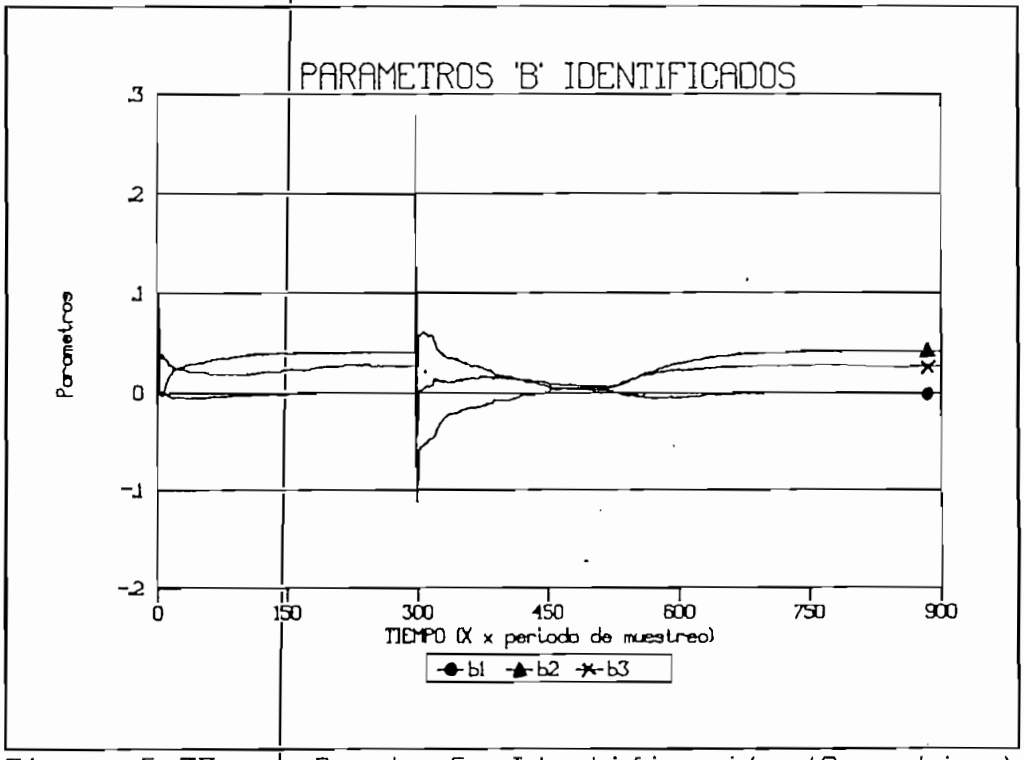


Figura 5.37 Prueba 5: Identificación (2 cambios)

Al inicio, convergencia a los valores de la planta inicial; en un tiempo intermedio, hacia los valores de la planta más lenta; y finalmente se recuperan los parámetros iniciales.

El algoritmo está por tanto en capacidad de reaccionar y responder a las variaciones que se presenten en la planta. Esto es muy importante para cuando se realice control adaptivo.

b) Sexta Prueba

Habiendo realizado la identificación paramétrica de la planta ahora se procederá a controlar la planta de segundo orden y a comprobar el funcionamiento del control adaptivo tipo self-tuning.

1.- Identificación y control (3° orden)

La planta de segundo orden será controlada mediante una señal emanada del computador. La referencia está ubicada en 5 V y el polinomio de polos será el mismo que se utilizó para simulación. La identificación se realizará a través de un modelo de tercer orden para detectar la dinámica del sistema en su totalidad. Los resultados se muestran en el Reporte nº 15 y en las figuras 5.38 y 5.39 del grafizador (señales de control y de salida) y 5.40 y 5.41 de los datos del computador (parámetros).

TIEMPO REAL

1) PARAMETROS DE TIEMPO REAL

Tiempo de muestreo = 400 ms

2) PARAMETROS DE IDENTIFICACION Y CONTROL

Orden del modelo = 3
Parametro Alfa = 10000
Referencia = 5
U minimo = 0
U máximo = 10
Número de Iteraciones = 1202

3) UBICACION DE POLOS

Polinomio de polos: $T(z) = 1 + \Sigma\ t(i)z^{-i}$

t(1) = -1.32
t(2) = .5
t(3) = 0

4) RESULTADOS DE IDENTIFICACION Y CONTROL

A(1) = 2.033999759557338
A(2) = 4.590338472803325
A(3) = -7.0604030838984
B(1) = -.7580722969535504
B(2) = -1.470833328072254
B(3) = 2.791685020521848

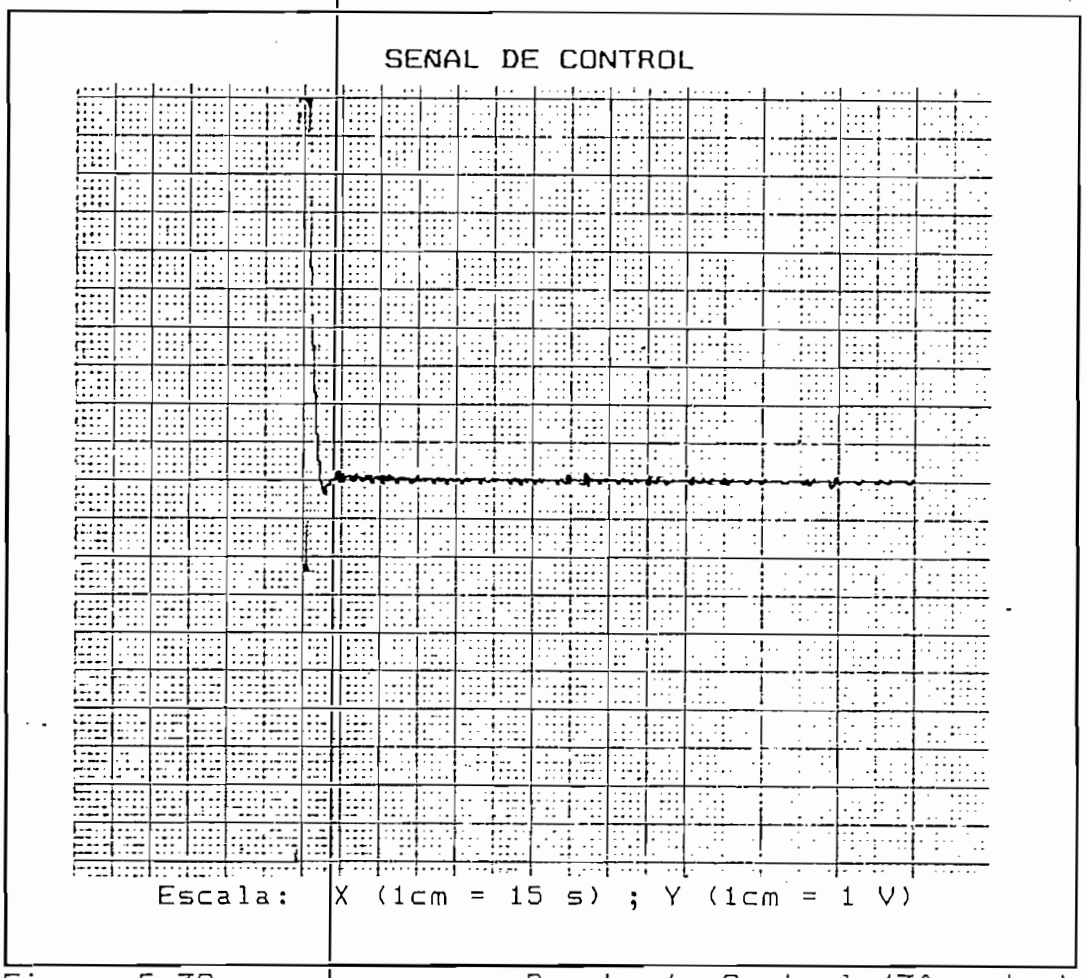


Figura 5.38 Prueba 6: Control (3° orden)

El control es muy estable y la salida alcanza rápidamente el valor de referencia prácticamente sin sobreimpulso. Es una respuesta muy satisfactoria en lo que a control se refiere.

Los parámetros tienen el comportamiento ya detectado en simulación. No hay una clara convergencia debido a la realimentación del ruido que impide un funcionamiento correcto del algoritmo de mínimos cuadrados. Pero aquí lo que interesa es el control, es decir la señal de salida y sus variaciones y el tipo de señal de control requerida para alcanzar el estado estable.

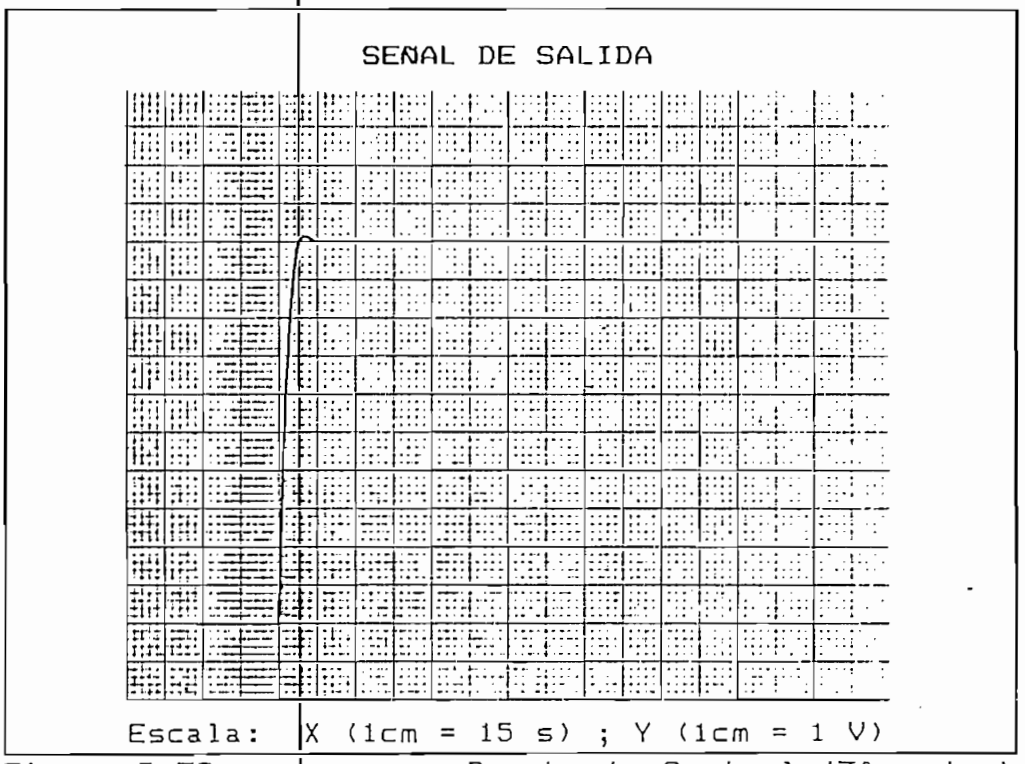


Figura 5.39 Prueba 6: Control (3° orden)

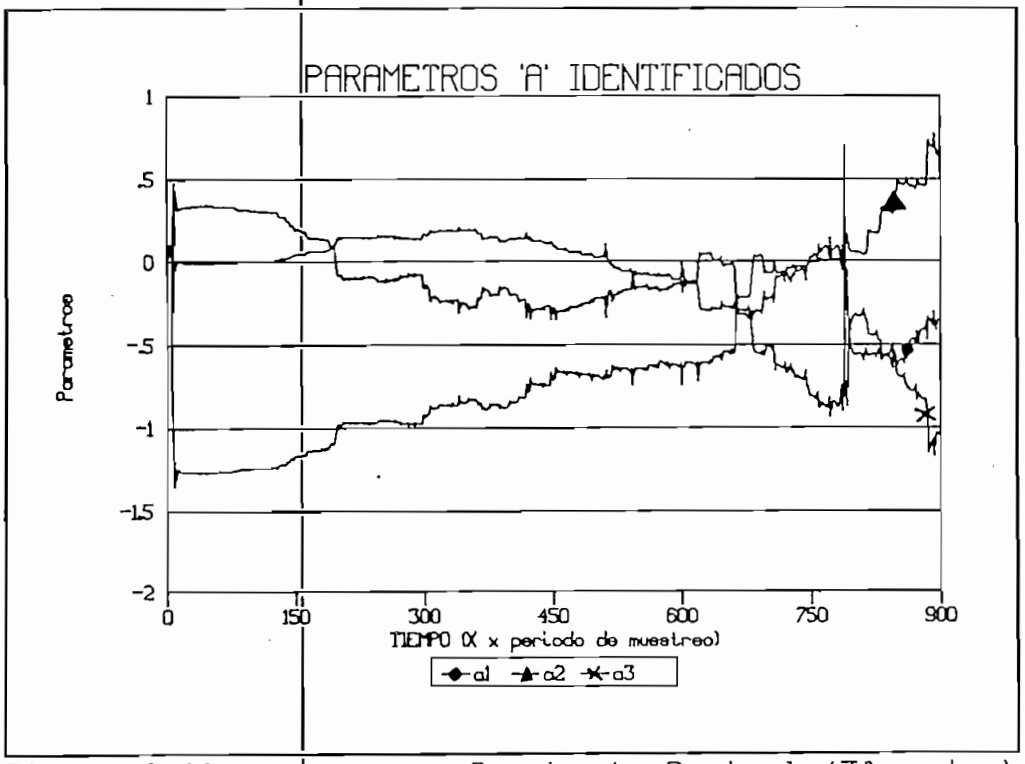


Figura 5.40 Prueba 6: Control (3° orden)

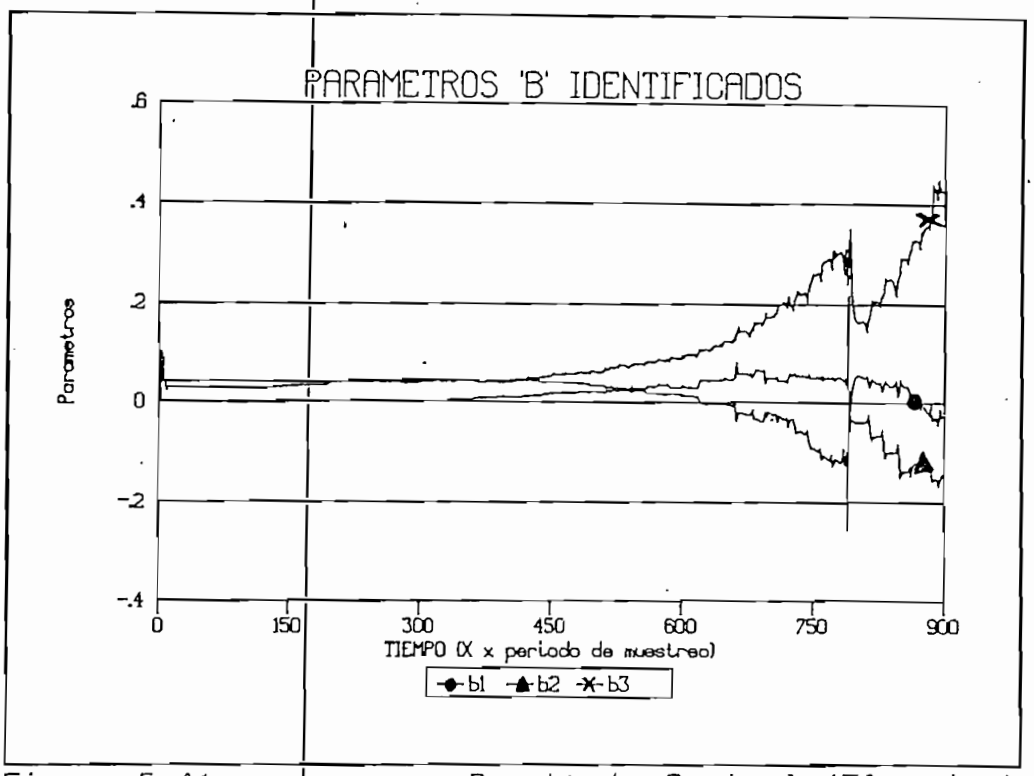


Figura 5.41 Prueba 6: Control (3° orden)

2.- Identificación y control con perturbación

La planta de segundo orden sufre una perturbación momentánea (un corto circuito interno) que cambia momentáneamente las características de la planta. Se va a probar ahora la efectividad del control adaptivo tipo self-tuning para superar este problema.

El corto-circuito se realiza sobre el primer condensador del circuito. Los resultados del control se muestran en la figura 5.42. No interesan los parámetros pues ya se conoce que no podrán tener una convergencia adecuada.

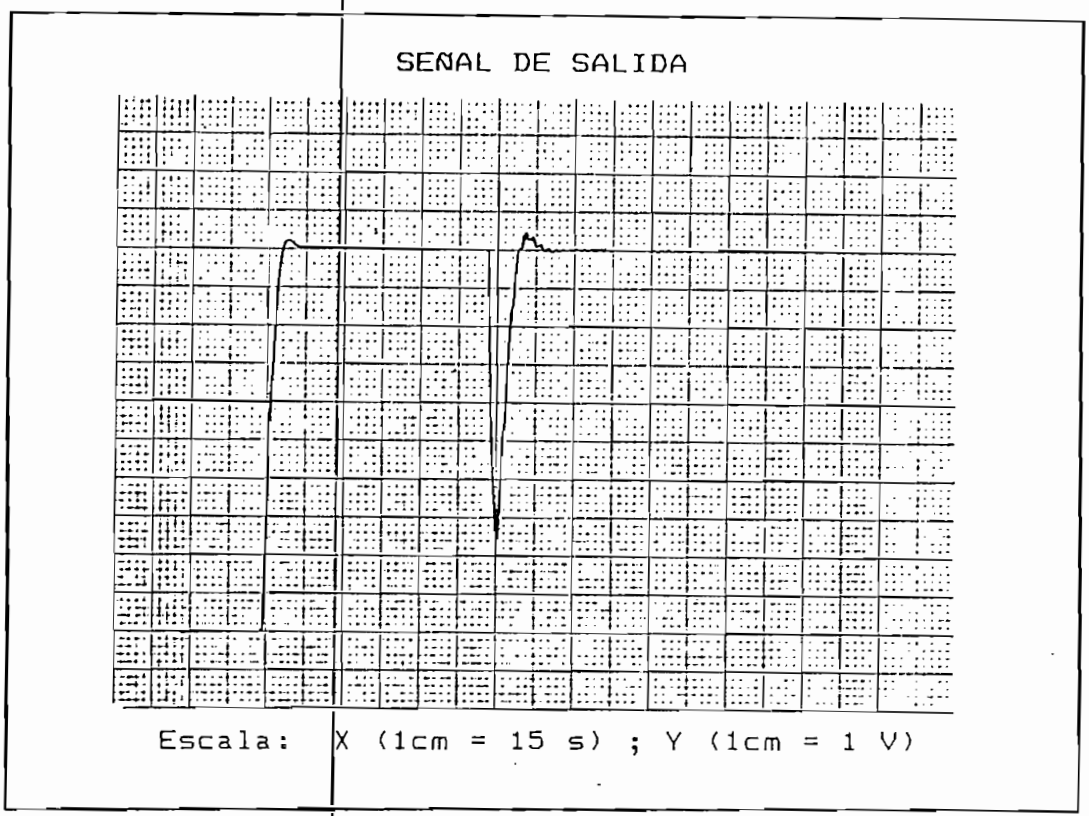


Figura 5.42 Prueba 6: Control (con perturbación)

Mientras dura el corto-circuito, el sistema no se puede controlar pues la información que trae la señal de control no puede llegar a la salida. Pero como éste dura muy poco, hay una súbita pérdida de control que al acabarse la perturbación es corregida por el sistema. y la salida se ve forzada a recuperar su condición en estado estable. Como se puede ver en la figura 5.42, la respuesta es bastante buena.

3.- Identificación y control con cambio de planta.

La perturbación anterior era momentánea. Ahora se va a introducir una perturbación permanente en la planta cambiando su dinámica: tal como se hizo para identificación, se

introducirá un condensador a la salida. Después de que la planta alcance la estabilidad con su nueva dinámica, se retirará nuevamente el condensador, restituyendo la planta a su dinámica inicial. El objetivo es analizar el comportamiento del control adaptivo en este caso. Un control clásico tipo PID, normalmente deja de actuar adecuadamente frente a este tipo de variaciones estructurales en la planta.

El resultado se muestra en la figura 5.43, donde se ha graficado la señal de salida. La primera perturbación corresponde al condensador añadido; la segunda es el retiro del condensador. Como se puede apreciar, el control supera fácilmente los cambios de la planta, pues rápidamente se alcanza el estado estable.

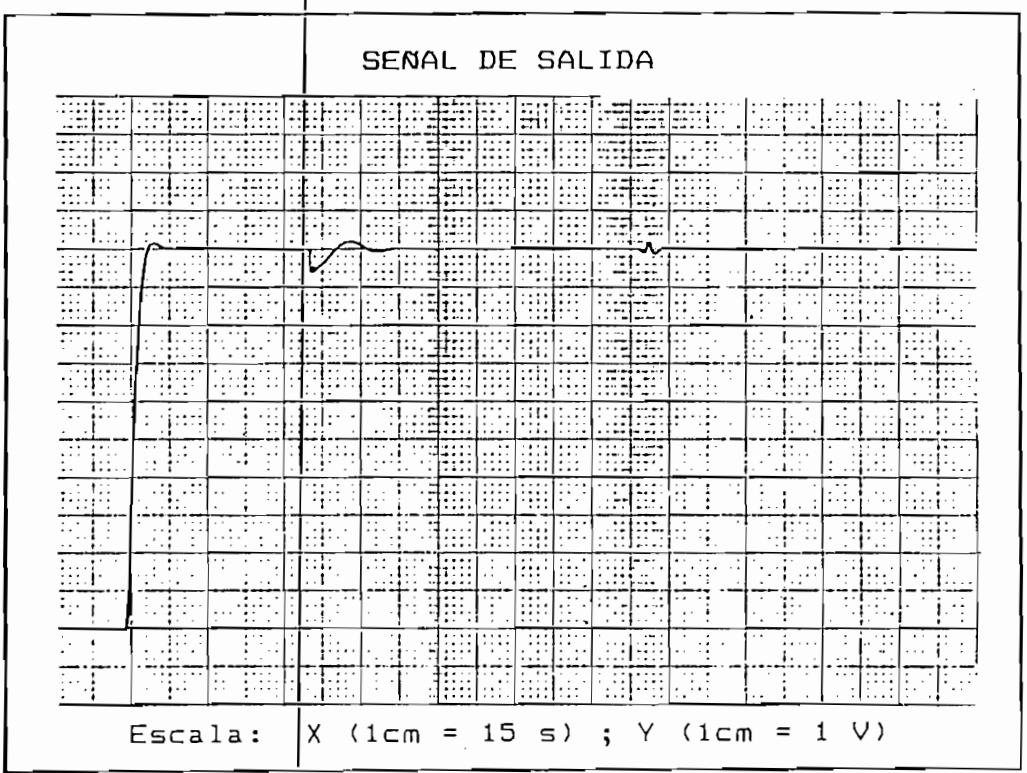


Figura 5.43 Prueba 6: Control (cambio de planta)

4.- Identificación y control (300 ms)

Finalmente se va a verificar la influencia del periodo de muestreo sobre el control adaptivo. Se realizará ahora el control con características similares a los anteriores, con la planta sin perturbaciones forzadas pero con un tiempo de muestreo más pequeño: 300 ms. La variación fundamental está ahora en la señal de control que se vuelve más inestable (ver figura 5.44). Esta inestabilidad si se vuelve muy grande, puede provocar deterioro en los transductores así como introducir perturbaciones en el sistema.

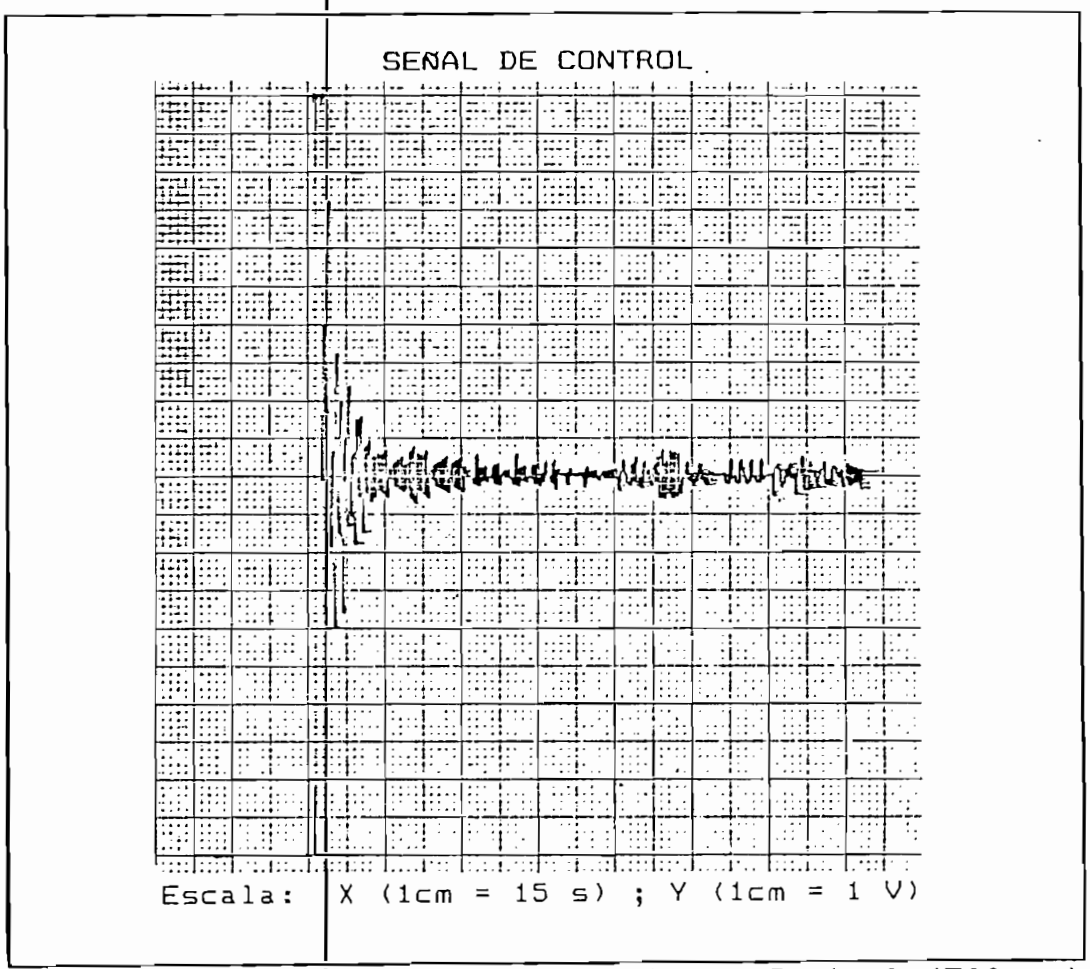


Figura 5.44 Prueba 6: Control (300 ms)

c) Séptima Prueba

La última prueba en tiempo real se realizará con la planta de primer orden. Interesa ahora el control de esta planta mediante un modelo de identificación de segundo orden para abarcar la dinámica global del sistema. Con los polos prefijados y un tiempo de muestreo de 200 ms (pues la planta es más rápida que la anterior), se obtienen los resultados de esta prueba (Reporte nº 16 las figuras 5.45, 5.46 y 5.47).

La respuesta resulta muy buena, con un mínimo sobreimpulso y una gran estabilidad, con lo que se reafirma la calidad del control adaptivo por ubicación de polos.

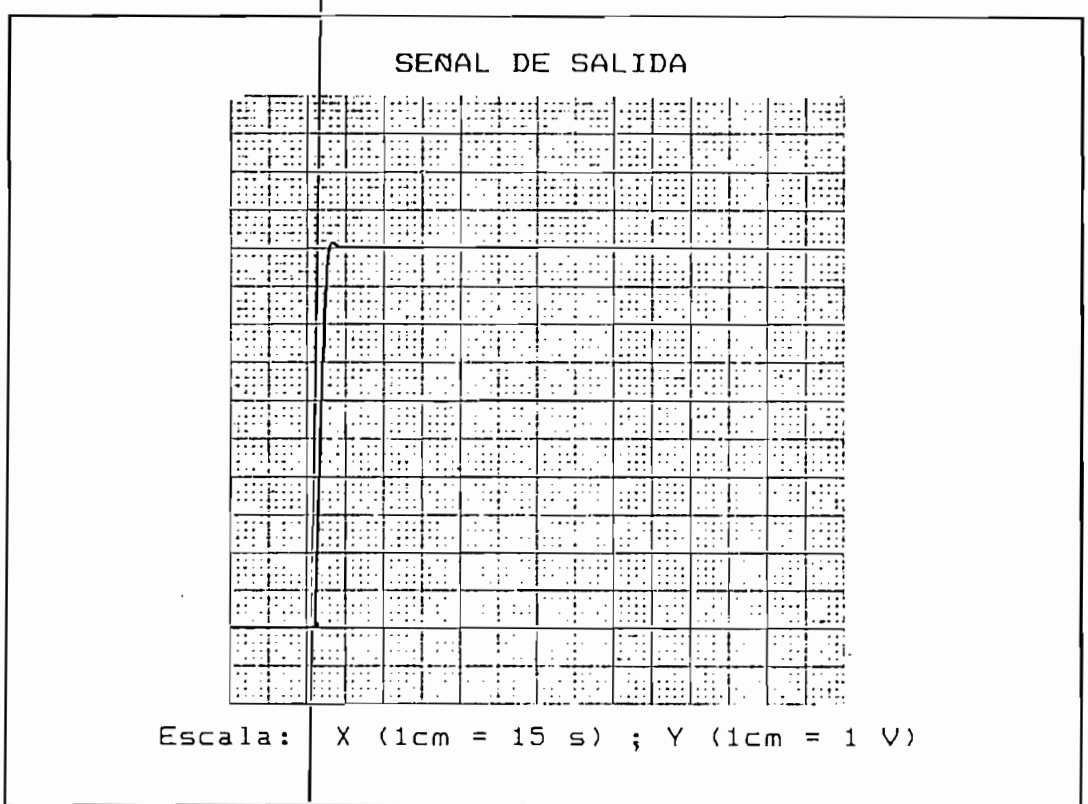


Figura 5.45 Prueba 7: Control

TIEMPO REAL

1) PARAMETROS DE TIEMPO REAL

Tiempo de muestreo = 200 ms

2) PARAMETROS DE IDENTIFICACION Y CONTROL

Orden del modelo = 2
Parametro Alfa = 10000
Referencia = 5
U minimo = 0
U máximo = 10
Número de Iteraciones = 1224

3) UBICACION DE POLOS

Polinomio de polos: $T(z) = 1 + \Sigma\ t(i)z^{-i}$

t(1) = -1.32
t(2) = .5

4) RESULTADOS DE IDENTIFICACION Y CONTROL

A(1) = 8.931557264242183
A(2) = -9.801012873040841
B(1) = -.4886093003025068
B(2) = .6188056637986288

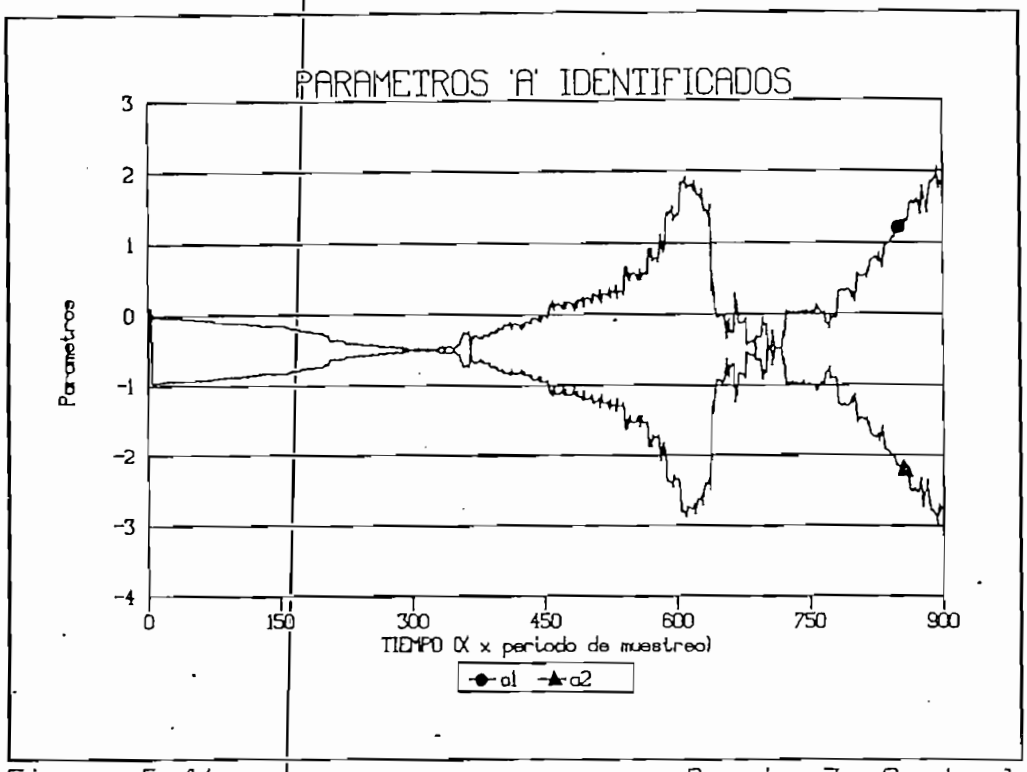


Figura 5.46 Prueba 7: Control

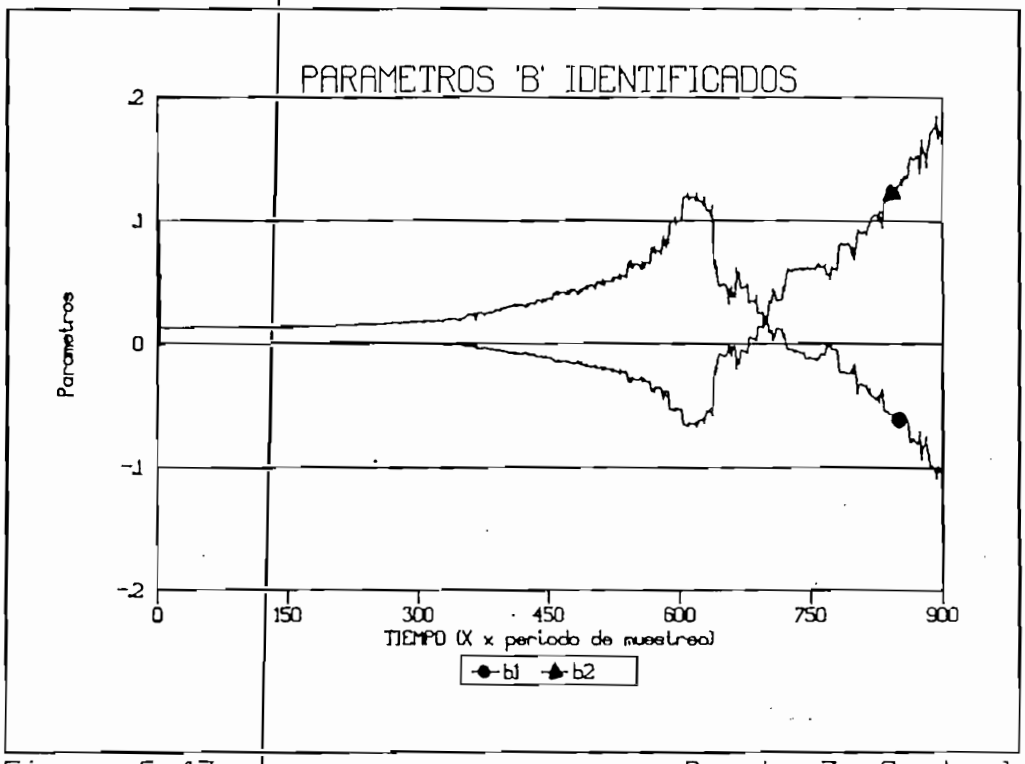


Figura 5.47 Prueba 7: Control

5.3 CONCLUSIONES

La necesidad de contar con modelos de plantas reales para su posterior control llevó a la realización de este trabajo enfocándolo hacia la identificación de sistemas, es decir la modelación de una planta mediante el uso de un computador. Partiendo del método de mínimos cuadrados determinístico, se lo amplió hacia modelos estocásticos con el método de mínimos cuadrados estocástico, mostrando que al existir ruido blanco en la salida, existe de todas maneras convergencia hacia los valores deseados.

Con el modelo de la planta ya encontrado, bastó con cerrar el lazo y diseñar una ley de control en base a los parámetros estimados, para estar en capacidad de realizar control adaptivo digital de tipo self-tuning, que además de utilizar la información de la identificación permite realizar pruebas de la más diversa índole, con plantas complejas y de dinámica variable en el tiempo. El control se diseñó en base al algoritmo de ubicación de polos, siendo el computador el que diseña el control sobre la base de la información de la planta.

El sistema completo consta entonces de un programa que realiza identificación y control en simulación y tiempo real, quedando a cargo del computador la identificación de la planta y la generación de la señal de control en base al diseño de la ley de control.

El programa realiza en simulación y en tiempo real las tareas de identificación (en lazo abierto) y de identificación y control (lazo cerrado). Este ha sido desarrollado para un computador PS/60 con tarjeta de gráficos VGA a colores y coprocesador matemático, al que ha sido conectado un equipo de adquisición de datos KEITHLEY 500A. El lenguaje de programación utilizado es el QUICK BASIC 4.5 con su extensión a rutinas de adquisición de datos QUICK 500.

Además de realizar las pruebas debidas para la simulación y el tiempo real, de manera a obtener resultados de identificación y control en plantas de diverso tipo, el programa tiene una clara orientación pedagógica y de muy fácil interacción con el usuario. Esto ha sido realizado para que pueda ser utilizado en los laboratorios de la Facultad con miras a una formación más profunda en el área de control y sistemas. Por esto se ha puesto mucho énfasis en la presentación, en los métodos de ingreso de datos y en la visualización permanente de la información sobre el proceso. El uso intenso de gráficos, ventanas y menus hace muy fácil el uso de este programa.

Para las pruebas de simulación, se escogieron plantas con modelos diversos. En todos los casos los resultados de la simulación mostraron que el algoritmo de mínimos cuadrados estocástico está efectivamente en capacidad de identificar con una grandísima exactitud los parámetros de las plantas.

Al pasar al tiempo real, intervienen algunas condiciones que no aparecen en simulación. Los sistemas reales presentan variaciones, perturbaciones no sólo debido a la dinámica de la planta sino a interacciones con otros sistemas cercanos, ruidos indeseados en las señales, mediciones con limitacones por las características de los instrumentos. Pese a esto, el algoritmo demostró estar en capacidad de identificar también plantas en tiempo real, pues en todos los casos se obtiene una convergencia de los parámetros entorno a valores medios.

Para el control adaptivo, las pruebas de simulación mostraron la eficiencia de este tipo de control y los problemas en la identificación de parámetros en lazo cerrado. Esto era previsible pues como se ha mencionado, la realimentación de la señal de salida Y provoca una realimentación del error y por tanto el ruido se vuelve correlacionado con la consiguiente falla del algoritmo de identificación. Pese a esto, el control mantiene estable a la señal de salida y por tanto se cumplen los objetivos esenciales del control adaptivo. Este resultado que ya se comprobó en simulación, fue verificado también en tiempo real donde pese a las variaciones en los parámetros, se obtuvo una muy buena respuesta en el control.

Se realizaron además pruebas con plantas de dinámica variable y se comprobó que incluso en estos casos el control trabaja adecuadamente: la variación de parámetros en la planta provoca

una variación en el control que ajusta sus parámetros para mantener la señal de salida en el valor referencial.

Es necesario hacer ahora algunas consideraciones sobre el tiempo real como tal, pues los resultados de simulación mostraron su total apego a lo esperado.

Para el tiempo real resulta esencial determinar un periodo de muestreo que permita un control adecuado de la planta y una estabilidad de la señal de control. Lo primero se puede obtener con pequeños periodos de muestreo; lo segundo requiere grandes periodos de mustreo. Es pues necesario llegar a un compromiso. Un periodo de muestreo muy grande estabiliza al control pero puede provocar que la información que llega al computador no describa totalmente la dinámica de la planta y que por tanto no haya estabilidad en la señal de salida. Lo contrario, un tiempo de muestreo muy pequeño hace a la planta muy poco sensible a las variaciones del control por lo que éste último puede alcanzar oscilaciones muy grandes. No existen reglas definitivas para escoger este periodo de muestreo, pero un elemento de aproximación es el que aquí se ha utilizado: estar cercano a una décima parte del tiempo de respuesta de la planta.

El programa tiene limitaciones al respecto pues para plantas muy rápidas, el periodo de muestreo está limitado por el tiempo que demora el algoritmo implementado. Este tiempo se ve incrementado por la utilización de gráficos de entrada y

salida. Si se desea trabajar con este tipo de plantas, es necesario no sólo eliminar los gráficos sino optimizar el programa con algoritmos más veloces, como el self-tuning directo que no requiere la identificación paramétrica explícita.

Otro de los problemas aparece con la realimentación del error en el control si se desea tener el modelo de la planta al mismo tiempo que el control adaptivo. Esto requiere de otros algoritmos de identificación como el de "Maximum Likelihood" que permite superar el problema del ruido correlacionado. Existen además muchas otras rutinas de control adaptivo que permiten corregir otros problemas como la estabilidad de la señal de control.

Durante el tiempo de pruebas fue posible además verificar el funcionamiento adecuado del control adaptivo en una planta a pequeña escala diseñada y construída por J. Garzón en el marco del proyecto CONUEP-EPN 87.01. El control adaptivo se desempeñó muy bien en el momento de controlar esta planta.

En síntesis este trabajo ha permitido incursionar en el campo del tiempo real realizando identificación de sistemas pero también control adaptivo de tipo self-tuning. Es necesario indicar que es el primer trabajo de este tipo que se realiza en la Facultad.

5.4. RECOMENDACIONES

La realización de identificación y control adaptivo en tiempo real abre las puertas para un sinnúmero de aplicaciones y de incursiones teóricas en el campo del control discreto y adaptivo.

Para la identificación propiamente dicha sería interesante desarrollar programas basados en otros algoritmos como el de "Maximum Likelihood" o aquellos en donde además es posible identificar, en ciertas condiciones el orden de la planta.

En el área del control, el ámbito es mucho mayor: se puede pensar en el self-tuning directo, en control adaptivo tipo MRAS[1], y otros más. Con la capacidad del equipo de adquisición de datos para trabajar con sistemas multivariables, sería provechoso intentar realizar control adaptivo multivariable.

Dada la orientación pedagógica de este programa, sería recomendable su utilización en prácticas de laboratoro para materias como modelación y simulación, control discreto, control estocástico, de manera que se complemente la formación teórica de los estudiantes con un sólido sustento en la práctica.

---

[1]) Ya desarrollado en simulación. Ver la Tesis de Hugo Ortiz Tulcán: Control Adaptivo con modelo de referencia para sistemas discretos, 1990, Quito, EPN

6. BIBLIOGRAFIA

6. BIBLIOGRAFIA

1. ASTROM KARL JOHAN y WITTENMARK BJORN, "Adaptive control", Addison-Wesley Publishing Company, Reading, Massachusetts, 1989.

2. ASTROM KARL JOHAN, "Adaptive feedback control", Proceedings of the IEEE, Vol. 75, nº 2, February 1987.

3. BENNETT S. y LINKERS D. A. Eds., "Computer control of industrial processes", Peter Peregrinus Ltd., Stevenage, 1982.

4. BURDEN RICHARD I. y FAIRES J. DOUGLAS, "Análisis Numérico", Grupo Editorial Iberoamerica, Mexico, 1985.

4. D'AZZO JOHN y HOUPIS CONSTANTINE, "Sistemas Lineales de Control", Paraninfo, Madrid, 1987.

5. FRANKLIN GENE F. y POWELL J. DAVID, "Digital control of dynamic systems", Addison-Wesley Publishing Company, Reading, Massachusetts, 1980.

6. HARRIS C. J., "Self-tuning and adaptive control", Peter Peregrinus Ltd., Stevenage, 1981.

7. KEITHLEY, "Quick 500 Data Aquisition and Control Software", Keithley Data Acquisition and Control, Cleveland, 1987.

8. KEITHLEY, "500 Series Measurement and Control System", Keithley Data Acquisition and Control, Cleveland, 1989.

9. KUO BENJAMIN, "Digital Control Systems", Holt-Saunders International Editions, Tokyo, 1981.

10. LEIGH J. R., "Modelling and Simulation", Peter Peregrinus Ltd., London, 1983.

11. MICROSOFT, "QUICK BASIC Version 4.5", Microsoft Corporation, USA, 1988.

12. OGATA KATSUHIKO, "Ingeniería de control moderna", Editorial Prentice-Hall Internacional, New York, 1974.

13. RAO C. R., "Linear Statistical Inference and its Applications", John Wiley and Sons, New York, 1973.

14. SEBER G. A. F., "Linear Regression Analysis", John Wiley and Sons, New York, 1977.

APENDICE A:

LISTADO DEL PROGRAMA

LISTADO DE RUTINAS

MODULO PRINCIPAL: tesisad.bas

```
'=================================================================
' MODULO PRINCIPAL tesisad.bas
' Establece las rutinas y funciones para la ejecución del programa.
' Define las variables y constantes del programa.
' Define las rutinas de error del sistema.
' Crea el Menu Principal.
'   Opciones:
'   1. Ejecutar los procesos en SIMULACION
'   2. Ejecutar los procesos en TIEMPO REAL
'=================================================================

'Definición de Rutinas, Subrutinas y Funciones
DECLARE FUNCTION convert$ (D#)
DECLARE SUB clerr ()
DECLARE SUB ersist ()
DECLARE SUB printer (para$(), T#(), P#(), DI#, K%)
DECLARE SUB calcu (X1#(), V1#())
DECLARE SUB ident (para$(), val$())
DECLARE SUB trcdat ()
DECLARE SUB sidat ()
DECLARE SUB scdat ()
DECLARE SUB tridat ()
DECLARE SUB parin (mp$())
DECLARE FUNCTION menuda% (supx#, supy#, N%, mat() AS STRING, b!
dato$())
DECLARE FUNCTION valid! (b!, V$(), P%)
DECLARE SUB datin (para$(), val$(), pol$())
DECLARE FUNCTION menu! (supx!, supy!, N%, mat() AS STRING, b!)
DECLARE SUB datos ()
DECLARE SUB graf (I%)
DECLARE SUB defgraf ()
DECLARE SUB trcontrol (para$(), val$(), pol$())
DECLARE SUB salidau ()
DECLARE SUB trparam ()
DECLARE SUB simula ()
DECLARE SUB treal ()
DECLARE SUB trident (para$(), val$())
DECLARE SUB traery ()
DECLARE SUB control (para$(), val$(), pol$())
DECLARE SUB sist (A#(), V1#(), N%, K%)
DECLARE SUB modelo ()
DECLARE SUB calcy ()
DECLARE FUNCTION pesc (V1#(), V2#())
DECLARE SUB pmat (M1#(), M2#(), M3#())
DECLARE SUB pvect (V1#(), M1#(), V2#())

'Definición de Tipo de Variables
DEFINT I-N
DEFDBL A-H, O-Z

'Variables Comunes a todas las rutinas
'N = Orden del Modelo * 2 (Identificación)
'NA: Grado del polinomio A(z) identificado
```

```
'NB: Grado del polinomio B(z) identificado
'NS = Orden del Modelo * 2 (Planta en Simulación)
'NAS: Grado del Polinomio A(z)
'NBS: Grado del Polinomio B(z)
'XS(): Datos de Entrada-Salida en simulación
'TS(): Parámetros del Modelo de la Planta para simulación
'Y: Señal de Salida de la planta
'U: Señal de entrada a la planta
'R: Referencia
'dep!: longitud del Arreglo de datos para Tiempo Real
'bintv%: intervalo de muestreo
'DAT(): Matriz de datos
'MaxDat: Número máximo de datos
'ERF%: Detección de Error
COMMON SHARED N, NA, NB, NS, NAS, NBS, XS(), TS(), Y, U, R, dep!
COMMON SHARED bintv%, DAT(), MaxDat, ERF%

'Definición de Constantes para los Colores
CONST negro = 0
CONST azul = 1
CONST verde = 2
CONST cian = 3
CONST rojo = 4
CONST violeta = 5
CONST marron = 6
CONST blanco = 7
CONST gris = 8
CONST celeste = 9
CONST tomate = 12
CONST magenta = 13
CONST amarillo = 14
CONST blancoi = 15

CLS

'Cambio de color
PALETTE 3, 25

'Inicialización de rutinas de error
ON ERROR GOTO errores

'Definición de Menu Principal
DIM mprin(4) AS STRING
mprin(1) = "   M E N U     P R I N C I P A L "
mprin(2) = " Simulación"
mprin(3) = " Tiempo Real"
mprin(4) = " Salida"

'Inicialización de Pantalla y Colores para Menu Principal
SCREEN 0
WIDTH 80, 25
COLOR blanco, negro

'Menu Principal
DO
```

```
    'Accesible: Menu Principal
    I = menu(8, 17, 4, mprin(), 1)
    SELECT CASE I

        'Simulación
        CASE 1

            'Pasar a Simulación
            CALL simula

            CLS

        'Tiempo Real
        CASE 2

            'Detección de Sistema de adquisición apagado
            DEF SEG = &HCFF9

            'Sistema de Adquisición de Datos apagado
            IF PEEK(&HB) = 255 THEN
               COLOR negro, blanco
               BEEP
               LOCATE 24, 30
               PRINT "Error: Sistema de Adquisición apagado";
               COLOR blanco, negro
               DEF SEG

            'Sistema de Adquisición de Datos encendido
            ELSE
               DEF SEG

               'Pasar a Tiempo Real
               CALL treal

               CLS

        END IF

        CASE ELSE

    END SELECT

'Repetir hasta escoger Terminar
LOOP UNTIL I = 3

'Borrar, salir a DOS
CLS

END

'Rutinas de error
errores:

'Definición de Colores y Ubicación para errores
```

```
COLOR negro, blancoi
LOCATE 24, 40

'Sonido para indicar detección de error
BEEP

'Inicialización variable de detección de error
ERF% = 1

'Rutinas de selección del error
SELECT CASE ERR
   CASE 24, 25, 27
      'Errores en la impresora
      PRINT "Error: verificar impresora";

   CASE 53
      'Archivo no presente.
      PRINT "Error: Archivo no presente";

   CASE ELSE
      'Otros errores
      ON ERROR GOTO 0

END SELECT

'Redefinición Colores y Regreso
COLOR azul, blanco
RESUME NEXT

END
```

SUBRUTINA calcu

```
'=============================================================================
' SUBRUTINA calcu
' Calcula la señal de control u(t) para la planta
' Parámetros de Entrada:
'    X1(): Datos de Entrada-Salida de la planta
'    V1(): Parámetros de Control
'=============================================================================
DEFINT I-N
DEFDBL A-H, O-Z
SUB calcu (X1(), V1())

'Calculo de V1*X1
U1 = 0
FOR I = 2 TO NB - 1
   U1 = U1 + (V1(I - 1) - V1(I)) * X1(NA + I)
NEXT I

'Cálculo de U para orden mayor a 2
IF N > 2 THEN
   U1 = U1 + (1 - V1(1)) * X1(NA + 1) + V1(NB - 1) * X1(NA + NB)
   G = 0
   FOR I = 1 TO NA
      U1 = U1 + V1(NB + I) * X1(I)
      G = G + V1(NB + I)
   NEXT I

'Calculo de U para orden igual a 1
ELSE
   U1 = X1(2)

END IF

U1 = U1 - V1(NB) * Y + (G + V1(NB)) * R
U = U1

END SUB
```

SUBRUTINA calcy

```
'=============================================================================
' SUBRUTINA calcy
' Calcula el valor de Y con los Parámetros del Modelo (SIMULACION)
'=============================================================================
DEFINT I-N
DEFDBL A-H, O-Z
SUB calcy

'Actualización de Datos
FOR I = NAS TO 2 STEP -1
   XS(I) = XS(I - 1)
NEXT I
XS(1) = -Y
FOR I = NS TO NS - NBS + 2 STEP -1
   XS(I) = XS(I - 1)
NEXT I
XS(NAS + 1) = U

'Cálculo de Y
N1 = N
N = NS
Y = pesc(XS(), TS())
N = N1

END SUB
```

SUBRUTINA clerr

```
'=============================================================================
' SUBRUTINA clerr
' Define Colores y Borra la zona de mensajes de Error
'=============================================================================
DEFINT I-N
DEFDBL A-H, O-Z
SUB clerr

COLOR blanco, negro
LOCATE 24, 40
PRINT SPACE$(30);
COLOR blanco, azul

END SUB
```

SUBRUTINA control

```
'=============================================================================
' SUBRUTINA control
' Ejecuta el proceso de IDENTIFICACION y CONTROL en SIMULACION en
' base a los parámetros y valores iniciales definidos.
' Produce los gráficos en pantalla de la señal de entrada ( u(t) )
' y de salida ( y(t) ) y calcula los parámetros de la planta.
' Parámetros de Entrada:
'    para$(): Parámetros definidos en "scdat"
'    val$(): Valores Iniciales definidos en "scdat"
'    pol$(): Polos definidos en "scdat"
'=============================================================================
DEFINT I-N
DEFDBL A-H, O-Z
SUB control (para$(), val$(), pol$())

'Recuperación de Parámetros
'Par: Valor inicial de α
'R: Referencia
'Umin: valor mínimo permitido de U
'Umax: valor máximo permitido de U
'MaxDat: Número máximo de Datos para la Matriz de Datos
'ERP: Ruido Porcentual
Par = VAL(para$(3))
R = VAL(para$(4))
Umin = VAL(para$(5))
Umax = VAL(para$(6))
MaxDat = VAL(para$(7))
ERP = VAL(para$(8))

'Calculo del grado del polinomio de Polos
IF NA > 2 THEN
   Nt = 3
ELSE
   Nt = 2
END IF

'Dimensionamiento de Matrices y Vectores
'P(): Matriz de Covarianzas
'MA(), MB(): Matrices auxiliares de cálculo
'T(): Vector de Parámetros (modelo ARMA)
'X(): Vector de Datos Entrada-Salida (modelo ARMA)
'V(): Vector auxiliar de cálculo
's(): Matrix de Control
'a(): Vector de Parámetros a para Control
'pol(): Vector de coeficientes de polos
'TM(): Matriz con las últimos 50 cálculos de parámetros
'TF(): Vector con los Parámetros Promedio
'DAT(): Matriz con los Parámetros y las señales de entrada y salida
REDIM P(N, N)
REDIM MA(N, N) AS DOUBLE, MB(N, N) AS DOUBLE
REDIM T(N), X(N), V(N), s(N, N + 1), A(NA + 1), pol(N)
REDIM TM(0 TO 49, N), TF(N)
REDIM DAT(MaxDat, N + 2)
```

```
'Inicialización de T
FOR I = 2 TO N + 1
   T(I - 1) = VAL(val$(I))
NEXT I

'Inicialización de "pol"
FOR I = 2 TO Nt + 1
  pol(I - 1) = VAL(pol$(I))
NEXT I

'Inicialización de P, TF y X
FOR I = 1 TO N
   FOR J = 1 TO N
      s(I, J) = 0
      IF I = J THEN
         P(I, J) = Par
      ELSE
         P(I, J) = 0
      END IF
   NEXT J
   X(I) = 0
   TF(I) = 0
NEXT I

'Inicialización de XS
FOR I = 1 TO NAS + NBS
   XS(I) = 0
NEXT I

'Inicialización de Variables y Constantes
'K: cuenta el número de iteraciones para graficación
'DK: cuenta el número total de iteraciones
'U: señal de entrada a la planta
'F: Factor de olvido
'Y: señal de salida de la planta
K = 0
DK = 0
U = 0
F = .98
Y = 0
DAT(DK, 1) = Y
DAT(DK, 2) = U
RANDOMIZE TIMER

'Definición de Pantalla y Colores para graficación
SCREEN 9
WIDTH 80, 43
CLS

'Iniciar Identificación
LOCATE 24, 30
PRINT "PRESIONE UNA TECLA PARA INICIAR CONTROL"
DO
   q$ = INKEY$
LOOP WHILE q$ = ""
```

```
'Impresión en pantalla de los gráficos y mensajes iniciales
CLS
CALL defgraf
LOCATE 21, 30
PRINT "SIMULACION"
COLOR 2
LOCATE 42, 10
PRINT "CONTROL EN PROCESO";
COLOR 15
LOCATE 42, 40
PRINT "Presione una Tecla para terminar...";

'Inicio del lazo de Identificación
DO

   'Rutina de Actualización de Datos Entrada-Salida
   FOR I = NA TO 2 STEP -1
      X(I) = X(I - 1)
   NEXT I
   X(1) = -Y
   FOR I = N TO N - NB + 2 STEP -1
      X(I) = X(I - 1)
   NEXT I
   X(NA + 1) = U

   'Cálculo de Y
   CALL calcy
   Y = Y + R * (ERP / 100 * RND - ERP / 200)

   'Inicio Algoritmo de Mínimos Cuadrados Recursivo
   CALL pvect(V(), P(), X())
   A = F + pesc(X(), V())
   FOR I = 1 TO N
      V(I) = V(I) / A
   NEXT I
   E = Y - pesc(X(), T())

   'Cálculo Nuevos Parámetros
   FOR I = 1 TO N
      T(I) = T(I) + V(I) * E
   NEXT I

   'Ingreso de valores a la Matriz de Datos
   IF DK < MaxDat THEN
      DAT(DK + 1, 1) = Y
      DAT(DK + 1, 2) = U
      FOR I = 3 TO N + 2
         DAT(DK + 1, I) = T(I - 2)
      NEXT I
   END IF

   'Ingreso de valores a la Matriz de Parámetros
   FOR I = 3 TO N + 2
      TM(DK MOD 50, I - 2) = T(I - 2)
   NEXT I
```

```
'Cálculo de Matriz de Covarianzas P.
FOR I = 1 TO N
   FOR J = 1 TO N
      IF I = J THEN
         MA(I, J) = 1 - V(I) * X(J)
      ELSE
         MA(I, J) = -V(I) * X(J)
      END IF
   NEXT J
NEXT I
CALL pmat(MB(), MA(), P())
FOR I = 1 TO N
   FOR J = 1 TO N
      P(I, J) = MB(I, J) / F
   NEXT J
NEXT I

'Cálculo nuevo vector de Parámetros a para Control
A(1) = T(1) - 1
FOR I = 2 TO NA
   A(I) = T(I) - T(I - 1)
NEXT I
A(NA + 1) = -T(NA)

'Definición Matriz s para orden mayor a 2
IF N > 2 THEN
   FOR J = 1 TO N + 1
      FOR I = 1 TO N
         IF J < NB THEN
            IF I < J THEN
               s(I, J) = 0
            ELSEIF I = J THEN
               s(I, J) = 1
            ELSEIF I <= NA + 1 + J THEN
               s(I, J) = A(I - J)
            ELSE
               s(I, J) = 0
            END IF
         ELSEIF J < N + 1 THEN
            IF I < 1 - NB + J THEN
               s(I, J) = 0
            ELSEIF I <= J THEN
               s(I, J) = T(NA + NB + I - J)
            ELSE
               s(I, J) = 0
            END IF
         ELSE
            IF I <= Nt AND I <= NA + 1 THEN
               s(I, N + 1) = pol(I) - A(I)
            ELSEIF I <= Nt AND I > NA + 1 THEN
               s(I, N + 1) = pol(I)
            ELSEIF I > Nt AND I <= NA + 1 THEN
               s(I, N + 1) = -A(I)
            ELSE
```

```
                s(I, N + 1) = 0
            END IF
        END IF
    NEXT I
NEXT J

'Resolución Sistema de Ecuaciones para orden mayor a 2
CALL sist(s(), V(), N, ERS%)

'Error en sistema de ecuaciones
IF ERS% = 1 THEN

    'Impresión mensajes de error
    CALL ersist

    'Redefinición Pantalla y colores
    SCREEN 0
    WIDTH 80, 25
    COLOR blanco, negro
    PALETTE 3, 25

    'Terminar: volver al Menu de Control
    EXIT SUB

END IF

'Calculo del control para un orden del modelo igual a 1
ELSE

    'Grado del polinomio de polos es 1
    IF Nt = 1 THEN
        V(1) = pol(1) - A(1) / T(2)
        V(2) = -A(2) / T(2)

    'Grado del polinomio de polos es 2
    ELSE
        V(1) = pol(1) - A(1) / T(2)
        V(2) = pol(2) - A(2) / T(2)

    END IF

END IF

'Calculo de la señal de control u
CALL calcu(X(), V())

'U debe estar entre Umin y Umax
IF U > Umax THEN
    U = Umax
ELSEIF U < Umin THEN
    U = Umin
END IF

'Nuevo Gráfico si más de 580 valores
IF K > 580 THEN
```

```
        'Nuevo Gráfico
        CALL defgraf

        'Inicialización de K
        K = 0

    END IF

    'Graficar Nuevo Punto
    CALL graf(K)

    'Incrementar contadores
    K = K + 1
    DK = DK + 1

'Repetir el lazo hasta la detección de una tecla
LOOP WHILE INKEY$ = ""

'Fin de Identificación
LOCATE 42, 2
PRINT SPACE$(8); "Presione una Tecla para continuar..."; SPACE$(29);
DO
LOOP WHILE INKEY$ = ""

'Cálculo de Parámetros Promedio
FOR I = 1 TO N
    IF DK > 50 THEN
        FOR J = 0 TO 49
            TF(I) = TF(I) + TM(J, I)
        NEXT J
        TF(I) = TF(I) / 50
    ELSE
        TF(I) = T(I)
    END IF
NEXT I

'Definición de Pantalla y Colores para Menús y Cuadros
SCREEN 0
WIDTH 80, 25
COLOR blanco, negro
PALETTE 3, 25

'************************************************
' Menu Final.
' Opciones:
'    1. Imprimir Resultados en Papel
'    2. Regresar a Menu de Identificación
'************************************************
'Borrar Pantalla
CLS

'Definición de Arreglos para la impresión de Resultados
'datim(): Nombres Cuadro Resultados
'cimpr(): Nombres Menu Final
```

```
'cdat$(): Datos Cuadro Resultados
'cim$(): Datos Menu Final
DIM datim(20) AS STRING
DIM cimpr(10) AS STRING
DIM cdat$(20)
DIM cim$(10)

'Definición Cuadro de Resultados
NS = NAS + NBS
datim(1) = "          RESULTADOS          "

'Nombre de Parámetros del Modelo
FOR J = 1 TO NAS
   datim(J + 1) = " Am(" + LTRIM$(RTRIM$(STR$(J))) + ") = "
NEXT J
FOR J = 1 TO NBS
   datim(NAS + J + 1) = " Bm(" + LTRIM$(RTRIM$(STR$(J))) + ") = "
NEXT J

'Nombre de Parámetros Calculados
FOR J = 1 TO NA
   datim(NS + J + 1) = " A(" + LTRIM$(RTRIM$(STR$(J))) + ") = "
NEXT J
FOR J = 1 TO NB
   datim(NS + NA + J + 1) = " B(" + LTRIM$(RTRIM$(STR$(J))) + ") = "
NEXT J

'Iteraciones
datim(NS + N + 2) = " Iteraciones = "

'Conversión de Datos a Caracteres
FOR J = 1 TO NS
   cdat$(J + 1) = convert$(TS(J))
NEXT J
FOR J = 1 TO N
   cdat$(NS + J + 1) = convert$(TF(J))
NEXT J
cdat$(NS + N + 2) = convert$(DK)

'Definición Menu Final
cimpr(1) = "          FIN DEL CONTROL          "
cimpr(2) = " Imprimir resultados"
cimpr(3) = " Terminar "

'Menu Final y Cuadro de Resultados
DO

   'Accesible: Menú Final
   'Solo visible: Cuadro de Resultados
   ERF% = 0
   J = menuda(2, 2, NS + N + 2, datim(), 1, cdat$())
   I = menuda(5, 41, 3, cimpr(), 2, cim$())

   SELECT CASE I
```

```
        'Impresión de Resultados en Papel
        CASE 1

            'Borrar zona de mensajes de Error
            clerr

            'Imprimir
            CALL printer(para$(), TF(), pol(), DK, 2)

    END SELECT

'Repetir hasta la obtener Opción Terminar
LOOP UNTIL I = 2

END SUB
```

FUNCION convert$

```
'=============================================================================
' FUNCION convert$
' Transforma un dato numérico en un arreglo de caracteres
' Parámetro de Entrada:
'    d: dato numérico
' Salida: Arreglo de caracteres
'=============================================================================
DEFINT I-N
DEFDBL A-H, O-Z
FUNCTION convert$ (D)

'Transformación de dato a caracteres
A$ = LTRIM$(STR$(D))

'Verificación Tamaño y tipo de Dato
IF D >= 1E+08 AND INSTR(A$, "D") = 0 THEN
   D$ = LEFT$(A$, 1) + "." + MID$(A$, 2, 2) + "E+"
   IF LEN(A$) < 10 THEN
      D$ = D$ + "0" + LTRIM$(STR$(LEN(A$) - 1))
   ELSE
      D$ = D$ + LTRIM$(STR$(LEN(A$) - 1))
   END IF
ELSE
   IF INSTR(A$, "D") <> 0 THEN
      D$ = LEFT$(A$, 4) + RIGHT$(RTRIM$(STR$(D)), 4)
   ELSE
      D$ = LEFT$(A$, 8)
   END IF
END IF

'Salida
convert$ = D$

END FUNCTION
```

SUBRUTINA datin

```
'=================================================================
' SUBRUTINA datin
' Recupera los valores Predefinidos para los Parámetros, Valores
' Iniciales y Polos del proceso de Identificación o Control
' Parámetros de Entrada:
'    para$(): Arreglo con los PARAMETROS
'    val$(): Arreglo con los  VALORES INICIALES
'    pol$(): Arreglo con los Coeficientes del Polinomio de POLOS
'=================================================================
DEFINT I-N
DEFDBL A-H, O-Z
SUB datin (para$(), val$(), pol$())

'Recuperación Parámetros predefinidos
para$(2) = "2"
para$(3) = "10000"
para$(4) = "1"
para$(5) = "0"
para$(6) = "10"
para$(7) = "100"
para$(8) = "10"

'Recuperación Valores Iniciales predefinidos
val$(2) = "0.1"
val$(3) = "0.1"
val$(4) = "0.1"
val$(5) = "0.1"
val$(6) = "0.1"
val$(7) = "0.1"
val$(8) = "0.1"
val$(9) = "0.1"

'Recuperación coeficientes del polinomio de Polos
pol$(2) = "-1.32"
pol$(3) = "0.5"
pol$(4) = "0"

END SUB
```

SUBRUTINA datos

```
'=================================================================
' SUBRUTINA datos
' Crea un archivo con los datos del proceso y accede a LOTUS para su
' graficación.
' Genera el Menu de Datos y el Cuadro de definición del Archivo
' Menu de Datos:
'   Opciones:
'   1. Definir Nuevo Nombre de Archivo
'   2. Salida de datos al archivo
'   3. Cambio al LOTUS
'   4. Volver al Menu de Control
'=================================================================
DEFINT I-N
DEFDBL A-H, O-Z
SUB datos STATIC

'Definición arreglos para Menu de Datos
DIM DT(10) AS STRING
DIM nom(10) AS STRING
DIM dd$(5)
DIM dn$(2)

'Definición Menu de Datos
DT(1) = "      DATOS PARA IMPRESION       "
DT(2) = ". Nuevo Archivo"
DT(3) = " Sacar datos al disco"
DT(4) = " Imprimir (LOTUS)"
DT(5) = " Terminar"

'Definición Cuadro Archivo
nom(1) = "       DEFINICION DEL ARCHIVO       "
nom(2) = " Nombre Archivo = "

'Nombre del archivo predefinido
dn$(2) = "datos"

'Inicializar pantalla y puntero
'd1% = 0: Todo por hacer
'd2% = 1: Datos ya sacados al archivo
'd3% = 2: Matriz de Datos Borrada
CLS
d1% = 0

'Menu de Datos
DO

   'Accesible: Menu de Datos
   I = menuda(2, 2, 5, DT(), 2, dd$())
   SELECT CASE I

      'Nuevo Nombre del Archivo
      CASE 1
```

```
    'Borrar zona de mensajes de error
    clerr

    'Matriz de Datos Existente
    IF d1% <> 2 THEN

       'Accesible: Cuadro de definición de Archivo
       'Solo visible: Menu de Datos
       J = menuda(2, 2, 5, DT(), 1, dd$())
       J = menuda(15, 15, 2, nom(), 3, dn$())
       CLS

    'Matriz de Datos Borrada
    ELSE

       'Error
       BEEP

    END IF

'Sacar datos al archivo
CASE 2

    'Borrar zona de mensajes de error
    clerr

    'Matriz de Datos Existente
    IF d1% <> 2 THEN

       'Datos sacados al archivo
       d1% = 1

       'Solo visible: Menu de Datos
       J = menuda(2, 2, 5, DT(), 1, dd$())

       'Impresión Mensaje de Salida de datos
       COLOR blancoi, azul
       LOCATE 19, 15
       PRINT "Archivo: "; dn$(2) + ".dat";
       LOCATE 21, 15
       COLOR verde + 16, negro
       PRINT "SALIDA DE DATOS EN PROCESO"

       'Salida de datos
       arch$ = "d:\adaptivo\" + dn$(2) + ".dat"
       OPEN arch$ FOR OUTPUT AS #1
       FOR I = 0 TO MaxDat
          FOR J = 1 TO N + 1
             PRINT #1, USING "+##.########"; DAT(I, J);
          NEXT J
          PRINT #1, USING "+##.########"; DAT(I, N + 2)
       NEXT I
       COLOR blanco, negro
       CLOSE #1
       CLS
```

```
            'Matriz de Datos Borrada
            ELSE

               'Error
               BEEP

            END IF

         'Ejecutar LOTUS
         CASE 3

            'Borrar zona de mansajes de error
            clerr

            'Datos sacados a archivo
            IF d1% = 1 THEN

               'Verificación existencia de archivos
               NFF% = 0
               ERF% = 0
               OPEN "d:\adaptivo\tsgraf.bat" FOR INPUT AS #2
               IF ERF% = 0 THEN
                  OPEN "d:\adaptivo\tesisad.wk1" FOR INPUT AS #3
               END IF
               CLOSE
               COLOR blanco, negro

               'Archivos existentes
               IF ERF% = 0 THEN

                  'Matriz de Datos Borrada
                  d1% = 2
                  REDIM DAT(1, 1)
                  CLS

                  'Ir a LOTUS
                  SHELL "tsgraf.bat"
                  CLS

               END IF

            'Datos todavía no sacados a archivo
            ELSE

               'Error
               BEEP

            END IF

      END SELECT

   'Repitir hasta que se escoja la opción 3
   LOOP UNTIL I = 4

END SUB
```

SUBRUTINA defgraf

```
'=============================================================================
' SUBRUTINA defgraf
' Define e Imprime los gráficos de Entrada y Salida en pantalla
'=============================================================================
DEFINT I-N
DEFDBL A-H, O-Z
SUB defgraf

'Construcción Gráfico de Y
LINE (10, 10)-(629, 140), 1, BF
LINE (10, 10)-(629, 140), 2, B
LINE (20, 125)-(619, 125), 7

'Impresión Nombres de Ejes
LOCATE 17, 30
PRINT "TIEMPO"
LOCATE 5, 3
PRINT "Y"
LINE (30, 20)-(30, 130), 7

'Construcción Gráfico de U
LINE (10, 185)-(629, 315), 1, BF
LINE (10, 185)-(629, 315), 2, B
LINE (20, 300)-(619, 300), 7

'Impresión Nombres de Ejes
LOCATE 39, 30
PRINT "TIEMPO"
LOCATE 27, 3
PRINT "U"
LINE (30, 195)-(30, 305), 7

END SUB
```

SUBRUTINA ersist

```
'=============================================================================
' SUBRUTINA ersist
' Genera mensajes de error si hay un error en la solución del
' sistema de ecuaciones.
'=============================================================================
DEFINT I-N
DEFDBL A-H, O-Z
SUB ersist

'Error detectado
BEEP

'Impresión en pantalla de mensajes de error
COLOR rojo
LOCATE 42, 3
PRINT "Matrix Singular: Control Suspendido";
COLOR 15

'Fin del control
LOCATE 42, 40
PRINT "Presione una Tecla para terminar...";
DO
LOOP WHILE INKEY$ = ""

END SUB
```

SUBRUTINA graf

```
'=================================================================
' SUBRUTINA graf
' Grafica el nuevo valor de Y y U en el gráfico de pantalla.
' Parámetro de Entrada:
'    I: Posición de los nuevos puntos
'=================================================================
DEFINT I-N
DEFDBL A-H, O-Z
SUB graf (I)

'Si Y>10 Imprimir en rojo
IF Y > 10 THEN
   PSET (I + 30, 25), 4

' Si Y<0 Imprimir en rojo
ELSEIF Y < 0 THEN
   PSET (I + 30, 125), 4

'Si 0<Y<10 Imprimir en amarillo
ELSE
   YG = -10 * Y + 125
   PSET (I + 30, YG), 14
END IF

'Si U>10 imprimir en rojo
IF U > 10 THEN
   PSET (I + 30, 200), 4

'Si U<0 imprimir en rojo
ELSEIF U < 0 THEN
   PSET (I + 30, 300), 4

'Si 0<U<10 imprimir en amarillo
ELSE
   UG = -10 * U + 300
   PSET (I + 30, UG), 14
END IF

END SUB
```

SUBRUTINA ident

```
'=================================================================
' SUBRUTINA ident
' Ejecuta el proceso de IDENTIFICACION en SIMULACION en base a los
' parámetros y valores iniciales definidos.
' Produce los gráficos en pantalla de la señal de entrada ( u(t) ) y
' de salida ( y(t) ) y calcula los parámetros de la planta.
' Parámetros de Entrada:
'   para$(): Parámetros definidos en "sidat"
'   val$(): Valores Iniciales definidos en "sidat"
'=================================================================
DEFINT I-N
DEFDBL A-H, O-Z
SUB ident (para$(), val$())

'Recuperación de Parámetros
'Par: Valor inicial de α
'R: Referencia
'Umin: valor mínimo permitido de U
'Umax: valor máximo permitido de U
'MaxDat: Número máximo de Datos para la Matriz de Datos
'ERP: Ruido Porcentual
Par = VAL(para$(3))
R = VAL(para$(4))
Umin = VAL(para$(5))
Umax = VAL(para$(6))
MaxDat = VAL(para$(7))
ERP = VAL(para$(8))

'Dimensionamiento de Matrices y Vectores
'P(): Matriz de Covarianzas
'MA(), MB(): Matrices auxiliares de cálculo
'T(): Vector de Parámetros (modelo ARMA)
'X(): Vector de Datos Entrada-Salida (modelo ARMA)
'V(): Vector auxiliar de cálculo
'TM(): Matriz con las últimos 50 cálculos de parámetros
'TF(): Vector con los Parámetros Promedio
'DAT(): Matriz con los Parámetros y las señales de entrada y salida
REDIM P(N, N)
REDIM MA(N, N) AS DOUBLE, MB(N, N) AS DOUBLE
REDIM T(N), X(N), V(N), TM(0 TO 49, N), TF(N)
REDIM DAT(MaxDat, N + 2)

'Inicialización de T
FOR I = 2 TO N + 1
   T(I - 1) = VAL(val$(I))
NEXT I

'Inicialización de P, TF y X
FOR I = 1 TO N
   FOR J = 1 TO N
      IF I = J THEN
         P(I, J) = Par
      ELSE
```

```
            P(I, J) = 0
        END IF
    NEXT J
    TF(I) = 0
    X(I) = 0
NEXT I

'Inicialización de XS
FOR I = 1 TO NAS + NBS
    XS(I) = 0
NEXT I

'Inicialización de Variables y Constantes
'K: cuenta el número de iteraciones para graficación
'DK: cuenta el número total de iteraciones
'U: señal de entrada a la planta
'F: Factor de olvido
'Y: señal de salida de la planta
K = 0
DK = 0
U = 1
F = .98
Y = 0
DAT(DK, 1) = Y
DAT(DK, 2) = U
RANDOMIZE TIMER

'Definición de Pantalla y Colores para graficación
SCREEN 9
WIDTH 80, 43
CLS

'Iniciar Identificación
LOCATE 24, 30
PRINT "PRESIONE ESPACIO PARA INICIAR IDENTIFICACION"
DO
    q$ = INKEY$
LOOP WHILE q$ = ""

'Impresión en pantalla de los gráficos y mensajes iniciales
CLS
CALL defgraf
COLOR 15
LOCATE 21, 30
PRINT "SIMULACION"
LOCATE 42, 10
COLOR 2
PRINT "IDENTIFICACION EN PROCESO";
COLOR 15
LOCATE 42, 40
PRINT "Presione una Tecla para terminar...";

'Inicio del lazo de Identificación
DO
```

```
'Rutina de Actualización de Datos Entrada-Salida
FOR I = NA TO 2 STEP -1
   X(I) = X(I - 1)
NEXT I
X(1) = -Y
FOR I = N TO N - NB + 2 STEP -1
   X(I) = X(I - 1)
NEXT I

'Cálculo de U
U = R * (1 + ERP / 100 * RND - ERP / 200)

'U debe estar entre Umin y Umax
IF U > Umax THEN
   U = Umax
ELSEIF U < Umin THEN
   U = Umin
END IF

'Cálculo de Y
CALL calcy

'Actualización de U
X(NA + 1) = U

'Inicio Algoritmo de Mínimos Cuadrados Recursivo
CALL pvect(V(), P(), X())
A = F + pesc(X(), V())
FOR I = 1 TO N
   V(I) = V(I) / A
NEXT I
E = Y - pesc(X(), T())

'Cálculo Nuevos Parámetros
FOR I = 1 TO N
   T(I) = T(I) + V(I) * E
NEXT I

'Ingreso de valores a la Matriz de Datos
IF DK < MaxDat THEN
   DAT(DK + 1, 1) = Y
   DAT(DK + 1, 2) = U
   FOR I = 3 TO N + 2
      DAT(DK + 1, I) = T(I - 2)
   NEXT I
END IF

'Ingreso de valores a la Matriz de Parámetros
FOR I = 3 TO N + 2
   TM(DK MOD 50, I - 2) = T(I - 2)
NEXT I

'Cálculo de Matriz de Covarianzas P.
FOR I = 1 TO N
   FOR J = 1 TO N
```

```
            IF I = J THEN
               MA(I, J) = 1 - V(I) * X(J)
            ELSE
               MA(I, J) = -V(I) * X(J)
            END IF
         NEXT J
      NEXT I
      CALL pmat(MB(), MA(), P())
      FOR I = 1 TO N
         FOR J = 1 TO N
            P(I, J) = MB(I, J) / F
         NEXT J
      NEXT I

      'Nuevo Gráfico si más de 580 valores
      IF K > 580 THEN

         'Nuevo Gráfico
         CALL defgraf

         'Inicialización de K
         K = 0

      END IF

      'Graficar Nuevo Punto
      CALL graf(K)

      'Incrementar contadores
      K = K + 1
      DK = DK + 1

'Repetir el lazo hasta la detección de una tecla
LOOP WHILE INKEY$ = ""

'Fin de Identificación
LOCATE 42, 2
PRINT SPACE$(8); "Presione una Tecla para continuar..."; SPACE$(29);
DO
LOOP WHILE INKEY$ = ""

'Cálculo de Parámetros Promedio
FOR I = 1 TO N
   IF DK > 50 THEN
      FOR J = 0 TO 49
         TF(I) = TF(I) + TM(J, I)
      NEXT J
      TF(I) = TF(I) / 50
   ELSE
      TF(I) = T(I)
   END IF
NEXT I

'Definición de Pantalla y Colores para Menús y Cuadros
SCREEN 0
```

```
WIDTH 80, 25
COLOR blanco, negro
PALETTE 3, 25

'************************************************
' Menu Final.
' Opciones:
'    1. Imprimir Resultados en Papel
'    2. Regresar a Menu de Identificación
'************************************************

'Borrar Pantalla
CLS

'Definición de Arreglos para la impresión de Resultados
'datim(): Nombres Cuadro Resultados
'cimpr(): Nombres Menu Final
'cdat$(): Datos Cuadro Resultados
'cim$(): Datos Menu Final
DIM datim(20) AS STRING
DIM cimpr(10) AS STRING
DIM cdat$(20)
DIM cim$(10)

'Definición Cuadro de Resultados
NS = NAS + NBS
datim(1) = "         RESULTADOS          "

'Nombre de Parámetros del Modelo
FOR J = 1 TO NAS
   datim(J + 1) = " Am(" + LTRIM$(RTRIM$(STR$(J))) + ") = "
NEXT J
FOR J = 1 TO NBS
   datim(NAS + J + 1) = " Bm(" + LTRIM$(RTRIM$(STR$(J))) + ") = "
NEXT J

'Nombre de Parámetros Calculados
FOR J = 1 TO NA
   datim(NS + J + 1) = " A(" + LTRIM$(RTRIM$(STR$(J))) + ") = "
NEXT J
FOR J = 1 TO NB
   datim(NS + NA + J + 1) = " B(" + LTRIM$(RTRIM$(STR$(J))) + ") = "
NEXT J

'Iteraciones
datim(NS + N + 2) = " Iteraciones = "

'Conversión de Datos a Caracteres
FOR J = 1 TO NS
   cdat$(J + 1) = convert$(TS(J))
NEXT J
FOR J = 1 TO N
   cdat$(NS + J + 1) = convert$(TF(J))
NEXT J
cdat$(NS + N + 2) = convert$(DK)
```

```
'Definición Menu Final
cimpr(1) = "    FIN DE LA IDENTIFICACION      "
cimpr(2) = " Imprimir resultados"
cimpr(3) = " Terminar "

'Menu Final y Cuadro de Resultados
DO

   'Accesible: Menú Final
   'Solo visible: Cuadro de Resultados
   ERF% = 0
   J = menuda(2, 2, NS + N + 2, datim(), 1, cdat$())
   I = menuda(5, 39, 3, cimpr(), 2, cim$())

   SELECT CASE I

      'Impresión de Resultados en Papel
      CASE 1

         'Borrar zona de mensajes de Error
         clerr

         'Imprimir
         CALL printer(para$(), TF(), T(), DK, 1)

   END SELECT

'Repetir hasta la obtener Opción Terminar
LOOP UNTIL I = 2

END SUB
```

FUNCION menu

```
'=============================================================================
' FUNCION menu
' Genera los menus Principal, Simulación y Tiempo Real
' Parámetros de Entrada:
'    supx: abscisa de la posición superior izquierda del recuadro
'    supy: ordenada de la posición superior izquierda del recuadro
'    N%: número de opciones posibles + 1
'    mat(): arreglo con los nombres del Titulo y las opciones
'         mat(1): Titulo
'         mat(2) a mat(N): Opciones
'    b!: Detector de procedimientos
'         Valores posibles:
'         0: Modelo o Parámetros no definidos
'         1: Modelo definido
'         2: Parámetros Definidos
'         3: Proceso ejecutado
' Salida: Opción escogida
'=============================================================================
FUNCTION menu (supx, supy, N%; mat() AS STRING, b!) STATIC

'Calculo de la Longitud del Título
LON% = LEN(mat(1))

'Definición de Colores y Ubicación del cursor
COLOR blanco, azul
LOCATE supx, supy

'Impresión encabezado
PRINT "╔"; STRING$(LON% + 10, 205); "╗"
LOCATE supx + 1, supy
PRINT "║"; SPACE$(5); mat(1); SPACE$(5); "║"
LOCATE supx + 2, supy
PRINT "╠"; STRING$(LON% + 10, 205); "╣"

'Ubicación del puntero para la opción 1
pt% = 2

'Proceso de impresión del resto del menu y detección de la opción
DO
   J = 3
   FOR K = 2 TO N%
      piv% = 5 + LON% - LEN(mat(K))
      LOCATE supx + J, supy
      PRINT "║";
         IF pt% = K THEN

            'Colores para la opción escogida
            COLOR blancoi, magenta

         ELSE

            'Colores para el resto de opciones
            COLOR blanco, azul
```

```
      END IF

   PRINT SPACE$(5); mat(K); SPACE$(piv%);
   COLOR blanco, azul
   PRINT "||"
   J = J + 1
NEXT K

'Impresión línea final
LOCATE supx + 2 + N%, supy
PRINT "╚"; STRING$(LON% + 10, 205); "╝"

'Detección de tecla
s$ = INKEY$

'Selección de rutina en función de la tecla detectada
SELECT CASE s$

   'Flecha hacia arriba
   CASE CHR$(0) + CHR$(72)

      'Modelo o Parámetros definidos
      IF b! >= 1 THEN

         'Si el puntero está en la primera opción, ir a la última
         IF pt% = 2 THEN
            pt% = N%

         ELSE

            'Si el puntero está en la última opción, y no se ha
            'ejecutado el proceso, saltar sobre la MATRIZ DE DATOS
            IF (b! = 2 AND pt% = 7) OR (b! = 1 AND pt% = 6) THEN
               pt% = pt% - 2

            'En otro caso, ir a la opción inmediatamente anterior
            ELSE
               pt% = pt% - 1

            END IF

         END IF

      'Si el Modelo o los Parámetros no están definidos
      ELSE

         'Si el puntero está en opción 1 ir a la última opción
         IF pt% = 2 THEN
            pt% = N%

         'Si no, ir a la primera opción
         ELSE
            pt% = 2

         END IF
```

```
      END IF

   'Flecha hacia abajo
   CASE CHR$(0) + CHR$(80)

      'Modelo o Parámetros definidos
      IF b! >= 1 THEN

         'Si el puntero está en la primera opción, ir a la última
         IF pt% = N% THEN
            pt% = 2

         ELSE

            'Si el puntero está en la opción anterior a MATRIZ DE
            'DATOS y no se ha ejecutado el proceso, ir a la últim
            'opción
            IF (b! = 2 AND pt% = 5) OR (b! = 1 AND pt% = 4) THEN
               pt% = pt% + 2

            'En otro caso, ir a la opción inmediatamente anterior
            ELSE
               pt% = pt% + 1

            END IF

         END IF

      'Modelo o parámetros no definidos
      ELSE

         'Si el puntero está en la última opción, ir a la primera
         IF pt% = N% THEN
            pt% = 2

         'Si no, ir a la última opción
         ELSE
            pt% = N%

         END IF

      END IF

   'Enter
   CASE CHR$(13)

      'Opción seleccionada
      menu = pt% - 1

      'Redefinición de colores
      COLOR blanco, negro

   'Resto de Teclas
   CASE ELSE
```

```
    END SELECT

'Repetir el proceso hasta que se teclee ENTER
LOOP UNTIL s$ = CHR$(13)

END FUNCTION
```

FUNCION menuda

```
'=============================================================================
' FUNCION menuda
' Genera los menus y cuadros que no se generan con "menu"
' Parámetros de Entrada:
'   supx: abscisa de la posición superior izquierda del recuadro
'   supy: ordenada de la posición superior izquierda del recuadro
'   N%: número de opciones posibles + 1
'   mat(): arreglo con los nombres del Titulo y las opciones
'       mat(1): Titulo
'       mat(2) a mat(N): Opciones
'   b!: Detector de tipo de datos
'       Valores posibles:
'       1: Datos solo visibles
'       2: Menu de opciones
'       3: Palabra (Números y Letras)
'       10: Parámetros
'       20: Valores Iniciales
'       30: Otros Datos Numéricos (Parámetros Modelo, Polos)
'   dato$(): arreglo de datos de los cuadros en pantalla
' Salida: Opción escogida
'=============================================================================
DEFINT I-N
DEFDBL A-H, O-Z
FUNCTION menuda (supx, supy, N%, mat() AS STRING, b!, dato$())

'Calculo de la Longitud del Título
LON% = LEN(mat(1))

'Detección del acceso al cuadro o menu
IF b! = 1 THEN

   'Solo visible
  COLOR blanco, cian
ELSE

   'Accesible
   COLOR blanco, azul
END IF

'Ubicación del cursor
LOCATE supx, supy

'Impresión encabezado
PRINT "╔"; STRING$(LON% + 6, 205); "╗"
LOCATE supx + 1, supy
PRINT "║"; SPACE$(3); mat(1); SPACE$(3); "║"
LOCATE supx + 2, supy
PRINT "╠"; STRING$(LON% + 6, 205); "╣"

'Ubicación del puntero para la opción 1
pt% = 2

'Proceso de impresión del resto del menu y detección de la opción
```

```
DO
   J = 3
   FOR K = 2 TO N%
      piv% = 3 + LON% - LEN(mat(K))
      LOCATE supx + J, supy
      PRINT "║";

         'Colores para la opción escogida
         IF pt% = K THEN

            'Colores de acceso
            IF b! <> 1 THEN
               COLOR blancoi, rojo
            ELSE
               COLOR blanco, cian
            END IF

         'Colores para el resto de opciones
         ELSE

            'Colores de acceso
            IF b! = 1 THEN
               COLOR blanco, cian
            ELSE
               COLOR blanco, azul
            END IF

         END IF

      PRINT SPACE$(3); mat(K); SPACE$(piv% - 12); dato$(K); SPACE$(12)
- LEN(dato$(K)));
      IF b! = 1 THEN
         COLOR blanco, cian
      ELSE
         COLOR blanco, azul
      END IF
      PRINT "║"
      J = J + 1
   NEXT K

   'Impresión línea final
   LOCATE supx + 2 + N%, supy
   PRINT "╚"; STRING$(LON% + 6, 205); "╝";

   'Acceso Permitido
   IF b! <> 1 THEN

      'Detección de tecla
      s$ = ""
      s$ = INKEY$

      'Selección de rutina en función de la tecla detectada
      SELECT CASE s$

         'Flecha hacia arriba
```

```
CASE CHR$(0) + CHR$(72)
   'Validación datos presentes
   valor = valid(b!, dato$(), pt%)

   'Validación aceptada
   IF valor = 0 THEN

      'Si el puntero está en la primera opción,
      'ir a la última
      IF pt% = 2 THEN
         pt% = N%

      'Si no, pasar la anterior
      ELSE
         pt% = pt% - 1

      END IF

   'Validación negada
   ELSE

      'Error: el puntero no cambia de posición
      BEEP

   END IF

'Flecha hacia abajo
CASE CHR$(0) + CHR$(80)

   'Validación datos presentes
   valor = valid(b!, dato$(), pt%)

   'Validación aceptada
   IF valor = 0 THEN

      'Si el puntero está en la primera opción,
      'ir a la última
      IF pt% = N% THEN
         pt% = 2

      'Si no, ir a la siguiente
      ELSE
         pt% = pt% + 1

      END IF

   'Validación negada
   ELSE

      'Error: el puntero no cambia de posición
      BEEP

   END IF

'Enter
```

```
        CASE CHR$(13)

            'Validación Datos Presentes
            valor = valid(b!, dato$(), pt%)

            'Validación aceptada
            IF valor = 0 THEN

                'Salida
                menuda = pt% - 1
                COLOR blanco, negro

            'Validación negada
            ELSE

                'Error: el puntero no cambia de posición
                BEEP

            END IF

        'Ninguna tecla detectada: seguir esperando
        CASE ""

        'Tecla diferente de las anteriores: Ingreso de Datos
        CASE ELSE

            'Longitud máxima del dato
            NUMAX = 8

            'Detección tipo de dato
            SELECT CASE b!

                'Menu de opciones
                CASE 2

                'Palabra (letras o números)
                CASE 3

                    'caracter ingresado es una letra
                    IF (ASC(s$) >= 65 AND ASC(s$) <= 90) OR (ASC(s$) >
97 AND ASC(s$) <= 122) THEN

                        'Ingreso posible de nueva letra
                        IF LEN(dato$(pt%)) < NUMAX THEN
                            dato$(pt%) = dato$(pt%) + s$

                        'Ingreso imposible de nueva letra
                        ELSE

                            'Error: el cursor se mantiene en la misma
                            'posición
                            BEEP

                        END IF
```

```
                    'Detección de "Retorno de un espacio":
                    'válido si hay datos
                    ELSEIF s$ = CHR$(8) AND LEN(dato$(pt%)) <> 0 THEN

                       'Borrar última letra
                       dato$(pt%) = LEFT$(dato$(pt%), LEN(dato$(pt%)) -
1)

                    'Detección de número
                    'válido si no está al inicio de la palabra
                    ELSEIF (ASC(s$) >= 48 AND ASC(s$) <= 57 AND
LEN(dato$(pt%)) >= 1) THEN
                       dato$(pt%) = dato$(pt%) + s$

                    'Caracter diferente de Enter
                    ELSEIF s$ <> CHR$(13) THEN

                       'Error: cursor se mantiene en la misma posición
                       BEEP

                    END IF

                 'Número
                 CASE ELSE

                    'Caracter es un número o el punto
                    IF (ASC(s$) >= 48 AND ASC(s$) <= 57) OR ASC(s$) = 46
THEN

                       'Es posible ingresar nuevo caracter
                       IF LEN(dato$(pt%)) < NUMAX THEN

                          'Punto
                          IF ASC(s$) = 46 THEN

                             'No hay datos todavía:
                             'Ingresar el punto
                             IF INSTR(dato$(pt%), s$) = 0 THEN
                                dato$(pt%) = dato$(pt%) + s$

                             'Sí hay datos
                             ELSE

                                'Error: el cursor se mantiene en la
                                'misma posición
                                BEEP

                             END IF

                          'Numero
                          ELSE

                             'Ingresar nuevo dígito
                             dato$(pt%) = dato$(pt%) + s$
```

```
                    END IF

                'Imposible ingresar nuevo caracter
                ELSE

                    'Error: el cursor se mantiene en la misma
                    'posición
                    BEEP

                END IF

            'Signo menos
            ELSEIF ASC(s$) = 45 THEN

                'Si no hay datos, ingresar signo menos
                IF LEN(dato$(pt%)) = 0 THEN
                    dato$(pt%) = dato$(pt%) + s$

                'Si hay datos, error
                ELSE

                    'Error: el cursor se mantiene en la misma
                    'posición
                    BEEP

                END IF

            'Detección de "Retorno de un espacio":
            'válido si hay datos
            ELSEIF s$ = CHR$(8) AND LEN(dato$(pt%)) <> 0 THEN

                'Borrar última letra
                dato$(pt%) = LEFT$(dato$(pt%), LEN(dato$(pt%))
1)

            'Caracter diferente de Enter
            ELSEIF s$ <> CHR$(13) THEN

                'Error: el puntero se mantiene en la misma
                'posición
                BEEP

            END IF

        END SELECT

    END SELECT

  END IF

'Repetir hasta detectar Enter y los datos están validados
LOOP UNTIL (s$ = CHR$(13) AND valor = 0) OR b! = 1

END FUNCTION
```

```
SUBRUTINA modelo

'=============================================================================
' SUBRUTINA modelo
' Genera el Menu de Modelo, el de Orden y permite la definición
' de los parámetros
' Menu de Orden:
'    Opciones:
'    1. Definir el NUEVO ORDEN del modelo
'    2. Definir o cambiar los PARAMETROS del modelo
'    3. Recuperar el MODELO PREDEFINIDO
'    4. Retornar al Menu de Simulación
'=============================================================================
DEFINT I-N
DEFDBL A-H, O-Z
SUB modelo STATIC

'Definición de arreglos para la definición del modelo
DIM model(10) AS STRING      'Titulo y opciones
DIM modo(10) AS STRING       '
DIM modp(10) AS STRING       '
DIM modin$(5)                '
DIM mord$(2)                 '
DIM mpar$(10)                '

'Definición del Menu de Modelo
model(1) = "            MODELO              "
model(2) = " Nuevo orden"
model(3) = " Parámetros"
model(4) = " Modelo Predefinido"
model(5) = " Terminar"

'Definición del Menu de Orden
modo(1) = "   ORDEN DEL MODELO    "
modo(2) = " Orden = "

'Modelo Predefinido
IF F = 0 THEN
   NAS = 2
   NBS = 2
   NS = NAS + NBS
   mord$(2) = "2"
   CALL parin(mpar$())
   F = 1
END IF

'Definición Cuadro de Parámetros
modp(1) = "       PARAMETROS        "
FOR K = 1 TO NAS
   modp(K + 1) = " Am(" + LTRIM$(RTRIM$(STR$(K))) + ") = "
NEXT K
FOR K = 1 TO NBS
   modp(NAS + K + 1) = " Bm(" + LTRIM$(RTRIM$(STR$(K))) + ") = "
NEXT K
```

```
CLS

'Menu de Modelo, Orden y Cuadro de Parámetros
DO

    'Menu de Modelo accesible
    'Orden y Parámetros no accesibles pero sí visibles
    J = menuda(3, 50, 2, modo(), 1, mord$())
    J = menuda(10, 50, NS + 1, modp(), 1, mpar$())
    I = menuda(2, 2, 5, model(), 2, modin$())

    SELECT CASE I

        'Menu de Orden accesible
        'Modelo y Parámetros no accesibles pero sí visibles
        CASE 1
            J = menuda(10, 50, NS + 1, modp(), 1, mpar$())
            J = menuda(2, 2, 5, model(), 1, modin$())
            J = menuda(3, 50, 2, modo(), 10, mord$())

            'Nuevo Orden y recuperación parámetros predefinidos
            NS = 2 * VAL(mord$(2))
            CALL parin(mpar$())
            NAS = NS / 2
            NBS = NAS
            FOR K = 1 TO NAS
              modp(K + 1) = " Am(" + LTRIM$(RTRIM$(STR$(K))) + ") = "
            NEXT K
            FOR K = 1 TO NBS
               modp(NAS + K + 1) = " Bm(" + LTRIM$(RTRIM$(STR$(K))) + ")
= "
            NEXT K
            CLS

        'Cuadro de Parámetros accesible
        'Modelo y Orden no accesibles pero sí visibles
        CASE 2
            J = menuda(2, 2, 5, model(), 1, modin$())
            J = menuda(3, 50, 2, modo(), 1, mord$())
            J = menuda(10, 50, NS + 1, modp(), 30, mpar$())
            CLS

        'Modelo, Orden y Parámetros sólo visibles
        CASE 3

            'Recuperación paámetros predefinidos
            CALL parin(mpar$())
            CLS

        CASE ELSE

    END SELECT

'Repetir hasta la opción Terminar
LOOP UNTIL I = 4
```

```
'Redefinir tamaños de XS() y TS()
REDIM XS(NS), TS(NS)

'Transferencia de nuevos parámetros
FOR I = 1 TO NS
   TS(I) = VAL(mpar$(I + 1))
NEXT I

END SUB
```

SUBRUTINA parin

```
'=============================================================================
' SUBRUTINA parin
' Recupera los Parámetros del Modelo Predefinido
' Están definidos Modelos de 1º, 2º, 3º y 4º orden.
' Parámetro de Salida:
'   mp$(): arreglo con los parámetros recuperados.
'=============================================================================
DEFINT I-N
DEFDBL A-H, O-Z
SUB parin (mp$())

'Calculo del orden del modelo
M1 = NS / 2

'Recuperación de Parámetros según el orden
SELECT CASE M1

   'Primer Orden
   CASE 1
      mp$(2) = "-.5"
      mp$(3) = ".5"

   'Segundo Orden
   CASE 2
      mp$(2) = "1"
      mp$(3) = ".4"
      mp$(4) = ".5"
      mp$(5) = ".9"

   'Tercer Orden
   CASE 3
      mp$(2) = ".5"
      mp$(3) = ".4"
      mp$(4) = "-.5"
      mp$(5) = ".9"
      mp$(6) = "-.3"
      mp$(7) = ".2"

   'Cuarto Orden
   CASE 4
      mp$(2) = ".2"
      mp$(3) = ".4"
      mp$(4) = "-.5"
      mp$(5) = "-.3"
      mp$(6) = ".8"
      mp$(7) = ".5"
      mp$(8) = "-.3"
      mp$(9) = ".4"

END SELECT

END SUB
```

FUNCION pesc

```
'=================================================================
' FUNCION pesc
' Calcula el Producto Escalar entre dos Vectores
' Parámetros de Entrada:
'    V1#(): Primer Vector
'    V2#(): Segundo Vector
' Salida: Producto Escalar entre V1 y V2
'=================================================================
DEFINT I-N
FUNCTION pesc (V1#(), V2#())

'Inicialización variable auxiliar
Z# = 0

'Cálculo Producto Escalar
FOR I = 1 TO N
   Z# = V1#(I) * V2#(I) + Z#
NEXT I

'Sacar Resultado
pesc = Z#

END FUNCTION
```

SUBRUTINA pmat

```
'=============================================================================
' SUBRUTINA pmat
' Calcula el Producto de Dos Matrices
' Parámetros de Entrada:
'   M2#(): Primera Matriz
'   M3#(): Segunda Matriz
' Parámetro de Salida
'   M1#(): Matriz Resultado (M1 = M2*M3)
'=============================================================================
DEFINT I-N
SUB pmat (M1#(), M2#(), M3#())

'Cálculo del Producto de M2 por M3
FOR I = 1 TO N
   FOR K = 1 TO N
      M1#(I, K) = 0
      FOR J = 1 TO N
         M1#(I, K) = M1#(I, K) + M2#(I, J) * M3#(J, K)
      NEXT J
   NEXT K
NEXT I

END SUB
```

SUBRUTINA printer

```
'=================================================================
' SUBRUTINA printer
' Imprime los Resultados en Papel manejando una impresora
' Parámetros de Entrada:
'   para$(): Parámetros usados en la Identificación o Control
'   T(): Parámetros Identificados
'   P(): Polos
'   DI: Número de Iteraciones
'   K: Tipo de Proceso Ejecutado
'      Valores Posibles:
'      1: Identificación en SIMULACION
'      2: Identificación y Control en SIMULACION
'      3: Identificación en TIEMPO REAL
'      4: Identificación y Control en TIEMPO REAL
'=================================================================
DEFINT I-N
DEFDBL A-H, O-Z
SUB printer (para$(), T(), P(), DI, K)

'Prueba de impresión
LOCATE 24, 40
COLOR rojo + 16, negro
PRINT "Intentando Imprimir";
LPRINT CHR$(0)

'La impresión se puede realizar
IF ERF% = 0 THEN

   'Mensaje de Impresión en proceso
   LOCATE 24, 40
   COLOR verde + 16, negro
   PRINT "IMPRESION EN PROCESO";

   'Impresión encabezados
   LPRINT SPACE$(10); "            ESCUELA POLITECNICA NACIONAL
 "
   LPRINT CHR$(13);
   LPRINT SPACE$(10); "          FACULTAD DE INGENIERIA ELECTRICA
  "
   LPRINT CHR$(13)
   LPRINT SPACE$(10); "   TESIS: IDENTIFICACION DE SISTEMAS EN TIEMPO
REAL"
   LPRINT CHR$(13)

   'NLIN: Contador de número de líneas
   NLIN = 8

   'Simulación
   IF K < 3 THEN

      'Imprimir Simulación
      LPRINT CHR$(27); CHR$(45); CHR$(1)
      LPRINT "SIMULACION"
```

```
    LPRINT CHR$(27); CHR$(45); CHR$(0)

    'Imprimir Forma del Modelo
    LPRINT "                                        Σ b(i)z";
    LPRINT CHR$(27); CHR$(83); CHR$(0); "-i"; CHR$(27); CHR$(84)
    LPRINT "MODELO DE LA FORMA:     Y(z) = ---------------- U(z)"
    LPRINT "                                      1 + Σ a(i)z";
    LPRINT CHR$(27); CHR$(83); CHR$(0); "-i"; CHR$(27); CHR$(84)

    'Impresión Modelo y Parámetros
    LPRINT CHR$(13)
    LPRINT "1) MODELO DE LA PLANTA"; CHR$(13)
    LPRINT "Orden del Modelo = "; NS / 2

    FOR I = 1 TO NAS
       LPRINT "A("; I; ") = "; TS(I)
    NEXT I
    FOR I = 1 TO NBS
       LPRINT "B("; I; ") = "; TS(I + NAS)
    NEXT I
    NLIN = NLIN + 9 + NS

'Tiempo Real
ELSE

    'Imprimir Tiempo Real
    LPRINT CHR$(27); CHR$(45); CHR$(1)
    LPRINT "TIEMPO REAL"
    LPRINT CHR$(27); CHR$(45); CHR$(0)

    'Imprimir parámetros en tiempo real
    LPRINT CHR$(13)
    LPRINT "1) PARAMETROS DE TIEMPO REAL "; CHR$(13)
    LPRINT "Tiempo de muestreo = "; bintv%; " ms"
    NLIN = NLIN + 6
END IF

LPRINT CHR$(13)

'Identificación
IF K = 1 OR K = 3 THEN
   LPRINT "2) PARAMETROS DE IDENTIFICACION"; CHR$(13)

'Identificación y Control
ELSE
   LPRINT "2) PARAMETROS DE IDENTIFICACION Y CONTROL"; CHR$(13)
END IF

'Impresión parámetros del proceso
LPRINT "Orden del modelo = "; VAL(para$(2))
LPRINT "Parametro Alfa = "; VAL(para$(3))
LPRINT "Referencia = "; VAL(para$(4))
LPRINT "U minimo = "; VAL(para$(5))
LPRINT "U máximo = "; VAL(para$(6))
NLIN = NLIN + 15
```

```
    'Imprimir Ruido si no es Control en Tiempo Real
    IF K <> 4 THEN
       LPRINT "Ruido Porcentual = "; VAL(para$(8)); " %"
       NLIN = NLIN + 1
    END IF
    LPRINT "Número de Iteraciones = "; DI
    NLIN = NLIN + 1

    LPRINT CHR$(13)

    'Identificación
    IF K = 1 OR K = 3 THEN
       LPRINT "3) RESULTADOS DE IDENTIFICACION"; CHR$(13)

    'Identificación y Control
    ELSE
       LPRINT "3) UBICACION DE POLOS"
       LPRINT
       LPRINT "     Polinomio de polos: T(z) = 1 + Σ t(i)z"";
       LPRINT CHR$(27); CHR$(83); CHR$(0); "-i"; CHR$(27); CHR$(84)
       LPRINT
       LPRINT "t(1) = "; P(1)
       LPRINT "t(2) = "; P(2)
       NLIN = NLIN + 8
       IF NA > 2 THEN
          LPRINT "t(3) = "; P(3)
          NLIN = NLIN + 1
       END IF
       LPRINT CHR$(13)
       LPRINT "4) RESULTADOS DE IDENTIFICACION  Y CONTROL"; CHR$(13)
    END IF

    'Impresión Resultados de Identificación Paramétrica
    FOR I = 1 TO NA
       LPRINT "A("; I; ") = "; T(I)
    NEXT I
    FOR I = 1 TO NB
       LPRINT "B("; I; ") = "; T(I + NA)
    NEXT I

    'Cálculo número total de líneas
    NLIN = NLIN + 4 + N - 3
    NF = 66 - NLIN MOD 66

    'Pasar a la página siguiente
    FOR I = 1 TO NF
       LPRINT
    NEXT I

    'Imprimir en pantalla mensaje de Fin de Impresión
    COLOR verde, negro
    LOCATE 24, 40
    PRINT "FIN DE LA IMPRESION      ";
END IF
END SUB
```

SUBRUTINA pvect

```
'=============================================================================
' SUBRUTINA pvect
' Calcula el Producto de una Matriz por un Vector
' Parámetros de Entrada:
'    M1#(): Matriz
'    V2(): Vector
' Parámetro de Salida:
'    V1(): Vector Resultado (V1 = M1*V1)
'=============================================================================
DEFINT I-N
DEFDBL A-H, O-Z
SUB pvect (V1(), M1#(), V2())

'Calculo del Producto de M1 por V1
FOR I = 1 TO N
   V1(I) = 0
   FOR J = 1 TO N
      V1(I) = V1(I) + M1#(I, J) * V2(J)
   NEXT J
NEXT I

END SUB
```

SUBRUTINA salidau

```
'=============================================================================
' SUBRUTINA salidau
' Envía al equipo de adquisición de Datos la señal de salida u(t).
'=============================================================================
DEFINT I-N
DEFDBL A-H, O-Z
SUB salidau
'Ubicación del último valor enviado
CALL arlastp("salida%", lp!)
lp! = lp! + 1
IF lp! > dep! THEN
   lp! = 1
END IF

'Salida de Datos
U1! = U
CALL arputvalf("salida%", lp!, -1, "ANOUTO", U1!, 0)

END SUB
```

SUBRUTINA scdat

```
'=================================================================
' SUBRUTINA scdat
' Genera Menus y Cuadros para Identificación y Control en SIMULACION
' a) Menu de Control
'    Opciones:
'    1. Visualización y cambio de PARAMETROS
'    2. Visualización y cambio de VALORES INICIALES
'    3. Recuperación de Parámetros y Valores Iniciales Predefinidos
'    4. Ubicación de los POLOS para el Control
'    5. Inicio del proceso de IDENTIFICACION
'    6. Retorno al Menu de Simulación
' b) Cuadro de Parámetros
'    1. Orden del Modelo a Identificar
'    2. α = valor inicial matrix de covarianzas P
'    3. R = Referencia (0 ≤ R ≤ 10)
'    4. Umin = Valor mínimo permitido de la señal de control
'    5. Umax = Valor máximo permitido de la señal de control
'       (0 ≤ Umin ≤ Umax ≤ 10)
'    6. Numero de Datos a guardarse para la Matriz de Datos
'    7. Ruido (%) = Porcentaje de ruido respecto a Y
'       (0% ≤ Ruido ≤ 100%)
' c) Cuadro de Valores Iniciales
' d) Cuadro de Polos
'    Se ingresan los coeficientes del Denominador de la Función de
'    Transferencia en lazo cerrado.
'=================================================================
DEFINT I-N
DEFDBL A-H, O-Z
SUB scdat STATIC

'Definición de Arreglos para la Identificación y Control
'sid(): Nombres Menu Control
'scval(): Nombres Cuadro Valores Iniciales
'scpar(): Nombres Cuadro Parámetros
'spol(): Nombres Cuadro Polos
'dato$(): Datos Menu Control
'para$(): Datos Parámetros
'val$(): Datos Valores Iniciales
'pol$(): Datos Polos (No usado)
DIM sid(10) AS STRING
DIM scval(10) AS STRING
DIM scpar(10) AS STRING
DIM spol(10) AS STRING
DIM dato$(7)
DIM para$(8)
DIM val$(10)
DIM pol$(10)

'Cambio de Color
PALETTE 3, 25

'Definición del Menu de Control
sid(1) = "    SIMULACION: IDENTIFICACION Y CONTROL     "
```

```
sid(2) = " Parámetros"
sid(3) = " Valores Iniciales"
sid(4) = " Parámetros y valores predefinidos"
sid(5) = " Ubicación de polos"
sid(6) = " Iniciar control"
sid(7) = " Terminar"

'Definición del cuadro de Parámetros
scpar(1) = "          PARAMETROS          "
scpar(2) = " Orden = "
scpar(3) = " α = "
scpar(4) = " R = "
scpar(5) = " Umin = "
scpar(6) = " Umax = "
scpar(7) = " Número Datos = "
scpar(8) = " Ruido (%) = "

'Definición del cuadro de Parámetros
IF F = 0 THEN
   Nt = 2
   CALL datin(para$(), val$(), pol$())
   F = 1
END IF

'Definición del Cuadro de Polos
spol(1) = "         POLOS          "
FOR I = 2 TO 4
   spol(I) = " Polo" + LTRIM$(RTRIM$(STR$(I - 1))) + " ="
NEXT I

'Datos Predefinidos
N = 2 * VAL(para$(2))
NA = N / 2
NB = NA

'Definición Cuadro de Valores Iniciales
scval(1) = " VALORES INICIALES "
FOR K = 1 TO NA
   scval(K + 1) = " A(" + LTRIM$(RTRIM$(STR$(K))) + ") = "
NEXT K
FOR K = 1 TO NB
   scval(NA + K + 1) = " B(" + LTRIM$(RTRIM$(STR$(K))) + ") = "
NEXT K
CLS

'Menu de Control y Cuadros de Parámeteros, Valores Iniciales y Polos
DO

   'Accesible: Menu de Control
   'Solo visibles: Parámetros y Valores Iniciales
   J = menuda(5, 54, N + 1, scval(), 1, val$())
   J = menuda(14, 10, 8, scpar(), 1, para$())
   I = menuda(2, 1, 7, sid(), 2, dato$())
   SELECT CASE I
```

```
'Accesible: Cuadro de Parámetros
'Solo visibles: Menu de Control y Valores Iniciales
CASE 1
   J = menuda(2, 1, 7, sid(), 1, dato$())
   J = menuda(5, 54, N + 1, scval(), 1, val$())
   J = menuda(14, 10, 8, scpar(), 10, para$())

   'Definición nuevo Orden
   N = 2 * VAL(para$(2))
   NA = N / 2
   NB = NA

   'Recuperación Valores Iniciales
   FOR K = 1 TO NA
      scval(K + 1) = " A(" + LTRIM$(RTRIM$(STR$(K))) + ") = "
   NEXT K
   FOR K = 1 TO NB
      scval(NA + K + 1) = " B(" + LTRIM$(RTRIM$(STR$(K))) + ")
"
   NEXT K
   CLS

'Accesible: Cuadro de Valores Iniciales
'Solo visibles: Menu de Control y Parámetros
 CASE 2
   J = menuda(14, 10, 8, scpar(), 1, para$())
   J = menuda(2, 1, 7, sid(), 1, dato$())
   J = menuda(5, 54, N + 1, scval(), 20, val$())
   CLS

'Recuperación valores predefinidos
CASE 3

   'Recuperación parámetros predefinidos
   CALL datin(para$(), val$(), pol$())
   N = 2 * VAL(para$(2))
   NA = N / 2
   NB = NA

   'Recuperación Valores Iniciales predefinidos
   FOR K = 1 TO NA
      scval(K + 1) = " A(" + LTRIM$(RTRIM$(STR$(K))) + ") = "
   NEXT K
   FOR K = 1 TO NB
      scval(NA + K + 1) = " B(" + LTRIM$(RTRIM$(STR$(K))) + ")
"
   NEXT K
   CLS

'Accesible: Cuadro de Polos
'Solo Visibles: Menu de Control, Parámetros y Valores Iniciales
CASE 4

   'Calculo Grado del Polinomio de Polos
   IF NA > 2 THEN
```

```
            Nt = 3
         ELSE
            Nt = 2
         END IF

         'Presentación Menus y Cuadros
         J = menuda(2, 1, 7, sid(), 1, dato$())
         J = menuda(5, 54, N + 1, scval(), 1, val$())
         J = menuda(14, 10, 8, scpar(), 1, para$())
         J = menuda(18, 54, Nt + 1, spol(), 30, pol$())
         CLS

      'Inicio Identificación y Control
      CASE 5

         'Pasar a la Rutina de Identificacción y Control en SIMULACION
         CALL control(para$(), val$(), pol$())
         CLS

      CASE ELSE

   END SELECT

'Repetir hasta la opción Terminar
LOOP UNTIL I = 6

END SUB
```

SUBRUTINA sidat

```
'=================================================================
' SUBRUTINA sidat
' Genera los Menus y Cuadros para Identificación en SIMULACION
' a) Menu de Identificación
'     Opciones:
'     1. Visualización y cambio de PARAMETROS
'     2. Visualización y cambio de VALORES INICIALES
'     3. Recuperación de Parámetros y Valores Iniciales Predefinidos
'     4. Inicio del proceso de IDENTIFICACION
'     5. Retorno al Menu de Simulación
' b) Cuadro de Parámetros
'     1. Orden del Modelo a Identificar
'     2. α = valor inicial matrix de covarianzas P
'     3. R = Referencia (0 ≤ R ≤ 10)
'     4. Umin = Valor mínimo permitido de la señal de control
'     5. Umax = Valor máximo permitido de la señal de control
'        (0 ≤ Umin ≤ Umax ≤ 10)
'     6. Numero de Datos a guardarse para la Matriz de Datos
'     7. Ruido (%) = Porcentaje de ruido respecto a Y
'        (0% ≤ Ruido ≤ 100%)
' c) Cuadro de Valores Iniciales
'=================================================================
DEFINT I-N
DEFDBL A-H, O-Z
SUB sidat STATIC

'Definición de Arreglos para la Identificación
'sim(): Nombres Menu Identificación
'sval(): Nombres Cuadro Valores Iniciales
'spar(): Nombres Cuadro Parámetros
'dato$(): Datos Menu Identificación
'para$(): Datos Parámetros
'val$(): Datos Valores Iniciales
'pol$(): Datos Polos (No usado)
DIM sim(10) AS STRING
DIM sval(10) AS STRING
DIM spar(10) AS STRING
DIM dato$(6)
DIM para$(8)
DIM val$(10)
DIM pol$(10)

'Nuevo Color
PALETTE 3, 25

'Definición del Menu Identificación
sim(1) = "       SIMULACION:  IDENTIFICACION          "
sim(2) = " Parámetros"
sim(3) = " Valores iniciales"
sim(4) = " Parámetros y valores predefinidos"
sim(5) = " Iniciar identificación"
sim(6) = " Terminar"
```

```
'Definición del cuadro de Parámetros
spar(1) = "          PARAMETROS          "
spar(2) = " Orden = "
spar(3) = " α = "
spar(4) = " R = "
spar(5) = " Umin = "
spar(6) = " Umax = "
spar(7) = " Número Datos = "
spar(8) = " Ruido (%) = "

'Datos Predefinidos
IF F = 0 THEN
   Nt = 2
   CALL datin(para$(), val$(), pol$())
   F = 1
END IF
N = 2 * VAL(para$(2))
NA = N / 2
NB = NA

'Definición Cuadro de Valores Iniciales
sval(1) = " VALORES INICIALES "
FOR K = 1 TO NA
   sval(K + 1) = " A(" + LTRIM$(RTRIM$(STR$(K))) + ") = "
NEXT K
FOR K = 1 TO NB
   sval(NA + K + 1) = " B(" + LTRIM$(RTRIM$(STR$(K))) + ") = "
NEXT K

CLS

'Menu de Identificación y Cuadros de Parámateros y Valores Iniciales
DO

   'Accesible: Menu de Identificación
   'Solo visibles: Parámetros y Valores Iniciales
   J = menuda(5, 54, N + 1, sval(), 1, val$())
   J = menuda(14, 10, 8, spar(), 1, para$())
   I = menuda(2, 1, 6, sim(), 2, dato$())
   SELECT CASE I

      'Accesible: Cuadro de Parámetros
      'Solo visibles: Menu de Identificación y Valores Iniciales
      CASE 1
         J = menuda(2, 1, 6, sim(), 1, dato$())
         J = menuda(5, 54, N + 1, sval(), 1, val$())
         J = menuda(14, 10, 8, spar(), 10, para$())

         'Definición nuevo Orden
         N = 2 * VAL(para$(2))
         NA = N / 2
         NB = NA

         'Recuperación Valores Iniciales
         FOR K = 1 TO NA
```

```
            sval(K + 1) = " A(" + LTRIM$(RTRIM$(STR$(K))) + ") = "
         NEXT K
         FOR K = 1 TO NB
            sval(NA + K + 1) = " B(" + LTRIM$(RTRIM$(STR$(K))) + ") =
"
         NEXT K
         CLS

      'Accesible: Cuadro de Valores Iniciales
      'Solo visibles: Menu de Identificación y Parámetros
      CASE 2
         J = menuda(14, 10, 8, spar(), 1, para$())
         J = menuda(2, 1, 6, sim(), 1, dato$())
         J = menuda(5, 54, N + 1, sval(), 20, val$())
         CLS

      'Recuperación valores predefinidos
      CASE 3

         'Recuperación parámetros predefinidos
         CALL datin(para$(), val$(), pol$())
         N = 2 * VAL(para$(2))
         NA = N / 2
         NB = NA

         'Recuperación Valores Iniciales predefinidos
         FOR K = 1 TO NA
            sval(K + 1) = " A(" + LTRIM$(RTRIM$(STR$(K))) + ") = "
         NEXT K
         FOR K = 1 TO NB
            sval(NA + K + 1) = " B(" + LTRIM$(RTRIM$(STR$(K))) + ")
"
         NEXT K
         CLS

      'Inicio Identificación
      CASE 4

         'Pasar a la Rutina de Identificación en SIMULACION
         CALL ident(para$(), val$())

         CLS

      CASE ELSE

   END SELECT

'Repetir hasta la opción Terminar
LOOP UNTIL I = 5

END SUB
```

```
SUBRUTINA simula

'=================================================================
' SUBRUTINA simula
' Permite la entrada a las rutinas de SIMULACION
' Genera el Menu de Simulación y las opciones posibles en SIMULACION
' Menu de Simulación
'   Opciones:
'   1. Definición del MODELO (Esta Rutina debe ejecutarse al inicio)
'   2. IDENTIFICACION de la planta predefinida
'   3. IDENTIFICACION Y CONTROL de la planta predefinida
'   4. Transferencia de la MATRIZ DE DATOS a un archivo en el disco
'   5. Regreso al Menu Principal
'=================================================================
DEFINT I-N
DEFDBL A-H, O-Z
SUB simula

'Definición del Menu de Simulación
DIM msim(6) AS STRING
msim(1) = "   S I M U L A C I O N   "
msim(2) = " Modelo"
msim(3) = " Identificación"
msim(4) = " Identificación y Control"
msim(5) = " Matriz de Datos"
msim(6) = " Terminar"

's1=0: Modelo no definido todavía; opciones posibles 1 y 5
's1=1: Modelo ya definido; opciones posibles 1, 2, 3 y 5
's1=3: Proceso ya ejecutado; todas las opciones son posibles

'Modelo no definido todavía
s1! = 0

'Menu de Simulación
DO
   CLS
   I = menu(8, 17, 6, msim(), s1!)
   SELECT CASE I
   CASE 1
      'Definir Modelo
      CALL modelo

      'Modelo ya definido
      s1! = 1

   CASE 2
      'Identificación
      CALL sidat

      'Proceso ejecutado
      s1! = 3

   CASE 3
      'Identificación y Control
```

```
        CALL scdat

        'Proceso ejecutado
        s1! = 3

    CASE 4
        'Matriz de Datos
        CALL datos

    CASE ELSE

    END SELECT

'Repetir hasta que se escoja la opción 4
LOOP UNTIL I = 5

'Modelo no definido (Regreso al Menu Principal)
s1! = 0


END SUB
```

SUBRUTINA sist

```
'=============================================================================
' SUBRUTINA sist
' Resuelve es Sistema De ecuaciones Dado por la Matriz A.
' Parámetros de Entrada:
'   A(): Matriz ampliada con los coeficientes del sistema de
'        ecuaciones
'   N: Orden
' Parámetros de salida:
'   V1(): Vector de soluciones
''  ERS%(): Error en el sistema de ecuaciones
'=============================================================================
DEFINT I-N
DEFDBL A-H, O-Z
SUB sist (A(), V1(), N, K)

'Definición de los Vectores y Matrices Auxiliares
DIM s(N), NFILA(N), W(N, N)

'Algoritmo de resolución
FOR I = 1 TO N
   VALMAX = ABS(A(I, 1))
   FOR J = 2 TO N
      IF VALMAX < ABS(A(I, J)) THEN
         VALMAX = ABS(A(I, J))
      END IF
   NEXT J
   s(I) = VALMAX
   IF s(I) = 0 THEN
      K = 1
      EXIT SUB
   END IF
   NFILA(I) = I
NEXT I
FOR I = 1 TO N - 1
   P = I
   VALMAX = ABS(A(NFILA(I), I)) / s(NFILA(I))
   FOR J = I TO N
      IF VALMAX < ABS(A(NFILA(J), I)) / s(NFILA(J)) THEN
         VALMAX = ABS(A(NFILA(J), I)) / s(NFILA(J))
         P = J
      END IF
   NEXT J
   IF A(NFILA(P), I) = 0 THEN
      K = 1
      EXIT SUB
   END IF
   IF NFILA(P) <> NFILA(I) THEN
      SWAP NFILA(P), NFILA(I)
   END IF
   FOR J = I + 1 TO N
      W(NFILA(J), I) = A(NFILA(J), I) / A(NFILA(I), I)
      FOR K = 1 TO N + 1
         A(NFILA(J), K) = A(NFILA(J), K) - W(NFILA(J), I) * A(NFILA(I)
```

```
K)
      NEXT K
   NEXT J
NEXT I
IF A(NFILA(N), N) = 0 THEN
   K = 1
   EXIT SUB
END IF
V1(N) = A(NFILA(N), N + 1) / A(NFILA(N), N)
FOR I = N - 1 TO 1 STEP -1
  . V1(I) = 0
   FOR J = I + 1 TO N
      V1(I) = V1(I) + A(NFILA(I), J) * V1(J)
   NEXT J
   V1(I) = (A(NFILA(I), N + 1) - V1(I)) / A(NFILA(I), I)
NEXT I

END SUB
```

SUBRUTINA traery

```
'=============================================================================
' SUBRUTINA traery
' Recupera el último valor de Y leído por el sistema de adquisición
' de datos
'=============================================================================
DEFINT I-N
DEFDBL A-H, O-Z
SUB traery

'Llamar al primer valor
DO
   CALL arlastp("entraday%", lp!)
LOOP WHILE lp! = 0

'Recuperar el último valor de Y
CALL argetvalf("entraday%", lp!, -1, "ANLGO", valor!, 0)
Y = valor!

END SUB
```

SUBRUTINA trcdat

```
'=================================================================
' SUBRUTINA trcdat
' Genera Menus y Cuadros para Identificación y Control en TIEMPO REAL
' a) Menu de Control
'    Opciones:
'    1. Visualización y cambio de PARAMETROS
'    2. Visualización y cambio de VALORES INICIALES
'    3. Recuperación de Parámetros y Valores Iniciales Predefinidos
'    4. Ubicación de los POLOS para el Control
'    5. Inicio del proceso de IDENTIFICACION
'    6. Retorno al Menu de Tiempo Real
' b) Cuadro de Parámetros
'    1. Orden del Modelo a Identificar
'    2. α = valor inicial matrix de covarianzas P
'    3. R = Referencia (0 ≤ R ≤ 10)
'    4. Umin = Valor mínimo permitido de la señal de control
'    5. Umax = Valor máximo permitido de la señal de control
'       (0 ≤ Umin ≤ Umax ≤ 10)
'    6. Numero de Datos a guardarse para la Matriz de Datos
' c) Cuadro de Valores Iniciales
' d) Cuadro de Polos
'    Se ingresan los coeficientes del Denominador de la Función de
'    Transferencia en lazo cerrado.
'=================================================================
DEFINT I-N
DEFDBL A-H, O-Z
SUB trcdat STATIC

'Definición de Arreglos para la Identificación y Control
'trcont(): Nombres Menu de Control
'trcval(): Nombres Cuadro Valores Iniciales
'trcpar(): Nombres Cuadro Parámetros
'trcpol(): Nombre Cuadro Polos
'dato$(): Datos Menu de Control
'para$(): Datos Parámetros
'val$(): Datos Valores Iniciales
'pol$(): Datos Polos (No usado)

DIM trcont(10) AS STRING
DIM trcval(10) AS STRING
DIM trcpar(10) AS STRING
DIM trpol(10) AS STRING
DIM dato$(7)
DIM para$(8)
DIM val$(10)
DIM pol$(10)

'Cambio de Color
PALETTE 3, 25

'Definición del Menu de Control
trcont(1) = "   TIEMPO REAL: IDENTIFICACION Y CONTROL    "
trcont(2) = " Parámetros"
```

```
trcont(3) = " Valores Iniciales"
trcont(4) = " Parámetros y valores predefinidos"
trcont(5) = " Ubicación de polos"
trcont(6) = " Iniciar control"
trcont(7) = " Terminar"

'Definición del cuadro de Parámetros
trcpar(1) = "          PARAMETROS          "
trcpar(2) = " Orden = "
trcpar(3) = " α = "
trcpar(4) = " R = "
trcpar(5) = " Umin = "
trcpar(6) = " Umax = "
trcpar(7) = " Número Datos = "

'Definición del cuadro de Parámetros
IF F = 0 THEN
   Nt = 2
   CALL datin(para$(), val$(), pol$())
   F = 1
END IF

'Definición del Cuadro de Polos
trpol(1) = "         POLOS          "
FOR I = 2 TO 4
   trpcl(I) = " Polo" + LTRIM$(RTRIM$(STR$(I - 1))) + " ="
NEXT I

'Datos Predefinidos
N = 2 * VAL(para$(2))
NA = N / 2
NB = NA

'Definición Cuadro de Valores Iniciales
trcval(1) = " VALORES INICIALES "
FOR K = 1 TO NA
   trcval(K + 1) = " A(" + LTRIM$(RTRIM$(STR$(K))) + ") = "
NEXT K
FOR K = 1 TO NB
   trcval(NA + K + 1) = " B(" + LTRIM$(RTRIM$(STR$(K))) + ") = "
NEXT K
CLS

'Menu de Control y Cuadros de Parámateros, Valores Iniciales y Polos
DO

   'Accesible: Menu de Control
   'Solo visibles: Parámetros y Valores Iniciales
   J = menuda(5, 54, N + 1, trcval(), 1, val$())
   J = menuda(14, 10, 7, trcpar(), 1, para$())
   I = menuda(2, 1, 7, trcont(), 2, dato$())
   SELECT CASE I

      'Accesible: Cuadro de Parámetros
      'Solo visibles: Menu de Control y Valores Iniciales
```

```
CASE 1
   J = menuda(2, 1, 7, trcont(), 1, dato$())
   J = menuda(5, 54, N + 1, trcval(), 1, val$())
   J = menuda(14, 10, 7, trcpar(), 10, para$())

   'Definición nuevo Orden
   N = 2 * VAL(para$(2))
   NA = N / 2
   NB = NA

   'Recuperación Valores Iniciales
   FOR K = 1 TO NA
      trcval(K + 1) = " A(" + LTRIM$(RTRIM$(STR$(K))) + ") = "
   NEXT K
   FOR K = 1 TO NB
      trcval(NA + K + 1) = " B(" + LTRIM$(RTRIM$(STR$(K))) + ")
= "
   NEXT K
   CLS

'Accesible: Cuadro de Valores Iniciales
'Solo visibles: Menu de Control y Parámetros
CASE 2
   J = menuda(14, 10, 7, trcpar(), 1, para$())
   J = menuda(2, 1, 7, trcont(), 1, dato$())
   J = menuda(5, 54, N + 1, trcval(), 20, val$())
   CLS

'Recuperación valores predefinidos
CASE 3

   'Recuperación parámetros predefinidos
   CALL datin(para$(), val$(), pol$())
   N = 2 * VAL(para$(2))
   NA = N / 2
   NB = NA

   'Recuperación Valores Iniciales predefinidos
   FOR K = 1 TO NA
      trcval(K + 1) = " A(" + LTRIM$(RTRIM$(STR$(K))) + ") = "
   NEXT K
   FOR K = 1 TO NB
      trcval(NA + K + 1) = " B(" + LTRIM$(RTRIM$(STR$(K))) + "
= "
   NEXT K
   CLS

'Accesible: Cuadro de Polos
'Solo Visibles: Menu de Control, Parámetros y Valores Iniciales
CASE 4

   'Calculo Grado del Polinomio de Polos
   IF NA > 2 THEN
      Nt = 3
   ELSE
```

```
            Nt = 2
         END IF

         'Presentación Menus y Cuadros
         J = menuda(2, 1, 7, trcont(), 1, dato$())
         J = menuda(5, 54, N + 1, trcval(), 1, val$())
         J = menuda(14, 10, 7, trcpar(), 1, para$())
         J = menuda(18, 54, Nt + 1, trpol(), 30, pol$())
         CLS

   'Inicio Identificación y Control
   CASE 5

      'Pasar a la Rutina de Identificacción y Control en TIEMPO REAL
      CALL trcontrol(para$(), val$(), pol$())
      CLS

   CASE ELSE

END SELECT

'Repetir hasta la opción Terminar
LOOP UNTIL I = 6

END SUB
```

```
SUBRUTINA trcontrol

'=============================================================================
' SUBRUTINA trcontrol
' Ejecuta el proceso de IDENTIFICACION y CONTROL en TIEMPO REAL en
' base a los parámetros y valores iniciales definidos.
' Produce los gráficos en pantalla de la señal de entrada ( u(t) )
' y de salida ( y(t) ) y calcula los parámetros de la planta.
' Parámetros de Entrada:
'    para$(): Parámetros definidos en "trcdat"
'    val$(): Valores Iniciales definidos en "trcdat"
'    pol$(): Polos definidos en "trcdat"
'=============================================================================
DEFINT I-N
DEFDBL A-H, O-Z
SUB trcontrol (para$(), val$(), pol$())

'Recuperación de Parámetros
'Par: Valor inicial de α
'R: Referencia
'Umin: valor mínimo permitido de U
'Umax: vàlor máximo permitido de U
'MaxDat: Número máximo de Datos para la Matriz de Datos
Par = VAL(para$(3))
R = VAL(para$(4))
Umin = VAL(para$(5))
Umax = VAL(para$(6))
MaxDat = VAL(para$(7))

'Calculo del grado del polonomio de Polos
IF NA > 2 THEN
   Nt = 3
ELSE
   Nt = 2
END IF

'Dimensionamiento de Matrices y Vectores
'P(): Matriz de Covarianzas
'MA(), MB(): Matrices auxiliares de cálculo
'T(): Vector de Parámetros (modelo ARMA)
'X(): Vector de Datos Entrada-Salida (modelo ARMA)
'V(): Vector auxiliar de cálculo
's(): Matrix de Control
'a(): Vector de Parámetros a para Control
'pol(): Vector de coeficientes de polos
'TM(): Matriz con las últimos 50 cálculos de parámetros
'TF(): Vector con los Parámetros Promedio
'DAT(): Matriz con los Parámetros y las señales de entrada y salida
'C(): Vector con los últimos 20 valores de U
'unit(): Vector con sus valores en 1
REDIM tim(0 TO 7) AS INTEGER, DT(0 TO 7)
REDIM P(N, N)
REDIM MA(N, N) AS DOUBLE, MB(N, N) AS DOUBLE
REDIM T(N), X(N), V(N), s(N, N + 1), A(NA + 1), pol(N)
REDIM TM(0 TO 49, N), TF(N)
```

```
REDIM DAT(MaxDat, N + 2)
REDIM C(dep!), unit(dep!)

'Inicialización de T
FOR I = 2 TO N + 1
   T(I - 1) = VAL(val$(I))
NEXT I

'Inicialización de "pol"
FOR I = 2 TO Nt + 1
   pol(I - 1) = VAL(pol$(I))
NEXT I

'Inicialización de P, TF y X
FOR I = 1 TO N
   FOR J = 1 TO N
      s(I, J) = 0
      IF I = J THEN
         P(I, J) = Par
      ELSE
         P(I, J) = 0
      END IF
   NEXT J
   X(I) = 0
   TF(I) = 0
NEXT I

'Definición Arreglos de Entrada y Salida de Datos
CALL anin("entraday%", dep!, "ANLG0", bintv%, -1, "nt", "tarea1")
CALL armake("salida%", dep!, -1, "ANOUTO")
CALL anout("salida%", "ANOUTO", bintv%, -1, "nt", "tarea3")

'Inicialización Arreglo de Salida de datos, Vectores C y unit
FOR I! = 1 TO dep!
    CALL arputvalf("salida%", I!, -1, "ANOUTO", 0!, 0)
    unit(I!) = 1
    C(I!) = 0
NEXT I!

'Inicialización de Variables y Constantes
'K: cuenta el número de iteraciones para graficación
'DK: cuenta el número total de iteraciones
'U: señal de entrada a la planta
'F: Factor de olvido
'Y: señal de salida de la planta
'NF: Falla del Control (NF=0: no falla; NF=1: falla)
K = 0
DK = 0
F = .98
Y = 0
U = 0
NF = 0
DAT(DK, 1) = Y
DAT(DK, 2) = U
RANDOMIZE TIMER
```

```
'Definición de Pantalla y Colores para graficación
SCREEN 9
WIDTH 80, 43

'Iniciar Identificación
CLS
LOCATE 24, 30
PRINT "PRESIONE UNA TECLA PARA INICIAR CONTROL"
DO
    q$ = INKEY$
LOOP WHILE q$ = ""

'Impresión en pantalla de los gráficos y mensajes iniciales
CLS
CALL defgraf
LOCATE 21, 30
PRINT "TIEMPO REAL"
COLOR 2
LOCATE 42, 10
PRINT "CONTROL EN PROCESO";
COLOR 15
LOCATE 42, 40
PRINT "Presione una Tecla para terminar...";

'Inicialización Timer
'tim(0)=1: conteo con timer 0
'ndf: detección cuenta final del timer
'DT: Conteo del número de muestras
tim(0) = 1
DT = 1
ndf = -1
CALL timerstart(tim(), "nt", "timer")

'Iniciar interrupciones
CALL inton(1, "mil")

'Inicio del lazo de Identificación
DO

   'Rutina de Actualización de Datos Entrada-Salida
   FOR I = NA TO 2 STEP -1
      X(I) = X(I - 1)
   NEXT I
   X(1) = -Y
   FOR I = N TO N - NB + 2 STEP -1
      X(I) = X(I - 1)
   NEXT I
   X(NA + 1) = U

   'Recuperar valor de Y
   CALL traery

   'Inicio Algoritmo de Mínimos Cuadrados Recursivo
   CALL pvect(V(), P(), X())
   A = F + pesc(X(), V())
```

```
FOR I = 1 TO N
   V(I) = V(I) / A
NEXT I
E = Y - pesc(X(), T())

'Cálculo Nuevos Parámetros
FOR I = 1 TO N
   T(I) = T(I) + V(I) * E
NEXT I

'Ingreso de valores a la Matriz de Datos
IF DK < MaxDat THEN
   DAT(DK + 1, 1) = Y
   DAT(DK + 1, 2) = U
   FOR I = 3 TO N + 2
      DAT(DK + 1, I) = T(I - 2)
   NEXT I
END IF

'Ingreso de valores a la Matriz de Parámetros
FOR I = 3 TO N + 2
   TM(DK MOD 50, I - 2) = T(I - 2)
NEXT I

'Cálculo de Matriz de Covarianzas P.
FOR I = 1 TO N
   FOR J = 1 TO N
      IF I = J THEN
         MA(I, J) = 1 - V(I) * X(J)
      ELSE
         MA(I, J) = -V(I) * X(J)
      END IF
   NEXT J
NEXT I
CALL pmat(MB(), MA(), P())
FOR I = 1 TO N
   FOR J = 1 TO N
      P(I, J) = MB(I, J) / F
   NEXT J
NEXT I

'Cálculo nuevo vector de Parámetros a para Control
A(1) = T(1) - 1
FOR I = 2 TO NA
   A(I) = T(I) - T(I - 1)
NEXT I
A(NA + 1) = -T(NA)

'Definición Matriz s para orden mayor a 2
IF N > 2 THEN
   FOR J = 1 TO N + 1
      FOR I = 1 TO N
         IF J < NB THEN
            IF I < J THEN
               s(I, J) = 0
```

```
            ELSEIF I = J THEN
               s(I, J) = 1
            ELSEIF I <= NA + 1 + J THEN
               s(I, J) = A(I - J)
            ELSE
               s(I, J) = 0
            END IF
         ELSEIF J < N + 1 THEN
            IF I < 1 - NB + J THEN
               s(I, J) = 0
            ELSEIF I <= J THEN
               s(I, J) = T(NA + NB + I - J)
            ELSE
               s(I, J) = 0
            END IF
         ELSE
            IF I <= Nt AND I <= NA + 1 THEN
               s(I, N + 1) = pol(I) - A(I)
            ELSEIF I <= Nt AND I > NA + 1 THEN
               s(I, N + 1) = pol(I)
            ELSEIF I > Nt AND I <= NA + 1 THEN
               s(I, N + 1) = -A(I)
            ELSE
               s(I, N + 1) = 0
            END IF
         END IF
      NEXT I
   NEXT J

   'Resolución Sistema de Ecuaciones para orden mayor a 2
   CALL sist(s(), V(), N, ERS%)

   'Error en sistema de ecuaciones
   IF ERS% = 1 THEN

      'Suspender el Control
      CALL intoff
      CALL ardel("entraday%")
      CALL ardel("salida%")

      'Impresión mensajes de error
      CALL ersist

      'Redefinición Pantalla y colores
      SCREEN 0
      WIDTH 80, 25
      COLOR blanco, negro
      PALETTE 3, 25

      'Terminar: volver al Menu de Control
      EXIT SUB

   END IF

'Calculo del control para un orden del modelo igual a 1
```

```
ELSE

    'Grado del polinomio de polos es 1
    IF Nt = 1 THEN
       V(1) = pol(1) - A(1) / T(2)
       V(2) = -A(2) / T(2)

    'Grado del polinomio de polos es 2
    ELSE
       V(1) = pol(1) - A(1) / T(2)
       V(2) = pol(2) - A(2) / T(2)

    END IF

END IF

'Calculo de la señal de control u
CALL calcu(X(), V())

'U debe estar entre Umin y Umax
IF U > Umax THEN
   U = Umax
ELSEIF U < Umin THEN
   U = Umin
END IF

'Falla del Control: enviar señal de recuperación
IF (pesc(C(), unit()) / dep!) = Umin THEN
   U = Umax
END IF

'Actualización Vector de Salida
C(DK MOD dep!) = U

'Sacar valor de U
CALL salidau

'Nuevo Gráfico si más de 580 valores
IF K > 580 THEN

    'Nuevo Gráfico
    CALL defgraf

    'Inicialización de K
    K = 0

END IF

'Graficar Nuevo Punto
CALL graf(K)

'Verificación de Interrupciones
DO

    'Recuperar valor del Timer
```

```
        CALL timerread(DT())

        'Muestreo siguiente detectado
        IF DT(0) / bintv% > DT THEN
           ndf = 1

        'Final del Conteo Detectado
        ELSEIF DT(0) / bintv% < -3 THEN
           ndf = 0

        END IF

    'Repetir hasta que se dé el siguiente muestreo
    LOOP UNTIL ndf >= 0

    'Actualizar Conteo de Muestras
    DT = ndf * INT(DT(0) / bintv%) + 1
    ndf = -1

    'Incrementar contadores
    K = K + 1
    DK = DK + 1

    'Falla del Control
    IF DK + 1 <> DT AND NF = 0 THEN
       LOCATE 42, 3
       BEEP
       NF = 1
       COLOR rojo
       PRINT "FALLA DEL CONTROL" + SPACE$(12);
       COLOR blancoi

    END IF

'Repetir el lazo hasta la detección de una tecla
LOOP WHILE INKEY$ = ""

'Suspender proceso en Tiempo Real
CALL intoff
CALL ardel("entraday%")
CALL ardel("salida%")

'Fin de Identificación
LOCATE 42, 2
PRINT SPACE$(8); "Presione una Tecla para continuar..."; SPACE$(29);
DO
LOOP WHILE INKEY$ = ""

'Cálculo de Parámetros Promedio
FOR I = 1 TO N
    IF DK > 50 THEN
       FOR J = 0 TO 49
          TF(I) = TF(I) + TM(J, I)
       NEXT J
       TF(I) = TF(I) / 50
```

```
   ELSE
      TF(I) = T(I)
   END IF
NEXT I

'Definición de Pantalla y Colores para Menús y Cuadros
SCREEN 0
WIDTH 80, 25
COLOR blanco, negro
PALETTE 3, 25

'************************************************
' Menu Final.
' Opciones:
'    1. Imprimir Resultados en Papel
'    2. Regresar a Menu de Identificación
'************************************************
'Borrar Pantalla
CLS

'Definición de Arreglos para la impresión de Resultados
'datim(): Nombres Cuadro Resultados
'cimpr(): Nombres Menu Final
'cdat$(): Datos Cuadro Resultados
'cim$(): Datos Menu Final
DIM datim(20) AS STRING
DIM cimpr(10) AS STRING
DIM cdat$(20)
DIM cim$(10)

'Definición Cuadro de Resultados
datim(1) = "          RESULTADOS          "

'Nombre de Parámetros Calculados
FOR J = 1 TO NA
   datim(J + 1) = " A(" + LTRIM$(RTRIM$(STR$(J))) + ") = "
NEXT J
FOR J = 1 TO NB
   datim(NA + J + 1) = " B(" + LTRIM$(RTRIM$(STR$(J))) + ") = "
NEXT J

'Iteraciones, Interrupciones
datim(N + 2) = " Iteraciones = "
datim(N + 3) = " Interrupciones = "

'Conversión de Datos a Caracteres
FOR J = 1 TO N
   cdat$(J + 1) = convert$(TF(J)) 'LEFT$(LTRIM$(RTRIM$(STR$(TF(J)))),
8)
NEXT J
cdat$(N + 2) = convert$(DK)
cdat$(N + 3) = convert$(DT - 1)

'Definición Menu Final
cimpr(1) = "         FIN DEL CONTROL          "
```

```
cimpr(2) = " Imprimir resultados"
cimpr(3) = " Terminar "

'Menu Final y Cuadro de Resultados
DO

    'Accesible: Menú Final
    'Solo visible: Cuadro de Resultados
    ERF% = 0
    J = menuda(2, 2, N + 3, datim(), 1, cdat$())
    I = menuda(5, 41, 3, cimpr(), 2, cim$())
    SELECT CASE I

        'Impresión de Resultados en Papel
        CASE 1

            'Borrar zona de mensajes de Error
            clerr

            'Imprimir
            CALL printer(para$(), TF(), pol(), DK, 4)

    END SELECT

'Repetir hasta la obtener Opción Terminar
LOOP UNTIL I = 2

END SUB
```

SUBRUTINA treal

```
'=============================================================================
' SUBRUTINA treal
' Permite la entrada a las rutinas en TIEMPO REAL
' Genera el Menu de Tiempo Real y las opciones posibles en TIEMPO
' REAL.
' Menu de Tiempo real
'   Opciones:
'   1. Definición de PARAMETROS para la adquisición de datos (se
'      ejecuta al inicio)
'   2. IDENTIFICACION de la planta
'   3. IDENTIFICACION Y CONTROL de la planta
'   4. INICIALIZACION del equipo de adquisición de datos
'   5. Transferencia de la MATRIZ de DATOS a un archivo en disco
'   6. Retornar al Menu Principal
'=============================================================================
DEFINT I-N
DEFDBL A-H, O-Z
SUB treal

'Inicialización de Rutinas para el equipo de adquisición de datos
CALL softinit
IF ERNUM% <> 0 THEN
   PRINT ERNUM%;
END IF
CALL init.

'Definición Menu de Tiempo Real
DIM mtreal(10) AS STRING
mtreal(1) = "   T I E M P O    R E A L   "
mtreal(2) = " Parámetros"
mtreal(3) = " Identificación"
mtreal(4) = " Identificación y Control"
mtreal(5) = " Inicialización"
mtreal(6) = " Matriz de datos"
mtreal(7) = " Terminar"

'TR!=0: Parámetros no definidos todavía: opciones posibles 1 y 6
'TR!=2: Parámetros definidos: opciones posibles 1, 2, 3, 4 y 6
'TR!=3: Proceso ya ejecutado: todas las opciones son posibles

'Parámetros no definidos
TR! = 0

'Menu de Tiempo Real
DO
   CLS
   I = menu(8, 17, 7, mtreal(), TR!)
   SELECT CASE I
   CASE 1
      'Definición de Parámetros
      CALL trparam

      'Parámetros definidos
```

```
        TR! = 2

    CASE 2
        'Identificación
        CALL tridat

        'Proceso ejecutado
        TR! = 3

    CASE 3
        'Identificación y Control
        CALL trcdat

        'Proceso ejecutado
        TR! = 3

    CASE 4
        'Inicialización equipo de adquisición de datos
        CALL init

    CASE 5
       'Matriz de Datos
       CALL datos

    CASE ELSE
    END SELECT
LOOP UNTIL I = 6

'Parámetros no definidos (Retorno al Menu Principal)
TR! = 0

END SUB
```

SUBRUTINA tridat

```
'=============================================================================
' SUBRUTINA tridat
' Genera los Menus y Cuadros para Identificación en TIEMPO REAL
' a) Menu de Identificación
'     Opciones:
'     1. Visualización y cambio de PARAMETROS
'     2. Visualización y cambio de VALORES INICIALES
'     3. Recuperación de Parámetros y Valores Iniciales Predefinidos
'     4. Inicio del proceso de IDENTIFICACION
'     5. Retorno al Menu de Tiempo Real
' b) Cuadro de Parámetros
'     1. Orden del Modelo a Identificar
'     2. α = valor inicial matrix de covarianzas P
'     3. R = Referencia (0 ≤ R ≤ 10)
'     4. Umin = Valor mínimo permitido de la señal de control
'     5. Umax = Valor máximo permitido de la señal de control
'        (0 ≤ Umin ≤ Umax ≤ 10)
'     6. Numero de Datos a guardarse para la Matriz de Datos
'     7. Ruido (%) = Porcentaje de ruido respecto a Y
'        (0% ≤ Ruido ≤ 100%)
' c) Cuadro de Valores Iniciales
'=============================================================================
DEFINT I-N
DEFDBL A-H, O-Z
SUB tridat STATIC

'Definición de Arreglos para la Identificación
'trid(): Nombres Menu Identificación
'trival(): Nombres Cuadro Valores Iniciales
'tripar(): Nombres Cuadro Parámetros
'dato$(): Datos Menu Identificación
'para$(): Datos Parámetros
'val$(): Datos Valores Iniciales
'pol$(): Datos Polos (No usado)

DIM trid(10) AS STRING
DIM trival(10) AS STRING
DIM tripar(10) AS STRING
DIM dato$(6)
DIM para$(8)
DIM val$(10)
DIM pol$(10)

'Nuevo Color
PALETTE 3, 25

'Definición del Menu Identificación
trid(1) = "        TIEMPO REAL:  IDENTIFICACION          "
trid(2) = " Parámetros"
trid(3) = " Valores iniciales"
trid(4) = " Parámetros y valores predefinidos"
trid(5) = " Iniciar identificación"
trid(6) = " Terminar"
```

```
'Definición del cuadro de Parámetros
tripar(1) = "          PARAMETROS          "
tripar(2) = " Orden = "
tripar(3) = " α = "
tripar(4) = " R = "
tripar(5) = " Umin = "
tripar(6) = " Umax = "
tripar(7) = " Número Datos = "
tripar(8) = " Ruido (%) = "

'Datos Predefinidos
IF F = 0 THEN
   Nt = 2
   CALL datin(para$(), val$(), pol$())
   F = 1
END IF
N = 2 * VAL(para$(2))
NA = N / 2
NB = NA

'Definición Cuadro de Valores Iniciales
trival(1) = " VALORES INICIALES "
FOR K = 1 TO NA
   trival(K + 1) = " A(" + LTRIM$(RTRIM$(STR$(K))) + ") = "
NEXT K
FOR K = 1 TO NB
   trival(NA + K + 1) = " B(" + LTRIM$(RTRIM$(STR$(K))) + ") = "
NEXT K

CLS

'Menu de Identificación y Cuadros de Parámateros y Valores Iniciales
DO

   'Accesible: Menu de Identificación
   'Solo visibles: Parámetros y Valores Iniciales
   J = menuda(5, 54, N + 1, trival(), 1, val$())
   J = menuda(14, 10, 8, tripar(), 1, para$())
   I = menuda(2, 1, 6, trid(), 2, dato$())
   SELECT CASE I

      'Accesible: Cuadro de Parámetros
      'Solo visibles: Menu de Identificación y Valores Iniciales
      CASE 1
         J = menuda(2, 1, 6, trid(), 1, dato$())
         J = menuda(5, 54, N + 1, trival(), 1, val$())
         J = menuda(14, 10, 8, tripar(), 10, para$())

         'Definición nuevo Orden
         N = 2 * VAL(para$(2))
         NA = N / 2
         NB = NA

         'Recuperación Valores Iniciales
         FOR K = 1 TO NA
```

```
            trival(K + 1) = " A(" + LTRIM$(RTRIM$(STR$(K))) + ") = "
         NEXT K
         FOR K = 1 TO NB
            trival(NA + K + 1) = " B(" + LTRIM$(RTRIM$(STR$(K))) + ")
= "
         NEXT K
         CLS

      'Accesible: Cuadro de Valores Iniciales
      'Solo visibles: Menu de Identificación y Parámetros
      CASE 2
         J = menuda(14, 10, 8, tripar(), 1, para$())
         J = menuda(2, 1, 6, trid(), 1, dato$())
         J = menuda(5, 54, N + 1, trival(), 20, val$())
         CLS

      'Recuperación valores predefinidos
      CASE 3

         'Recuperación parámetros predefinidos
         CALL datin(para$(), val$(), pol$())
         N = 2 * VAL(para$(2))
         NA = N / 2
         NB = NA

         'Recuperación Valores Iniciales predefinidos
         FOR K = 1 TO NA
            trival(K + 1) = " A(" + LTRIM$(RTRIM$(STR$(K))) + ") = "
         NEXT K
         FOR K = 1 TO NB
            trival(NA + K + 1) = " B(" + LTRIM$(RTRIM$(STR$(K))) + ")
= "
         NEXT K
         CLS

      'Inicio Identificación
      CASE 4

         'Pasar a la Rutina de Identificación en SIMULACION
         CALL trident(para$(), val$())
         CLS


      CASE ELSE

   END SELECT

'Repetir hasta la opción Terminar
LOOP UNTIL I = 5


END SUB
```

SUBRUTINA trident

```
'=============================================================================
' SUBRUTINA trident
' Ejecuta el proceso de IDENTIFICACION en TIEMPO REAL en base a los
' parámetros y valores iniciales definidos.
' Produce los gráficos en pantalla de la señal de entrada ( u(t) ) y
' de salida ( y(t) ) y calcula los parámetros de la planta.
' Parámetros de Entrada:
'   para$(): Parámetros definidos en "sidat"
'   val$(): Valores Iniciales definidos en "sidat"
'=============================================================================
DEFINT I-N
DEFDBL A-H, O-Z
SUB trident (para$(), val$())

'Recuperación de Parámetros
'Par: Valor inicial de α
'R: Referencia
'Umin: valor mínimo permitido de U
'Umax: valor máximo permitido de U
'MaxDat: Número máximo de Datos para la Matriz de Datos
'ERP: Ruido Porcentual
Par = VAL(para$(3))
R = VAL(para$(4))
Umin = VAL(para$(5))
Umax = VAL(para$(6))
MaxDat = VAL(para$(7))
ERP = VAL(para$(8))

'Dimensionamiento de Matrices y Vectores
'tim(): Timer
'DT(): Conteo de Timer
'P(): Matriz de Covarianzas
'MA(), MB(): Matrices auxiliares de cálculo
'T(): Vector de Parámetros (modelo ARMA)
'X(): Vector de Datos Entrada-Salida (modelo ARMA)
'V(): Vector auxiliar de cálculo
'TM(): Matriz con las últimos 50 cálculos de parámetros
'TF(): Vector con los Parámetros Promedio
'DAT(): Matriz con los Parámetros y las señales de entrada y salida
REDIM tim(0 TO 7) AS INTEGER, DT(0 TO 7)
REDIM P(N, N)
REDIM MA(N, N) AS DOUBLE, MB(N, N) AS DOUBLE
REDIM T(N), X(N), V(N), TM(0 TO 49, N), TF(N)
REDIM DAT(MaxDat, N + 2)

'Inicialización de P, TF y X
FOR I = 1 TO N
   FOR J = 1 TO N
      IF I = J THEN
         P(I, J) = Par
      ELSE
         P(I, J) = 0
      END IF
```

```
    NEXT J
    X(I) = 0
    TF(I) = 0
NEXT I

'Inicialización de T
FOR I = 2 TO N + 1
    T(I - 1) = VAL(val$(I))
NEXT I

'Inicialización de Variables y Constantes
'K: cuenta el número de iteraciones para graficación
'DK: cuenta el número total de iteraciones
'U: señal de entrada a la planta
'F: Factor de olvido
'Y: señal de salida de la planta
'NF: Falla del Control (NF=0: no falla; NF=1: falla)
K = 0
DK = 0
U = R
F = .98
Y = 0
NF = 0
DAT(DK, 1) = U
DAT(DK, 2) = Y
RANDOMIZE TIMER

'Definición de Pantalla y Colores para graficación
SCREEN 9
WIDTH 80, 43
CLS

'Definición Arreglos de Entrada y Salida de Datos
CALL armake("salida%", dep!, -1, "ANOUTO")
CALL anout("salida%", "ANOUTO", bintv%, -1, "nt", "tarea3")
CALL anin("entraday%", dep!, "ANLGO", bintv%, -1, "nt", "tarea1")

'Inicialización Arreglo de Salida de datos
FOR I! = 1 TO dep!
     CALL arputvalf("salida%", I!, -1, "ANOUTO", 0!, 0)
NEXT I!

'Iniciar Identificación
CLS
LOCATE 24, 30
PRINT "PRESIONE ESPACIO PARA INICIAR IDENTIFICACION"
DO
    q$ = INKEY$
LOOP WHILE q$ = ""

'Impresión en pantalla de los gráficos y mensajes iniciales
CLS
CALL defgraf
LOCATE 21, 30
PRINT "TIEMPO REAL"
```

```
COLOR 2
LOCATE 42, 10
PRINT "IDENTIFICACION EN PROCESO";
COLOR 15
LOCATE 42, 40
PRINT "Presione una Tecla para terminar...";

'Inicialización Timer
'tim(0)=1: conteo con timer 0
'ndf: detección cuenta final del timer
'DT: Conteo del número de muestras
tim(0) = 1
ndf = -1
DT = 1
CALL timerstart(tim(), "nt", "timer")

'Iniciar interrupciones
CALL inton(1, "mil")

'Inicio del lazo de Identificación
DO

   'Rutina de Actualización de Datos Entrada-Salida
   FOR I = NA TO 2 STEP -1
      X(I) = X(I - 1)
   NEXT I
   X(1) = -Y
   FOR I = N TO N - NB + 2 STEP -1
      X(I) = X(I - 1)
   NEXT I
   X(NA + 1) = U

   'Recuperar valor de Y
   CALL traery

   'Calculo de U
   U = R * (1 + ERP / 100 * RND - ERP / 200)

   'U debe estar entre Umin y Umax
   IF U > Umax THEN
      U = Umax
   ELSEIF U < Umin THEN
      U = Umin
   END IF

   'Sacar valor de U
   CALL salidau

   'Inicio Algoritmo de Mínimos Cuadrados Recursivo
   CALL pvect(V(), P(), X())
   A = F + pesc(X(), V())
   FOR I = 1 TO N
      V(I) = V(I) / A
   NEXT I
   E = Y - pesc(X(), T())
```

```
'Cálculo Nuevos Parámetros
FOR I = 1 TO N
   T(I) = T(I) + V(I) * E
NEXT I

'Ingreso de valores a la Matriz de Datos
IF DK < MaxDat THEN
   DAT(DK + 1, 1) = Y
   DAT(DK + 1, 2) = U
   FOR I = 3 TO N + 2
      DAT(DK + 1, I) = T(I - 2)
   NEXT I
END IF

'Ingreso de valores a la Matriz de Parámetros
FOR I = 3 TO N + 2
   TM(DK MOD 50, I - 2) = T(I - 2)
NEXT I

'Cálculo de Matriz de Covarianzas P.
FOR I = 1 TO N
   FOR J = 1 TO N
      IF I = J THEN
         MA(I, J) = 1 - V(I) * X(J)
      ELSE
         MA(I, J) = -V(I) * X(J)
      END IF
   NEXT J
NEXT I
CALL pmat(MB(), MA(), P())
FOR I = 1 TO N
   FOR J = 1 TO N
      P(I, J) = MB(I, J) / F
   NEXT J
NEXT I

'Nuevo Gráfico si más de 580 valores
IF K > 580 THEN

   'Nuevo Gráfico
   CALL defgraf

   'Inicialización de K
   K = 0

END IF

'Graficar Nuevo Punto
CALL graf(K)

'Incrementar contadores
K = K + 1
DK = DK + 1
```

```
    'Verificación de Interrupciones
    DO

        'Recuperar valor del Timer
        CALL timerread(DT())

        'Muestreo siguiente detectado
        IF DT(0) / bintv% > DT THEN
           ndf = 1

        'Final del Conteo Detectado
        ELSEIF DT(0) / bintv% < DT - 3 THEN
           ndf = 0

        END IF

    'Repetir hasta que se dé el siguiente muestreo
    LOOP UNTIL ndf >= 0

    'Actualizar Conteo de Muestras
    DT = ndf * INT(DT(0) / bintv%) + 1
    ndf = -1

    'Falla del Control
    IF DK + 1 <> DT AND NF = 0 THEN
        LOCATE 42, 3
        BEEP
        NF = 1
        COLOR rojo
        PRINT "FALLA IDENTIFICACION" + SPACE$(12);
        COLOR blancoi
    END IF

'Repetir el lazo hasta la detección de una tecla
LOOP WHILE INKEY$ = ""

'Suspender proceso en Tiempo Real
CALL intoff
CALL ardel("entraday%")
CALL ardel("salida%")

'Fin de Identificación
LOCATE 42, 2
PRINT SPACE$(8); "Presione una Tecla para continuar..."; SPACE$(29);
DO
LOOP WHILE INKEY$ = ""

'Cálculo de Parámetros Promedio
FOR I = 1 TO N
    IF DK > 50 THEN
        FOR J = 0 TO 49
            TF(I) = TF(I) + TM(J, I)
        NEXT J
        TF(I) = TF(I) / 50
    ELSE
```

```
      TF(I) = T(I)
   END IF
NEXT I

'Definición de Pantalla y Colores para Menús y Cuadros
SCREEN 0
WIDTH 80, 25
COLOR blanco, negro
PALETTE 3, 25

'************************************************
' Menu Final.
' Opciones:
'    1. Imprimir Resultados en Papel
'    2. Regresar a Menu de Identificación
'************************************************

'Borrar Pantalla
CLS

'Definición de Arreglos para la impresión de Resultados
'datim(): Nombres Cuadro Resultados
'cimpr(): Nombres Menu Final
'cdat$(): Datos Cuadro Resultados
'cim$(): Datos Menu Final
DIM datim(20) AS STRING
DIM cimpr(10) AS STRING
DIM cdat$(20)
DIM cim$(10)

'Definición Cuadro de Resultados
datim(1) = "         RESULTADOS          "

'Nombre de Parámetros Calculados
FOR J = 1 TO NA
   datim(J + 1) = " A(" + LTRIM$(RTRIM$(STR$(J))) + ") = "
NEXT J
FOR J = 1 TO NB
   datim(NA + J + 1) = " B(" + LTRIM$(RTRIM$(STR$(J))) + ") = "
NEXT J

'Iteraciones, Interrupciones
datim(N + 2) = " Iteraciones = "
datim(N + 3) = " Interrupciones ="

'Conversión de Datos a Caracteres
FOR J = 1 TO N
   cdat$(J + 1) = convert$(TF(J))
NEXT J
cdat$(N + 2) = convert$(DK)
cdat$(N + 3) = convert$(DT - 1)

'Definición Menu Final
cimpr(1) = "    FIN DE LA IDENTIFICACION    "
cimpr(2) = " Imprimir resultados"
```

```
cimpr(3) = " Terminar "

'Menu Final y Cuadro de Resultados
DO

    'Accesible: Menú Final
    'Solo visible: Cuadro de Resultados
    ERF% = 0
    J = menuda(2, 2, N + 3, datim(), 1, cdat$())
    I = menuda(5, 39, 3, cimpr(), 2, cim$())
    SELECT CASE I

        'Impresión de Resultados en Papel
        CASE 1

            'Borrar zona de mensajes de Error
            clerr

            'Imprimir
            CALL printer(para$(), TF(), T(), DK, 3)

    END SELECT

'Repetir hasta la obtener Opción Terminar
LOOP UNTIL I = 2

END SUB
```

SUBRUTINA trparam

```
'=============================================================================
' SUBRUTINA trparam
' Genera el Cuadro de Parámetros para Tiempo Real
'=============================================================================
DEFINT I-N
DEFDBL A-H, O-Z
SUB trparam

STATIC PR AS INTEGER

'Definición de arreglos para el Cuadro de Parámetros
DIM trpar(10) AS STRING
DIM trint$(2)

'Definición Cuadro de Parámetros
trpar(1) = "          PARAMETROS TIEMPO REAL          "
trpar(2) = " Interrupciones (ms) = "

'Valor Predefinido
IF PR = 0 THEN
   bintv% = 100
   PR = 1
END IF

'Conversión de dato a caracter
trint$(2) = LTRIM$(RTRIM$(STR$(bintv%)))
CLS

'Accesible: Cuadro de Parámetros en Tiempo Real
I = menuda(5, 20, 2, trpar(), 30, trint$())

'Definición constantes de tiempo real
dep! = 20!
bintv% = VAL(trint$(2))

END SUB
```

FUNCTION valid

```
'=============================================================================
' FUNCTION valid
' Permite la validación de los datos ingresados
' Parámetros de Entrada:
'   b!: Tipo de datos
'       Valores posibles:
'       1: Datos solo visibles
'       10: Parámetros
'       20: Valores Iniciales
'   P%: Posición del puntero de opciones
' Parámatros de Entrada-Salida
'   V$(): Vector de datos
' Salida: Validación
'   Valores Posibles:
'   0: Datos correctos
'   1: Datos incorrectos
'=============================================================================
FUNCTION valid (b!, V$(), P%)

'Detección del Tipo de Datos
SELECT CASE b!

   'Parámetros
   CASE 10

      'Cero si vacío o punto
      IF LEN(V$(P%)) = 0 OR V$(P%) = "." THEN
         V$(P%) = "0"
      END IF

      'Posición del Puntero
      SELECT CASE P%

         'Orden
         CASE 2
            T = VAL(V$(2))
            IF T - INT(T) = 0 AND T >= 1 AND T < 5 THEN
               D = VAL(V$(7))
               IF T = 3 AND D > 900 THEN
                  V$(7) = "900"
               ELSEIF T = 4 AND D > 740 THEN
                  V$(7) = "740"
               END IF
               valid = 0
            ELSE
               valid = 1
            END IF

         'Alfa
         CASE 3
            IF VAL(V$(3)) >= 0 THEN
               valid = 0
            ELSE
```

```
        valid = 1
    END IF

'Referencia
CASE 4
    T = VAL(V$(4))
    IF T >= 0 AND T <= 10 THEN
        valid = 0
    ELSE
        valid = 1
    END IF

'Umax
CASE 5
    T = VAL(V$(5))
    IF T >= 0 AND T <= 10 THEN
        valid = 0
    ELSE
        valid = 1
    END IF

'Umin
CASE 6
    T = VAL(V$(6))
    IF T >= 0 AND T <= 10 THEN
        valid = 0
    ELSE
        valid = 1
    END IF

'Número de Datos
CASE 7
    T = VAL(V$(7))
    IF T - INT(T) = 0 AND T >= 0 THEN
        D = VAL(V$(2))
        IF D <= 2 AND T <= 1000 THEN
            valid = 0
        ELSEIF D = 3 AND T <= 900 THEN
            valid = 0
        ELSEIF D = 4 AND T <= 740 THEN
            valid = 0
        ELSE
            valid = 1
        END IF
    ELSE
        valid = 1
    END IF

'Error Porcentual
CASE 8
    IF VAL(V$(8)) >= 0 AND VAL(V$(8)) <= 100 THEN
        valid = 0
    ELSE
        valid = 1
    END IF
```

```
      END SELECT

   'Valores iniciales
   CASE 20
      IF VAL(V$(P%)) = 0 THEN
         valid = 1
      ELSE
         valid = 0
      END IF

   'Otros Datos Numéricos
   CASE 30

      'Cero si vacío o punto
      IF LEN(V$(P%)) = 0 OR V$(P%) = "." THEN
         V$(P%) = "0"
      END IF

   'Menu de opciones
   CASE ELSE
      valid = 0

END SELECT

END FUNCTION
```

APENDICE B: PROGRAMAS ADICIONALES

1. PROGRAMA TSGRAF.BAT.

a) Su funcionamiento

Este programa con comandos del sistema operativo (tipo BATCH) permite la transferencia del control al programa LOTUS 123 desde el subdirectorio d:\ADAPTIVO. Esto se realiza copiando el archivo TESISAD.WK1 al directorio D:\LOTUS con un nuevo nombre (AUTO123.WK1) para que el LOTUS 123, al ejecutarse, ingrese directamente al archivo de graficación. Al salir, este archivo es borrado del directorio.

b) Listado

TDGRAF.BAT contiene:

```
@echo off
copy d:\adaptivo\tesisad.wk1 c:\lotus\auto123.wk1
c:\lotus
lotus
del auto123.wk1
d:\adaptivo
```

2. PROGRAMA COMPIL.BAT

Este programa igualmente desarrollado con comandos DOS, permite la compilación externa del programa TESISAD.BAS, dado que por su tamaño, no es posible compilarlo desde el QB.

Consta de los siguientes comandos:

```
bc d:\adaptivo\tesisad.bas,,nul.lst /O /e /x /s
link c:\qb45\tesisad.obj,,,qlib.lib /e
```

3. ARCHIVO TESISDAD.WK1

Este archivo es una hoja electrónica para LOTUS 123. Allí se encuentran predefinidos los valores que permiten la construcción de los gráficos, un listado de gráficos preestablecidos para facilitar su elaboración y un macro autoejecutable al entrar a la hoja electrónica que permite ingresar directamente los datos requeridos y que pasa automáticamente a la zona de graficación del LOTUS 123.

Los gráficos predefinidos son:

IO100: Entrada y Salida con 100 datos
IO900: Entrada y Salida con 900 datos
P2A100: Parámetros "a" 2° orden, 100 datos
P2A900: Parámetros "a" 2° orden, 900 datos

P2B100: Parámetros "b" 2° orden, 100 datos

P2B900: Parámetros "b" 2° orden, 900 datos

P3A100: Parámetros "a" 3° orden, 100 datos

P3A900: Parámetros "a" 3° orden, 900 datos

P3B100: Parámetros "b" 3° orden, 100 datos

P3B900: Parámetros "b" 3° orden, 900 datos

P4A100: Parámetros "a" 4° orden, 100 datos

P4A900: Parámetros "a" 4° orden, 900 datos

P4B100: Parámetros "b" 4° orden, 100 datos

P4B900: Parámetros "b" 4° orden, 900 datos

El ingreso de datos al LOTUS se realiza a través del comando IMPORT NUMBERS.

APENDICE C: PROGRAMA PID

1. CONTROL TIPO PID

Como una prueba adicional a las realizadas se realizó un pequeño programa para controlar las plantas con control tipo PID. El objetivo es mostrar el comportamiento y las definiencias del PID cuando existen variaciones en las características de la planta. Allí el control adaptivo tipo self-tuning resulta mucho más eficiente.

Para esta prueba se desarrolló un programa basado en los algoritmos de graficación desarrollados en este trabajo. Posteriormente se realizaron dos pruebas en tiempo real con la planta de segundo orden.

2. PID DISCRETIZADO.

En el dominio del tiempo, la señal de control de un controlador tipo PID está dada por:

$$u = K_p e + K_i \int e dt + K_d \frac{de}{dt}$$

siendo:

u: señal de control

e: error

K_p: constante proporcional

K_i : constante integral

K_d : constante derivativa

En el plano s (transformada de Laplace), la ecuación está dada por:

$$u(s) = \left(K_p + \frac{K_i}{s} + K_d s\right) e(s)$$

Realizando la transformada z con aproximando la integral por el método de las sumas de trapecios, se obtiene:

$$\frac{u(z)}{e(z)} = K_p + K_i \frac{T(z+1)}{2(z-1)} + \frac{K_d}{T} \frac{z-1}{z}$$

donde T es el periodo de muestreo.

Entonces, con la transformada z inversa, se tiene:

$$u(k) = u(k-1) + b_1 e(k) + b_2 e(k-1) + b_3 e(k-2)$$

donde:

$$b_1 = K_p + K_i \frac{T}{2} + \frac{K_d}{T}$$

$$b_2 = -K_p + K_i \frac{T}{2} - \frac{2K_d}{T}$$

$$b_3 = \frac{K_d}{T}$$

Con lo que queda definido el algoritmo a implementarse en QUICK BASIC.

2. EL PROGRAMA

El programa consta de un módulo principal (PID.BAS) y algunas rutinas adicionales elaboradas para este trabajo: SALIDAU, TRAERY, GRAF, DEFGRAF y una rutina elaborada aqui: SPID. Los parámetros b_i son almacenados en el vector T(). El diagrama de flujo se muestra en el diagrama C.1. y el listado del programa se muestra en la sección 4 de este apéndice.

3. PRUEBAS

Se reaizaron dos pruebas con el PID. Ambas utilizaron la planta de segundo orden ya descrita para el tirmpo real (circuito RC-RC). Se tomaron como parámetros para el PID, los siguientes valores: $K_p = 4$; $K_i = 2$ y $K_d = 1$.

1.- Control con perturbación momentánea,

Se trata aquí de realizar el control de la planta cuando ésta sufre una perturbación momentánea (corto-circuito). Los resultados de señal de control y señal de salida se muestran en los gráficos C.1 y C.2 respectivamente.

La planta no ha cambiado y el PID está en capacidad de responder adecuadamente cuando existe una perturbación momentánea. El resultado que se obtiene es similar al caso del control adaptivo.

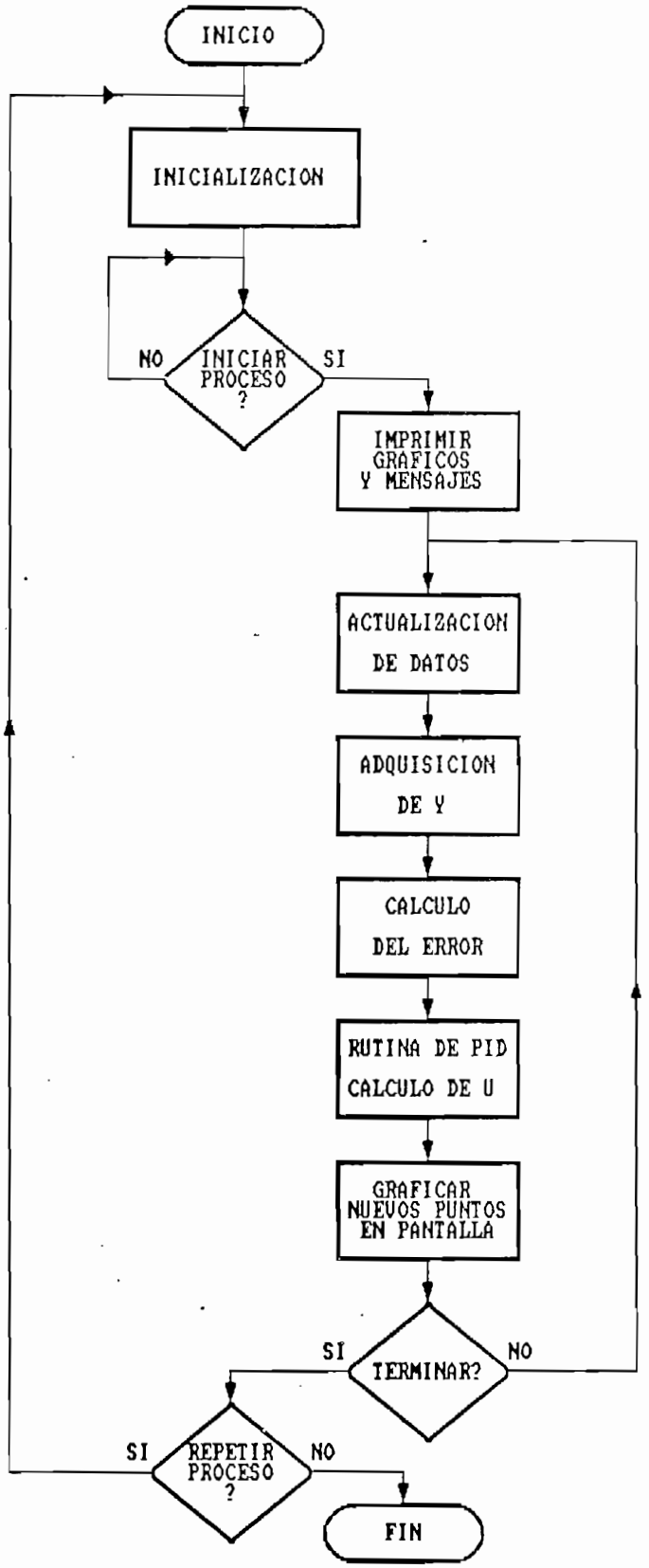


Diagrama C.1

PID

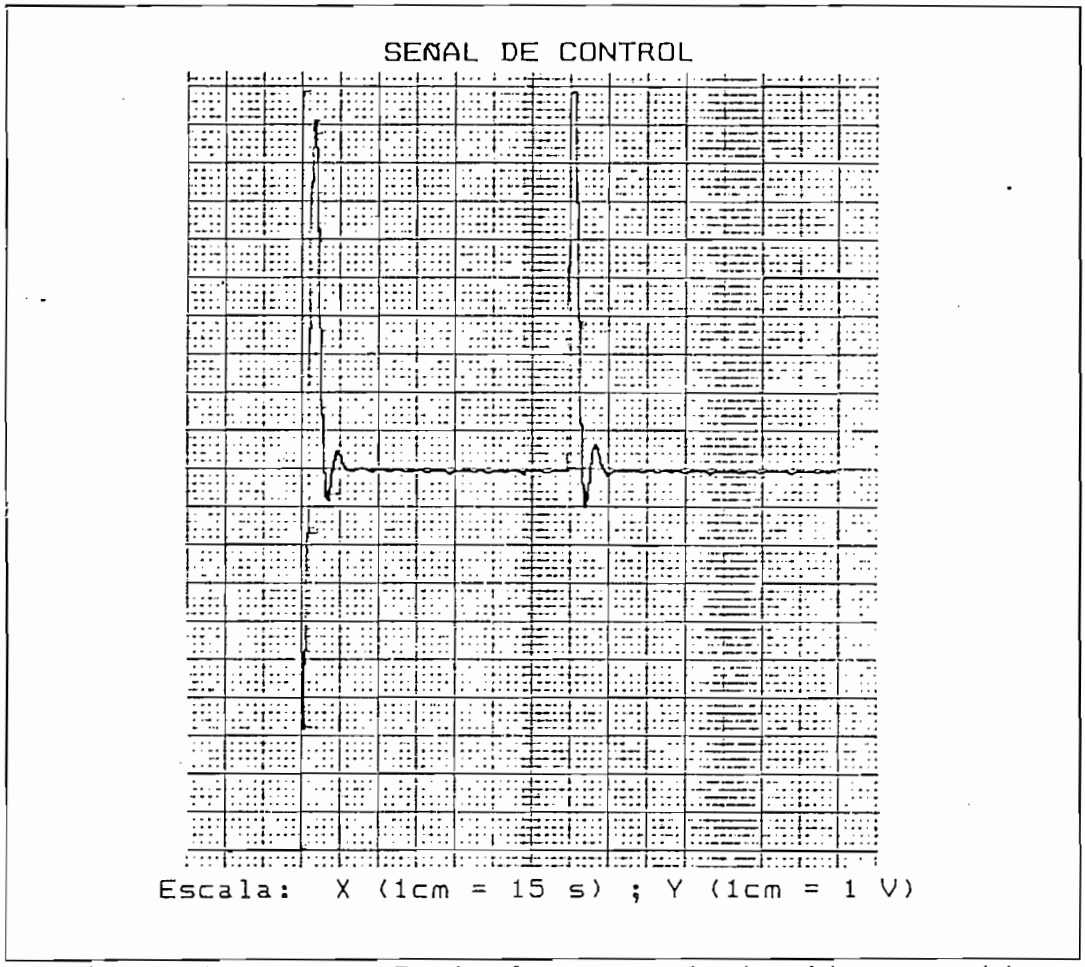


Gráfico C.1 Control con perturbación momentánea

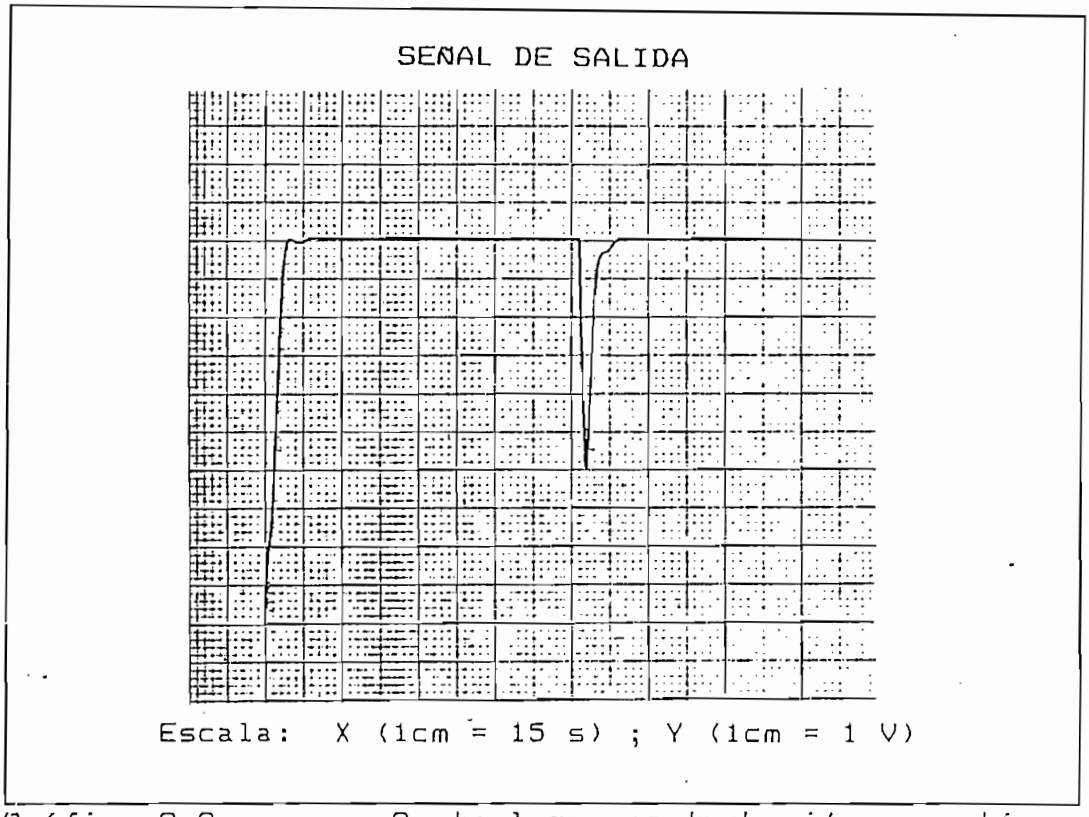


Gráfico C.2 Control con perturbación momentánea

2.- Control con cambio de planta.

En este caso la perturbación es permanente al cambiar físicamente la planta (se ha añadido un condensador). Los resultados se muestran en las figuras C.3 y C.4. Aquí se puede apreciar claramente la diferencia con el control adaptivo. Los parámetros definidos para el PID ya no corresponden a la nueva planta y la dinámica cambia. El PID finalmente ajusta la salida a la referencia pero después de una serie de oscilaciones que son bastante significativas especialmente en la señal de control.

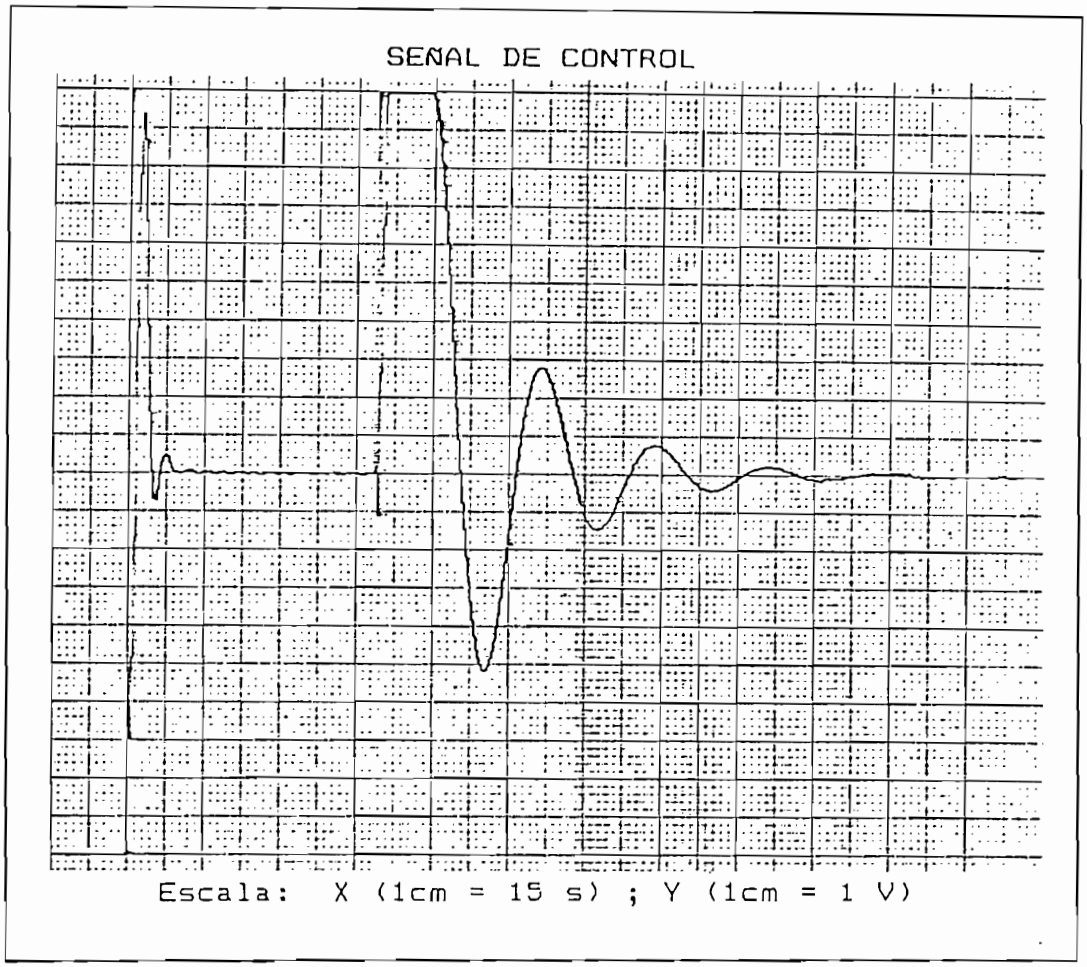


Gráfico C.3 Control con cambio de planta

Con esto queda verificada experimentalmente la utilidad del control adaptivo frente al PID clásico. El control adaptivo está en condiciones de responder mucho mejor a variaciones de la planta con una respuesta que definitivamente supera lo que puede hacer el PID.

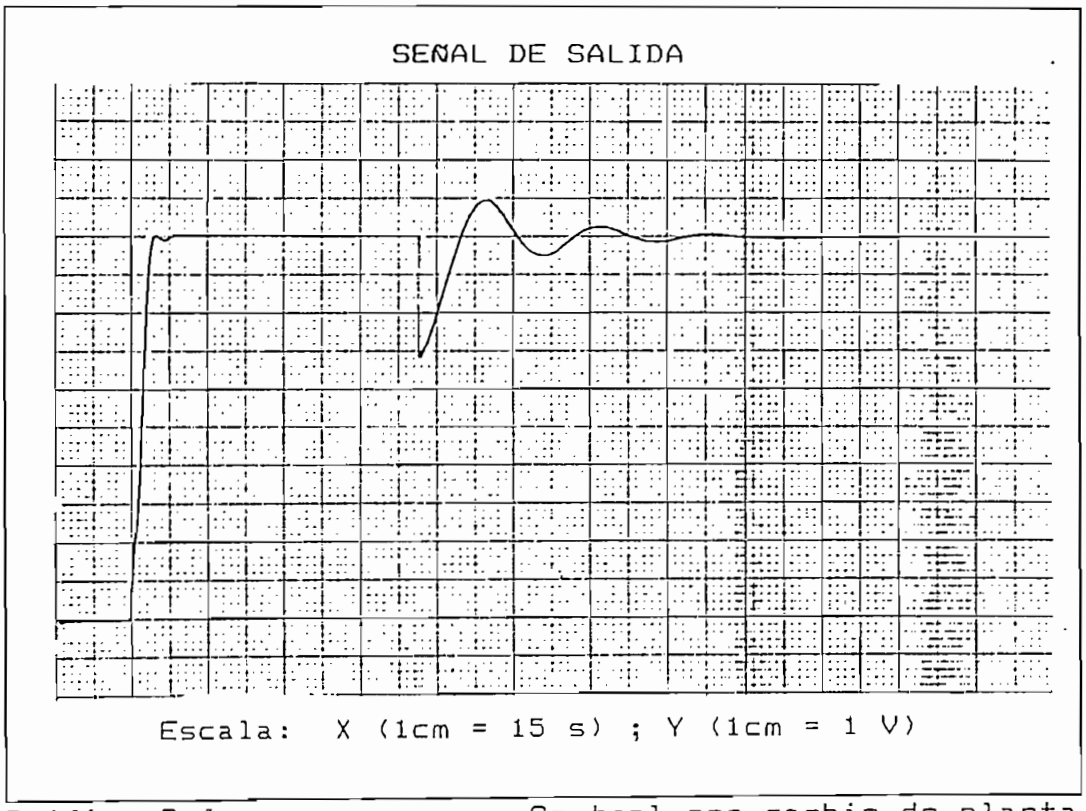


Gráfico C.4 Control con cambio de planta

APENDICE D: MANUAL DEL USUARIO

1. SOFTWARE Y REQUERIMIENTOS

a) Software

El sistema consiste en:

* un archivo TESISAD.EXE ejecutable
* un archivo TESISDAD.BAS realizado en QUICK BASIC 4.5
* un archivo TSGRAF.BAT (DOS 3.30)
* un archivo TESISAD.WK1 para LOTUS 2.0
* un archivo COMPIL.BAT para la compilación de TESISAD.BAS

b) Requerimientos

Para poder ser ejecutado, compilado o editado, el sistema requiere:

* Microsoft QUICK BASIC 4.5
* Tarjeta de gráficos VGA a colores
* Librerías y rutinas del QUICK 500 de KEITHLEY
* Equipo de adquisición de datos KEITHLEY 500A
* Todos los archivos que conforman el sistema deberán estar instalados en el directorio D:\ADAPTIVO.

2. EJECUCION DEL PROGRAMA PRINCIPAL

El programa TESISAD.EXE se debe ejecutar con el prefijo QRUN para activar las rutinas de tiempo real.

Inicio: D:\ADAPTIVO\QRUN TESISAD

Al ejecutarse correctamente el programa aparece un primer menú con tres opciones:

1. Simulación -----> Rutinas de Simulación
2. Tiempo Real -----> Rutinas de Tiempo Real
3. Salir -----> Terminar

NORMAS GENERALES PARA EL USO:

1) Las opciones se escogen ubicando el cursor sobre la que se desea (con las flechas) y presionando ENTER.

2) El ingreso de un nuevo valor se debe hacer borrando el anterior. Se usa para esto la tecla BACKSPACE, se ingresan los datos y se presiona ENTER. Una señal auditiva indicará un error en el ingreso de datos.

2.1. Simulación

En simulación se tiene el menu siguiente:

1. Modelo ----> Definición del modelo
2. Identificación ----> Iniciar identificación
3. Identificación y control ----> Iniciar control
4. Matriz de datos ----> Sacar datos al disco
5. Terminar ----> Terminar

Es obligatorio iniciar con la opción 1 antes de pasar a las otras.

2.1.1. Modelo

Aparecen tres ventanas, una de las cuales está activada. Las otras están sólo visibles (en celeste). En el Menu de Modelo hay 4 opciones:

1. Nuevo Orden ----> Ingresar nuevo orden
2. Parámetros ----> Ingresar nuevos parámetros
3. Modelo Predefinido ----> Recuperar valores predefinidos
4. Terminar ----> Volver a simulación

2.1.2. Identificación

Para la identificación aparecen tres ventanas. Sólo el Menu de Identificación está activado al inicio. Aparecen las opciones siguientes:

1. Parámetros ----> Nuevos parámetros
2. Valores Iniciales ----> Nuevos valores
3. Parámetros y valores predefinidos ----> Recuperar
4. Iniciar Identificación ----> Inicio
5. Terminar ----> Menú Simulación

Si se escoje la opción 4, se inicia la identificación y aparecen los siguientes mensajes, en su orden:

* PRESIONE ESPACIO PARA INICIAR IDENTIFICACION
* Presione una tecla para terminar ---> Queda el gráfico en pantalla
* Presione una tecla para terminar ---> Se pasa al Menu Final

Se tienen entonces los resultados en una ventana sólo visible y el menu final con dos opciones:

1. Imprimir Resultados ---> Reporte en la impresora
2. Terminar ---> Volver

Si la impresora no está conectada y se escoje la primera opción se verá un mensaje de error.

2.1.3. Identificación y control.

Las ventanas y las opciones son las mismas que para identificación, sólo que ahora se añade una más:

4. Ubicación de polos ---> Define los coeficientes de polos

Los mensajes de salida del control son similares (ver 2.1.2. identificación).

2.1.4. Matriz de datos

Aparece un Menu de Datos Para Impresión. Opciones:

1. Nuevo Archivo ----> Nuevo nombre
2. Sacar datos al disco ----> Transferir a disco duro
3. Imprimir (LOTUS) ----> Transferir a LOTUS
4. Terminar ----> Volver

Aparecerá un error si no se encuentran presentes los archivos TSGRAF.BAT y TESISAD.WK1 en D:\ADAPTIVO.

2.2. Tiempo Real

Al entrar a tiempo real debe estar encendido el equipo de adquisición de datos. Caso contrario, se tendrá un error.

Los menús para tiempo real son similares a los de identificación. Hay dos diferencias en el Menú de Tiempo Real:

1. Parámetros ----> Periodo de muestreo

...

4. Inicialización ----> Inicialización Equipo Adquisición

...

La opción 1 es obligatoria.