ESCUELA POLITECNICA NACIONAL

ESCUELA DE INGENIERIA

MODELO DEL MOTOMATIC CON REALIMENTACIÓN DE VELOCIDAD Y POSICIÓN ANTE UNA SEÑAL PASO MEDIANTE MODELACIÓN, IDENTIFICACIÓN Y REDES NEURONALES

PROYECTO PREVIO A LA OBTENCIÓN DEL TITULO DE INGENIERO EN ELECTRÓNICA Y CONTROL

HERNÁN PATRICIO JÁCOME GRANIZO

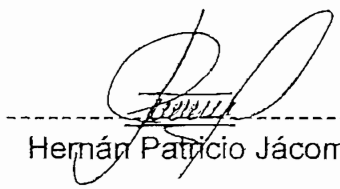
DIRECTOR: PROF. MSc. PATRICIO BURBANO P.

Quito, Julio del 2002

DECLARACIÓN

Yo Hernán Patricio Jácome Granizo, declaro bajo juramento que el trabajo aquí descrito es de mi autoría; que no ha sido previamente presentado para ningún grado o calificación profesional; y, que he consultado las referencias bibliográficas que se incluyen en este documento.

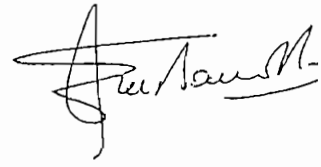
A través de la presenta cedo mis derechos de propiedad intelectual correspondiente a este trabajo, a la Escuela Politécnica Nacional, según lo establecido por la Ley de Propiedad Intelectual, por su Reglamento y por la normatividad institucional vigente.



Hernán Patricio Jácome Granizo

CERTIFICACIÓN

Certifico que el presente trabajo fue realizado en su totalidad por el señor Hernán Patricio Jácome Granizo.



ING. PATRICIO BURBANO.

Director de Tesis.

AGRADECIMIENTO

Especialmente al Ing. PATRICIO BURBANO mis más sinceros agradecimientos por su paciencia y acertada dirección, sin la cuál no hubiese llegado al término de este trabajo.

A todas las personas que me ayudaron en el desarrollo de toda mi carrera.

DEDICATORIA

A Dios por cuidar y guiar mi camino.

A mis padres que siempre confiaron en mí y me apoyaron en todas mis decisiones.

A mis hermanos y toda mi familia que en su momento me dieron su respaldo.

CONTENIDO

CAPITULO 1

ESTUDIO DEL COMPORTAMIENTO DINÁMICO Y MODELACIÓN DEL MOTOMATIC

CAPITULO 2

IDENTIFICACIÓN DEL MOTOMATIC CON REALIMENTACIÓN DE VELOCIDAD Y POSICIÓN

CAPITULO 3

SIMULACIÓN DEL MOTOMATIC CON REALIMENTACIÓN DE VELOCIDAD Y POSICIÓN MEDIANTE REDES NEURONALES

CAPITULO 4

DESARROLLO DE SOFTWARE EN MATLAB PARA LA SIMULACIÓN DEL MOTOMATIC

CAPITULO 5

RESULTADOS Y EVALUACIÓN DE LAS DIFERENTES METODOLOGÍAS

RESUMEN

Para realizar análisis y diseño de sistemas de control es necesario partir de un modelo del sistema. Se pueden aplicar distintas estrategias para obtener dicho modelo.

Se utiliza la Modelación cuando se conoce la estructura y los componentes de la planta, entrega un modelo analítico a ecuaciones diferenciales o ecuaciones de diferencia, función de transferencia o a variables de estado.

La Identificación es otra alternativa, en la que se utiliza un algoritmo computacional, manejando mediciones de entrada y salida, también se obtiene un modelo analítico.

Como última alternativa se presenta en este trabajo la aplicación de Redes Neuronales, como un método de inteligencia artificial, que se fundamenta en la utilización de las señales de entrada y salida, una estructura y el entrenamiento de una conexión de neuronas artificiales.

En el presente trabajo se emplea modelación, identificación y redes neuronales para modelar el Motomatic con realimentación de velocidad y posición.

Para dicho propósito se desarrolla un programa en Matlab para la simulación mediante ecuaciones diferenciales, ecuaciones de diferencia y redes neuronales.

Se plantea la aplicación de las redes neuronales para el modelo del Motomatic como una alternativa a los métodos de identificación y modelación.

PRESENTACIÓN

Este trabajo se centra en la comparación de los distintos métodos para obtener el modelo de sistemas como son: por modelación, identificación y redes neuronales.

En la modelación es necesario conocer la estructura y componentes de la planta a modelar, por tal razón en el capítulo 1 se trata del estudio del comportamiento dinámico de la máquina de corriente continua, para con esto aplicar conceptos y principios básicos de ingeniería con la finalidad de obtener el modelo de dicho sistema.

En el capítulo 2 se aplica dos técnicas para la identificación de la máquina de corriente continua: identificación en tiempo por el método de determinación de constantes (Motomatic con realimentación de velocidad) y aproximando a un sistema de segundo grado (Motomatic con realimentación de velocidad) e identificación paramétrica discreta en base a mínimos cuadrados ordinarios y recursivos.

El modelo del Motomatic con realimentación de velocidad y posición mediante redes neuronales es explicado e implementado en el capítulo 3.

Con los resultados obtenidos se procede a implementar un software de aplicación en el capítulo 4, para que se evalúe las diferentes técnicas de modelación en el capítulo 5.

CAPITULO 1

ESTUDIO DEL COMPORTAMIENTO DINÁMICO Y MODELACIÓN DEL MOTOMATIC

1.1 ESTUDIO DEL COMPORTAMIENTO DINÁMICO DE LA MÁQUINA DE CORRIENTE CONTINUA.

Una máquina eléctrica es un dispositivo que puede convertir energía eléctrica en energía mecánica en dicho caso funcionaría como motor, sin embargo también puede convertir energía mecánica en energía eléctrica por lo cuál funcionaría como generador. Esta conversión de energía eléctrica en mecánica o mecánica en eléctrica se realiza a través de la acción de campos magnéticos [1].

En este trabajo solo se centra al funcionamiento de la máquina eléctrica como motor, específicamente al motor de corriente continua controlado por voltaje de armadura.

La máquina de corriente continua es en la actualidad la más común si se requiere controlar sobre un amplio rango de velocidades. Esto es debido a que tiene excelentes propiedades de operación y características de control. La desventaja es la presencia del colector el cual restringe la potencia y la velocidad, incrementa la inercia y por tanto requiere un mantenimiento periódico [2].

Una máquina de C.C. (corriente continua) tiene dos componentes básicos, el bobinado de campo, montado en el estator y el bobinado de armadura montado en el rotor.

Los dos bobinados son alimentados con fuentes de corriente continua. El conmutador o colector situado en el rotor está conectado a los conductores de la armadura. Este actúa como un cambiador de frecuencia mecánico o como rectificador, para mantener una corriente unidireccional a través de las escobillas en todos los rangos de velocidades.

En la figura 1 se representa a la máquina rotatoria DC más sencilla la cuál posee una sola espira de alambre que rota alrededor de un eje fijo. La parte rotante de la máquina se llama el rotor; la parte estacionaria se denomina estator. El campo

magnético de la máquina es suministrado por los polos norte y sur magnéticos mostrados sobre el estator en la figura 1.

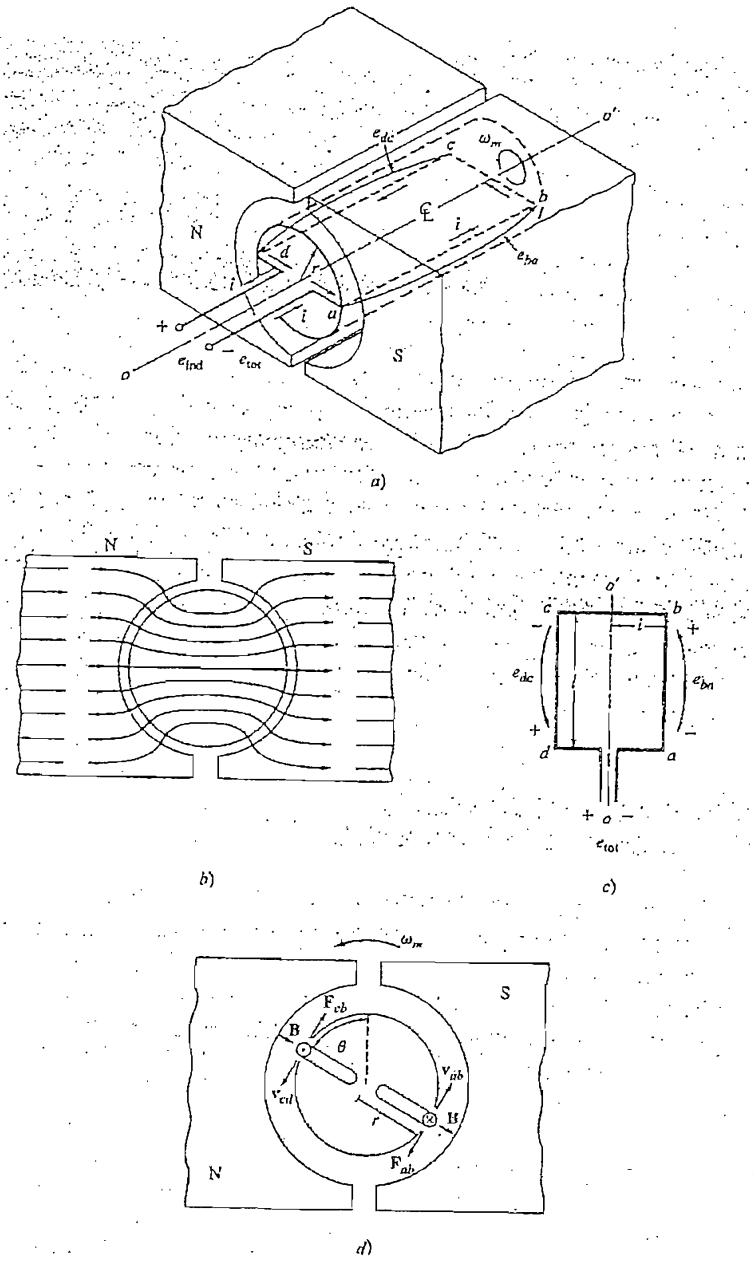


Figura 1.

Lazo sencillo rotacional entre caras polares curvas. a) Vista en perspectiva; b) vista de las líneas de campo; c) vista superior; d) vista frontal.

Como se puede observar en la figura 1 la anchura del entrehierro entre el rotor y el estator es constante por construcción [1]. Puesto que el flujo magnético debe

tomar el camino más corto a través del aire, éste es perpendicular a la superficie del rotor en todos los puntos situados bajo las superficies polares.

Al ser la reluctancia uniforme por la anchura del entrehierro uniforme, significa que la densidad de flujo magnético es constante en todo punto situado bajo las caras polares.

1.1.1 VOLTAJE INDUCIDO EN EL LAZO ROTACIONAL

Si se gira el rotor de esta máquina, se inducirá un voltaje en la espira. En la figura 2 se muestra una espira de alambre rectangular, los lados ab y cd son perpendiculares al plano de la página y los lados bc y da son paralelos a este plano. El campo magnético es constante y perpendicular a la superficie del rotor en todo punto situado bajo las caras polares y cae con rapidez a cero más allá de los extremos de los polos.

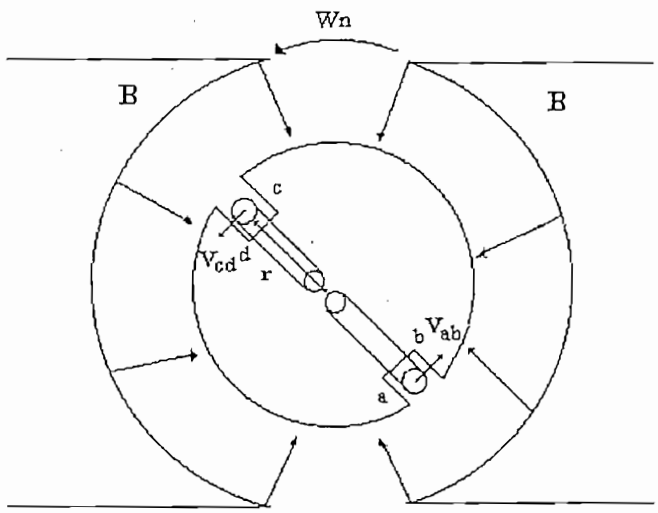


Figura 2.

Deducción de una ecuación para los voltajes inducidos en la espira.

Para determinar el voltaje total e_{tot} en la espira, se examina cada segmento de ella por separado y se suman los voltajes resultantes. El voltaje de cada segmento está dado por la ecuación (1).

$$e_{ind} = (vxB)l \tag{1}$$

- *Segmento ab.* En este segmento, la velocidad del alambre es tangencial a la trayectoria de rotación. El campo magnético **B** apunta en dirección perpendicular hacia fuera de la superficie del rotor en todo punto situado debajo de la cara polar, y es cero más allá de los extremos de ésta. Bajo la cara polar, la velocidad es **v** es perpendicular a **B** y la cantidad **vxB** apunta hacia adentro de la página. Entonces, el voltaje inducido en este segmento es

$$e_{ba} = (vxB)l \tag{2}$$

$$= \begin{cases} vBl \text{ positivo hacia dentro de la página} & \text{bajo la cara polar} \\ 0 & \text{más allá de los extremos de los polos} \end{cases}$$

- *Segmento bc.* En este segmento, la cantidad **vxB** apunta hacia dentro o hacia fuera de la página mientras que la longitud **l** está en el plano de la página, de modo que **vxB** es perpendicular a **l** y, por tanto, el voltaje en este segmento será cero:

$$e_{cb} = 0 \tag{3}$$

- *Segmento cd.* En este segmento, la velocidad del alambre es tangencial a la trayectoria de rotación. El campo magnético **B** apunta en dirección perpendicular hacia adentro de la superficie del rotor en todo punto situado bajo la cara polar, y es cero más allá de los extremos de ésta. Bajo la cara polar, la velocidad **v** es perpendicular a **B** y la cantidad **vxB** apunta hacia fuera de la página. Entonces, el voltaje inducido es este segmento es

$$e_{dc} = (vxB)l \tag{4}$$

$$= \begin{cases} vBl \text{ positivo hacia fuera de la página} & \text{bajo la cara polar} \\ 0 & \text{más allá de los extremos de los polos} \end{cases}$$

- Segmento da. Como el segmento dc, **vxB** es perpendicular a l. Entonces, el voltaje en el segmento también será cero:

$$e_{ad} = 0 \tag{5}$$

El voltaje total inducido en la espira e_{ind} está dado por

$$e_{ind} = e_{ba} + e_{cb} + e_{dc} + e_{ad} \tag{6}$$

$$e_{ind} = \begin{cases} 2vbl & \text{bajo las caras polares} \\ 0 & \text{más allá de los extremos de los polos} \end{cases}$$

Cuando la espira rota 180°, el segmento ab está bajo la cara polar norte y no en la cara polar sur. En este momento, la dirección del voltaje del segmento se invierte, pero su magnitud permanece constante. La figura 3 muestra el voltaje resultante e_{tot} en función del tiempo.

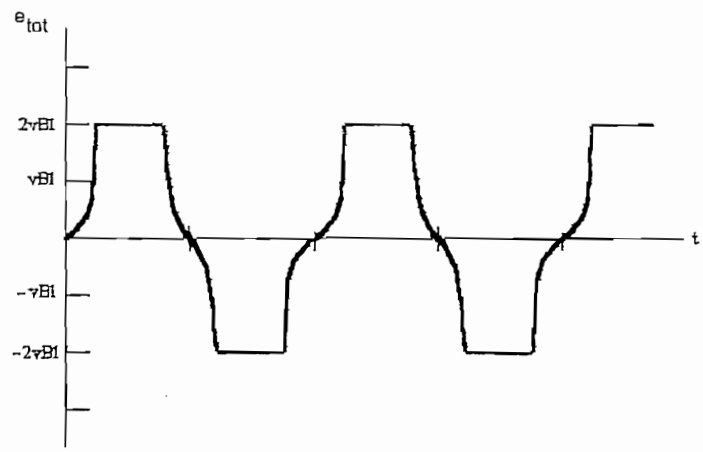


Figura 3.

Voltaje de salida de la espira

1.1.2 OBTENCIÓN DEL VOLTAJE DC DE SALIDA DE LA ESPIRA ROTATORIA

En la figura 4a se puede observar que se adicionan al extremo de la espira dos segmentos conductores semicirculares y se sitúan dos contactos fijos en un ángulo tal que en el instante cuando el voltaje en la espira es cero, los contactos cortocircuitan los dos segmentos. De este modo, cada vez que el voltaje de la espira cambia de dirección, los contactos también cambian las conexiones, y la salida de los contactos esta siempre construida de la misma manera (figura 4b). Este proceso de cambio de conexión se conoce como conmutación. Los segmentos semicirculares rotantes se denominan segmentos de conmutación [1] y los contactos fijos se llaman escobillas.

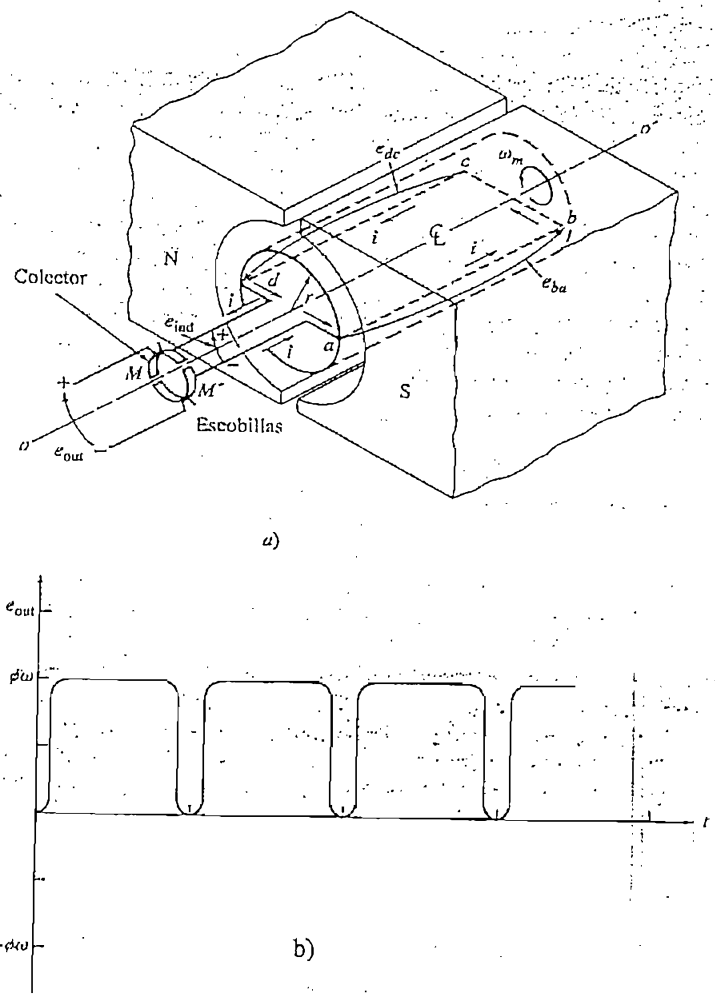


Figura 4.

Producción de una salida dc de la máquina con colector y escobillas

1.1.3. PAR INDUCIDO EN LA ESPIRA ROTATORIA

Si se conecta una batería a la máquina de la figura 4. La configuración resultante se muestra en la figura 5. Para determinar el par, obsérvese en detalle la espira mostrada en la figura 5b.

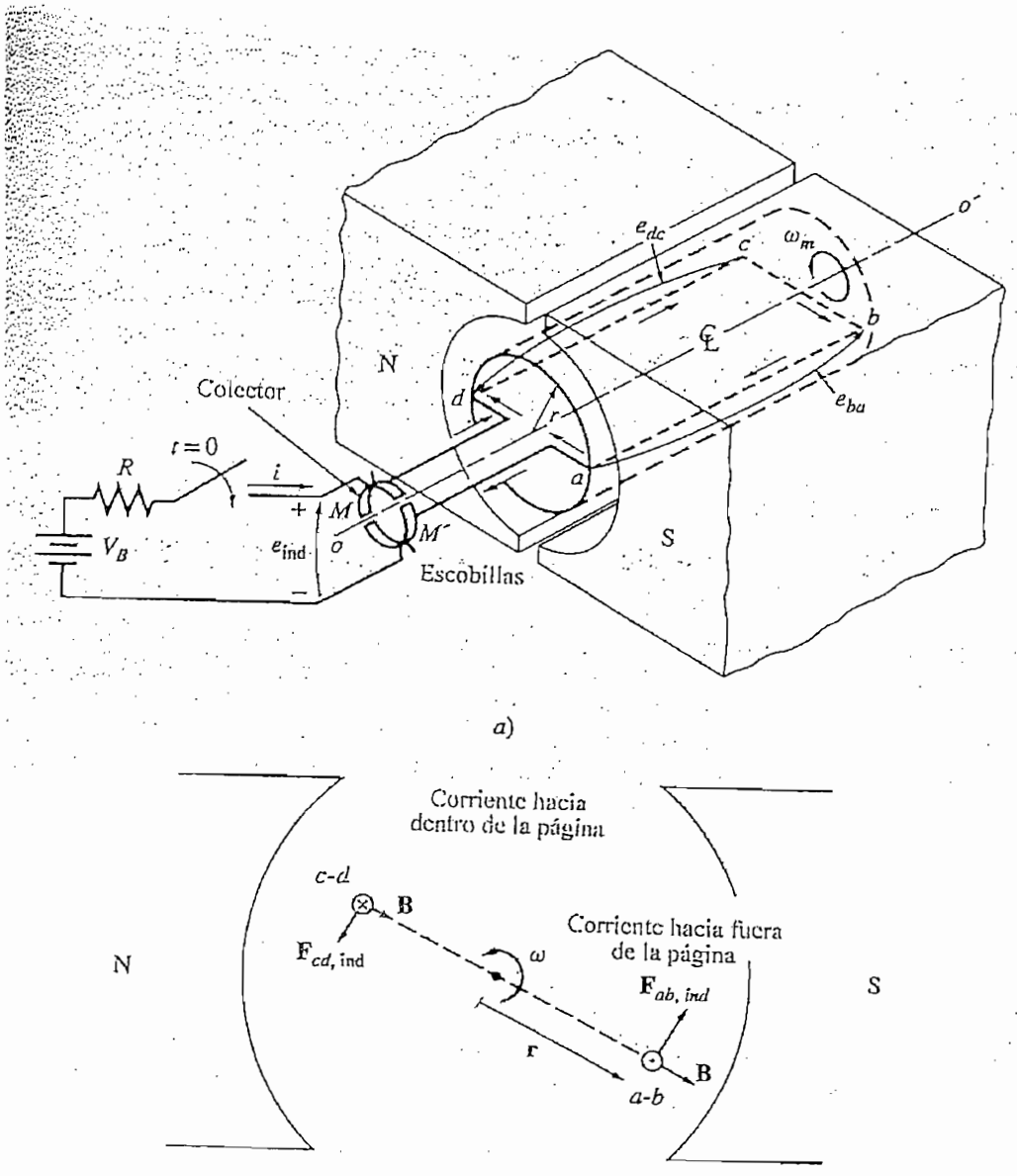


Figura 5.

Deducción de una ecuación para el par inducido en la espira. Nótese que el hierro del núcleo no se muestra en el literal b) por claridad.

El método que debe emplearse para determinar el par sobre la espira consiste en tener por separado cada segmento de ésta y luego sumar los efectos de los segmentos individuales. La fuerza inducida sobre el segmento de la espira está dada por la ecuación (7).

$$F = i(lxB) \tag{7}$$

y el par sobre el segmento está dado por

$$\tau = rFsen\theta \tag{8}$$

donde θ es el ángulo entre r y F. El par es cero en todos los puntos en los que la espira está situada fuera de las caras polares.

Mientras la espira está bajo las caras polares el par es

- *Segmento ab.* En el segmento ab la corriente de la batería está dirigida hacia fuera de la página. El campo magnético bajo la cara polar apunta radialmente hacia fuera del rotor; por tanto la fuerza inducida sobre el alambre está dada por

$$F_{ab} = i(lxB)$$
$$F_{ab} = ilB \quad \textit{Tangente a la dirección del movimiento} \tag{9}$$

El par sobre el rotor, causado por esta fuerza es

$$\tau_{ab} = rFsen\theta$$
$$\tau_{ab} = r(ilB)sen\,90° \tag{10}$$
$$\tau_{ab} = rilB \quad \textit{en sentido contrario a las manecillas del reloj}$$

- *Segmento bc.* En el segmento bc la corriente de la batería fluye desde la parte superior izquierda hacia la inferior derecha del dibujo. La fuerza inducida en el alambre está dada por

$$F_{bc} = i(lxB) \tag{11}$$
$$F_{bc} = 0 \text{ puesto que } l \text{ es paralelo a } B$$
$$\text{Entonces}$$
$$\tau_{bc} = 0 \tag{12}$$

- *Segmento cd.* En el segmento cd la corriente de la batería está dirigido hacia dentro de la página. El campo magnético bajo la cara polar apunta radialmente hacia adentro del rotor, por tanto la fuerza sobre el alambre está dada por

$$F_{cd} = i(lxB) \tag{13}$$
$$F_{cd} = ilB \quad \text{Tangente a la dirección del movimiento}$$

- El par sobre el rotor causado por esta fuerza es

$$\tau_{cd} = rFsen\,\theta$$
$$\tau_{cd} = r(ilB)sen\,90° \tag{14}$$
$$\tau_{cd} = rilB \quad \text{en sentido contrario a las manecillas del reloj}$$

- *Segmento da.* En el segmento da la corriente de la batería fluye desde el extremo superior izquierdo hacia el inferior derecho en el dibujo. La fuerza inducida sobre el alambre está dada por

$$F_{da} = i(lxB)$$
$$F_{da} = 0 \text{ puesto que } l \text{ es paralelo a } B \tag{15}$$
$$\text{Entonces}$$
$$\tau_{da} = 0 \tag{16}$$

El par inducido resultante total en la espira está dado por

$$\tau_{ind} = \tau_{ab} + \tau_{bc} + \tau_{cd} + \tau_{da}$$
$$\tau_{ind} = \begin{cases} 2rilB \text{ bajo las caras polares} \\ 0 \quad \text{por fuera de las caras polares} \end{cases} \tag{17}$$

Realizando las sustituciones convenientes la ecuación 17 puede ser expresada como [1].

$$\tau_{ind} = \begin{cases} \frac{2}{\pi}\phi\, i & bajo\ las\ caras\ polares \\ 0 & por\ fuera\ de\ las\ caras\ polares \end{cases} \tag{18}$$

Entonces, el par producido en la máquina es el producto del flujo y la corriente en ella multiplicada por una cantidad que representa la construcción mecánica de la máquina (el porcentaje del rotor cubierto por las caras polares). En general, el par en cualquier máquina real dependerá de los mismos tres factores:

- El flujo en la máquina.
- La corriente en la máquina.
- Una constante que representa la construcción de la máquina.

1.1.4 ECUACIONES BÁSICAS DEL MOTOR DE C.C.

La inyección de corriente continua a través de los bobinados de campo establece una corriente de excitación, la cual incrementa el flujo del campo en el entrehierro del motor.

En términos de valores instantáneos se tiene:

$$V_f = i_f R_f + L_f \frac{di_f}{dt} \tag{19}$$
$$\phi_f = f(i_f)$$

Si el motor opera en la parte lineal de la curva de magnetización, como se puede ver en la figura 6, hay una relación lineal entre el flujo del campo en estado estable y la corriente de campo en estado estable.

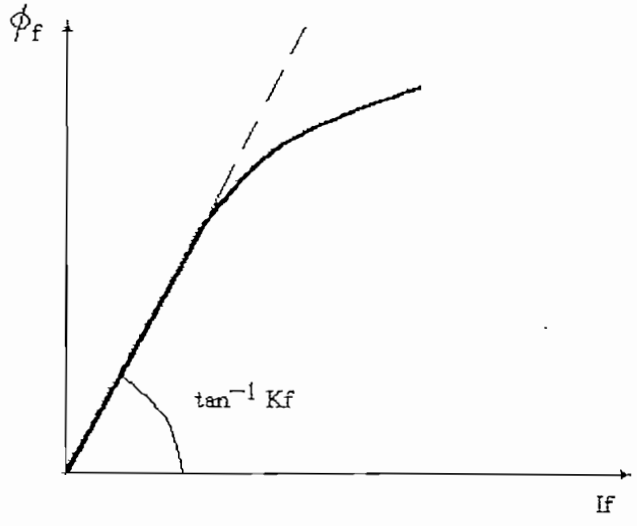


Figura 6.

Curva de magnetización de la máquina de C.C.

$$\phi_f = K_f I_f \tag{20}$$

En condiciones de estado estable no hay variación de la corriente de campo. Además no se induce fuerza electromotriz en los bobinados de campo debido al efecto del circuito de armadura, entonces:

$$V_f = I_f R_f = \frac{R_f}{K_f}\phi_f \tag{21}$$

El circuito de armadura y el circuito de campo pueden ser interconectados en las siguientes formas básicas (figura 7).

Cada conexión resulta en un funcionamiento particular del motor y cada una es relacionada a aplicaciones particulares de carga.

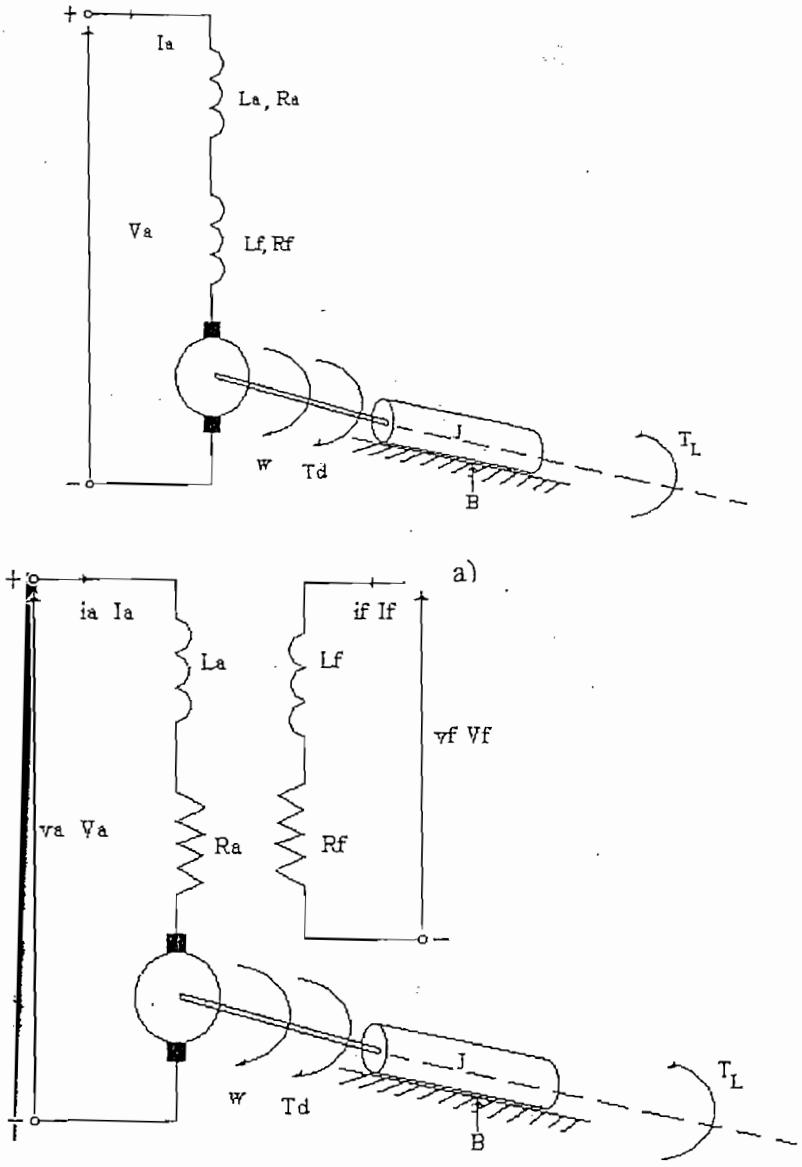


Figura 7.

a) Conexión serie, b) Conexión de excitación independiente

En la conexión serie, las corrientes de campo y armadura son las mismas y las características del motor (torque – velocidad) tienen la forma hiperbólica de la figura 8.

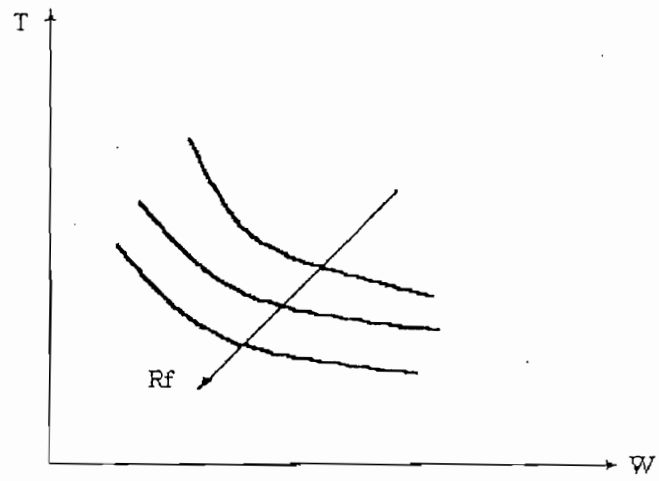


Figura 8.

Curva torque – velocidad en conexión en serie

Las características torque – velocidad en conexión de excitación independiente están dadas por la figura 9.

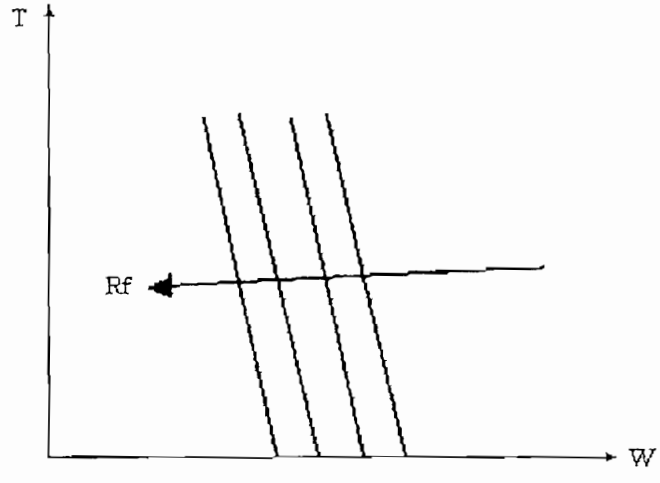


Figura 9.

Curva torque – velocidad en conexión de excitación independiente.

El torque instantáneo desarrollado por el motor está dado por:

$$T_{(i)} = K_T \phi \, I_a \tag{22}$$

En términos de estado estable, el valor medio del torque está dado por:

$$T = K_T \phi \, I_a \tag{23}$$

La rotación de los conductores de la armadura en el flujo del campo causa una f.e.m. (fuerza electromotriz) que es inducida en el circuito de armadura con polaridad opuesta al flujo de la corriente de armadura. Esta f.e.m. inducida es reconocida como f.c.e.m. (fuerza contraelectromotriz).

El valor instantáneo de f.c.e.m. está dado por:

$$e_a = K_v \phi \, w \tag{24}$$

donde w es la velocidad instantánea.

En operación en estado estable se tiene:

$$E_a = K_v \phi \, W \tag{25}$$

En el sistema internacional K_v, K_T tienen los mismos valores

$$K_v = \left[\frac{N.m}{Weber.A}\right] \qquad K_T = \left[\frac{V.\text{sec}}{Weber.rad}\right]$$

La potencia interna desarrollada por el motor es

$$P = E_a I_a \tag{26}$$

El valor instantáneo del voltaje de armadura está dado por:

$$V_a = i_a R_a + L_a \frac{di_a}{dt} + e_a \tag{27}$$

(Para un motor de excitación independiente)

$$V_a = i_a\left(R_a + R_f\right) + \left(L_a + L_f\right)\frac{di_a}{dt} + e_a \qquad (28)$$

(Para un motor en serie)

En estado estable, el efecto de las inductancias son pequeños, luego:

$$V_a = I_a R_a + E_a \qquad (motor\ excit.\ indep.)$$
$$V_a = I_a\left(R_a + R_f\right) + E_a \qquad (motor\ serie)$$

1.1.5 MÁQUINA DC CON CAMPO CONSTANTE O MAGNETO PERMANENTE

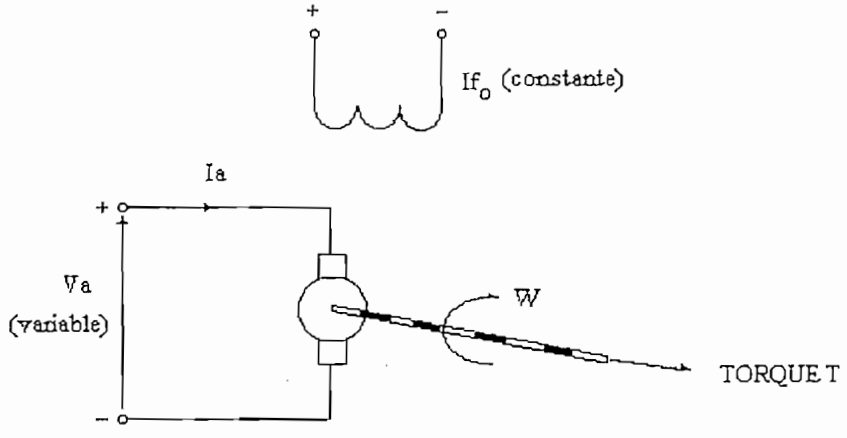


Figura 10.

Motor DC con campo constante o magneto permanente.

El flujo ϕ es proporcional a I_f (sin tomar en cuenta la saturación magnética)

$$E_a = K_v W \frac{I_f}{I_{f0}} \qquad (29)$$

K_v = constante de la fuerza contraelectromotriz funcionando a flujo nominal, esto es cuando $I_f = I_{f0}$

$$K_v = \left[\frac{V}{rad.seg^{-1}}\right]$$

$$T_a \alpha I_a \; y \; \phi$$

$$T = K_T i_a \frac{I_f}{I_{f0}} \tag{30}$$

$$K_T = cons\tan te\ de\ torque$$

$$K_T = \left[\frac{N.m}{A}\right]$$

Por tanto para la máquina de DC con flujo nominal las ecuaciones son:

$$V_a = i_a R_a + K_v w + L_a \frac{di_a}{dt} \tag{31}$$

$$T = K_T i_a \tag{32}$$

La característica de velocidad en función del torque está dada por la figura 11.

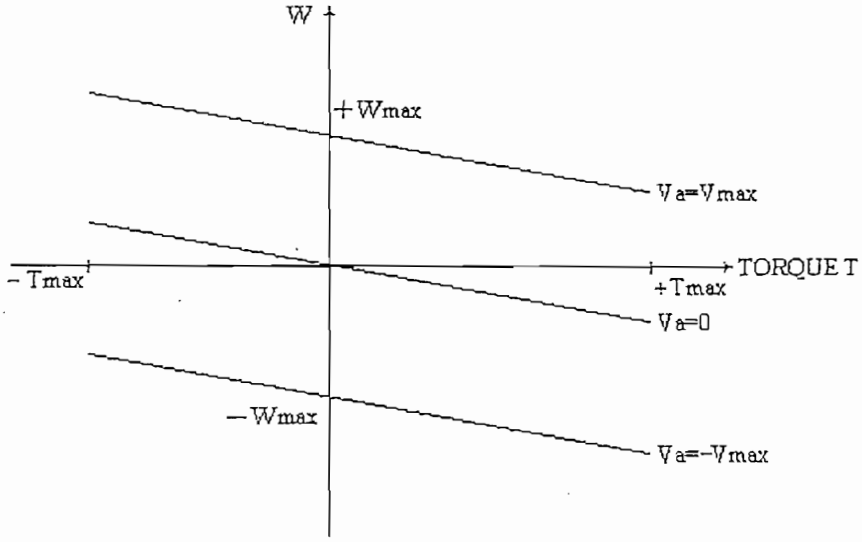


Figura 11.

Curva velocidad – torque de la máquina DC con campo constante.

1.1.6 MOTOR DC CON CAMPO VARIABLE, VOLTAJE DE ARMADURA CONSTANTE

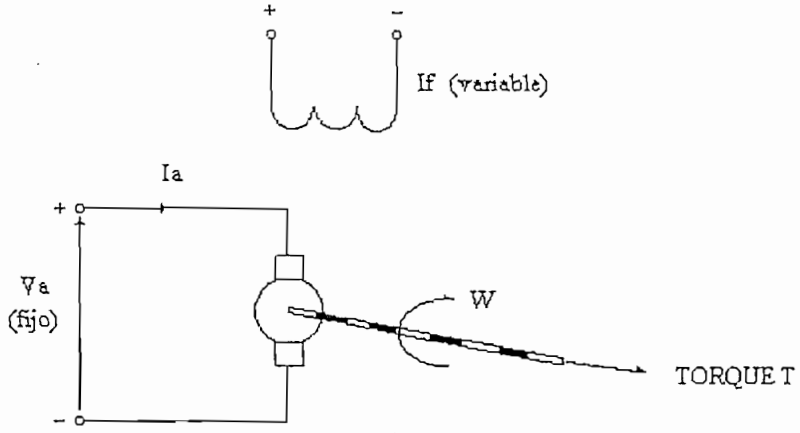


Figura 12.

Motor DC con campo variable y voltaje de armadura constante

Ecuaciones

$$V_a = I_a R_a + K_v w \frac{I_f}{I_{f0}} \tag{33}$$

$$T = K_T I_a \frac{I_f}{I_{f0}} \tag{34}$$

Características

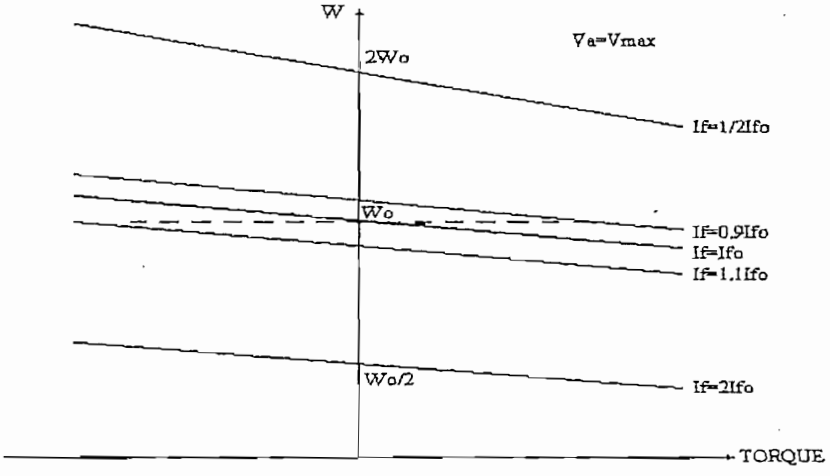


Figura 13.

Curva velocidad – torque del motor DC con campo variable, voltaje de armadura constante.

1.1.7 MOTOR DC SERIE

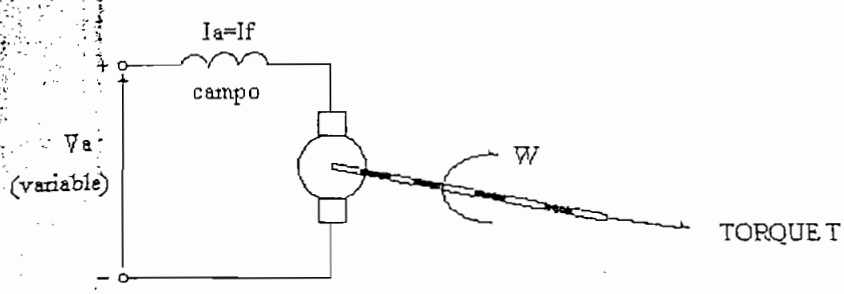


Figura 14.

Motor DC serie

Ecuaciones

$$V_a = I_a K_T + K_v w \frac{I_a}{I_{a0}} \tag{35}$$

I_{a0} = *condición de máximo flujo*

$$T = I_a K_T \left(\frac{I_a}{I_{a0}} \right) \rightarrow T \alpha \, I_a^{\,2} \tag{36}$$

Características

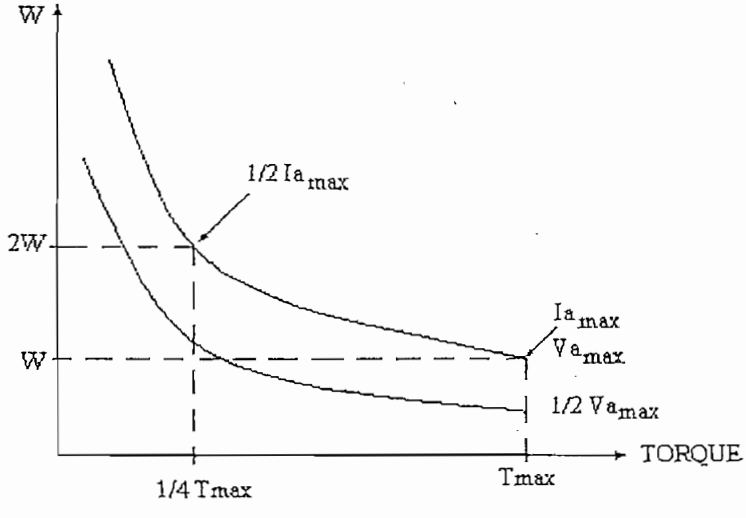


Figura 15.

Curva velocidad – torque del motor DC serie

1.1.8 MOTOR DC CON V_a VARIABLE Y CAMPO VARIABLE

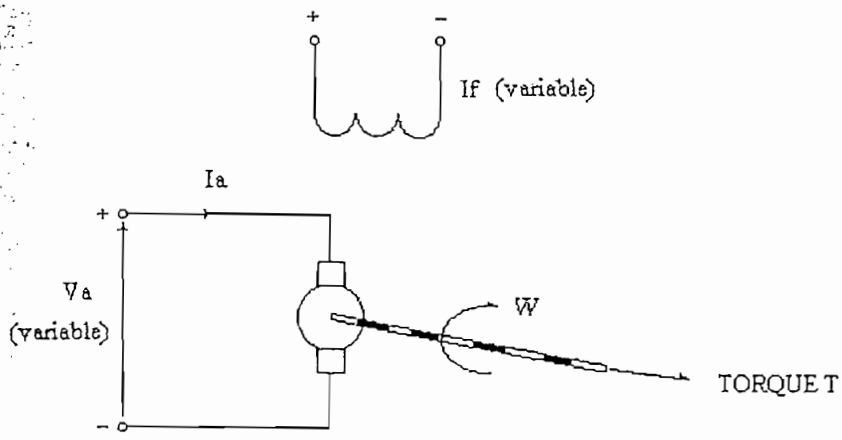


Figura 16.

Motor DC con V_a variable y campo variable

Ecuaciones

$$V_a = I_a R_a + K_v w \left(\frac{I_f}{I_{f0}} \right) \tag{37}$$

$$T = I_a K_T \left(\frac{I_f}{I_{f0}} \right) \tag{38}$$

Características

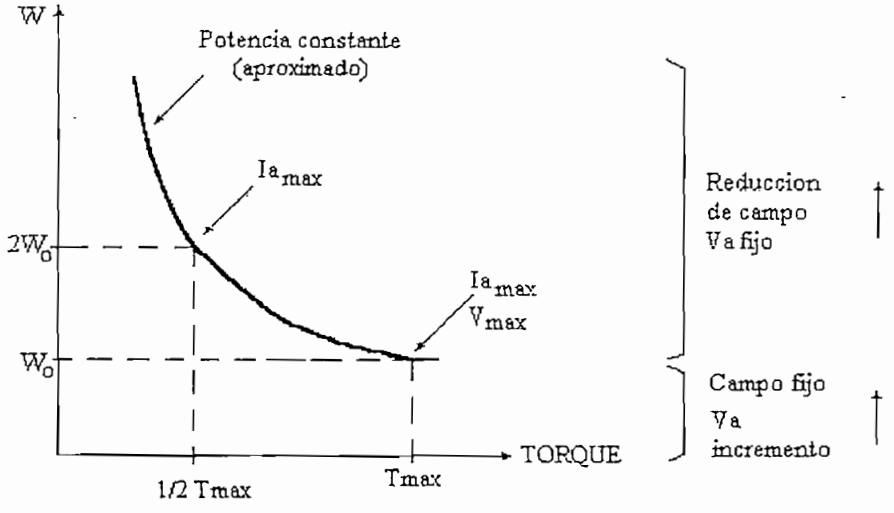


Figura 17.

Curva velocidad – torque motor DC con V_a variable y campo variable

1.2 MODELO DEL MOTOR DC REGULADO POR VOLTAJE DE ARMADURA

En el presente trabajo se centrará al estudio de la máquina de corriente continua regulada por voltaje de armadura para lo cuál se toma como modelo el sistema MCSL – 100 (*Motomatic Control System Laboratory*). El sistema MCSL – 100 es un sistema de control, con fines de laboratorio por su flexibilidad, dicho sistema esta compuesto por:

- Chasis o equipo electrónico
- Riel (para el montaje de los componentes electromecánicos)
- Sistema motor – generador de DC
- Unidad de reducción de velocidad
- Rueda para la aplicación de perturbaciones de carga
- Potenciómetro indicador de posición
- Acopladores mecánicos
- Chasis de medición con tacómetro, voltímetro y amperímetro.

1.2.1 EQUIPO ELECTRÓNICO

Este equipo se encuentra montado en un chasis de aluminio de 10" x 16" como indica la figura 18, en el equipo electrónico se encuentra la fuente de poder, el amplificador operacional, el amplificador de potencia, el medidor de corriente de armadura, el generador de entrada paso, terminales para los instrumentos de medición, terminales eléctricos para los componentes mecánicos, terminales de entrada al sistema y los circuitos de protección e indicadores.

La ubicación funcional de cada uno de los componentes se presenta en el diagrama de la figura 19.

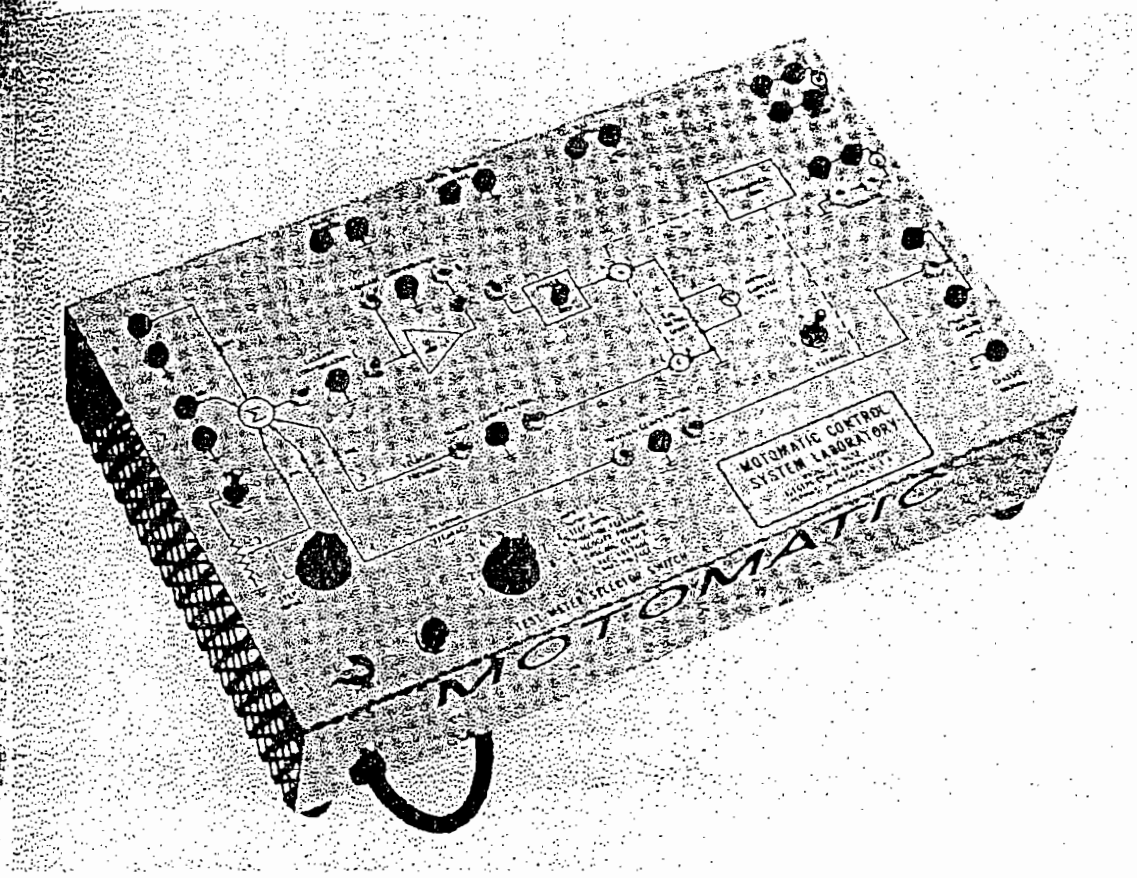


Figura 18.

Chasis electrónico

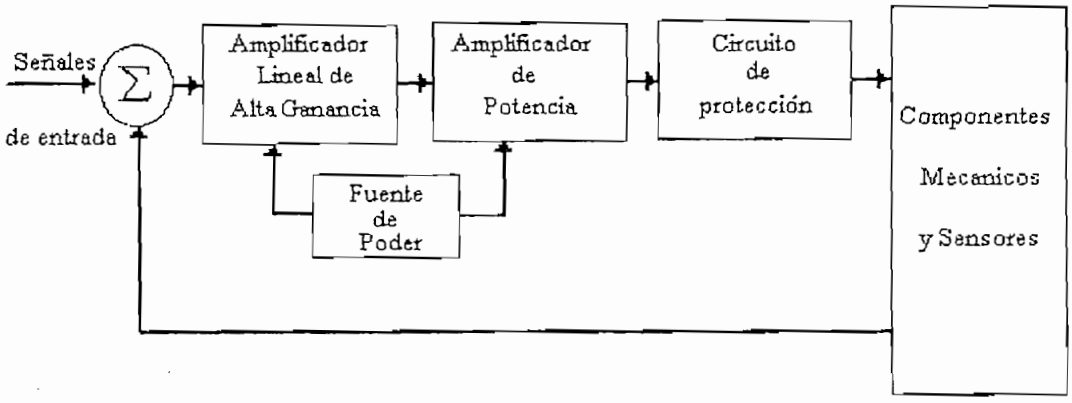


Figura 19.

Diagrama de bloques funcional del Sistema de Control de Laboratorio MCSL – 100

En la figura 20 se muestra un diagrama de bloques un poco más detallado del Motomatic [3].

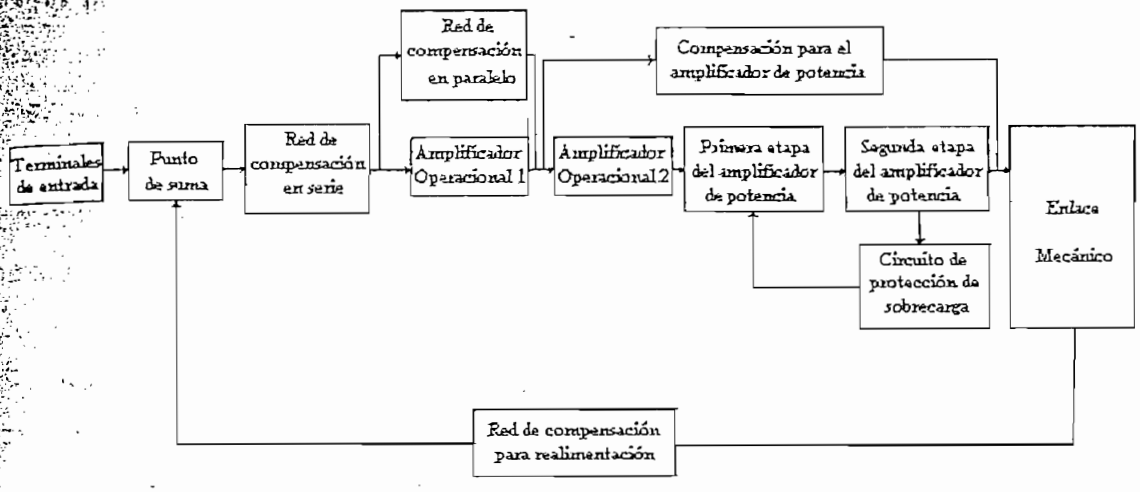


Figura 20.

Diagrama de bloques detallado del Sistema Motomatic

El bloque de suma es una red sumadora como se ve en la figura 21.

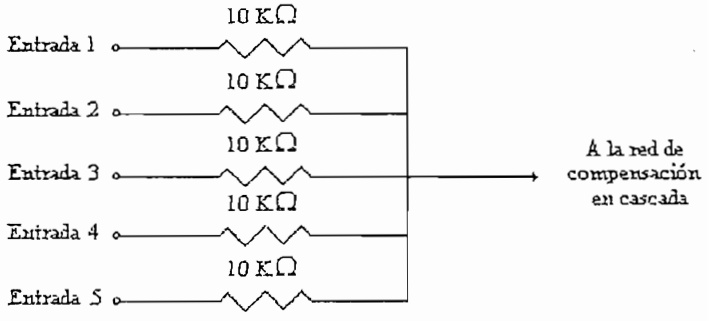


Figura 21.

Red sumadora

El amplificador operacional 1 es de propósito general, mientras que el segundo amplificador operacional forma la primera etapa de un amplificador de potencia DC compensado de tres etapas.

El amplificador de potencia usa transistores de potencia de Silicio en configuración complementaria, permitiendo así la inversión de giro del motor y tiene una ganancia de voltaje de aproximadamente 5 V. La compensación del amplificador de potencia se realiza mediante un circuito RC para producir esencialmente un amplificador lineal con ancho de banda de 40 KHz [3].

La corriente de salida está limitada a cerca de 2.25 A con 100 ms de duración y permitiendo picos de carga mayores por cortos intervalos de tiempo.

Un circuito en base a triacs cambia el estado del indicador de sobrecarga siempre que la corriente de carga exceda 1.5 A.

Existen tres fuentes de poder en la unidad, una primera fuente de alta corriente de $\pm 30VDC$ requerida por los amplificadores de potencia. Una segunda fuente de $\pm 21VDC$ usado como voltaje de referencia para la entrada y el sensor de posición. Para los amplificadores operacionales se utiliza una fuente de $\pm 15VDC$. El rango de requerimiento de potencia para el laboratorio es de 105 a 125 VAC y un consumo cerca de 75 W como máximo.

1.2.2 EL EQUIPO ELECTROMECÁNICO

Los componentes electromecánicos se ubican en rieles para un mejor aprovechamiento del espacio físico y para evitar desalineamientos de los componentes del sistema.

El conjunto motor – generador (figura 22) está montado sobre un eje común; ambos son de imán permanente lo que significa que el flujo producido es constante. Se tiene también que el torque es proporcional a la corriente de armadura y la fuerza electromotriz generada en la armadura del motor es proporcional a la velocidad de la armadura.

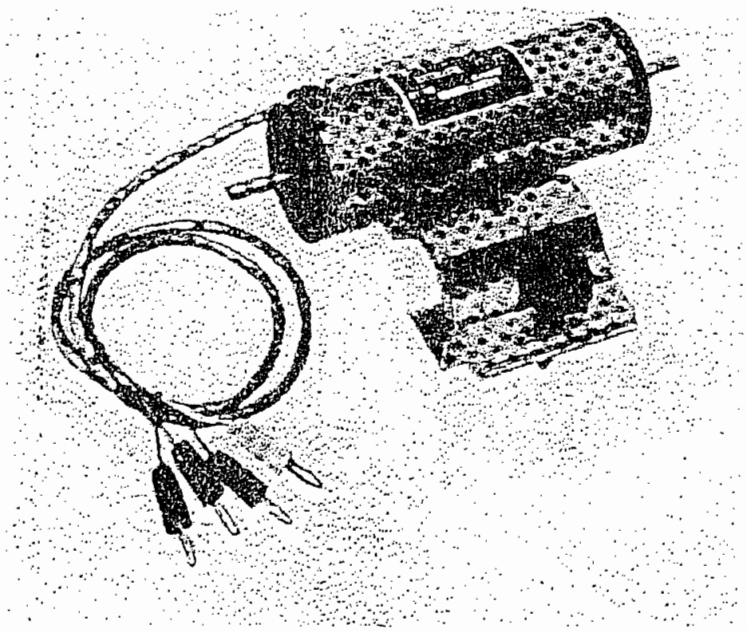
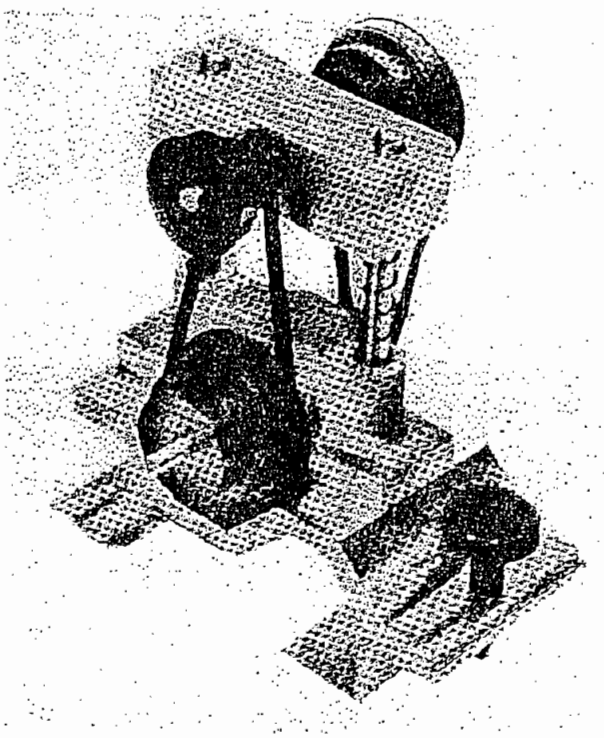


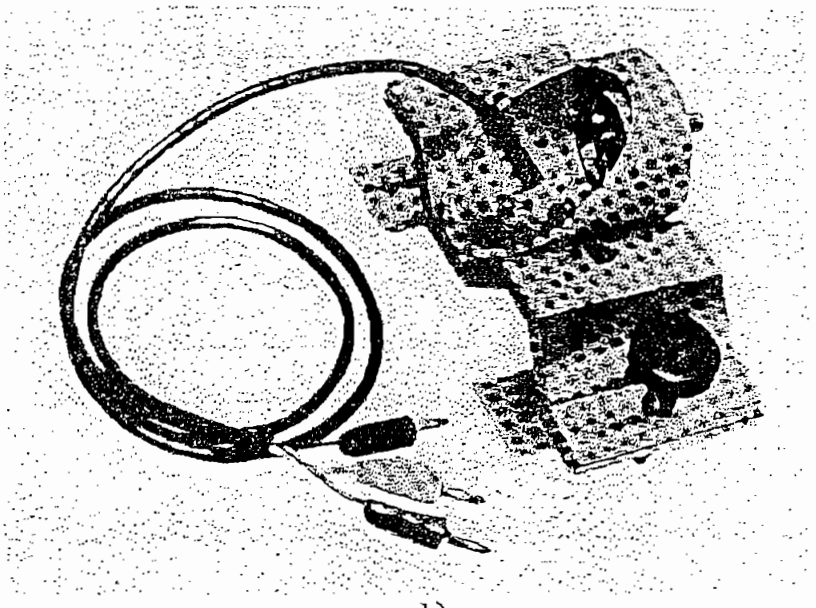
Figura 22.

Grupo motor – generador

En las figuras 23 a y 23 b se puede ver los dispositivos de reducción de velocidad y el sensor de posición potenciométrico que permite realimentar la señal de posición, respectivamente.



a)



b)

Figura 23.

a) Unidad de reducción de velocidad, b) Sensor de posición

La figura 24 muestra las diferentes posiciones de las bandas de transmisión de los reductores de velocidad, los cuales son requeridos para obtener proporciones deseadas de velocidad (mostradas en la tabla 1).

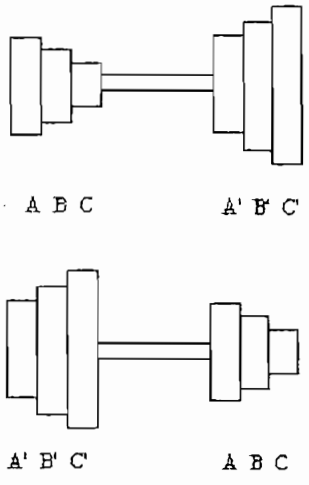


Figura 24.

Bandas de transmisión de las bandas de los reductores de velocidad

RELACION DE VELOCIDADES	POSICIÓN DE LA BANDA
1:1	AA'
$\sqrt{3}:1$	BA', AB'
3:1	CA', BB', AC'
$3\sqrt{3}:1$	CB', BC'
9:1	CC

Tabla 1.

Relaciones de velocidad para varias posiciones de la banda

1.2.3 CHASIS DE INSTRUMENTOS

El chasis de instrumentos se indica en la figura 25. Los tres medidores son usados para medir:

- Velocidad del motor
- Corriente de armadura del motor
- Un voltaje a seleccionar

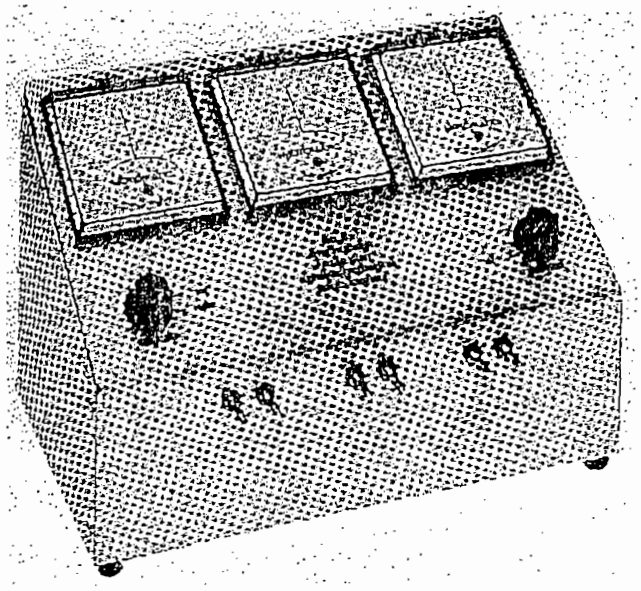


Figura 25.

Chasis de instrumentos

1.2.4 MODELACIÓN DEL SISTEMA

Desde el punto de vista del control el sistema Motomatic se lo puede representar de acuerdo al siguiente diagrama de bloques (figura 26).

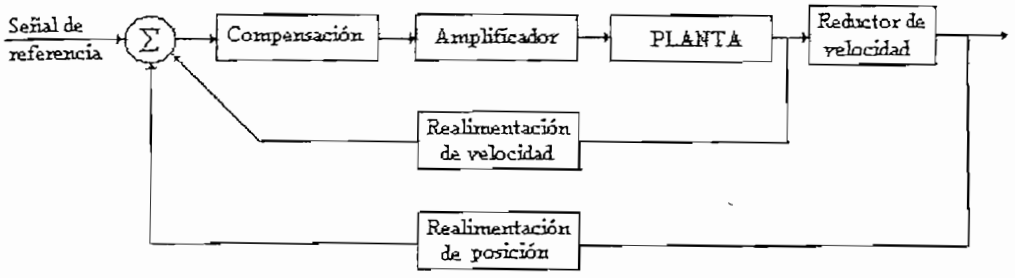


Figura 26.

Diagrama de bloques del sistema Motomatic

A continuación se detalla la función de transferencia de cada uno de los bloques mostrados en la figura anterior.

La planta: Está constituida por un motor DC. El circuito de armadura se lo puede representar mediante el siguiente diagrama (figura 27).

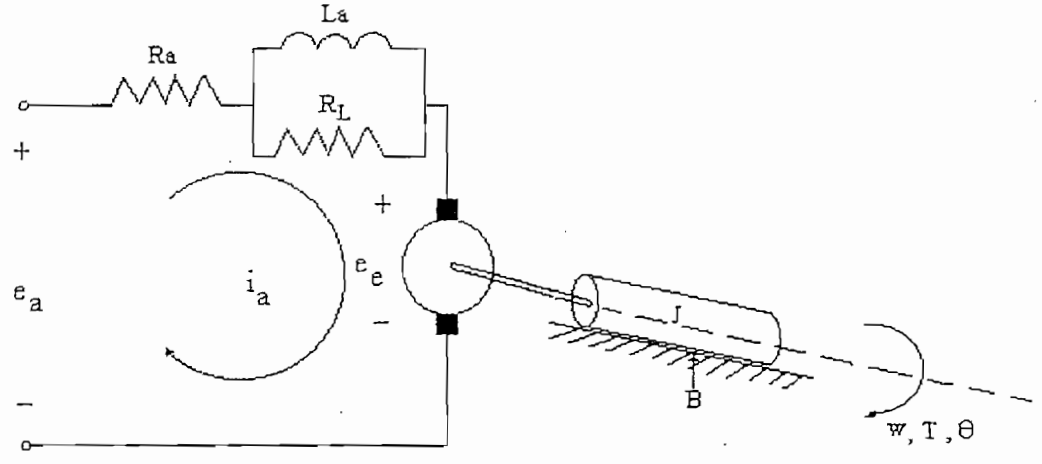


Figura 27.

Circuito de armadura de un motor de DC

Las ecuaciones que gobiernan el comportamiento del motor se escriben a continuación.

Ecuación eléctrica.- En la figura 27 R_L representa las pérdidas del circuito magnético, sin embargo en la operación del motor es insignificante por ser más grande que la resistencia de armadura Ra.

Las ecuaciones que rigen al circuito son:

$$Ec.\ Eléctricas \begin{cases} e_a = R_a i_a + L_a \dfrac{di_a}{dt} + e_e \\ e_e = K_v w \end{cases}$$

Donde e_e es la fuerza contra electromotriz inducida f.c.e.m., que aparece en los terminales de armadura como un voltaje generado internamente y es proporcional a la velocidad del motor w.

Ecuación dinámica.- Si consideramos que el campo magnético del motor es constante, el torque producido es proporcional a la corriente de armadura. Si asumimos que el motor está acoplado a la carga la relación entre torque y velocidad es:

$$T = (J_m + J_L)\frac{dw}{dt} + Bw + T_f + T_L$$

Conocida como la ecuación dinámica del motor, en donde J_m es el momento de inercia del motor, J_L momento de inercia de la carga, T_f constante de fricción del torque, T_L torque de carga y B coeficiente de rozamiento viscoso. Despreciando el efecto de T_f y trabajando en vacio (T_L=0, J=J_m) se tiene:

$$Ec.\ Mecánicas \begin{cases} T = K_T i_a \\ T = J\frac{dw}{dt} + Bw \end{cases}$$

Donde: [6]

$e_a = voltaje\ de\ armadura$

$R_a = resistencia\ de\ armadura$ 2.8Ω

$L_a = induc\tan cia\ de\ armadura$ $2.1mH$

$i_a = corriente\ de\ armadura$

$e_e = fuerza\ contraelectromotriz$

$J = representa\ el\ momento\ de\ inercia\ de\ la\ armadura$ $36.58*10^{-6}\ Kg-m^2$

$B = coeficiente\ de\ fricción\ vis\cos a\ de\ la\ armadura$ $0.0000153\frac{N-m}{rad/s}$

$T = torque\ generado\ por\ el\ motor$

$\theta = representa\ la\ posición\ angular$

$w = velocidad\ angular$

$K_T = cons\tan te\ de\ torque$ $0.0375\frac{N-m}{A}$

$K_v = cons\tan te\ de.\ voltaje$ $0.0382\frac{V}{rad/s}$

Los modelos matemáticos pueden adoptar formas distintas. Dependiendo del sistema que se trate y de las circunstancias específicas, un modelo matemático puede ser más conveniente que otro, por tal razón se presenta dos alternativas: modelo a diagrama de bloque y modelo a diagrama de flujo.

1.2.4.1 Modelo a diagrama de bloques

Un diagrama de bloques de un sistema es una representación gráfica de las funciones que lleva a cabo cada componente y el flujo de señales. Tal diagrama muestra las relaciones existentes entre los diversos componentes.

Utilizando la transformada de Laplace en las ecuaciones eléctricas y mecánicas antes mencionadas se obtiene:

$$E_a = R_a I_a + L_a s I_a + E_e \tag{39}$$

$$E_e = K_v \tag{40}$$

$$T = K_T I_a \tag{41}$$

$$T = JsW + BW \tag{42}$$

Si de la Ec. (39) se despeja la corriente de armadura I_a se obtendrá:

$$I_a = \frac{E_a - E_e}{R_a + sL_a} \tag{43}$$

A partir de las ecuaciones (39), (40), (41), (42) y (43) se obtiene el diagrama de bloque (figura28) de la planta [4]:

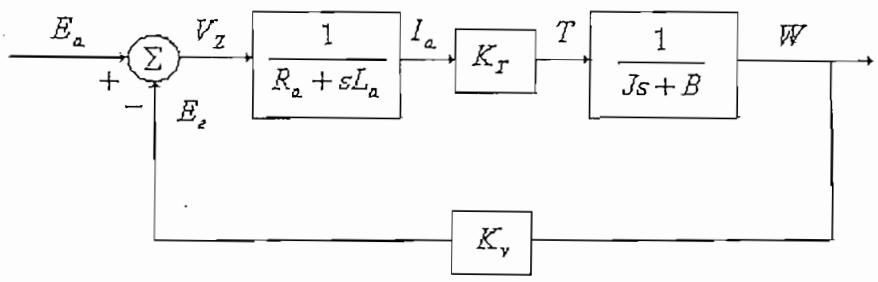


Figura 28.

Diagrama de bloque de la planta

Para el sistema que aparece en la figura 28 se puede obtener la función de transferencia en lazo cerrado, es decir la salida W y la entrada E_a se relacionan del modo siguiente:

$$\frac{W(s)}{E_a(s)} = \frac{K_T}{(K_T K_v + R_a B) + s(R_a J + BL_a) + s^2 L_a J}$$

La función de transferencia en lazo cerrado se obtiene a partir del diagrama de bloques [4] aplicando la formula:

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

Donde:

$Y(s) =$ *representa la señal de entrada*

$R(s) =$ *señal de salida*

$G(s) =$ *función de transferencia en lazo abierto*

$H(s) =$ *realimentación*

1.2.4.2 Modelo a diagrama de flujo

Se construye el modelo a diagrama de flujo despejando la mayor derivada tanto de las ecuaciones mecánicas como eléctricas, para posteriormente utilizar la regla de MASON [5]:

De (39) se tiene:

$$\frac{di_a}{dt} = -\frac{R_a}{L_a} i_a - \frac{e_e}{L_a} + \frac{1}{L_a} e_a$$

De (42):

$$\frac{dw}{dt} = -\frac{Bw}{J} + \frac{1}{J} T$$

Entonces se construye el diagrama de flujo (figura 29).

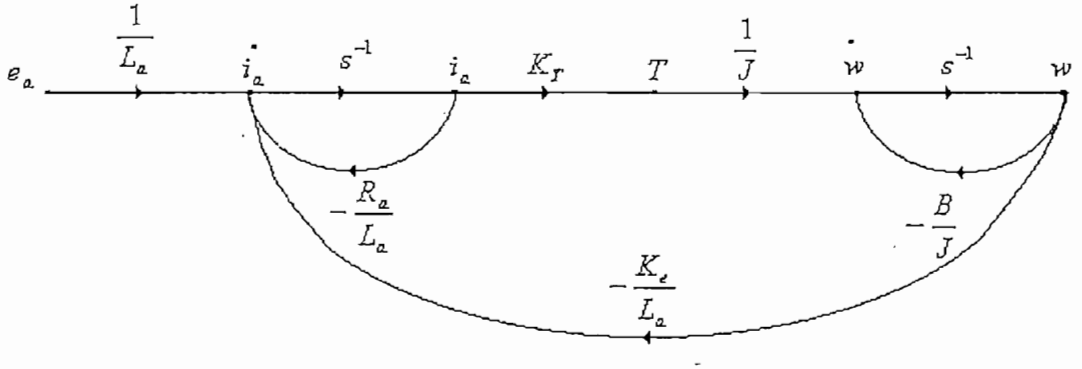


Figura 29.

Diagrama de flujo de la planta

Con el diagrama de flujo se puede obtener la función de transferencia de la planta aplicando:

$$\frac{Y(s)}{R(s)} = \frac{\sum P_i \Delta_i}{\Delta}$$

Donde:

$Y(s) =$ *señal de entrada*

$R(s) =$ *señal de salida*

$\Delta = 1 - \sum l_i + \sum l_i l_j - \sum l_i l_j l_k + \ldots$

$l_i =$ *ganancia de lazos individuales*

$l_i l_j =$ *productos binarios de ganacia de lazos que no se tocan* *(no tienen en común ni nodos ni ramas)*

$\Delta_i = \Delta$ *menos ter*minos *que contienen lazos que tocan a* P_i

$P_i =$ *iava trayectoria directa*

De lo que se tiene que la relación entre W y E_a es:

$$\frac{W(s)}{E_a(s)} = \frac{K_T}{(K_T K_v + R_a B) + s(R_a J + B L_a) + s^2 L_a J}$$

1.2.4.3 Modelo a variables de estado

Es importante conocer algunos conceptos básicos para determinar el modelo de un sistema a variables de estado, como por ejemplo que es estado, variable de estado, espacio de estado.

Estado. Es un conjunto de valores que caracteriza el comportamiento dinámico de un sistema.

Variables de Estado. Son un conjunto de funciones del tiempo linealmente independientes cuyo conocimiento junto con la señal de entrada y la condición inicial permiten determinar el estado futuro y las salidas.

Se tiene una variable de estado a la salida de un integrador como se muestra en la figura 30.

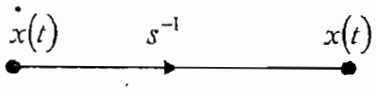


Figura 30.

Variable de estado

Espacio de Estado. Es un espacio n dimensional cuyas bases son las variables de estado.

Descripción interna

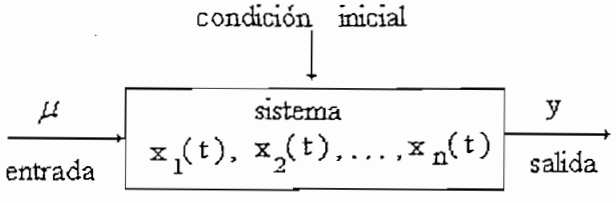


Figura 31.

Descripción a variables de estado de un sistema

Del modelo a diagrama de flujo (figura 29) se puede construir el modelo a variables de estado, a partir del sistema físico [4]:

$$\begin{cases} \dot{x} = A\underline{x} + B\mu & \textit{Ecuación de estado} \\ y = C\underline{x} & \textit{Ecuación de salida} \end{cases}$$

Entonces tenemos:

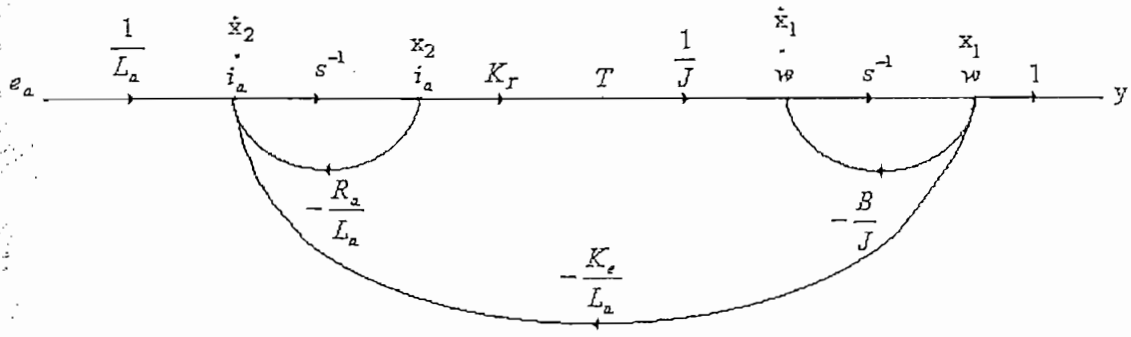


Figura 32.

Diagrama de flujo (estado)

De la figura 32 se forman las siguientes ecuaciones:

$$\dot{x}_1 = -\frac{B}{J}x_1 + \frac{K_T}{J}x_2$$

$$\dot{x}_2 = -\frac{K_e}{L_a}x_1 - \frac{R_a}{L_a}x_2 + \frac{1}{L_a}$$

$$y = x_1$$

y en forma matricial se obtiene:

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \underbrace{\begin{bmatrix} -\frac{B}{J} & \frac{K_T}{J} \\ -\frac{K_e}{L_a} & -\frac{R_a}{L_a} \end{bmatrix}}_{A} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \underbrace{\begin{bmatrix} 0 \\ \frac{1}{L_a} \end{bmatrix}}_{B} \mu$$

$$y = \underbrace{\begin{bmatrix} 1 & 0 \end{bmatrix}}_{C} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

1.2.5 FUNCIONES DE TRANSFERENCIA DE LOS COMPONENTES DE SERVOS DC.

1.2.5.1 Unidad de reducción de velocidad

La función de transferencia se la puede determinar de la siguiente relación:

$$\frac{W(s)_{salida}}{W(s)_{entrada}} = N \quad y \quad \frac{\theta(s)_{salida}}{\theta(s)_{entrada}} = N$$

En donde el valor de N esta dado en la tabla 1.

1.2.5.2 Tacómetro

El tacómetro tiene el bobinado acoplado al mismo eje de armadura del motor, se puede considerar que el flujo de campo es constante lo cuál indica que el voltaje es proporcional a la velocidad angular, es decir se tiene:

$$E_e = K_v w$$

Cuya función de transferencia es:

$$\frac{E_e(s)}{W(s)} = K_v$$

El valor de la constante de voltaje del generador K_v se la obtiene del gráfico voltaje generado versus velocidad angular [6].

$$K_v = 0.0154\left[\frac{v}{rpm}\right] \quad o \quad K_v = 0.148\left[\frac{v}{rad/s}\right]$$

1.2.5.3 Amplificador

Se dispone de un servoamplificador es decir se tiene un amplificador operacional con ganancia alta, al cual se acopla un amplificador de potencia con ganancia baja (figura 33).

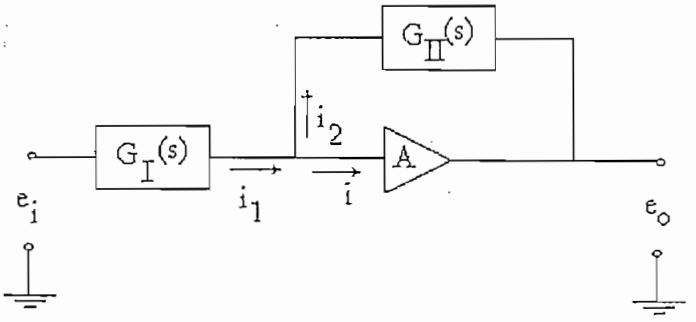


Figura 33.

Diagrama de un amplificador

Este amplificador d.c. de alta ganancia tiene la posibilidad de compensación en cascada por medio de $G_I(s)$ y compensación en realimentación por medio de $G_{II}(s)$.

1.2.5.4 Potenciómetro

Se utiliza sensores de posición como se indica en la figura 34 de 5 o 10 KΩ.

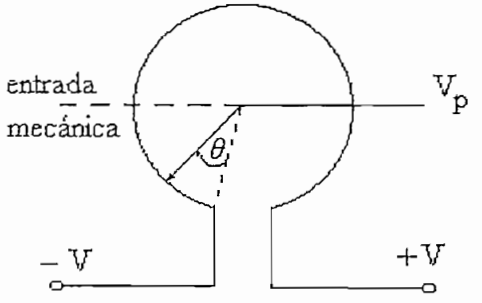


Figura 34.

Potenciómetro de posición de simple giro

Este potenciómetro es lineal, por tanto la señal de salida V_p (voltaje de salida) es directamente proporcional a la entrada angular de posición mecánica θ, por consiguiente tenemos:

$$V_p = K_p\theta \qquad 0° < \theta < 340°$$

Como se puede ver existe una zona muerta de aproximadamente 20°. Es necesario tener cuidado al pasar por esta zona muerta ya que el sistema se comporta inadecuadamente porque la señal producida V_p cambia de su máximo a su mínimo valor. Tomando la transformada de Laplace tenemos:

$$\frac{V_p(s)}{\theta(s)} = K_p$$

El valor de K_p se lo determina experimentalmente [6]:

$$K_p = 0.118\left[\frac{v}{grado}\right] \qquad 0 \qquad K_p = 6.74\left[\frac{v}{rad}\right]$$

A continuación en la figura 35 se muestra el diagrama de bloques del Motomatic con la función de transferencia detallada de cada bloque.

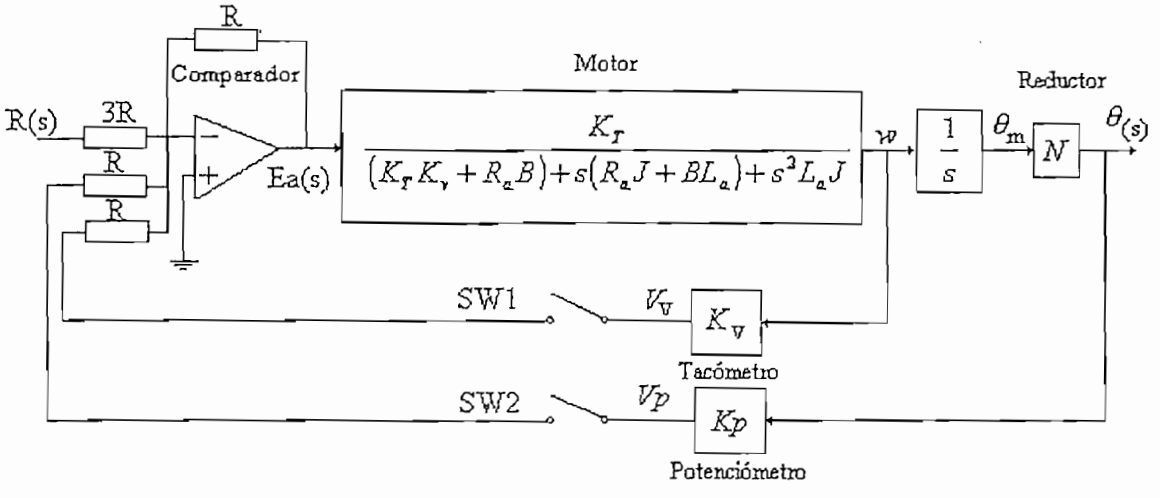


Figura 35.

Diagrama de bloques detallado del Motomatic

2.6 MODELOS PARA EL CONTROL DE VELOCIDAD Y DE POSICIÓN

2.6.1 Modelo para el control de velocidad

De la figura 35 se obtiene el modelo a diagrama de bloques para el control de velocidad, sustituyendo el valor de las constantes.

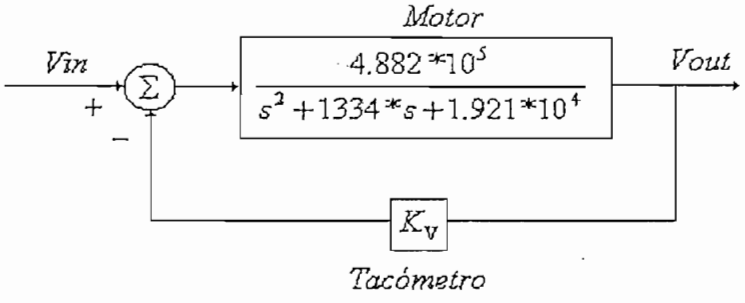


Figura 36.

Diagrama de bloques del Motomatic con realimentación de velocidad

De la figura 36 se tiene que la función de transferencia es:

$$\frac{Vout(s)}{Vin(s)} = \frac{4.882*10^5}{s^2+1334*s+9.145*10^4}$$

A partir de la función de transferencia es posible obtener el diagrama de estado de la siguiente forma:

$$\frac{y}{\mu} = \frac{4.882*10^5}{s^2+1334*s+9.146*10^4}$$

Donde:

$y = Voltaje\ de\ salida$

$\mu = Voltaje\ de\ entrada$

Entonces:

$$y\left(s^2+1334*s+9.145*10^4\right)=\mu\left(4.882*10^5\right)$$
$$\ddot{y}+1334*\dot{y}+9.145*10^4*y=4.882*10^5*\mu$$

Despejando la mayor derivada se tiene:

$$\ddot{y}=-1334*\dot{y}-9.145*10^4*y+4.882*10^5*\mu$$

Con lo cual el diagrama de estado es:

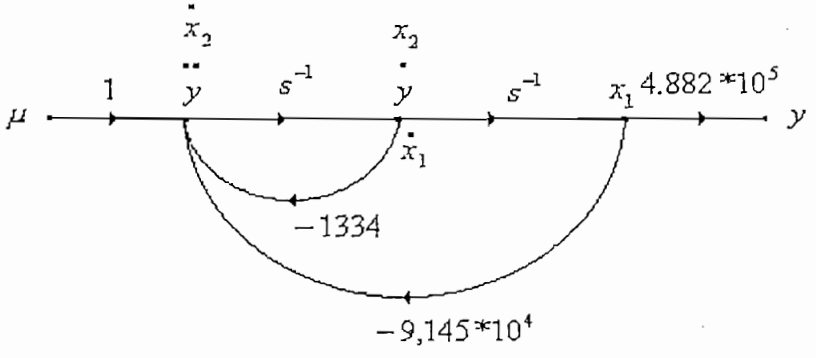


Figura 37.

Diagrama de flujo (estado) de la figura 36

De la figura 37 se tiene las siguientes ecuaciones:

$$\dot{x}_1=x_2$$
$$\dot{x}_2=-9.145*10^4*x_1-1334*x_2+\mu$$
$$y=4.882*10^5*x_1$$

En forma matricial se tiene:

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -9.145*10^4 & -1334 \end{bmatrix} * \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} * \mu$$

$$y = \begin{bmatrix} 1.8*10^5 & 0 \end{bmatrix} * \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

Con el diagrama de bloques de la figura 36 se puede simular en Matlab su respuesta a una entrada paso unitario.

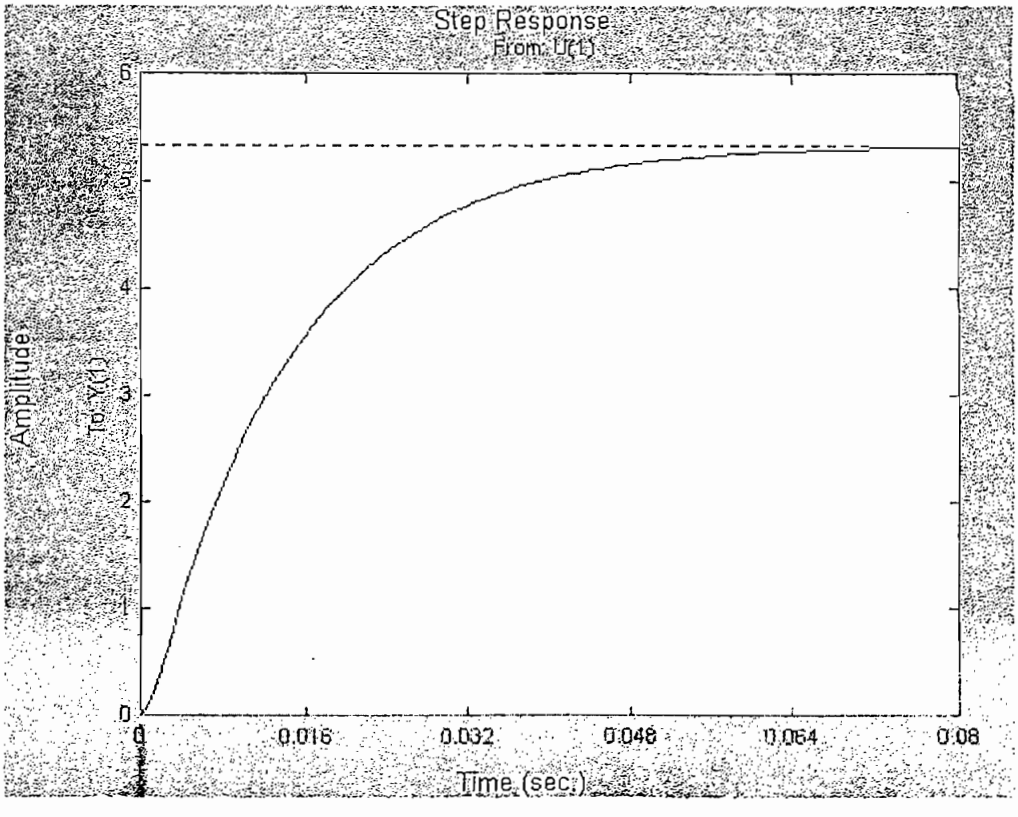


Figura 38.

Respuesta a una entrada paso del modelo para control de velocidad

1.2.6.2 Modelo para el control de posición

Como en el caso anterior se puede construir el modelo a diagrama de bloques del sistema Motomatic con realimentación de posición, reemplazando los valores de las constantes en la figura 35.

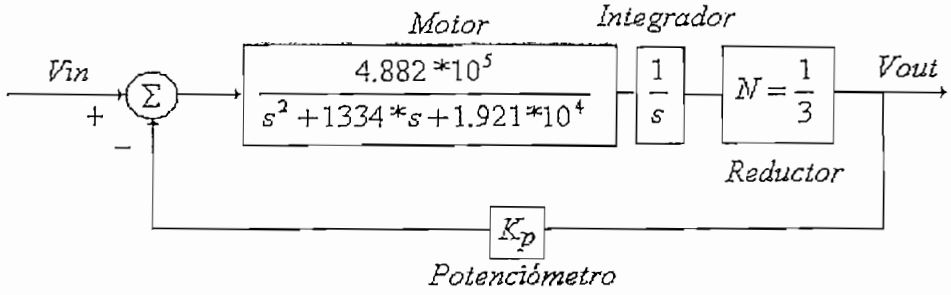


Figura 39.

Diagrama de bloques del Motomatic con realimentación de posición

De la figura 39 se tiene la función de transferencia:

$$\frac{Vout(s)}{Vin(s)} = \frac{1.627*10^5}{s^3+1334*s^2+1.921*10^4*s+1.097*10^6}$$

De la función de transferencia se tiene la ecuación diferencial siguiendo el mismo procedimiento para el caso del sistema Motomatic con realimentación de velocidad:

$$\dddot{y}+1334*\ddot{y}+1.921*10^4*\dot{y}+1.097*10^6*y=1.627*10^5*\mu$$

Y de la ecuación diferencial despejando la mayor derivada, se tiene el siguiente diagrama de estado:

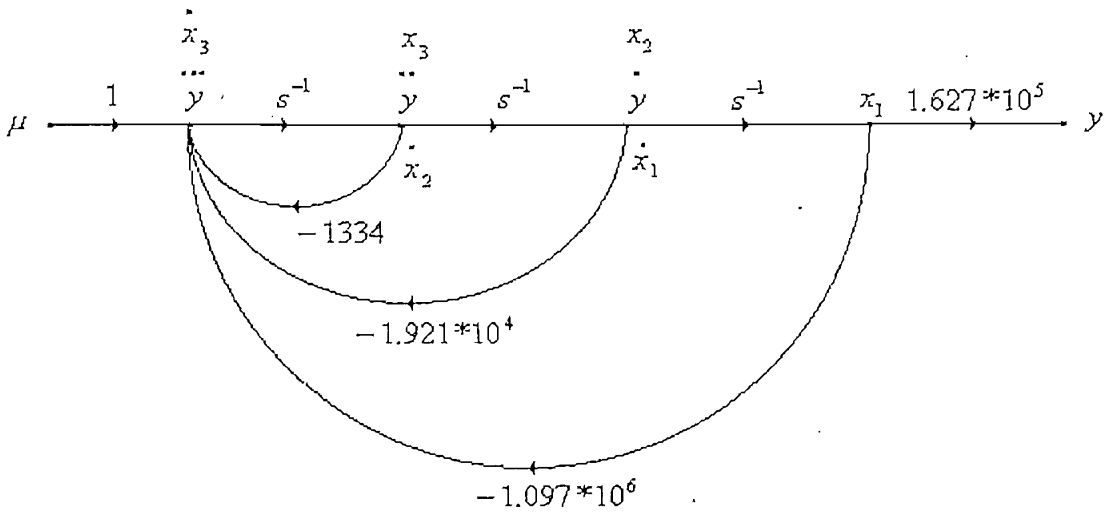


Figura 40.

Diagrama de flujo (estado) de la figura 39

De la figura 40 se generan las siguientes ecuaciones:

$$\dot{x}_1 = x_2$$

$$\dot{x}_2 = x_3$$

$$\dot{x}_3 = -1.097*10^6 * x_1 - 1.921*10^4 * x_2 - 1334 * x_3 + \mu$$

$$y = 1.627*10^5 * x_1$$

En forma matricial se tiene:

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ -1.097*10^6 & -1.921*10^4 & -1334 \end{bmatrix} * \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} * \mu$$

$$y = \begin{bmatrix} 1.627*10^5 & 0 & 0 \end{bmatrix} * \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}$$

n la figura 41 se observa la respuesta del diagrama de bloques de la figura 39 a
ıa entrada paso unitario.

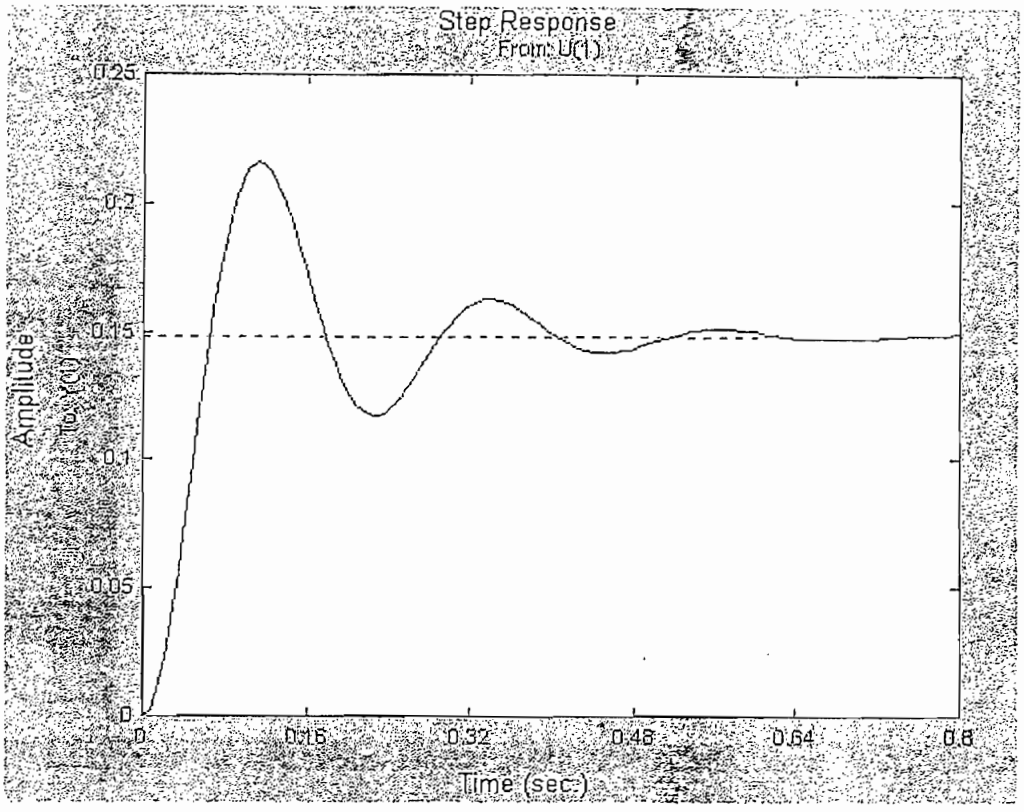


Figura 41.

Respuesta a una entrada paso del modelo para control de posición

n los dos modelos analizados se puede observar que al representarlos en forma
ıatricial, se obtiene la forma canónica controlable.

a modelación es un procedimiento mediante el cual se obtiene un modelo
ıatemático o analítico de una planta, conociendo la estructura y componentes de
sa planta, aplicando conceptos y principios básicos de Ciencia e Ingeniería. La
ıodelación mediante el método directo puede ser a ecuaciones diferenciales,
ınción de transferencia y a variables de estado.

El procedimiento de modelación es muy largo y laborioso como se pudo observar, no funciona cuando hay ruido en exceso y se tiene plantas complejas. Una alternativa es la Identificación que se detalla en el siguiente capítulo.

CAPITULO 2

IDENTIFICACIÓN DEL MOTOMATIC CON REALIMENTACIÓN DE VELOCIDAD Y POSICIÓN

2.1 INTRODUCCIÓN

La ***identificación*** es un procedimiento mediante el cuál se obtiene un modelo analítico o matemático de la planta, mediante mediciones de entrada y salida, basados en ciertos tipos de consideraciones iniciales. Las técnicas de identificación pueden ser: en el tiempo (en base a la curva de reacción), en frecuencia, y paramétrica discreta (mínimos cuadrados).

Un ***modelo*** es la representación de las características más importantes o relevantes de un proceso. Un ***modelo analítico*** puede ser representado a: ecuaciones diferenciales, función de transferencia o variables de estado.

2.1.1 IDENTIFICACIÓN EN EL TIEMPO

Al realizar la identificación de cualquier sistema, lo que se obtiene es el orden de la ecuación diferencial que lo describe y su expresión matemática.

2.1.1.1 Modelo para el control de velocidad

Como se mencionó anteriormente para realizar la identificación es necesario disponer de datos de entrada y salida del sistema; la señal de entrada en este caso es una función paso, que da como salida los datos mostrados en la tabla 2.1, para el sistema conformado por el motomatic y el tacogenerador (figura 2.1) en lazo cerrado.

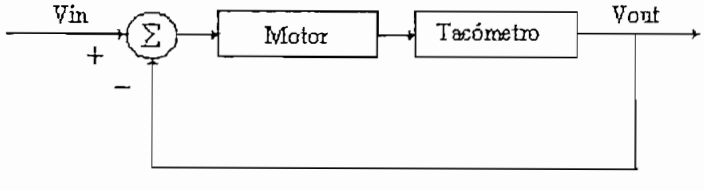


Figura 2.1

Sistema: motor, tacogenerador

Tiempo(s)	V_out(V)
0	0
0,01	0,4
0,02	1,4
0,03	2,6
0,04	3,4
0,05	4
0,06	4,9
0,07	5,4
0,08	5,7
0,09	6
0,1	6,1
0,11	6,2
0,12	6,2
0,13	6,2
0,14	6,2
0,15	6,2

Tabla 2.1

Variación del voltaje de salida en función del tiempo

La tabla 2.1 muestra que la respuesta del sistema es no oscilatoria por lo cuál para identificar el sistema se utiliza el método de determinación de las constantes de tiempo.

La respuesta de un sistema no oscilatorio en el dominio del tiempo es:

$$C(t) = A + k_1 e^{-\frac{t}{\tau_1}} + k_2 e^{-\frac{t}{\tau_2}} + k_3 e^{-\frac{t}{\tau_3}} + \ldots + k_n e^{-\frac{t}{\tau_n}} \qquad (2.1)$$

Donde:

$A =$ Amplitud de variación en la señal de salida
$\tau_i =$ Constante de tiempo del sistema
$k_i =$ Constante de proporcionalidad
$n =$ Orden del sistema

Se debe determinar las constantes de tiempo τ_i y las ganancias k_i.

Si la constante de tiempo τ_1 es grande con respecto a las demás, para tiempos grandes la ecuación 2.1 se aproxima a:

$$C(t) \cong A + k_1 e^{-\frac{t}{\tau_1}} + 0 + 0 + \ldots + 0 \qquad (2.2)$$

De la tabla 2.1 se tiene que la amplitud de variación en la señal de salida es de 6.2 [Voltios], correspondiente al valor de estabilización de la salida en

respuesta a la señal paso dada en la entrada. Por consiguiente A será igual a 6.2 [V].

De la Ec. (2.2)

$$C(t) - A = k_1 e^{-\frac{t}{\tau_1}} \tag{2.3}$$

C(t) es la variación de voltaje en función del tiempo, entonces se puede calcular la ecuación 2.3 lo que genera los siguiente datos:

Tiempo(s)	V_out(V)	C(t) - A	| C(t) – A |
0	0	-6,2	6,2
0,01	0,4	-5,8	5,8
0,02	1,4	-4,8	4,8
0,03	2,6	-3,6	3,6
0,04	3,4	-2,8	2,8
0,05	4	-2,2	2,2
0,06	4,9	-1,3	1,3
0,07	5,4	-0,8	0,8
0,08	5,7	-0,5	0,5
0,09	6	-0,2	0,2
0,1	6,1	-0,1	0,1
0,11	6,2	0	0
0,12	6,2	0	0
0,13	6,2	0	0
0,14	6,2	0	0
0,15	6,2	0	0

Tabla 2.2
Operaciones realizadas según la ecuación 2.3

Si se realiza la gráfica $|C(t) - A|$ vs. t en escala semilogarítmica (figura 2.2) se tiene que cuando el tiempo es grande, los valores del voltaje de salida se puede aproximar a una recta por medio de una interpolación mediante el método de mínimos cuadrados [7].

La función a interpolar es:

$$f(t) = k_1 e^{-\frac{t}{\tau_1}} \tag{2.4}$$

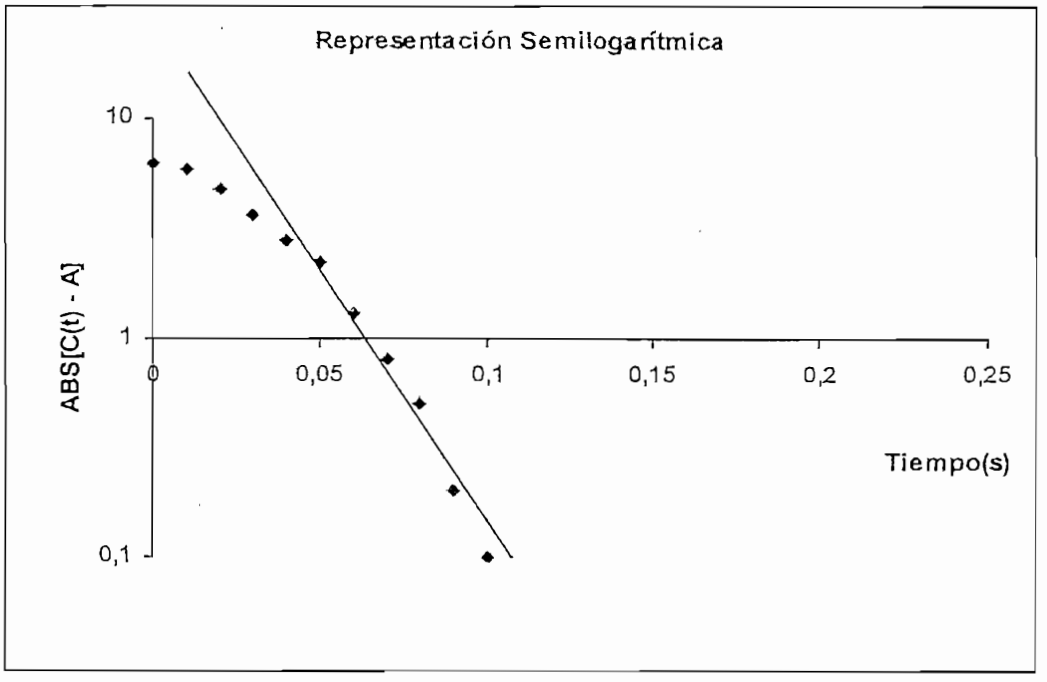


Figura 2.2
Representación semilogarítmica

La ecuación 2.4 mediante manipulaciones matemáticas se la puede transformar a una forma lineal de la siguiente forma:

$$f(t) = k_1 e^{-\frac{t}{\tau_1}}$$

$$\ln[f(t)] = \ln(k_1) - \left(\frac{1}{\tau_1}\right)t \qquad (2.5)$$

Por lo tanto una gráfica semilogarítmica de $\ln[f(t)]$ contra t genera una línea recta con una pendiente de $-\left(\frac{1}{\tau_1}\right)$ y una intersección de $\ln(k_1)$.

Para el ajuste de una recta utilizando mínimos cuadrados se determina los valores de las constantes a_0 y a_1 de la ecuación general de una recta.

$$y = a_0 + a_1 x \tag{2.6}$$

$$a_0 = \frac{\sum x_i \sum y_i - \sum x_i \sum x_i y_i}{n\sum x_i^2 - \left(\sum x_i\right)^2} \tag{2.7}$$

$$a_1 = \frac{n\sum x_i y_i - \sum x_i \sum y_i}{n\sum x_i^2 - \left(\sum x_i\right)^2} \tag{2.8}$$

Donde:

$n = número\ de\ puntos\ tomados$

$x_i = valores\ de\ tiempo$

$y_i = valores\ de\ voltaje$

De la tabla 2.3 se calcula a_0 y a_1 utilizando las ecuaciones 2.7 y 2.8 con lo que se tiene k_1 y τ_1 igualando la ecuación 2.6 con la ecuación 2.5.

n	xi=tiempo	f(t)	yi=ln[f(t)]	Xi*yi	xi^2
1	0,05	2,2	0,78845736	0,03942287	0,0025
2	0,06	1,3	0,26236426	0,01574186	0,0036
3	0,07	0,8	-0,22314355	-0,01562005	0,0049
4	0,08	0,5	-0,69314718	-0,05545177	0,0064
5	0,09	0,2	-1,60943791	-0,14484941	0,0081
6	0,1	0,1	-2,30258509	-0,23025851	0,01
$\sum$	0,45		-3,77749211	-0,39101502	0,0355

.a0=	3,98626564	K_1=	53,8534056
.a1=	-61,5446355	τ_1 =	0,01624837

Tabla 2.3

Análisis de regresión

De donde $f(t) = k_1 e^{-\frac{t}{\tau_1}} = 53{,}8534056 * e^{-\frac{t}{0{,}01624837}}$, para realizar la figura 2.2 es necesario tener sólo valores positivos, es por ello que se saco el valor absoluto de los datos negativos, es por tal razón que la constante k_1 tiene signo negativo.

La ecuación C(t) se representa como:

$$C(t) = 6{,}2 - 53{,}8534056 * e^{-\frac{t}{0{,}01624837}} + k_2 e^{-\frac{t}{\tau_2}} + k_3 e^{-\frac{t}{\tau_3}} + \ldots + k_n e^{-\frac{t}{\tau_n}} \qquad (2.9)$$

Para determinar el valor de k_2 y τ_2 se despejan de la ecuación 2.9, ya que los términos siguientes tienden a cero muy rápidamente se genera la tabla 2.4

$$C(t) - 6{,}2 + 53{,}8534056 * e^{-\frac{t}{0{,}01624837}} = k_2 e^{-\frac{t}{\tau_2}} + 0 + \ldots + 0 \qquad (2.10)$$

Tiempo(s)	V_out(V)	C(t) - 6,2+53,8534056*e(-t/0,01624837)
0	0	47,6534056
0,01	0,4	23,3023607
0,02	1,4	10,92690508
0,03	2,6	4,898813757
0,04	3,4	1,792755847

Tabla 2.4

Operaciones realizadas según la ecuación 2.10

Al realizar una representación semilogarítmica de los datos de la tabla 2.4 se obtiene la recta de ecuación $f(t) = k_2 e^{-\frac{t}{\tau_2}}$ (figura 2.3).

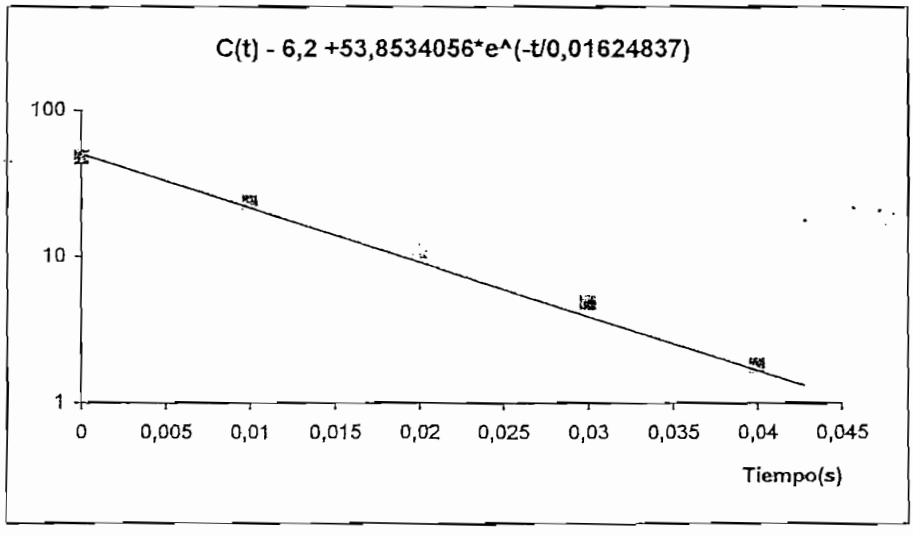


Figura 2.3

Representación semilogarítmica de la ec. 2.10

La figura 2.3 se la puede aproximar a una recta por medio de una interpolación siguiendo el mismo procedimiento anterior.

n	xi=tiempo(s)	V_out(V)	f(t)	yi=ln[f(t)]	xi*yi	xi*xi
1	0	0	47.6534056	3.8639541	0	0
2	0.01	0.4	23.3023607	3.14855467	0.03148555	0.0001
3	0.02	1.4	10.9269051	2.3912281	0.04782456	0.0004
4	0.03	2.6	4.89881376	1.58899309	0.04766979	0.0009
5	0.04	3.4	1.79275585	0.58375402	0.02335016	0.0016
	0.1			11.576484	0.15033006	0.003

a0=	3.93928915	K2=	51.3820632
a1=	-81.1996175	T2=	0.01231533

Tabla 2.5

Análisis de regresión

De la tabla 2.5 se tiene que:

$$f(t)= k_2 e^{-\frac{t}{\tau_2}} = 51{,}3820632 * e^{\frac{t}{0{,}01231533}}$$

Por consiguiente la ecuación C(t) es:

$$C(t)= 6{,}2 - 53{,}8534056 * e^{\frac{t}{0{,}01624837}} + 51{,}3820632 * e^{\frac{t}{0{,}01231533}} + k_3 e^{\frac{t}{\tau_3}} + \ldots + k_n e^{\frac{t}{\tau_n}}$$

Para este caso no se puede realizar una tercera aproximación por tanto la ecuación definitiva es:

$$C(t)= 6{,}2 - 53{,}8534056 * e^{\frac{t}{0{,}01624837}} + 51{,}3820632 * e^{\frac{t}{0{,}01231533}} \qquad (2.11)$$

En la tabla 2.6 se hace una comparación entre la variación de los valores reales del voltaje de salida al aplicar una función paso a la entrada del sistema, con los valores identificados y se observa que la diferencia es pequeña por lo tanto la identificación es válida.

Una mejor apreciación se la puede hacer en la figura 2.4

Tiempo(s)	C(t)real	C(t)ident
0	0	0.0000
0.01	0.4	0.4161
0.02	1.4	1.3481
0.03	2.6	2.7730
0.04	3.4	3.9726
0.05	4	4.8212
0.06	4.9	5.3732
0.07	5.4	5.7151
0.08	5.7	5.9202
0.09	6	6.0405
0.1	6.1	6.1100
0.11	6.2	6.1495
0.12	6.2	6.1719
0.13	6.2	6.1844
0.14	6.2	6.1914
0.15	6.2	6.1953

Tabla 2.6

Comparación entre los valores medidos e identificados

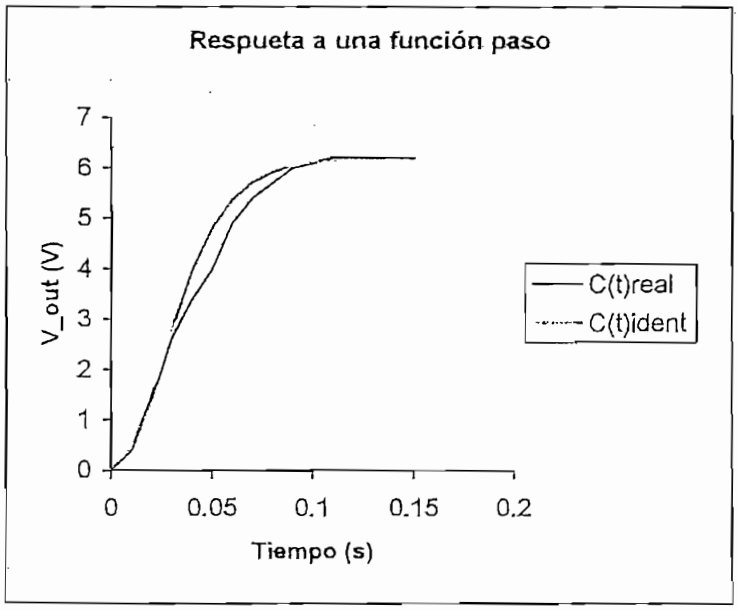


Figura 2.4

Representación gráfica de la variación de voltaje en función del tiempo real e identificado, al ser excitado el sistema en lazo cerrado por una función paso.

El modelo a función de transferencia de la ecuación 2.11 aplicando la transformada de Laplace está dado por:

Si:

$$f(t)_1 = 6{,}2 \Rightarrow F(s)_1 = \frac{6{,}2}{s}$$

$$f(t)_2 = -53{,}8534056 * e^{\frac{t}{0{,}01624837}} \Rightarrow F(s)_2 = -\frac{53{,}8534056}{s + \frac{1}{0{,}01624837}}$$

$$f(t)_3 = 51{,}3820632 * e^{\frac{t}{0{,}01231533}} \Rightarrow F(s)_3 = \frac{51{,}3820632}{s + \frac{1}{0{,}01231533}}$$

Entonces:

$$C(s) = F(s)_1 + F(s)_2 + F(s)_3$$

$$C(s) = \frac{3.729 * s^2 - 325.6 * s + 3.098 * 10^4}{s^3 + 142.7 * s^2 + 4997 * s}$$

2.1.1.2 Modelo para el control de posición.

Para el sistema indicado en la figura 2.5 la identificación se la realiza aproximando el sistema a uno de segundo grado oscilatorio amortiguado.

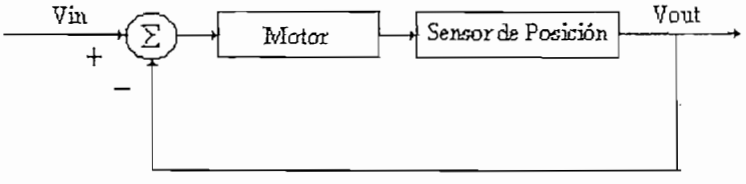


Figura 2.5

Sistema motor – sensor de posición.

De la tabla 2.7 se tiene el gráfico 2.6 con el cual se puede determinar el máximo sobrepico o sobre impulso y tiempo de establecimiento de la siguiente manera:

$$Mp\% = sobreimpulso$$
$$Mp\% = \frac{y_p - y(\infty)}{y(\infty) - y(0)} * 100 \qquad (2.12)$$

Donde:

y_p = Valor pico o valor máximo de y

$y(\infty)$ = Valor final

$y(0)$ = Valor inicial

Entonces:

$$Mp\% = \frac{11.2 - 6.8}{6.8 - 0} * 100$$
$$Mp\% = 64.71\%$$

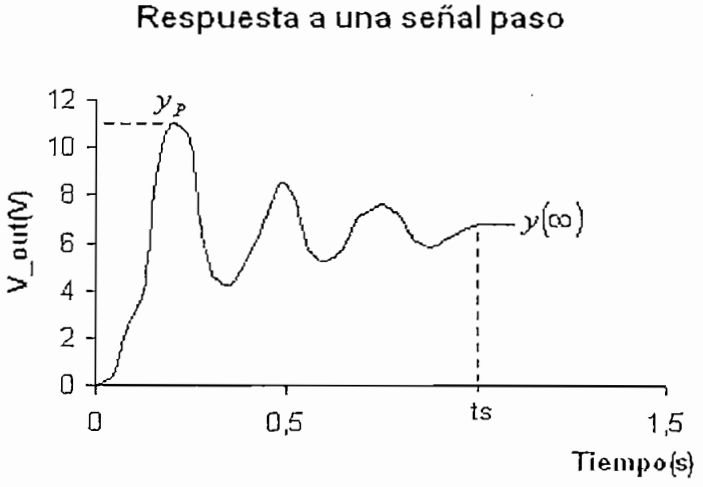


Figura 2.6

Curva de respuesta del sistema motor – sensor de posición realimentado

Tiempo(s)	V_out(V)
0	0
0,025	0,2
0,05	0,6
0,075	2
0,1	3
0,125	4
0,15	8
0,175	10
0,2	11
0,225	10,8
0,25	10
0,275	7
0,3	5
0,325	4,4
0,35	4,2
0,375	4,8
0,4	5,4
0,425	6,2
0,45	7,2
0,475	8,4
0,5	8,4
0,525	7,8
0,55	6
0,575	5,4
0,6	5,2
0,625	5,4
0,65	5,8
0,675	6,8
0,7	7,2
0,725	7,4
0,75	7,6
0,775	7,4
0,8	7
0,825	6,4
0,85	6
0,875	5,8
0,9	6
0,925	6,2
0,95	6,4
0,975	6,6
1	6,8
1,025	6,8
1,05	6,8
1,075	6,8
1,1	6,8

Tabla 2.7

Variación del voltaje de salida en función del tiempo

El tiempo de establecimiento ts es el tiempo que se requiere para que la curva de respuesta alcance un rango alrededor del valor final del tamaño especificado por el porcentaje absoluto del valor final (por lo general, de 2 a 5%) y permanezca dentro de él [4].

De la figura 2.6 se tiene que el tiempo de establecimiento es de 1 segundo. Este sistema tiene una respuesta oscilatoria amortiguada por lo que no es posible realizar la identificación por medio de la determinación de las constantes de tiempo, sin embargo este sistema se aproxima al modelo general de un sistema de segundo orden (figura 2.7).

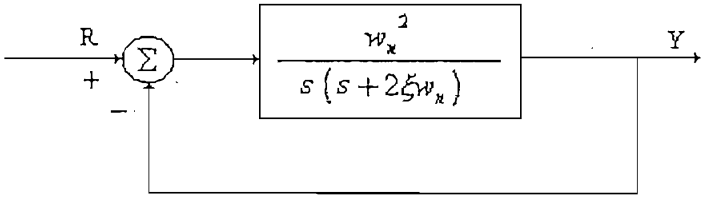


Figura 2.7

Sistema de segundo orden

Para un sistema de segundo orden como el de la figura anterior se tiene:

$$\frac{Y}{R}(s) = \frac{w_n^2}{s^2 + 2\xi w_n s + w_n^2} \qquad (2.13)$$

Donde:

w_n = *frecuencia natural* sin *amortiguamiento*
ξ = *coeficiente de amortiguamiento*

De las especificaciones de diseño obtenidas anteriormente se pueden obtener w_n y ξ ya que se tiene que:

$$Mp\% = e^{\frac{-\xi * \pi}{\sqrt{1-\xi^2}}} * 100 \qquad (2.14)$$

$$ts = \frac{4}{\xi * w_n} \qquad (2.15)$$

Por lo tanto si igualamos la ecuación 2.14 a 64.71% tenemos que ξ = 0.1372 e igualando la ecuación 2.15 a 1segundo da que w_n = 29.15, por lo que la función de transferencia en lazo cerrado es:

$$\frac{Vout}{Vin}(s) = \frac{849.7}{s^2 + 8s + 849.7} \qquad (2.16)$$

Si para la ecuación 2.16 el voltaje de entrada V_{in} es una función paso de tiene que la respuesta es oscilatoria amortiguada (figura 2.8).

Como se puede observar en la figura 2.8 se tiene la simulación de la respuesta del modelo identificado. Para ajustar la ganancia K con los valores medidos se obtiene K= 5600.

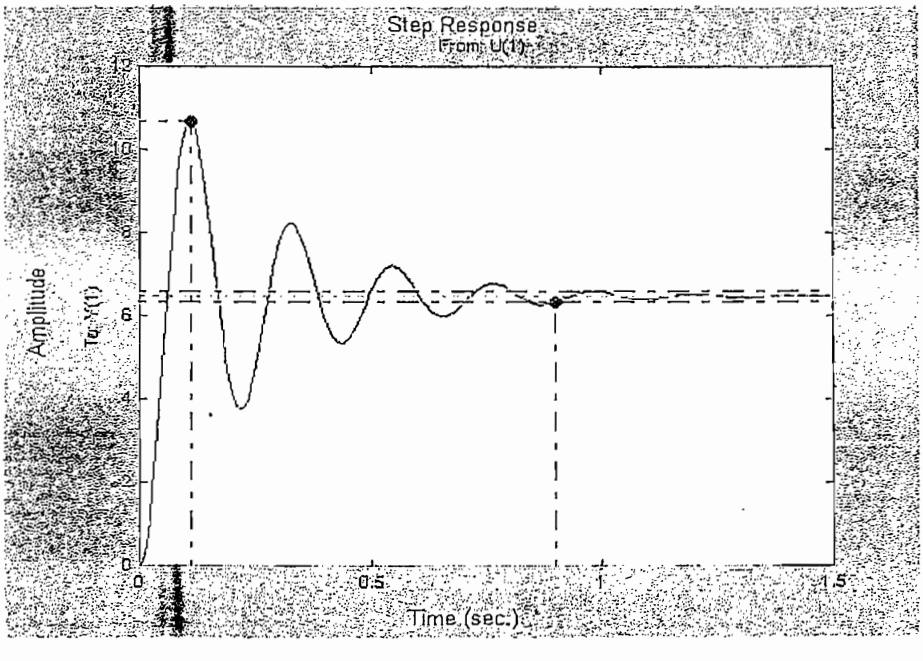


Figura 2.8

Respuesta a una función paso del sistema motor – sensor de posición realimentado

La figura 2.9 muestra la comparación de la respuesta del sistema real y del identificado.

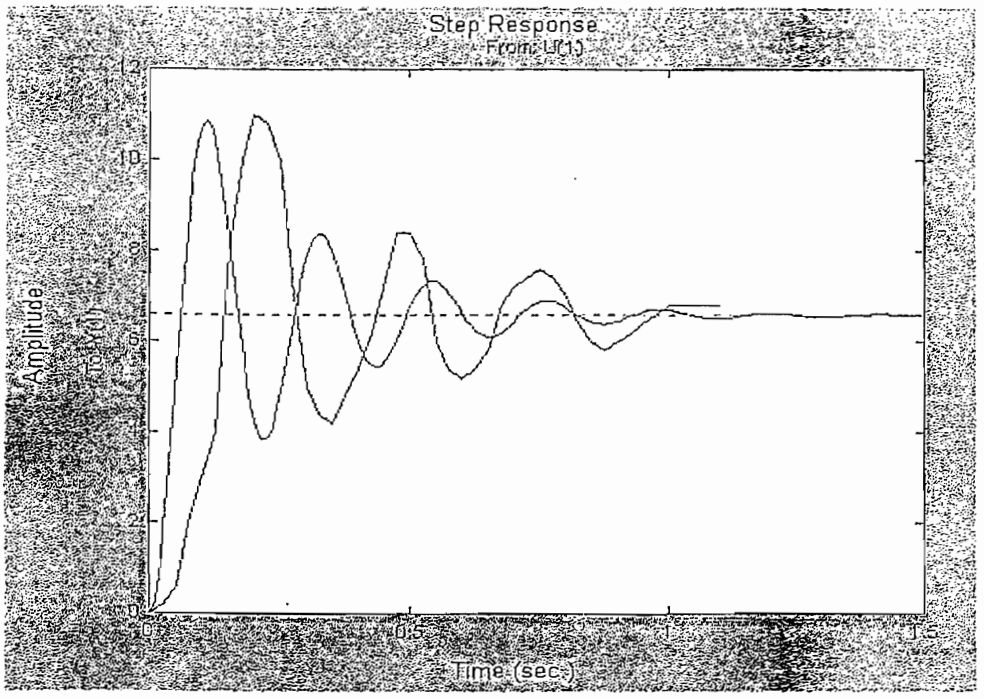


Figura 2.9

Respuesta a una función paso del sistema motor – sensor de posición realimentado del modelo real e identificado.

2.1.2 IDENTIFICACIÓN PARAMÉTRICA DISCRETA

La identificación paramétrica discreta que se utiliza en este proyecto se la realiza por medio de mínimos cuadrados ordinarios o recursivos.

2.1.2.2 Mínimos cuadrados ordinarios (MCO)

Se utiliza un modelo ARMA (Autorepresine Morring Average)

$$y(k)+a_1y(k-1)+a_2y(k-2)+\ldots+a_ny(k-n)=b_1\mu(k-1)+\ldots+b_n\mu(k-n) \qquad (2.17)$$

generalmente, b_0 es igual a cero por que existe un tiempo de retardo de la señal de salida con respecto a la señal de entrada.

En tiempo real se realiza las mediciones de entrada y salida utilizando conversores analógicos digitales (A/D).

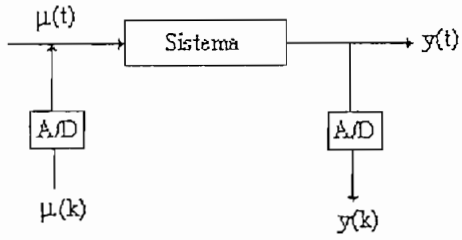


Figura 2.10

Implementación de un sistema en tiempo real

$x(k) = vector\ de\ \text{inf}\ ormación,\ es\ un\ vector\ fila$

$x(k) = [y(k-1), y(k-2), \ldots, y(k-n), \mu(k-1), \mu(k-2), \ldots, \mu(k-n)]$

$\theta(k) = vector\ de\ parámetros,\ vector\ columna$

$$\theta(k) = \begin{bmatrix} -a_1 \\ -a_2 \\ \vdots \\ -a_n \\ b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix}$$

Para determinar los valores de salida del sistema planteado se tiene que:

$$y(k) = x(k) * \theta(k) \tag{2.18}$$

$y(k) = x(k) * \theta(\dot{k}) = 0 \quad si\ \theta(\dot{k}) = parámetros\ verdaderos$

De la ecuación 2.18 se puede calcular el error de estimación ya que se puede estimar el vector de parámetros.

$$y(k)-x(k)*\theta(\hat{k})=e(k) \tag{2.19}$$

Donde:

$e(k)=$ *error de estimación*

$\theta(\hat{k})=$ *parámetros estimados* $=\theta(k)$

Si los parámetros estimados tienden a los parámetros verdaderos el error tenderá a cero, es decir el error se minimizara.

Para minimizar errores es conveniente elevar la función de error al cuadrado, para que errores positivos no se compensen con errores negativos.

$$J=\sum_{k=1}^{k} e(k)^2 = e(1)^2+e(2)^2+\cdots+e(k)^2 \tag{2.20}$$

$$\frac{\partial J}{\partial \theta}=0$$

De la ecuación 2.19 se tiene que:

$$e(k)=y(k)-x(k)*\theta(k) \tag{2.21}$$

Aplicando la ecuación 2.21 se puede obtener el vector de error:

$$e(n)=y(n)-x(n)*\theta(n)$$
$$e(n+1)=y(n+1)-x(n+1)*\theta(n+1)$$
$$\vdots$$
$$e(N)=y(N)-x(N)*\theta(N)$$

$N =$ *número de mediciones*

Se tiene:

Vector de error:

$$E(N)=\begin{bmatrix} e(n) \\ e(n+1) \\ \vdots \\ e(N) \end{bmatrix}$$

Vector de salida:

$$Y(N)=\begin{bmatrix} y(n) \\ y(n+1) \\ \vdots \\ y(N) \end{bmatrix}$$

Matriz de información:

$$X(N)=\begin{bmatrix} x(n) \\ x(n+1) \\ \vdots \\ x(N) \end{bmatrix}$$

Ecuación matricial del error:

$$E(N)=Y(N)-X(N)*\Theta(N) \qquad (2.22)$$

Aplicando la ecuación 2.20 se obtiene:

$$J=\sum_{k=n}^{N} e(k)^2 = E(N)^T * E(N) \qquad (2.23)$$

2.22 en 2.23

$$J=[Y(N)-X(N)*\Theta(N)]^T [Y(N)-X(N)*\Theta(N)]$$

Para mayor facilidad en la nomenclatura se omite N

$$J = \left[Y^T - X^T * \Theta^T\right]\left[Y - X * \Theta\right]$$

$$J = \left[Y^T * Y - Y^T * X * \Theta - X^T * \Theta^T * Y + X^T * \Theta^T * X * \Theta\right]$$

$$\frac{\partial J}{\partial \Theta} = -X^T * Y - X^T * Y + X^T * X * \Theta + X^T * X * \Theta = 0$$

$$-2X^T * Y + 2X^T * X * \Theta = 0$$

$$X^T * X * \hat{\Theta} = X^T * Y$$

$$\hat{\Theta} = \left[X^T * X\right]^{-1} X^T * Y$$

$$\hat{\Theta}(N) = \left[X^T(N) * X(N)\right]^{-1} X^T(N) * Y(N) \qquad (2.24)$$

Con la ecuación 2.24 se puede determinar los parámetros del sistema propuesto, conociendo el vector de salida y la matriz de información, con lo cuál queda identificado dicho sistema.

CARACTERÍSTICAS DEL MCO

1. Es un procesamiento en lote, porque toma N mediciones, esto impide que se pueda realizar en tiempo real.
2. Se tiene que evaluar una matriz inversa.
3. No es en tiempo real.

Se puede aplicar este algoritmo para trabajar en tiempo real, cuando el computador es el que controla, es decir está en lazo cerrado (ON LINE), realizando modificaciones que se exponen a continuación.

2.1.2.3 Mínimos cuadrados recursivos (MCR) [8]

El método de mínimos cuadrados recursivos se puede aplicar en tiempo real, no es un procesamiento en lotes y no se evalúa una matriz inversa, ya que se la simplifica mediante el Lema de Inversión de Matrices.

Satisface que:

$$\hat{\Theta}(k+1)=\hat{\Theta}(k)+corrección \tag{2.25}$$

$$\left.\begin{aligned} y(k)&=x(k)*\theta(k)\\ E(k)&=Y(k)-X(k)*\Theta(k)\\ \hat{\Theta}(k)&=\underbrace{\left[X^T(k)*X(k)\right]^{-1}}_{P(k)}X^T(k)*Y(k)\end{aligned}\right\}\ MCO$$

$$\hat{\Theta}(N+1)=\left[X^T(N)*X(N)\right]^{-1}X^T(N)*Y(N)$$

$$\hat{\Theta}(N+1)=\underbrace{\left[X^T(N+1)*X(N+1)\right]^{-1}}_{P(N+1)}X^T(N+1)*I*Y(N+1) \tag{2.26}$$

$$\hat{\Theta}(N+1)=\left\{\left[X^T(N):x^T(N+1)\right]\begin{bmatrix}X(N)\\ \cdots\cdots\\ x(N+1)\end{bmatrix}\right\}^{-1}\left[X^T(N):x^T(N+1)\right]\begin{bmatrix}Y(N)\\ \cdots\cdots\\ y(N+1)\end{bmatrix}$$

$$\hat{\Theta}(N+1)=\underbrace{\left[X^T(N)*X(N)+x^T(N+1)*x(N+1)\right]^{-1}}_{P(N+1)}\left[X^T(N)*Y(N)+x^T(N+1)*y(N+1)\right] \tag{2.27}$$

Donde:

P = Matriz de covarianza

I = Matriz identidad

De las ecuaciones 2.26 y 2.27 se tiene:

$$P(N+1)=\left[X^T(N+1)*X(N+1)\right]^{-1}$$

$$P(N+1)=\left[\overbrace{X^T(N)*X(N)}^{P^{-1}}+x^T(N+1)*x(N+1)\right]^{-1}$$

$$P(N+1)=\left[P^{-1}(N)+x^T(N+1)*x(N+1)\right]^{-1} \tag{2.28}$$

Lema de Inversión de Matrices:

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

Para aplica el lema de inversión de matrices para la ecuación 2.28 se recurre a un artificio matemático, que es el de multiplicar por la unidad, sin que esto altere el resultado de dicha ecuación.

$$P(N+1) = \left[P^{-1}(N) + x^T(N+1) * 1 * x(N+1)\right]^{-1}$$

Entonces:

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

Por tanto:

$$P(N+1) = \left\{P(N) - P(N) * x^T(N+1)\underbrace{\left[1 + x(N+1) * P(N) * x^T(N+1)\right]^{-1}}_{ESCALAR} x(N+1) * P(N)\right\} \quad (2.29)$$

Finalmente si sustituimos la ecuación 2.29 en la ecuación 2.26 y se incluye la matriz identidad:

$$I = \left[1 + x(N+1) * P(N+1) * x^T(N+1)\right]^{-1}\left[1 + x(N+1) * P(N+1) * x^T(N+1)\right] \quad (2.30)$$

Se tiene las ecuaciones que dan el algoritmo de identificación.

Algoritmo

Todo algoritmo computacional tiene una secuencia lógica para ser ejecutada, para los mínimos cuadrados recursivos el procedimiento a seguir es:

1. $L(k+1)=P(k)*x^T(k+1)\left[1+x(k+1)*P(k)*x^T(k+1)\right]^{-1}$
2. $\varepsilon(k+1)=y(k+1)-x(k+1)*\hat{\theta}(k)$
3. $\hat{\theta}(k+1)=\hat{\theta}(k)+L(k+1)*\varepsilon(k+1)$
4. $P(k+1)=\left[I-L(k+1)*x(k+1)\right]*P(k)$

Donde:

L = Es la ganancia

ε = Error de predicción

P(k) = Matriz de covarianza

La matriz de covarianza es una medida del error de predicción. Se puede inicializar con:

$$P(0)=\alpha*I \tag{2.31}$$

En la ecuación 2.31 se tiene que α representa la medida de incertidumbre, es por tal razón que su valor debe ser alto (generalmente 100), ya que los valores iniciales de los que se parte son de cero.

Cuando sé esta ejecutando el algoritmo de mínimos cuadrados recursivos, puede darse el caso de que en pocas iteraciones se identifique los parámetros que describen al sistema, esto hace que P tienda a cero y si P llega a ser cero todo va a tender a cero produciendo una inestabilidad numérica; Entonces es indispensable que para cada cálculo de P se tenga que dividir a P por el factor de olvido λ.

$$P=\frac{P}{\lambda}$$

$$\lambda = factor\ de\ olvido$$

$$\lambda = 0.98$$

En el presente trabajo, para aplicar lo expuesto anteriormente se realiza una rutina en Matlab llamada minimos_cuadrados. Para verificar dicha rutina se realiza un ejercicio con respuesta, que a continuación se detalla.

Sea:

$$y(k)-1.2y(k-1)+0.35y(k-2)=2\mu(k-1)+0\mu(k-2)$$

a_1 a_2 b_1 b_2

Despejando y(k) se tiene:

$$y(k)=1.2y(k-1)-0.35y(k-2)+2\mu(k-1) \qquad (2.32)$$

Dando una entrada a este sistema, es decir dando valores a μ(k) se puede calcular la salida y(k) mediante la ecuación 2.32 dando como resultado la tabla 2.7.

k	u(k)	y(k)
0	1	0
1	1	2
2	-1	4,4
3	1	2,58
4	-1	3,556
5	-1	1,3642
6	1	-1,60756

Tabla 2.7

Operaciones realizadas según la ecuación 2.32

Con la tabla 2.7 se puede utilizar la rutina de Matlab con los siguientes resultados:

» minimos_cuadrados

Ingrese el [vector de información u(k)]: [1 1 -1 1 -1 -1 1]

Ingrese el [vector de salida y(k)]: [0 2 4.4 2.58 3.556 1.3642 -1.60756]

Ingrese el orden del modelo: 2

Mínimos Cuadrados Ordinarios

a1 = -1.200000

a2 = 0.350000

b1 = 2.000000

b2 = -0.000000

Mínimos Cuadrados Recursivos

a1 = -1.189338

a2 = 0.340201

b1 = 1.997496

b2 = 0.018738

Los resultados que genera la rutina implementada son satisfactorios.

Para identificar los dos sistemas propuestos se utiliza la misma rutina, los resultados obtenidos son:

2.1.2.4 Para el sistema motor – tacogenerador (figura 2.1)

» minimos_cuadrados

Ingrese el [vector de información u (k)]: [0 20 20 20 20 20 20 20 20 20 20 20 20 20 20]

Ingrese el [vector de salida y (k)]: [0 0.4 1.4 2.6 3.4 4 4.9 5.4 5.7 6 6.1 6.2 6.2 6.2 6.2 6.2]

Ingrese el orden del modelo: 2

Mínimos Cuadrados Ordinarios

a1 = -0.916636

a2 = 0.125503

b1 = 0.051667

b2 = 0.015649

Mínimos Cuadrados Recursivos

a1 = -0.927153

a2 = 0.137626

b1 = 0.051456

b2 = 0.015850

Se tiene la ecuación de diferencias:

$$y(k) = -0.92y(k-1) + 0.13y(k-2) + 0.05\mu(k-1) + 0.02\mu(k-2)$$

2.1.2.5 Para el sistema motor – sensor de posición (figura 2.5)

» minimos_cuadrados

Ingrese el [vector de información u(k)]: [0 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20]

Ingrese el [vector de salida y(k)]: [0 0.2 0.6 2 3 4 8 10 11 10.8 10 7 5 4.4 4.2 4.8 5.4 6.2 7.2 8.4 8.4 7.8 6 5.4 5.2 5.4 5.8 6.8 7.2 7.4 7.6 7.4 7 6.4 6 5.8 6 6.2 6.4 6.6 6.8 6.8 6.8 6.8 6.8]

Ingrese el orden del modelo: 3

Mínimos Cuadrados Ordinarios

a1 = -1.197538

a2 = 0.320516

a3 = 0.239491

b1 = 0.033640

b2 = 0.033640

b3 = 0.051922

Mínimos Cuadrados Recursivos

a1 = -1.197538

a2 = 0.320516

a3 = 0.239491

b1 = 0.033640

b2 = 0.033640

b3 = 0.051922

Se tiene la ecuación de diferencias:

$$y(k) = -1.19y(k-1) + 0.32y(k-2) + 0.24y(k-3) + 0.034\mu(k-1) + 0.034\mu(k-2) + 0.052\mu(k-3)$$

CAPITULO 3

SIMULACIÓN DEL MOTOMATIC CON REALIMENTACIÓN DE VELOCIDAD Y POSICIÓN MEDIANTE REDES NEURONALES

3.1 INTRODUCCIÓN

Los intentos por imitar el funcionamiento del cerebro han seguido la evolución del estado de la tecnología. Por ejemplo, al finalizar el siglo 19 se le comparó con la operación de la bomba hidráulica; durante la década de 1920 a 1930 se intento utilizar la teoría de la conmutación telefónica como punto de partida de un sistema de conocimiento similar al del cerebro. Entre 1940 y 1950 los científicos comenzaron a pensar seriamente en las redes neuronales utilizando como concepto la noción de que las neuronas del cerebro funcionan como interruptores digitales (on – off) de manera también similar al recién desarrollado computador digital. Así nace la idea de "revolución cibernética" que maneja la analogía entre el cerebro y el computador digital.

En la actualidad se ha conseguido tener una buena aproximación del comportamiento y aprendizaje de una neurona mediante algoritmos computacionales, gracias a los diferentes trabajos realizados sobre esta temática.

Los sistemas neuronales artificiales o redes neuronales podrían ser utilizados para interpretar tramas complejas. Las redes neuronales pueden proporcionar una solución factible.

En este capítulo se desarrolla una red neuronal que no es más que la conexión entre neuronas, mediante un programa computacional (en este caso Matlab que posee una librería de redes neuronales). Esta red neuronal es entrenada para conseguir algorítmicamente transformar una entrada en una salida.

El entrenamiento de una red neuronal es simplemente una forma de codificar información acerca del problema que haya que resolver.

3.1.1 NEURONAS BIOLÓGICAS

En un neurona biológica se diferencian tres partes principales[9]: el cuerpo de la célula, las dendritas y el axón como se observa en la figura 3.1.

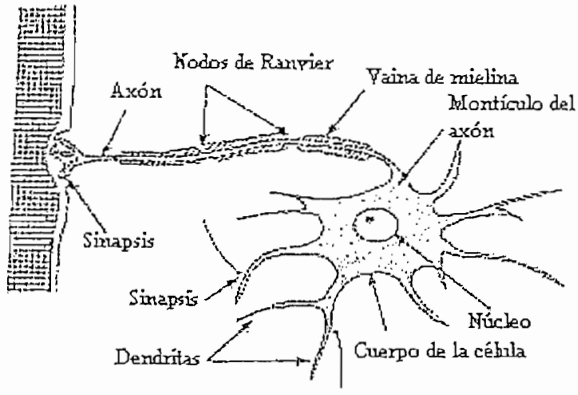


Figura 3.1

Neurona biológica.

El funcionamiento de la neurona comienza cuando las señales de otras neuronas son captadas por las dendritas a través de la unión sináptica, después en el cuerpo de la célula son promediadas mediante complicados procesos químicos, si el promedio resultante logra sobrepasar dentro de cierto intervalo de tiempo un valor conocido como umbral, la neurona se dispara emitiendo una señal que viaja a través del axón hacia otras neuronas; caso contrario no se producirá ningún tipo de estímulo.

3.1.2 NEURONAS ARTIFICIALES

La neurona artificial es un elemento de todo o nada, es decir, se puede o no tener una respuesta o estímulo. Su funcionamiento se asemeja a la neurona biológica ya que a la neurona artificial llega un conjunto de entradas (salidas de otras neuronas) que se multiplican con pesos (análogo a las conexiones sinápticas), el resultado de estos productos es comparado con otro llamado umbral (función de transferencia) y causará que la neurona se dispare o se inhiba, si el resultado es mayor o menor que el umbral respectivamente.

Las neuronas artificiales son unidades procesadoras de información que están constituidas por [10]:

1. Un conjunto de sinapsis o enlaces asociados o caracterizados por un peso o esfuerzo, donde son ponderadas las entradas. El peso es parte básica de una neurona artificial, pues sirve para determinar si esa entrada influirá de manera significativa en el resultado final.
2. Un sumador, para sumar las señales de entrada, pesadas por las respectivas sinapsis de la neurona.
3. Una función de transferencia para limitar la amplitud de la salida de la neurona.
4. Un umbral aplicado externamente, el cual tiene el efecto de incrementar o disminuir la red de entrada de la función de transferencia, dependiendo si el valor de ésta es positiva o negativa respectivamente. Funciona como un offset.

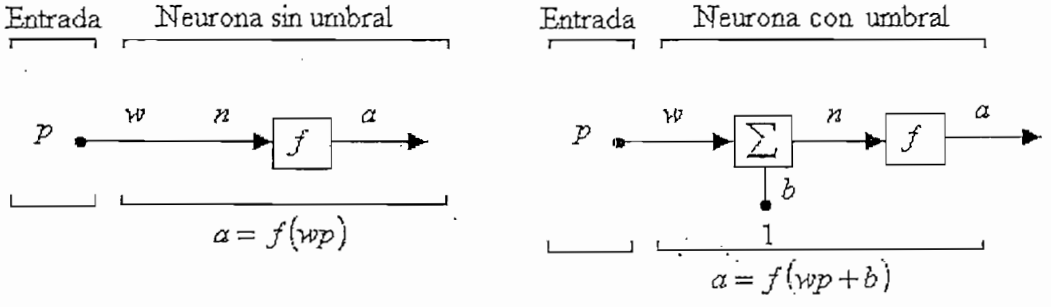


Figura 3.2

Modelo no lineal de una neurona artificial.

Las funciones de transferencia o de activación, definen la salida de una neurona en términos de los parámetros locales definidos. Se define tres tipos básicos de funciones de transferencia: Función escalón o limitador duro, función lineal y funciones sigmoidales [11].

Generalmente la función escalón se utiliza en la neurona llamada perceptrón que se definirá más adelante.

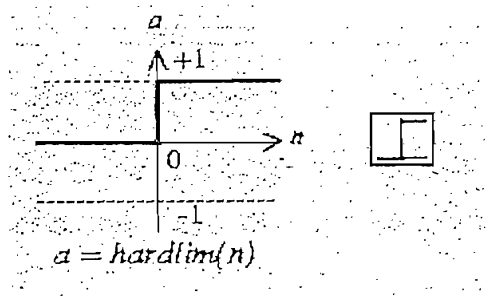


Figura 3.3

Función limitador duro

Esta función es la que da al perceptrón la habilidad de clasificar vectores de entrada.

Las funciones sigmoidales son utilizadas en neuronas de retropropagación.

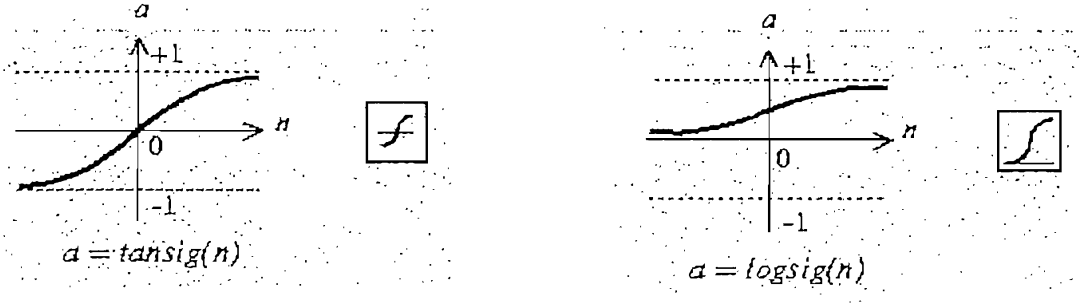


Figura 3.4

Funciones sigmoidales

En algunos casos, la función de transferencia lineal purelin es usada en las redes Backpropagation.

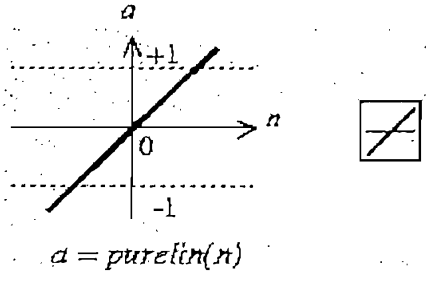


Figura 3.5

Función lineal

Aunque las tres funciones anteriores son las más utilizadas existen otras tipos de funciones que se puede consultar en [11].

3.1.2.1 Arquitectura de las redes neuronales

Los parámetros fundamentales de las redes neuronales son: El número de capas, el número de neuronas por capa, el grado de conectividad y el tipo de conexiones entre neuronas.

Entre las redes neuronales tenemos a la *Redes Monocapa,* que están conformadas por una capa de señal de entrada y una capa de señal de salida. *Redes Multicapa,* son aquellas que disponen de un conjunto de neuronas agrupadas en varios niveles o capas.

Es necesario realizar una acotación, para contabilizar el número de capas de una red, se toma en cuenta únicamente aquella donde se realiza alguna operación.

Las conexiones de las neuronas se puede realizar de dos formas:

1. Conexiones hacia delante o *feedforward*, en la que todas las neuronas de una capa reciben señales de entrada de otra capa anterior, más cercana a las entradas de la red, y envían las señales de salida a una capa posterior, más cercana a la salida de la red.
2. Conexión hacia atrás o *feedback* consiste en conectar las salidas de las neuronas posteriores a las entradas de las capas anteriores.

3.1.2.2 Redes neuronales con conexión hacia delante

Entre las más importantes tenemos: El perceptrón, ADALINE y MADALINE, Retropropagación (backpropagation).

El perceptrón inventado por Rosenblatt (1962), consiste en una sola neurona con pesos variables y un umbral, como función de transferencia se utiliza a la función limitador duro (figura 3.6).

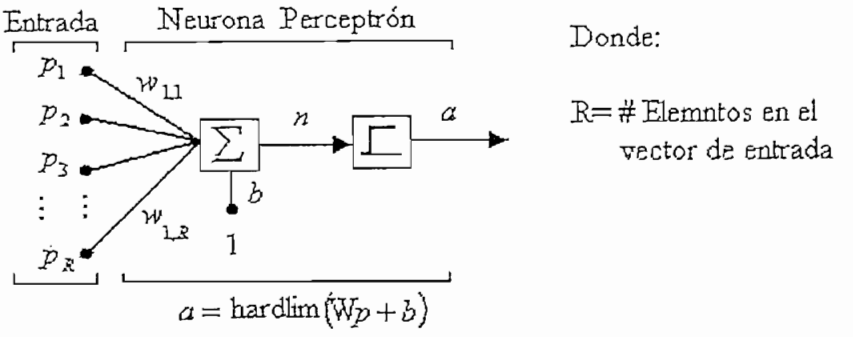


Figura 3.6

El perceptrón.

Los perceptrones, consisten en pesos entrenables multiplicativos, un sumador y una función de umbral. El aprendizaje es un proceso mediante el cual se modifica los valores de los pesos y del umbral [9].

El perceptrón fue concebido principalmente como un separador lineal, el éxito inicial lo obtuvo en el reconocimiento de tramas linealmente separables.

La separabilidad lineal, propias de las neuronas perceptrón, está limitado a los problemas en los cuales se tiene un conjunto de puntos (valores de entrada) que son geométricamente separables [9].

Las redes ADALINE (ADAptive LINear Element) y MADALINE (Múltiple ADALINE) fueron desarrolladas por Bernie Widrow en la Universidad de Stanford poco después de que Rosenblatt desarrollara el Peceptrón. Las arquitecturas de ADALINE y MADALINE son esencialmente las mismas que las del Perceptrón, difieren fundamentalmente en el mecanismo de aprendizaje. ADALINE y MADALINE utilizan la denominada regla de Widrow – Hoff o regla de error mínimo cuadrático medio (LMS).

La red ADALINE está limitada a una única neurona de salida, mientras que MADALINE puede tener varias.

La red ADALINE se muestra en la figura 3.7 a la que si se la compara con el perceptrón se aprecia que la única diferencia es la función de transferencia. Sin embargo la red ADALINE tiene la misma limitación del perceptrón, puesto que puede resolver únicamente problemas que sean linealmente separables.

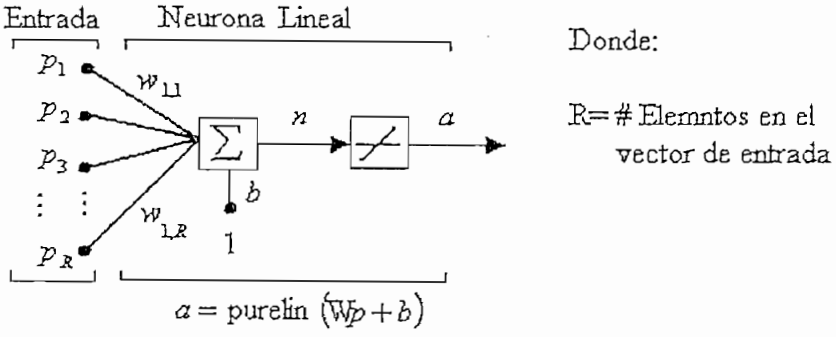


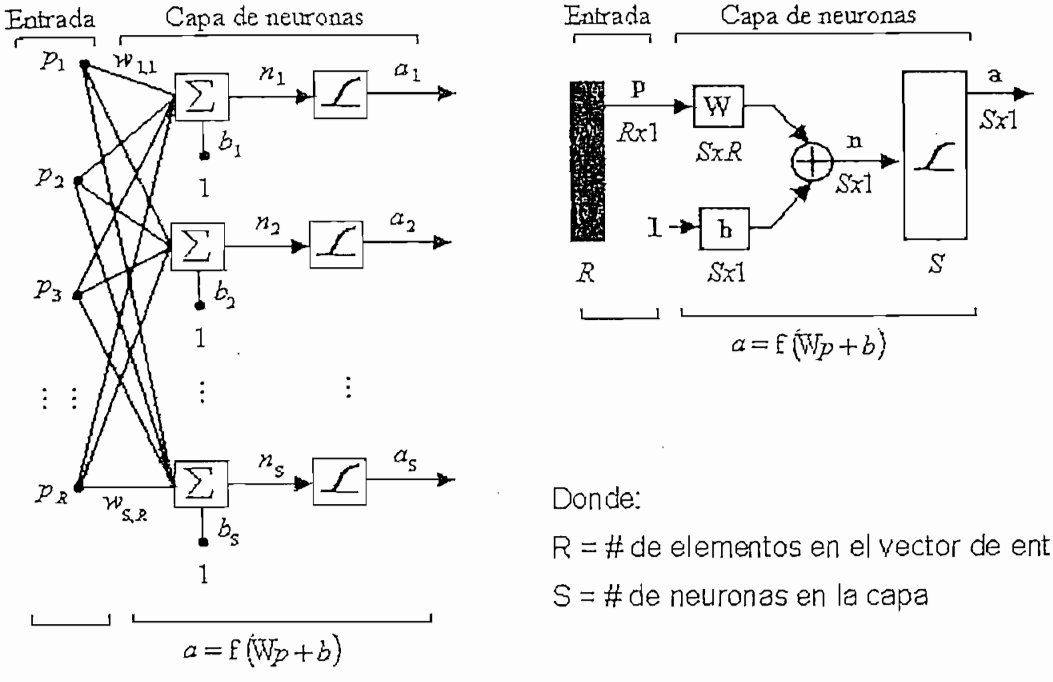
Figura 3.7

La red ADALINE.

La red de Retropropagación, conocida también como Perceptrón Multicapa (MultiLayer Perceptron), está formada por varias capas de neuronas. La estructura simplificada de una de sus neuronas se muestra en la figura 3.8 [11].

El algoritmo para entrenar redes Perceptrón Multicapa es el de propagación inversa, el cuál se hizo público gracias a los trabajos de Rumelhart y McClelland (1986). El algoritmo de propagación inversa calcula en forma directa la respuesta de la red, luego se calcula el error y de manera inversa se modifican los respectivos pesos.

La red de retropropagación no tienen la restricción de la separabilidad lineal ya que son capaces de representar cualquier función con un número finito de discontinuidades.



Donde:

R = # de elementos en el vector de entrada

S = # de neuronas en la capa

Figura 3.8

Red de retropropagación.

La función de activación es la sigmoide, esta introduce no linealidaes a la red de retropropagación, sin embargo puede usarse otras funciones, el único requisito es que sean derivables en todo su rango.

Matlab posee comandos especializados para crear: Un perceptrón (newp), una red ADALINE (mewlin) [11].

3.1.3 REDES NEURONALES ARTIFICIALES EN MATLAB

3.1.3.1 Marco teórico

En el paquete computacional MATLAB posee una librería especializada para redes neuronales en la que se pude entrenar y simular redes neuronales artificiales.

En general una red tiene la siguiente estructura y parámetros, que si se digita el comando net en la ventana de comandos de MATLAB se puede visualizar. Previamente se debe crear la red con el comando: net=network;

```
» net=network;
» net
net =
   Neural Network object:
   architecture:
      numInputs: 0                 número de vectores de entrada
      numLayers: 0                 número de capas de la red
     biasConnect: []               bias conectadas a las neuronas de la red
      inputConnect: []             conexión entre el vector de entrada y la
                                   primera capa
    layerConnect: []               conexión entre capas intermedias
   outputConnect: []               conexión entre la última capa y la salida
   targetConnect: []               conexión de un vector de salida objetivo

     numOutputs: 0  (read-only)    número de vectores de salida
     numTargets: 0  (read-only)    número de vectores de salida objetivo
  numInputDelays: 0  (read-only)   número de retardos a la entrada
  numLayerDelays: 0  (read-only)   número de retardos de capa

   subobject structures:

         inputs: {0x1 cell} of inputs
         layers: {0x1 cell} of layers
        outputs: {1x0 cell} containing no outputs
        targets: {1x0 cell} containing no targets
         biases: {0x1 cell} containing no biases
   inputWeights: {0x0 cell} containing no input weights
   layerWeights: {0x0 cell} containing no layer weights
```

functions:

adaptFcn: (none) Define la función a ser usada cuando la red adapta pesos y bias (no siempre se usa).

initFcn: (none) Define la función a ser usada para inicializar la matriz de pesos y el vector de bias de la red.

performFcn: (none) Defiene la función que se usará para medir el rendimiento de la red.

trainFcn: (none) Define la función a usar para entrenar la red.

parameters:

adaptParam: (none) Define los parámetros y valores de la actual función adapt.

initParam: (none) Define los parámetros y valores de la actual función de inicialización.

performParam: (none) Define los parámetros y valores de la actual función de rendimiento.

trainParam: (none) Define los parámetros y valores de la actual función de entrenamiento (iteraciones, meta, max. error, mostrar, resultados después de # iteraciones,etc.).

weight and bias values:

IW: {0x0 cell} containing no input weight matrices
LW: {0x0 cell} containing no layer weight matrices
b: {0x1 cell} containing no bias vectors

other:

userdata: (user stuff)

»

Para ampliar los conceptos anteriores se puede referir a [11].

3.1.3.2 Creación de una red de retropropagación

Una red de retropropagación puede ser creada usando la instrucción:

net =newff(PR,[S1 S2 ...SNl],{TF1 TF2...TFNl},BTF,BLF,PF)

Donde:

PR	matriz Rx2 de valores mínimos y máximos de R elementos de entrada
Si	tamaño de la i – ésima capa, para Nl capas
TFi	función de transferencia de la i – ésima capa, por defecto = 'tansig'
BTF	función de entrenamiento de la red, por defecto = 'trainlm'
BLF	función de aprendizaje de los pesos y las bias, por defecto = 'learngdm'
PF	función de rendimiento, por defecto = 'mse'

La función de transferencia, entrenamiento, el algoritmo de aprendizaje, la función de rendimiento, pueden ser cambiadas a las distintas alternativas que presenta Matlab.

Se inicializa los pesos y bias de cada capa y entonces la red esta lista para ser entrenada y posteriormente simulada.

Ejemplo [12]:

Para crear una red de retropropagación de tres capas se lo realiza con el siguiente código:

```
net=newff([-1 2;0 5],[3,2,1],{'tansing','logsing','purelin'},'traingd');
```

En está instrucción se aprecia que la red creada consta de un vector de entrada de dos elementos, posee tres capas. La primera capa con 3 neuronas y función de transferencia tansing, la segunda con 2 neuronas y función de

transferencia logsing, por último, la capa de salida con una sola neurona y función lineal purelin. La función de entrenamiento es traingd.

Se debe inicializar los pesos y umbrales antes de entrenar la red.

La función de inicialización predefinida de la red es 'initlay' (net.initFcn='initlay'), ésta permite a cada capa usar su propia función de inicialización. Ahora para definir las funciones de inicialización para la primera capa se usa las siguientes líneas:

net.layers{1}.initFcn='initwb';	función de inicialización de la primera capa.
net.inputWeights{1,1}.initFcn='rands';	pesos que unen la entrada 1 con la capa1.
net.biases{1,1}.initFcn='rands';	bia de la capa 1 neurona 1.
net.biases{2,1}.initFcn='rands';	bia de la capa 1 neurona 2.
net=init(net);	inicialización de pesos y bias.

net.layers{i}.initFcn determina la función de inicialización de cada capa, las funciones comúnmente usadas son: 'initwb' o 'initnw'.

net.inputWeights{i,j}.initFcn puede usar las funciones randómicas que asignan valores iniciales a los pesos entre un rango de –1 a 1.
Net.inputbiases{i,j}.initFcn también suele usarse como una función randómica.

Entrenamiento:

En general, los algoritmos de entrenamiento requieren que se defina los siguientes parámetros:

- epochs, número de épocas, el entrenamiento parará si el número de iteraciones es mayor al número aquí definido.
- show, indica cada cuantas épocas se desplegará información sobre el entrenamiento.

- goal, error meta, el entrenamiento se detendrá una vez que se ha alcanzado este error.
- time, tiempo, otra razón por la cual el entrenamiento puede detenerse si el tiempo que ha durado el proceso ha excedido este valor [segúndos].
- lr, razón de aprendizaje.
- Min_grad, mínimo gradiente, si la magnitud del gradiente es menor a este parámetro, el entrenamiento cesa.
- Max_fail, máximo valor de fracaso, es usado para mejorar la generalización de la red.

Los parámetros más importantes que deben ser ingresados son: epoch, goal, show y lr.

A continuación se presenta una tabla que resume los algoritmos de entrenamiento que presenta Matlab.

Función	Descripción
traingd	Gradiente descendente básico. Respuesta lenta, puede ser usado en el modo de entrenamiento incremental
traingdm	Gradiente descendente con momento. Generalmente más rápido que traingd. Puede ser usado en modo de entrenamiento incremental.
traingdx	Velocidad de entrenamiento adaptivo. Más rápido que traingd, pero solo puede ser usado en modo de entrenamiento por lotes (bach).
trainrp	Retropropagación elástica. Algoritmo simple en modo de entrenamiento incremental con rápida convergencia y mínimo requerimiento de memoria.
traincgf	Algoritmo de gradiente conjugado Fletcher – Reeves. Tiene un requerimiento de memoria mucho menor que otros algoritmos de gradiente desendente.
traincgp	Algoritmo de gradiente conjugado Polak – Ribiére. Requerimiento de memoria ligeramente mayor que traingcf. Convergencia más rápida en algunos problemas.
traincgb	Algoritmo de gradiente conjugado de Powell – Beale. Requerimiento de memoria ligeramente mayor a traincgp. Generalmente de convergencia más rápida.
trainscg	Algoritmo de gradiente conjugado escalado. Es el único algoritmo de gradiente conjugado que no requiere encontrar linealidad.
trainbfg	Método quasi – Newton BFGS. Requiere de memoria

	adicional para mantener la matriz Hessiana y tiene más trabajo computacional en cada iteración, pero generalmente converge en pocas iteraciones.
trainoss	Método de la secante en un paso. Es un arreglo entre los métodos del gradiente conjugado y de quasi – Newton.
trainlm	Algoritmo Lavenberg – Marquardt. Es un algoritmo más rápido para redes de tamaño moderado. Tiene una reducción de memoria apreciable cuando el entrenamiento es grande.
trainbr	Regularización Bayesiana. Es una modificación del algoritmo de entrenamiento Lavenberg – Maquardt. Reduce la dificultad para determinar la arquitectura óptima de la red

Tabla 3.1

Algoritmo de entrenamiento para redes de retropropagación en Matlab

Para entrenar una red utilizando cualquiera de los algoritmos de entrenamiento, se invoca la función 'train', ésta realiza el entrenamiento de acuerdo a net.trainFcn y net.trainParam, se escribe la siguiente instrucción:

[net,tr]=train(net,P,T)

donde:

P entradas a la red
T objetivo de la red, por defecto = ceros
net nombre de la red
tr registro de entrenamiento (épocas y rendimiento)

Simulación:

Finalmente se simula la red con el comando sim para comprobar los resultados:

a=sim(net,P)

3.1.4 SIMULACIÓN DEL SISTEMA MCSL – 100 (*Motomatic Control System Laboratory*) UTILIZANDO REDES NEURONALES

Existe una variedad de redes neuronales, la red de retropropagación se utilizó con excelentes resultados para simular la máquina de inducción [10]. En este caso se utiliza la misma red para realizar la simulación del sistema MCSL –100.

3.1.4.1 Construcción de la red neuronal de retropropagación en Matlab

La red neuronal a emplear para la simulación del sistema MCSL – 100 se representa en la figura 3.9.

La red neuronal está compuesta por:

- Cuatro entradas
- La capa uno está compuesta por seis neuronas, con una función de transferencia 'logsig'
- La segunda capa está formada por una sola neurona, con función de transferencia 'purelin'
- Se tiene una salida a continuación de la segunda capa.

En la inicialización de pesos y umbrales se utiliza el método de Nguyen – Widrow (initnw). La red neuronal es entrenada con el método de Relisient Backpropagation (trainrp), los valores del vector de salida son asociados con el vector objetivo o deseado con el error del mínimo cuadrático (mse).

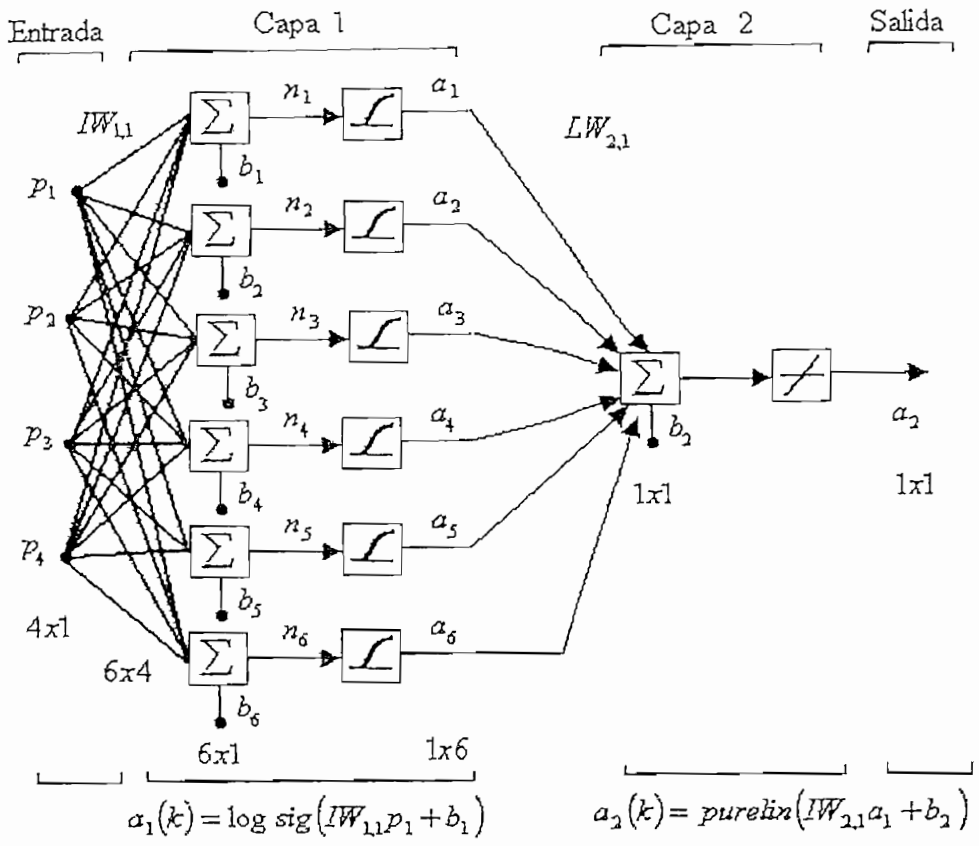


Figura 3.9

Red empleada en la simulación del sistema MCSL – 100.

3.1.4.1.1 Definición de la red neuronal

Para crear una nueva red neuronal en Matlab se utiliza el comando:

» net=network

3.1.4.1.2 Número de entradas y capas

De acuerdo al diagrama de la red propuesta (figura 3.9) el número de entradas y el número de capas se define con los comandos:

» net.numInputs=4;
» net.numLayers=2;

3.1.4.1.3 Conexión de umbrales

La presencia o ausencia de conexiones de umbrales, pesos de entrada, pesos de capa, salidas y objetivos se representa con 1 y 0 respectivamente.

Específicamente la primera y segunda capa tiene conexiones de umbrales en el diagrama de la red neuronal, entonces se escribe el siguiente código.

```
» net.biasConnect(1)=1;
» net.biasConnect(2)=1;
```

3.1.4.1.4 Conexión de pesos de entrada y de capas

Se conecta cada una de las entradas a la primera capa con el siguiente código.

```
» net.inputConnect(1,1)=1;
» net.inputConnect(1,2)=1;
» net.inputConnect(1,3)=1;
» net.inputConnect(1,4)=1;
```

3.1.4.1.5 Conexión a la salida y valores – objetivo

Se conecta la capa dos a la salida de la red neuronal con esta línea de código

```
» net.outputConnect=[0 1];
```

Y la conexión del objetivo con el código.

```
» net.targetConnect=[0 1];
```

El valor objetivo de la capa 2 será comparado con la salida de la capa dos para generar un error que será usado cuando se esté ejecutando la red neuronal, o cuando se esté actualizando la red neuronal durante el entrenamiento o la adaptación.

3.1.4.1.6 Número de salidas y valores – objetivo

Estos valores son solo de lectura (read – only) y se los puede visualizar al escribir net y al pulsar enter.

numOutputs: 1 (read-only)
numTargets: 1 (read-only)

3.1.4.1.7 Entradas

Para ver el arreglo de la estructura de entrada se escribe:

» net.inputs

ans =

[1x1 struct]
[1x1 struct]
[1x1 struct]
[1x1 struct]

La siguiente línea de código indica las propiedades asociadas con la primera entrada

» net.inputs{1}

ans =

range: [4x2 double]
size: 4
userdata: [1x1 struct]

Para especificar el rango de los valores de las entradas se escribe.

```
» net.inputs{1}.range=[0 7;0 20;0 20:0 20];
```

3.1.4.1.8 Capas

Al escribir la siguiente línea de código se puede ver las propiedades asociadas con la primera capa.

```
» net.layers{1}
```

Se escribe las siguientes líneas de código para cambiar el tamaño de la capa a 6 neuronas, su función de transferencia a logsig, y su función de inicialización a la función Nguyen – Widrow como lo requiere el diagrama de la red neuronal propuesta.

```
» net.layers{1}.size=6;
» net.layers{1}.transferFcn='logsig';
» net.layers{1}.initFcn='initnw';
```

La segunda capa tiene una neurona, la función de transferencia es purelin, y es inicializada con initnw entonces se tiene:

```
» net.layers{2}.size=1;
» net.layers{2}.transferFcn='purelin';
» net.layers{2}.initFcn='initnw';
```

3.1.4.1.9 Salidas y valores – objetivo

Para visualizar la salida y valor – objetivo se escribe.

```
» net.outputs
ans =
    []    [1x1 struct]
» net.targets
```

```
ans =
    []    [1x1 struct]
```

3.1.2.1.10 Funciones de la red neuronal

Se establece la función de inicialización como initlay con el siguiente código

```
» net.initFcn='initlay';
```

También se establece la función de desempeño a mse (mean square error) y la función de entrenamiento a trainrp (Relisient Backpropagation).

```
» net.performFcn='mse';
» net.trainFcn='trainrp';
» net.trainParam.epochs=2000;
» net.trainParam.goal=0.01;
» net.trainParam.show=50;
```

3.1.4.2 Funcionamiento de la red neuronal

3.1.4.2.1 Inicialización

Se inicializa la red neuronal con funciones randómicas para tener valores aleatorios en su inicio.

```
» net.inputWeights{1,1}.initFcn='rands';
» net.layerWeights{1,1}.initFcn='rands';
» net.biases{1,1}.initFcn='rands';
» net.biases{2,1}.initFcn='rands';
» net=init(net);
```

3.1.4.3 Entrenamiento de la red neuronal

El vector de entrada debe estar en el modo de conjunto de datos (batch mode) [10]. Para el entrenamiento es necesario dos vectores; el de entrada y el de salida.

Voltaje de salida al tiempo t

V0 = [0 0.4 1.4 2.6 3.4 4 4.9 5.4 5.7 6 6.1 6.2 6.2 6.2 6.2 6.2];

Voltaje de salida al tiempo t – 1

V1 = [0.4 1.4 2.6 3.4 4 4.9 5.4 5.7 6 6.1 6.2 6.2 6.2 6.2 6.2 6.2];

Voltaje de entrada al tiempo t

Vin = [0 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20];

Voltaje de entrada al tiempo t – 1

Vin1 = [20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20];

Voltaje de entrada al tiempo t – 2

Vin2 = [20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20] ;

El vector P de entrada resulta de la combinación en secuencia de vectores consecuentes de V1, Vin, Vin1, Vin2.

» P = {[0.4; 0; 20; 20][1.4; 20; 20; 20][2.6; 20; 20; 20]...[6.2; 20; 20; 20]}

El vector de salida T de salida de la red neuronal esta dado por:

» T = { 0 0.4 1.4 2.6 3.4 4 4.9 5.4 5.7 6 6.1 6.2 6.2 6.2 6.2 6.2 };

Con los valores de P y T que corresponden a los obtenidos del la mediciones del Motomatic con realimentación de velocidad, se procede a simular la red neuronal.

» Y=sim(net,P)

Y =

Columns 1 through 6

[3.8811e-006] [0.3630] [1.5452] [2.5037] [3.1908] [4.2788]

Columns 7 through 12

[4.9627] [5.3977] [5.8420] [5.9904] [6.1383] [6.1383]

Columns 13 through 16

[6.1383] [6.1383] [6.1383] [6.1383]

Se procede a configurar los parámetros de entrenamiento de la red neuronal, con el siguiente código se puede observar dichos parámetros:

```
» net.trainParam
ans =
epochs: 2000
show: 50
goal: 0.0100
time: Inf
min_grad: 1.0000e-006
max_fail: 5
delt_inc: 1.2000
delt_dec: 0.5000
delta0: 0.0700
deltamax: 50
```

Todos estos parámetros son propensos de cambios con su respectiva línea de código [11].

Posteriormente se entrena la red con siguiente código

```
» net=train(net,P,T);
```

Se observa el entrenamiento en la ventana de comandos de Matlab.

TRAINRP, Epoch 0/2000, MSE 10.0937/0.01, Gradient 18.3024/1e-006
TRAINRP, Epoch 50/2000, MSE 0.078669/0.01, Gradient 0.550605/1e-006
TRAINRP, Epoch 100/2000, MSE 0.0552871/0.01, Gradient 0.276856/1e-006
TRAINRP, Epoch 150/2000, MSE 0.0528667/0.01, Gradient 0.160679/1e-006
TRAINRP, Epoch 200/2000, MSE 0.0526312/0.01, Gradient 4.46072/1e-006
TRAINRP, Epoch 250/2000, MSE 0.0413126/0.01, Gradient 0.102067/1e-006
TRAINRP, Epoch 300/2000, MSE 0.0327837/0.01, Gradient 0.612294/1e-006
TRAINRP, Epoch 350/2000, MSE 0.0263793/0.01, Gradient 0.43644/1e-006
TRAINRP, Epoch 400/2000, MSE 0.0192549/0.01, Gradient 0.164681/1e-006
TRAINRP, Epoch 450/2000, MSE 0.0139981/0.01, Gradient 0.30179/1e-006
TRAINRP, Epoch 500/2000, MSE 0.0122925/0.01, Gradient 0.113125/1e-006
TRAINRP, Epoch 550/2000, MSE 0.0109965/0.01, Gradient 0.194821/1e-006
TRAINRP, Epoch 598/2000, MSE 0.00992915/0.01, Gradient 0.199753/1e-006
TRAINRP, Performance goal met.

En la figura 3.10 se aprecia el entrenamiento de la red neuronal.

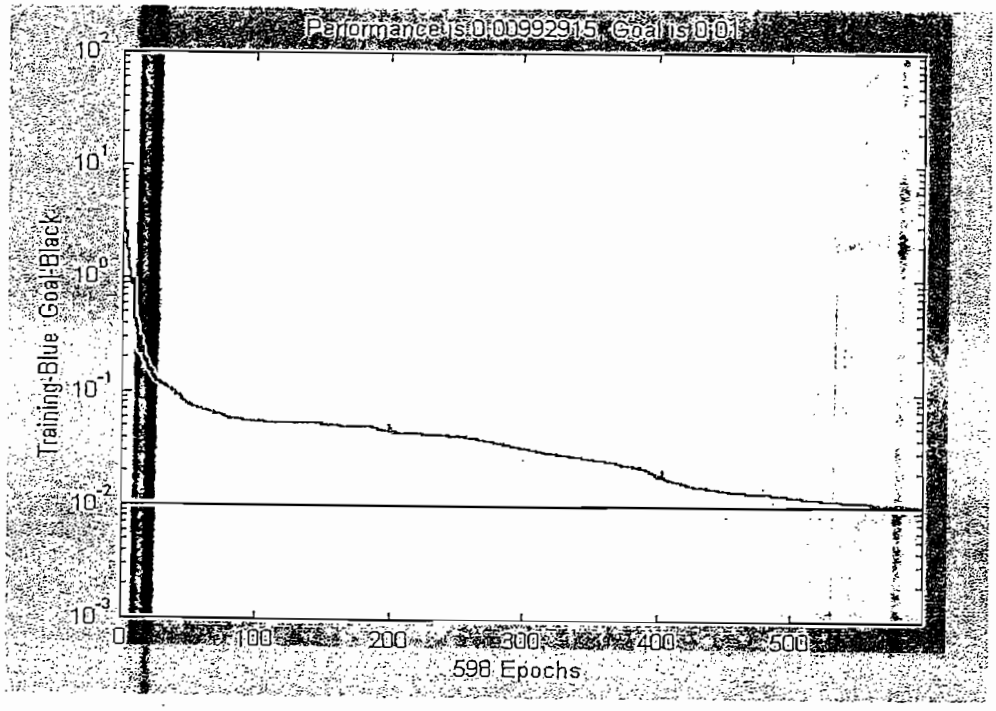


Figura 3.10

Desempeño en el entrenamiento de la red.

3.1.4.4 Simulación

3.1.4.4.1 Simulación del Motomatic con realimentación de velocidad en respuesta a una entrada paso.

Se puede simular la red neuronal con el comando:

```
»Y=sim(net,P)
Y =
  Columns 1 through 6
    [-0.0544]   [0.4697]   [1.4259]   [2.5614]   [3.1757]   [4.2392]
  Columns 7 through 12
    [4.9481]   [5.3968]   [5.8487]   [5.9978]   [6.1455]   [6.1455]
  Columns 13 through 16
    [6.1455]   [6.1455]   [6.1455]   [6.1455]
```

Con éstos valores podemos realizar una comparación (gráfico 3.11), entre los valores – objetivo del vector T (rojo) y los valores simulados del vector Y (azul).

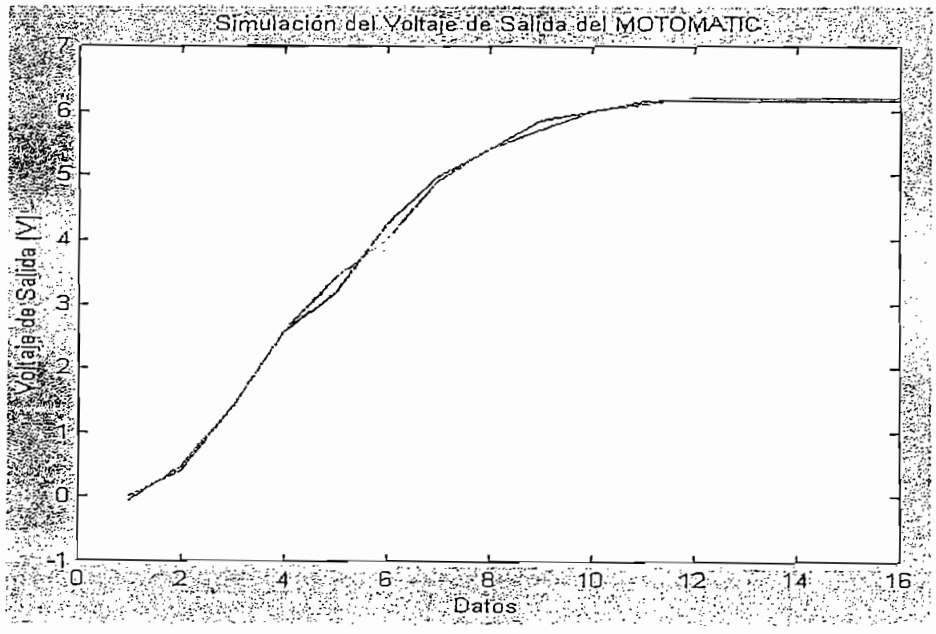


Figura 3.11

Simulación del Motomatic con realimentación de velocidad en respuesta a una entrada paso.

3.1.4.4.2 Simulación del Motomatic con realimentación de posición en respuesta a una entrada paso.

En este caso se sigue el procedimiento anterior, con los nuevos datos en entrada y salida de lo cual nos da como resultado la gráfica 3.12.

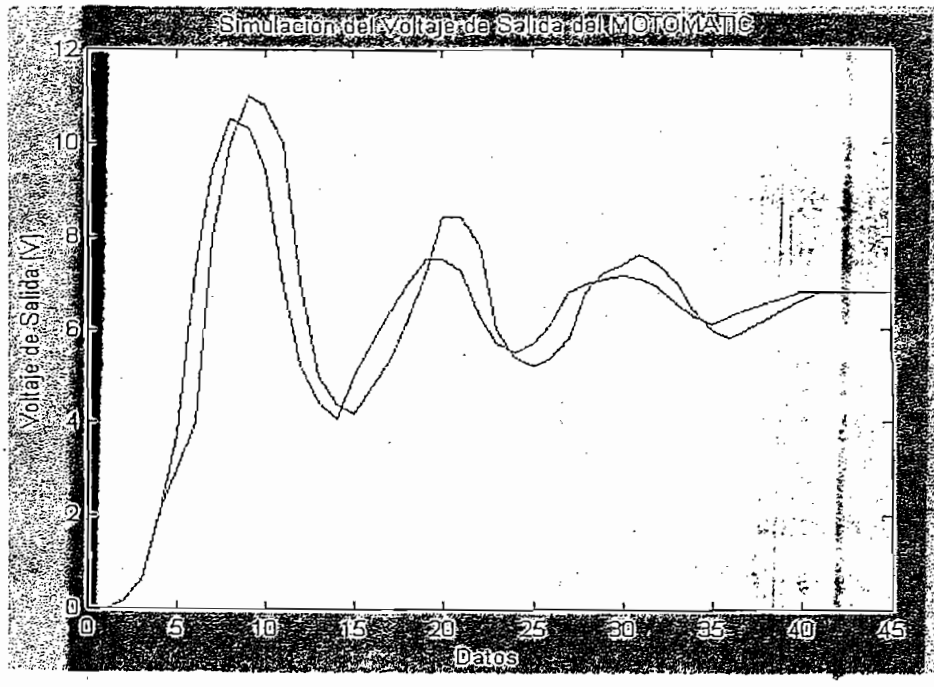


Figura 3.12

Simulación del Motomatic con realimentación de posición en respuesta a una entrada paso.

CAPITULO 4

DESARROLLO DE SOFTWARE EN MATLAB PARA LA SIMULACIÓN DEL MOTOMATIC

4.1 INTRODUCCIÓN

Para el desarrollo del software se utiliza un computador con las siguientes características o compatible.

- » Procesador 486 o superior
- » Memoria Ram de 16 Mb
- » CD ROM
- » Monitor SVGA o superior
- » Sistema operativo Windows 95, 98, 2000 o millenium

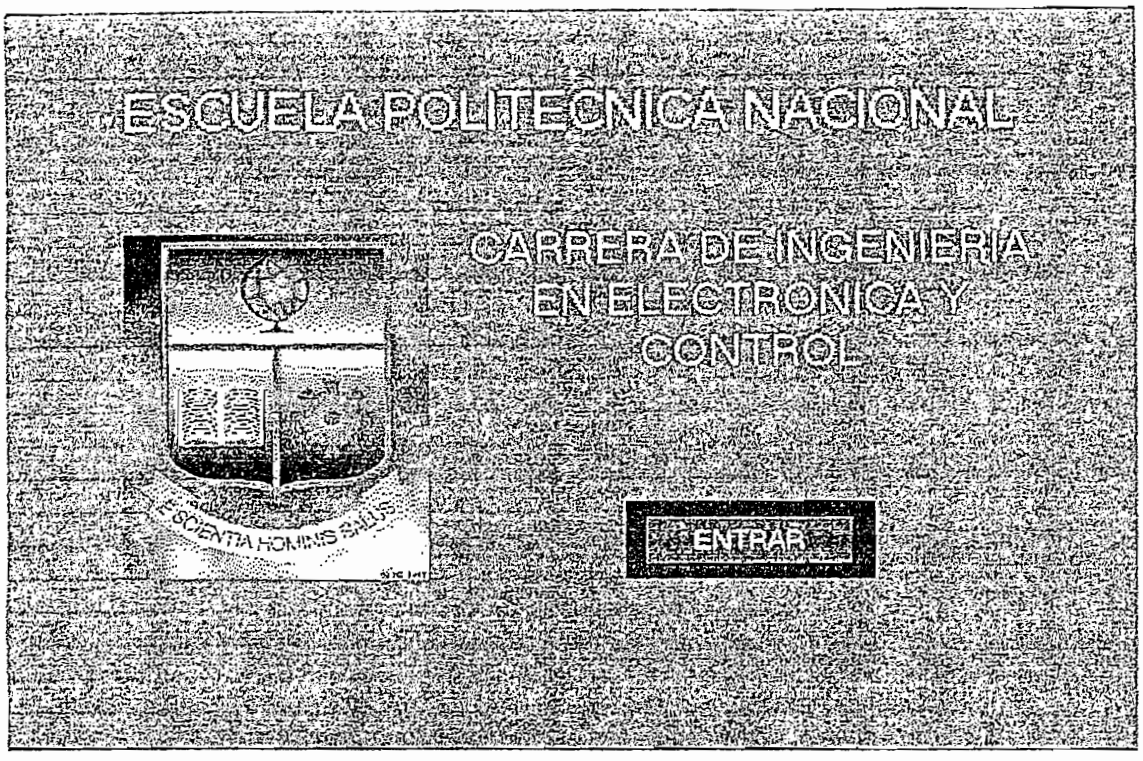
En dicho computador se dispone del programa Matlab 5.3 previamente instalado, necesario para el desarrollo de la simulación de la máquina de corriente continua (sistema MCSL – 100), también se utiliza la librería de redes neuronales propio de Matlab.

4.1.1 PANTALLAS DEL PROGRAMA

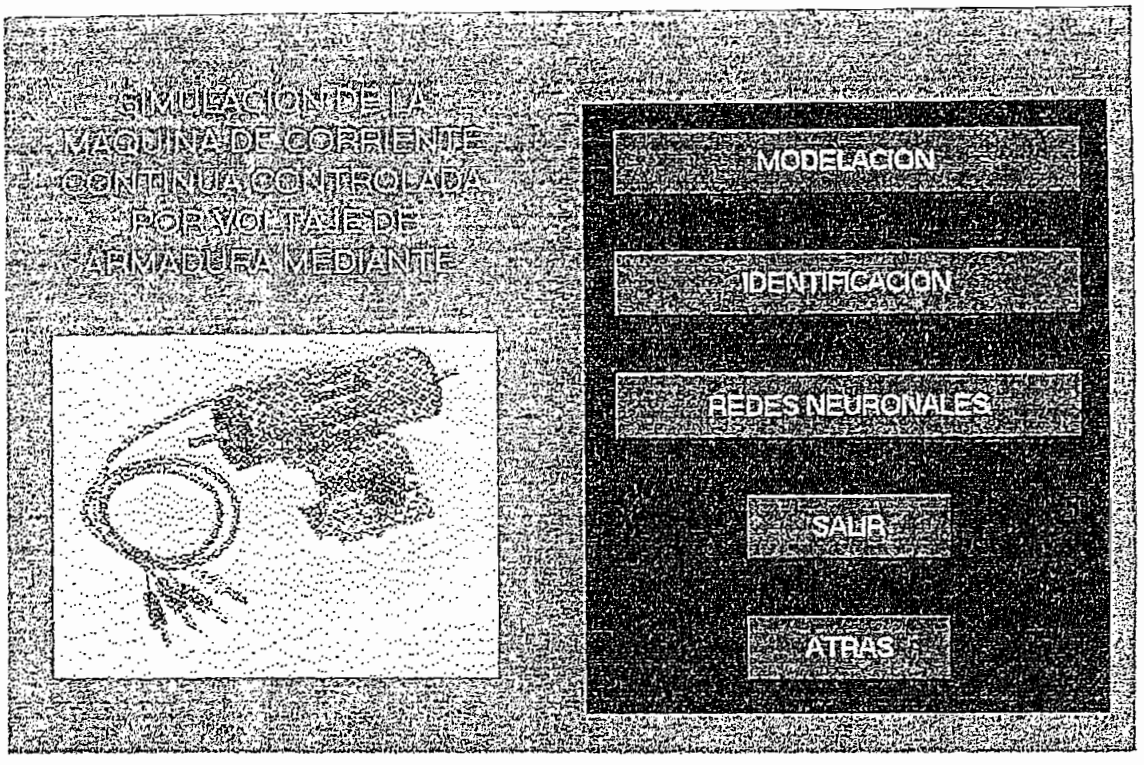
El programa desarrollado posee dos pantallas de presentación. Para ingresar al programa principal se digita en la Ventana de Comandos de Matlab:

» programa_principal

Dicho comando despliega la primera pantalla que tiene la única opción de entrar, esta opción despliega la segunda pantalla de presentación en la que se presenta dos opciones: la primera (siguiente) dará inicio al programa principal, mientras que la segunda (salir) da la oportunidad al usuario de salir del programa.

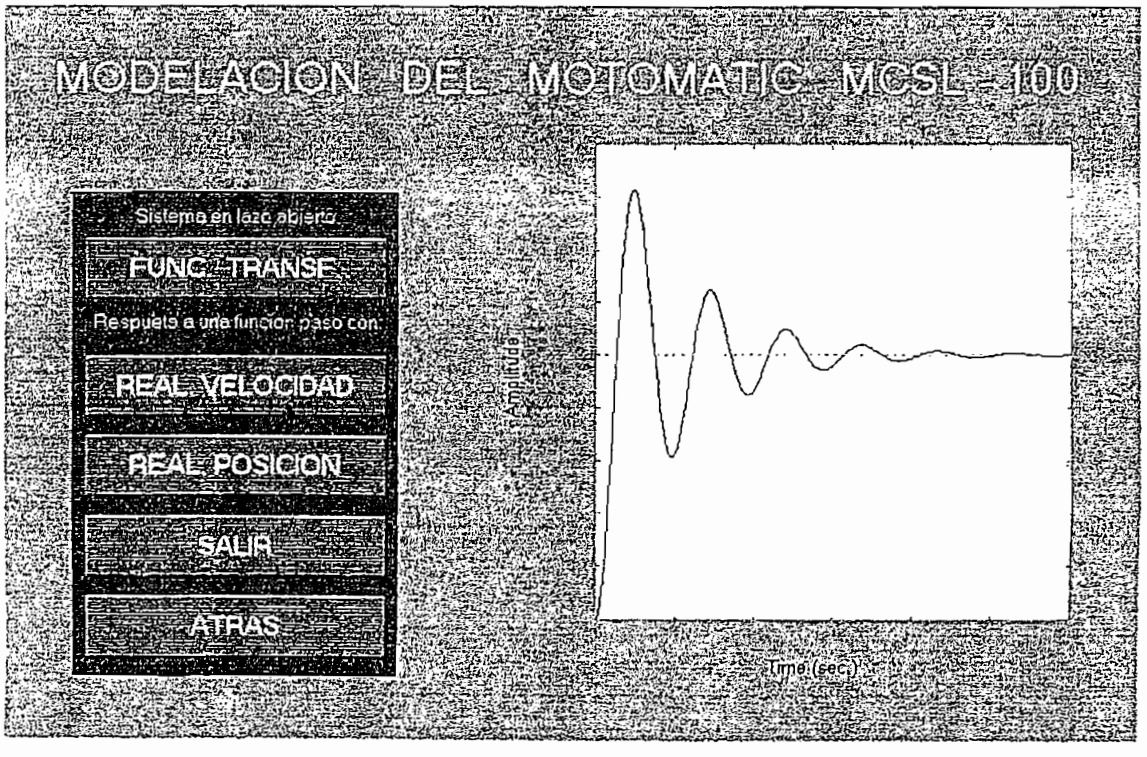


En estas pantallas el usuario tiene la opción de salir del programa o de continuar con los íconos respectivos. Si el usuario decide seguir en el programa, éste entra en un menú que se expone a continuación.



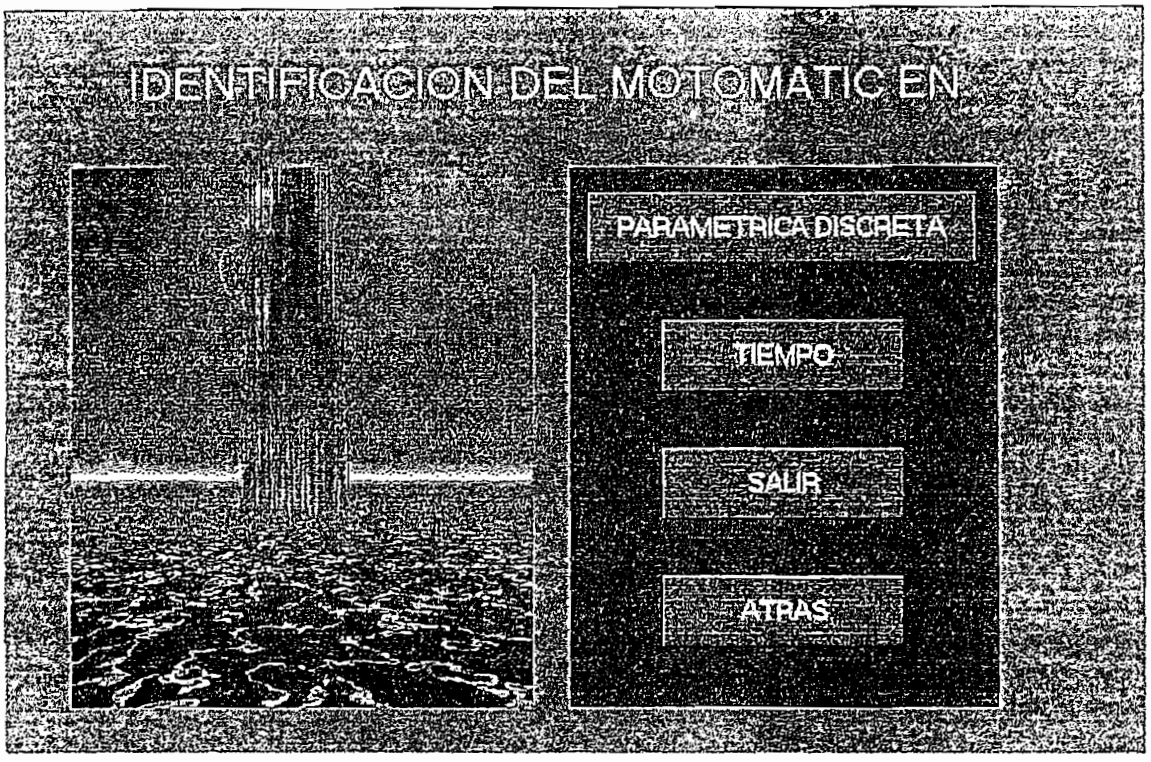
En este menú se tiene la opción de simular la máquina de corriente continua regulada por voltaje armadura (sistema MCSL – 100) mediante modelación, identificación y redes neuronales.

Si se escoge la opción de modelación se entra a la pantalla:



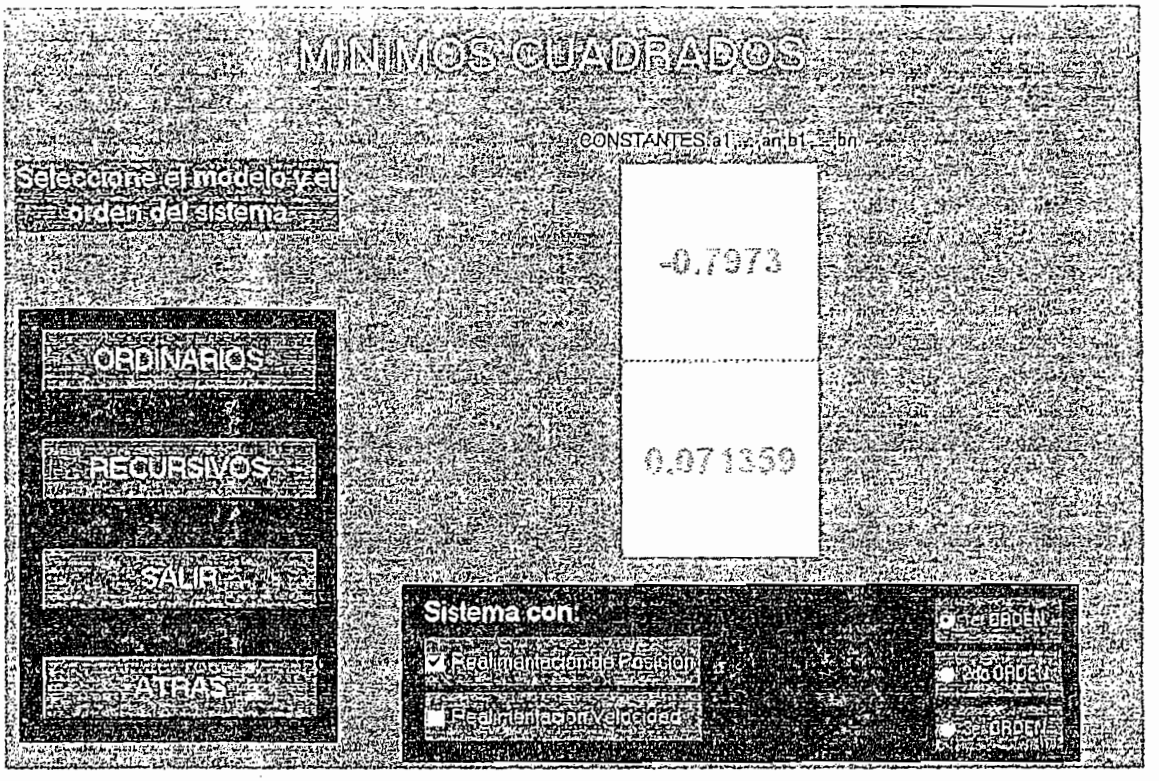
En esta pantalla se tiene las opciones de mostrar: la función de transferencia del Motomatic en lazo abierto, la respuesta a una función paso con realimentación de posición y velocidad.

Si se escoge la opción de identificación se presenta la pantalla:

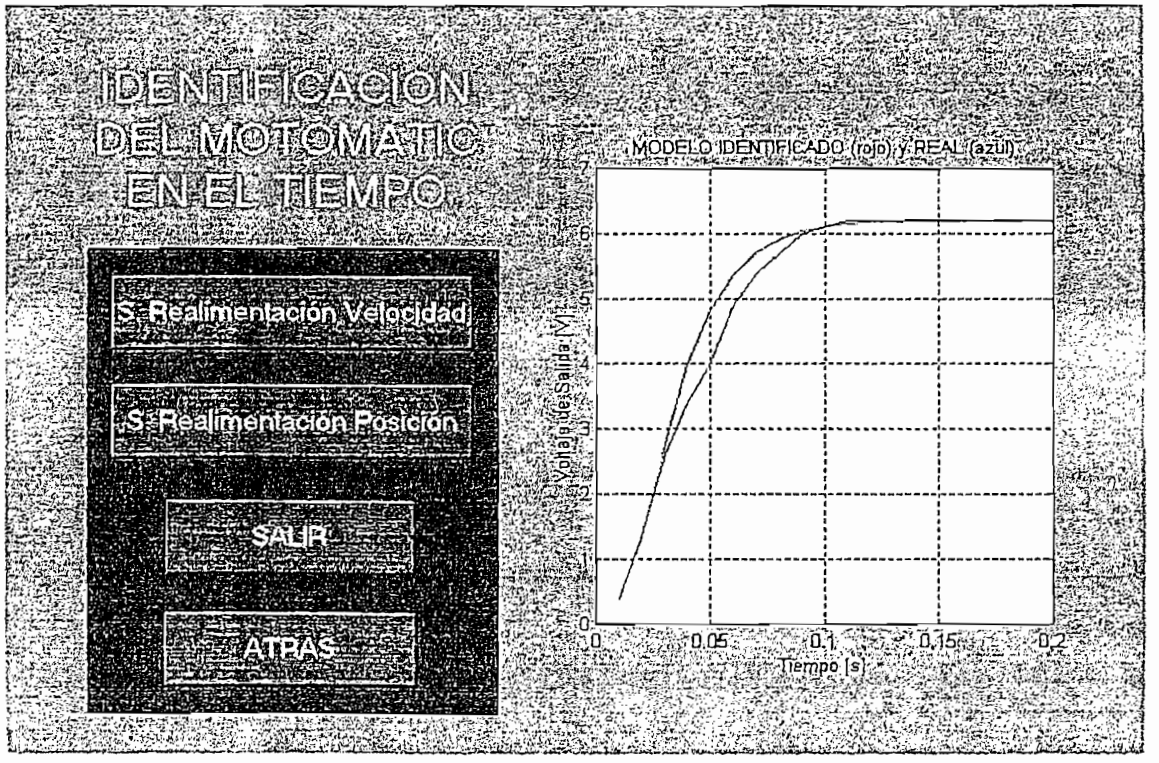


En la cuál se puede ingresar a realizar identificación en el tiempo o paramétrica discreta

En la identificación paramétrica discreta se ingresa al programa de mínimos cuadrados ordinarios y recursivos en el cual se puede escoger el sistema que se desea identificar, es preciso especificar el orden del modelo que se desea obtener ya que en esta pantalla existen tres opciones: 1er orden, 2do orden y 3er orden. Con todos estos parámetros seleccionados se procede a presionar el icono que desee ya sea ordinario o recursivo, que son los algoritmos con los cuales se obtienen las constantes a1,...,an,b1,...,bn del modelo discreto.

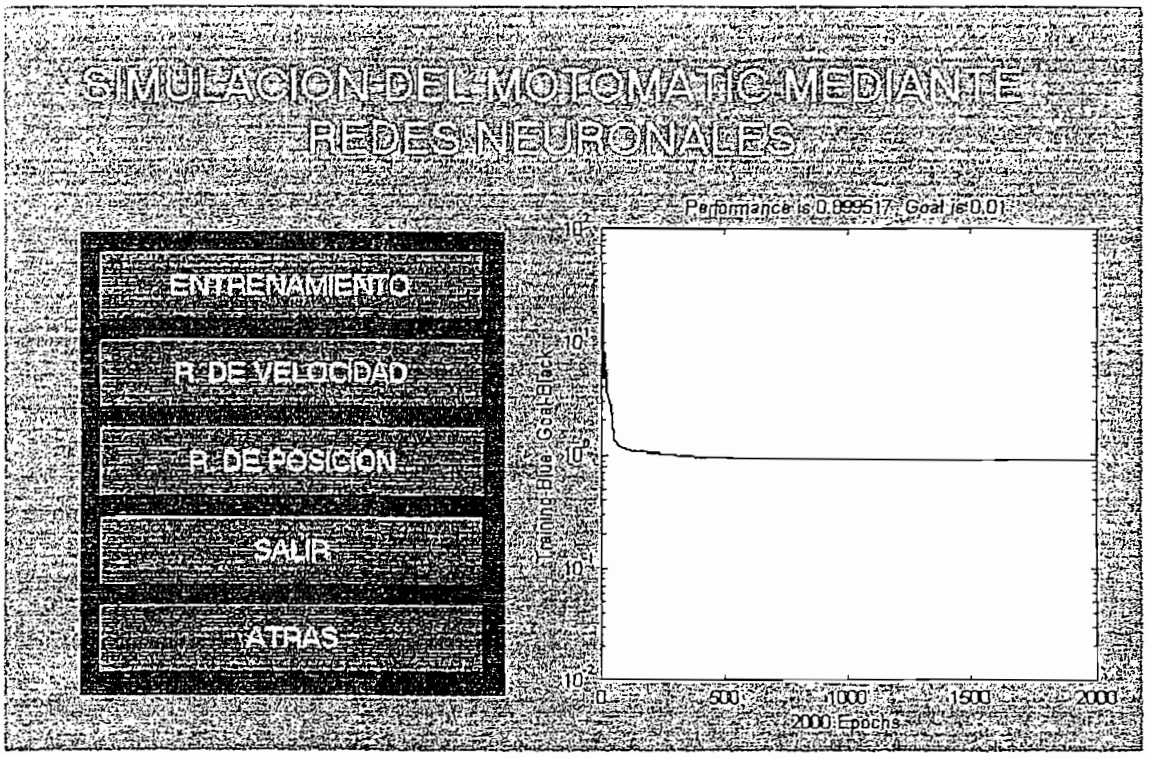


Para la identificación en el tiempo:



Se presenta la comparación de los modelos reales e identificados.

La última alternativa del menú principal es la simulación de la máquina de corriente continua controlada por voltaje de armadura mediante redes neuronales.



En la simulación del Motomatic por medio de redes neuronales, se debe entrenar la red neuronal para posteriormente proceder a simular los sistemas propuestos.

4.1.2 DESARROLLO DEL SOFTWARE

4.1.2.1 Desarrollo de una pantalla del menú

Matlab permite desarrollar de manera simple un conjunto de pantallas con botones, menús, ventanas, etc., que permite utilizar de manera muy simple programas realizados en el entorno Windows. Este conjunto de herramientas se denomina interface gráfico.

El comando GUIDE permite realizar interfaces de usuario de una forma muy cómoda y sencilla.

Los gráficos en Matlab tienen una estructura jerárquica formada por objetos de distintos tipos. Esta jerarquía tiene forma de árbol, con el aspecto mostrado en la figura 4.1.

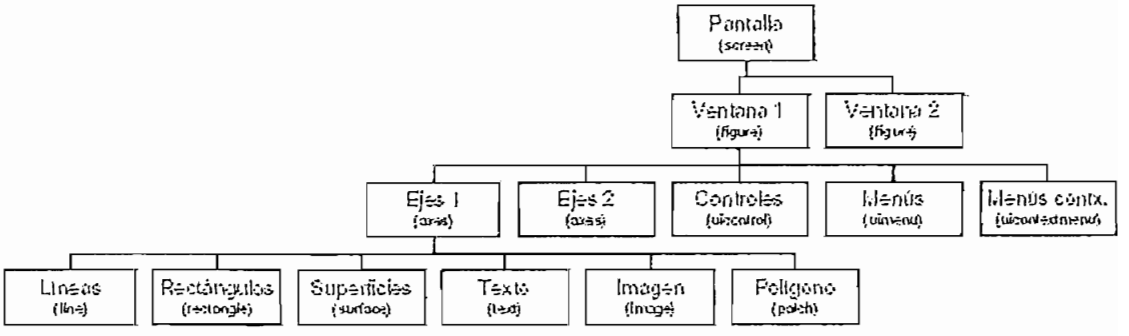


Figura 4.1

Jerarquía gráfica de Matlab

Según se muestra en la figura 4.1,el objeto más grande es la pantalla. Este objeto es la raíz de todos los demás y solo puede haber un objeto pantalla. Una pantalla puede contener una o más ventanas (figures). A su vez cada una de las ventanas puede tener uno o más ejes de coordenadas (axes) en los que presentan otros objetos de más bajo nivel. Una ventana también puede tener controles tales como botones, barras de desplazamiento, botones de selección o de opción, etc. Finalmente, los ejes pueden contener los seis tipos de elementos gráficos que permite Matlab: líneas (line), rectángulos (rectangle), polígonos (patches), superficies (surface), imágenes (image) y texto (text).

4.1.2.1.1 Ejemplo de la creación de una pantalla

Para la creación de cualquier pantalla se inicia digitando el comando GUIDE, este despliega: una pantalla que lleva el nombre de figura N° 1 por defecto (figura 4.2) y el panel de control (figura 4.3).

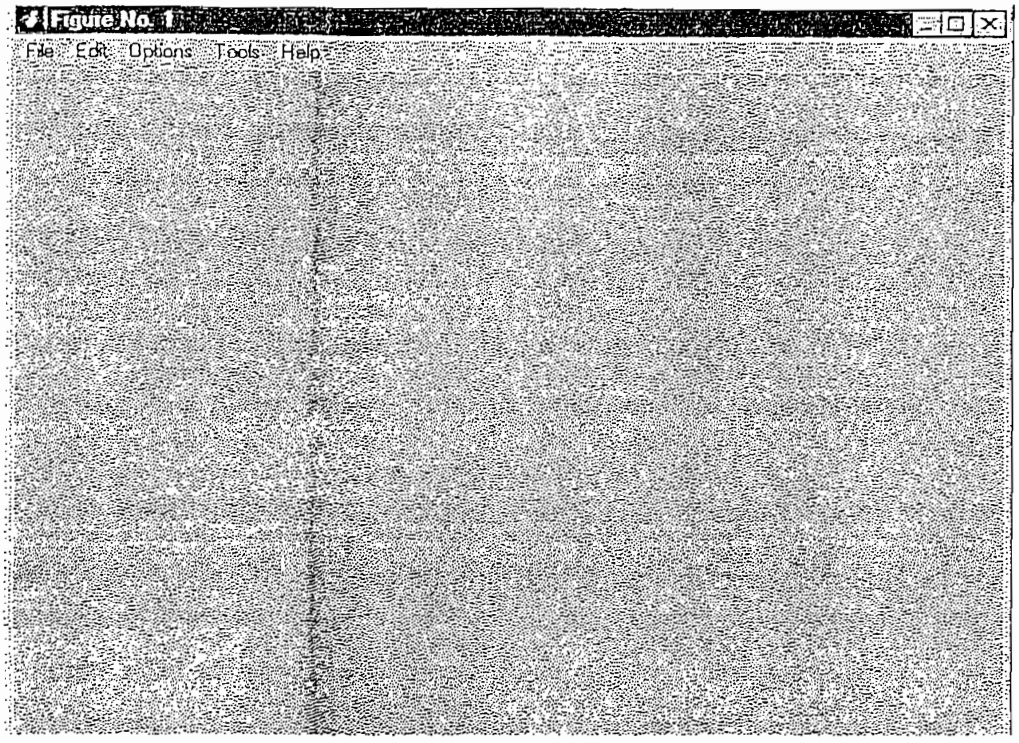


Figura 4.2

Pantalla de Matlab

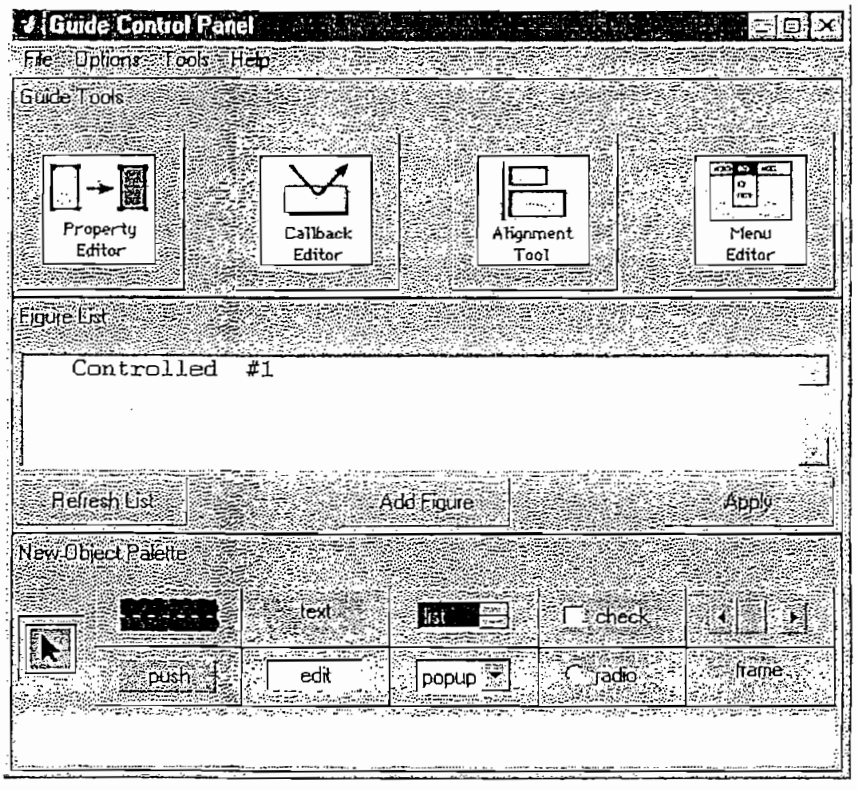


Figura 4.3

Panel de control

En el panel de control se dispone de todas las herramientas necesarias para diseñar un interface gráfico, ya que en la pantalla que se creo se puede ir insertando una o más ventanas simplemente con desplazarles del panel de control a la nueva pantalla. Es conveniente conocer como editar algunas propiedades de los objetos de Matlab como por ejemplo:

- *Color del objeto (BackgroundColor).* Controla el color del objeto. Por defecto este suele ser gris claro aunque este puede tomar varios colores.
- *Acción a efectuar por el comando (CallBack).* Este comando especifica la acción a efectuar por Matlab al actuar sobre el control. Se trata de una expresión u orden, almacenada en una cadena de caracteres o en una función, que se ejecutara al activar el control. CallBack sirve también para llamar otras pantallas que estén gravadas con la extensión .m.
- *Control activado / desactivado (Enable).* Este comando permite desactivar un control, de tal forma que una acción sobre el mismo (por ejemplo, apretar sobre el mismo con el ratón) no produce ningún efecto.
- *Posición del objeto (Position).* En esta opción se determina la posición y el tamaño del control dentro de la pantalla nueva.
- *Nombre del objeto (String).* Está opción define el nombre que aparecerá en el control. Cuando el control sea una opción de menú desplegable (popup menu), los nombres se sitúan en el orden dentro del string, separados por un carácter barra vertical (|).
- *Tipo de control (Style).* Está opción puede tomar los siguientes valores: pushbutton, togglebutton, radiobutton, checkbox, slider, edit, popupmenu, list, frames y text.
- *Visible (Visible).* Puede tomar dos valores: on / off. Indica si el control es visible o no en la ventana.

Una mejor descripción de los diferentes tipos de controles se puede observar en la ayuda de Matlab. La utilización de cada uno vendrá dada en función de sus características y aplicaciones.

Las propiedades mencionadas anteriormente se las puede visualizar si se realiza un doble clic en el objeto a editar. Las propiedades de los objetos se muestran en el editor de las propiedades de los gráficos (figura 4.4).

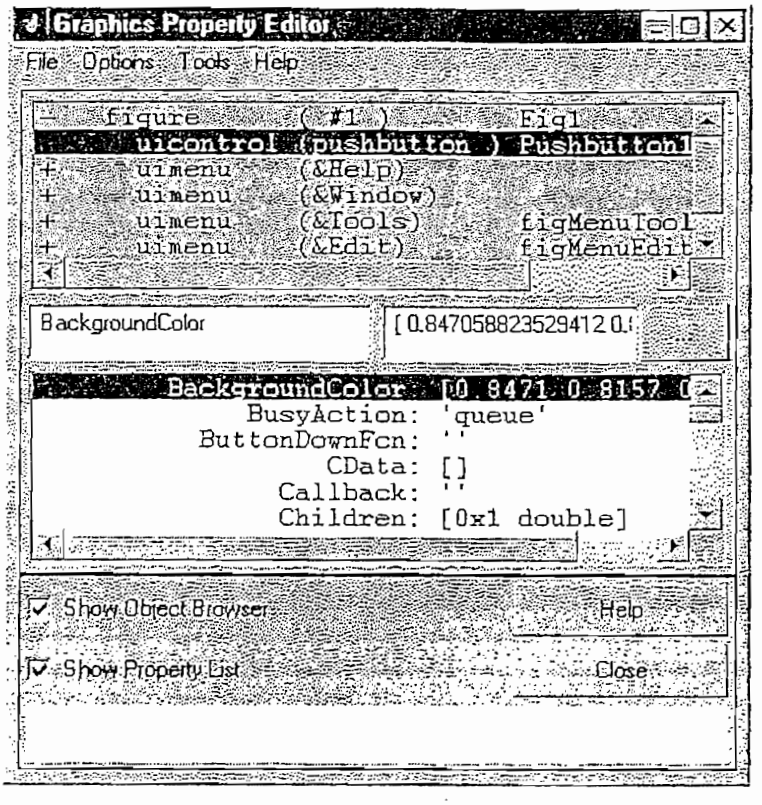


Figura 4.4

Editor de propiedades

Finalmente se grava la pantalla con un nombre, el mismo que por defecto tiene extensión .m esto significa que se la puede ejecutar desde la ventana de comandos de Matlab.

4.1.2.2 Rutina para el gráfico de la respuesta a un escalón para el sistema con realimentación de velocidad

La rutina a implementar se basa en el modelo a diagrama de bloques expuesto en el capítulo uno. Utilizando los comandos respectivos propios de Matlab se tiene la siguiente rutina:

Rutina rfprv

```
g=tf([4.882e5],[1 1334 1.921e4]);
Kg=0.148;
G=feedback(g,Kg);
step(G)
```

Para el sistema con realimentación de posición se realiza el mismo procedimiento antes mencionado.

4.1.2.3 Rutina de identificación

En la rutina de identificación es necesario ingresar: un vector de información $\mu(k)$, vector de salida y(k) y el orden del sistema. Para aplicar los respectivos algoritmos y obtener el modelo discreto del sistema ingresado.

Rutina minimos_cuadrados

```
%Escuela Politécnica Nacional
%Proyecto de Titulación
%Algoritmo Minimos Cuadrados
%Hernán Patricio Jácome Granizo

close all
msgbox('Algoritmo para Mínimos Cuadrados Ordinarios y
Recursivos','Patricio Jácome');
entrada = input('\nIngrese el [vector de información u(k)]: ');
salida = input('Ingrese el [vector de salida y(k)]: ');
orden = input('Ingrese el orden del modelo: ');
size(entrada); k = ans(2); b = 0; c = 0;
%formacion de X
if orden == 1
   a1=entrada(:,1:k-1);
   b1=salida(:,1:k-1);
   x=[b1;a1];
else
   if orden == 2
      a1=entrada(:,2:k-1);
      a2=entrada(:,1:k-2);
      b1=salida(:,2:k-1);
      b2=salida(:,1:k-2);
      x=[b1; b2;a1; a2];
   else
      if orden == 3
         a1=entrada(:,3:k-1);
         a2=entrada(:,2:k-2);
         a3=entrada(:,1:k-3);
         b1=salida(:,3:k-1);
         b2=salida(:,2:k-2);
         b3=salida(:,1:k-3);
         x=[b1; b2 ;b3;a1; a2;a3];
      else
```

```
        if orden == 4
           a1=entrada(:,4:k-1);
           a2=entrada(:,3:k-2);
           a3=entrada(:,2:k-3);
           a4=entrada(:,1:k-4);
           b1=salida(:,4:k-1);
           b2=salida(:,3:k-2);
           b3=salida(:,2:k-3);
           b4=salida(:,1:k-4);
           x=[b1; b2 ;b3;b4;a1; a2;a3;a4];
        else
           if orden == 5
              a1=entrada(:,5:k-1);
              a2=entrada(:,4:k-2);
              a3=entrada(:,3:k-3);
              a4=entrada(:,2:k-4);
              a5=entrada(:,1:k-5);
              b1=salida(:,5:k-1);
              b2=salida(:,4:k-2);
              b3=salida(:,3:k-3);
              b4=salida(:,2:k-4);
              b5=salida(:,1:k-5);
              x=[b1; b2 ;b3;b4;b5;a1; a2;a3;a4;b5];
           else
              msgbox('Solo hasta orden 5','E R R O R');
              return
           end
        end
     end
  end
end

y=salida(:,orden+1:k);
Y=transpose(y);
X=transpose(x);

%Minimos Cuadrados Ordinarios
teta=inv(x*X)*x*Y;
fprintf('\nMínimos Cuadrados Ordinarios\n')
if orden == 1
   fprintf('a1 = %f',-teta(1,:));
   fprintf('\nb1 = %f\n',teta(2,:));

else
   if orden == 2
      fprintf('a1 = %f',-teta(1,:));
      fprintf('\na2 = %f',-teta(2,:));
      fprintf('\nb1 = %f',teta(3,:));
      fprintf('\nb2 = %f\n',teta(4,:));
   else
      if orden == 3
         fprintf('a1 = %f',-teta(1,:));
         fprintf('\na2 = %f',-teta(2,:));
         fprintf('\na3 = %f',-teta(3,:));
         fprintf('\nb1 = %f',teta(4,:));
         fprintf('\nb2 = %f',teta(5,:));
         fprintf('\nb3 = %f\n',teta(6,:));
      else
         if orden == 4
            fprintf('a1 = %f',-teta(1,:));
            fprintf('\na2 = %f',-teta(2,:));
```

```
            fprintf('\na3 = %f',-teta(3,:));
            fprintf('\na4 = %f',-teta(4,:));
            fprintf('\nb1 = %f',teta(5,:));
            fprintf('\nb2 = %f',teta(6,:));
            fprintf('\nb3 = %f',teta(7,:));
            fprintf('\nb4 = %f\n',teta(8,:));
         else
            fprintf('a1 = %f',-teta(1,:));
            fprintf('\na2 = %f',-teta(2,:));
            fprintf('\na3 = %f',-teta(3,:));
            fprintf('\na4 = %f',-teta(4,:));
            fprintf('\na5 = %f',-teta(5,:));
            fprintf('\nb1 = %f',teta(6,:));
            fprintf('\nb2 = %f',teta(7,:));
            fprintf('\nb3 = %f',teta(8,:));
            fprintf('\nb4 = %f',teta(9,:));
            fprintf('\nb5 = %f\n',teta(10,:));
         end
      end
   end
end

%Minimos Cuadrados Recursivos
alfa=100; I=eye(2*orden);
P=alfa*I; teta=zeros(2*orden,1);
for i = 1:1:k-orden
   L = P * x(:,i) *inv(1+X(i,:) * P * x(:,i));
   e=Y(i,:)-X(i,:)*teta;
   teta=teta+L*e;
   P=((eye(2*orden)-L*X(i,:))*P)/0.95;
end

fprintf('\n\nMinimos Cuadrados Recursivos\n')

if orden == 1
   fprintf('a1 = %f',-teta(1,:));
   fprintf('\nb1 = %f\n',teta(2,:));

else
   if orden == 2
      fprintf('a1 = %f',-teta(1,:));
      fprintf('\na2 = %f',-teta(2,:));
      fprintf('\nb1 = %f',teta(3,:));
      fprintf('\nb2 = %f\n',teta(4,:));
   else
      if orden == 3
         fprintf('a1 = %f',-teta(1,:));
         fprintf('\na2 = %f',-teta(2,:));
         fprintf('\na3 = %f',-teta(3,:));
         fprintf('\nb1 = %f',teta(4,:));
         fprintf('\nb2 = %f',teta(5,:));
         fprintf('\nb3 = %f\n',teta(6,:));
      else
         if orden == 4
            fprintf('a1 = %f',-teta(1,:));
            fprintf('\na2 = %f',-teta(2,:));
            fprintf('\na3 = %f',-teta(3,:));
            fprintf('\na4 = %f',-teta(4,:));
            fprintf('\nb1 = %f',teta(5,:));
            fprintf('\nb2 = %f',teta(6,:));
            fprintf('\nb3 = %f',teta(7,:));
```

```
            fprintf('\nb4 = %f\n',teta(8,:));
        else
            fprintf('a1 = %f',-teta(1,:));
            fprintf('\na2 = %f',-teta(2,:));
            fprintf('\na3 = %f',-teta(3,:));
            fprintf('\na4 = %f',-teta(4,:));
            fprintf('\na5 = %f',-teta(5,:));
            fprintf('\nb1 = %f',teta(6,:));
            fprintf('\nb2 = %f',teta(7,:));
            fprintf('\nb3 = %f',teta(8,:));
            fprintf('\nb4 = %f',teta(9,:));
            fprintf('\nb5 = %f\n',teta(10,:));
        end
    end
  end
end
```

Esta misma rutina es implementada en el programa principal.

4.1.2.4 Rutina para redes neuronales

Para redes neuronales se tiene primeramente que definir: el número de capas, el número de neuronas por capa, el grado de conectividad y el tipo de conexiones entre neuronas. Para poder configurar todos los parámetros, como se tiene a continuación.

Rutina iniciar

Creación de una nueva red neuronal en Matlab.

```
net=network;
```

Número de entradas y capas.

```
net.numInputs=4;
net.numLayers=2;
```

Conexión de umbrales.

```
net.biasConnect(1)=1;
net.biasConnect(2)=1;
net.biasConnect=[1;1];
```

Conexión de pesos, de entrada y de capas.

```
net.inputConnect(1,1)=1;
net.inputConnect(1,2)=1;
net.inputConnect(1,3)=1;
net.inputConnect(1,4)=1;
net.inputConnect=[1 0 0 0; 0 0 0 0];
net.layerConnect(2,1)=1;
net.layerConnect=[0 0; 1 0];
```

Conexión a la salida y valores objetivo.

```
net.outputConnect=[0 1];
```

Conexión del objetivo con el código.

```
net.targetConnect=[0 1];
```

Especificación del rango de los valores de las entradas.

```
net.inputs{1}.range=[0 7; 0 20; 0 20; 0 20];
```

Tamaño de cada capa.

```
net.layers{1}.size=6;
net.layers{2}.size=1;
```

Función de transferencia e inicialización de cada capa.

```
net.layers{1}.transferFcn='logsig';
net.layers{1}.initFcn='initnw';
net.layers{2}.transferFcn='purelin';
net.layers{2}.initFcn='initnw';
```

Función de desempeño y de entrenamiento.

```
net.initFcn='initlay';
net.performFcn='mse';
```

Número de iteraciones al entrenar.

```
net.trainParam.epochs=2000;
```

Iteraciones entre las cuales se observa el progreso.

```
net.trainParam.show=50;
```

Tipo de entrenamiento de la red.

```
net.trainFcn='trainrp';
```

Inicialización de la red neuronal con funciones randómicas.

```
net.inputWeights{1,1}.initFcn='rands';
net.layerWeights{2,1}.initFcn='rands';
net.biases{1,1}.initFcn='rands';
net.biases{2,1}.initFcn='rands';
net=init(net);
```

Llamada a la subrutina datos.

```
datos
```

Meta de desempeño

```
net.trainParam.goal=(0.1^2);
```

Para comprobar el funcionamiento de la red, se realiza simulación

```
Y=sim(net,P);
```

Se entrena la red.

```
net=train(net,P,T);
```

Subrutina datos

```
%Voltaje de salida al tiempo t

V0 = [0 0.2 0.6 2 3 4 8 10 11 10.8 10 7 5 4.4 4.2 4.8 5.4 6.2 7.2 8.4
8.4 7.8 6 5.4 5.2 5.4 5.8 6.8 7.2 7.4 7.6 7.4 7 6.4 6 5.8 6 6.2 6.4
6.6 6.8 6.8 6.8 6.8 6.8];

%Voltaje de salida al tiempo t - 1

V1 = [0.2 0.6 2 3 4 8 10 11 10.8 10 7 5 4.4 4.2 4.8 5.4 6.2 7.2 8.4
8.4 7.8 6 5.4 5.2 5.4 5.8 6.8 7.2 7.4 7.6 7.4 7 6.4 6 5.8 6 6.2 6.4
6.6 6.8 6.8 6.8 6.8 6.8 6.8];

%Voltaje de entrada al tiempo t

Vin = [0 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
20];

%Voltaje de entrada al tiempo t - 1

Vin1 = [20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
20];

%Voltaje de entrada al tiempo t - 2

Vin2 = [20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
20] ;


Datos = 1:1:45;

Q = length(V0);

   for i=1:Q
      P1(i)={[V1(i); Vin(i); Vin1(i); Vin2(i)]};
   end

   cerox = zeros(1,Q);
   ceronx = con2seq(cerox);


P = [P1; ceronx; ceronx; ceronx];
T = con2seq(V0);
```

Finalmente se simula la red neuronal y compara con los datos medidos

Rutina simula

```
Y=sim(net,P);
Tobj=[T{:}];
Ysim=[Y{:}];
plot(Datos,Tobj,'r',Datos,Ysim,'b')
xlabel('Datos');
ylabel('Voltaje de Salida [V]');
title('Simulación del Voltaje de Salida del MOTOMATIC');
```

CAPITULO 5

RESULTADOS Y EVALUACIÓN DE LAS DIFERENTES METODOLOGÍAS

5.1 RESULTADOS DE LA MODELACIÓN

5.1.1 SISTEMA MCSL – 100 CON REALIMENTACIÓN DE VELOCIDAD

Al realizar la simulación del diagrama de bloques (figura 5.1) del sistema MCSL – 100 con realimentación de velocidad en respuesta a una entrada paso unitario se observa que el sistema propuesto es no oscilatorio (figura 5.2). La realimentación se la realiza con el comando feedback, por medio de un tacómetro (Kv=0.148).

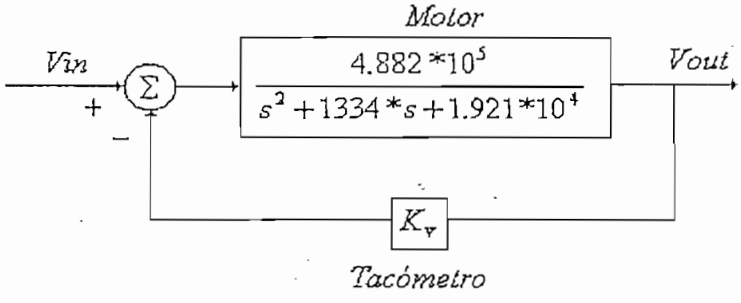


Figura 5.1

Diagrama de bloques del sistema MCSL – 100 con realimentación de velocidad

En la figura 5.2 se aprecia la comparación entre la simulación del diagrama de bloques del Motomatic con realimentación de velocidad (color azul) y la gráfica de los datos tomados experimentalmente de dicho sistema.

De la figura 5.2 se puede decir que las gráficas no coinciden y es debido a que los experimentos realizados para determinar el modelo a diagrama de bloques se lo realizo tiempos atrás, razón por la cuál en la actualidad dicho modelo no va a tener las mismas características.

Sin embargo como se aprecia el orden del modelo no cambia. Se debe considerar en todo caso que existe imprecisión en las mediciones.

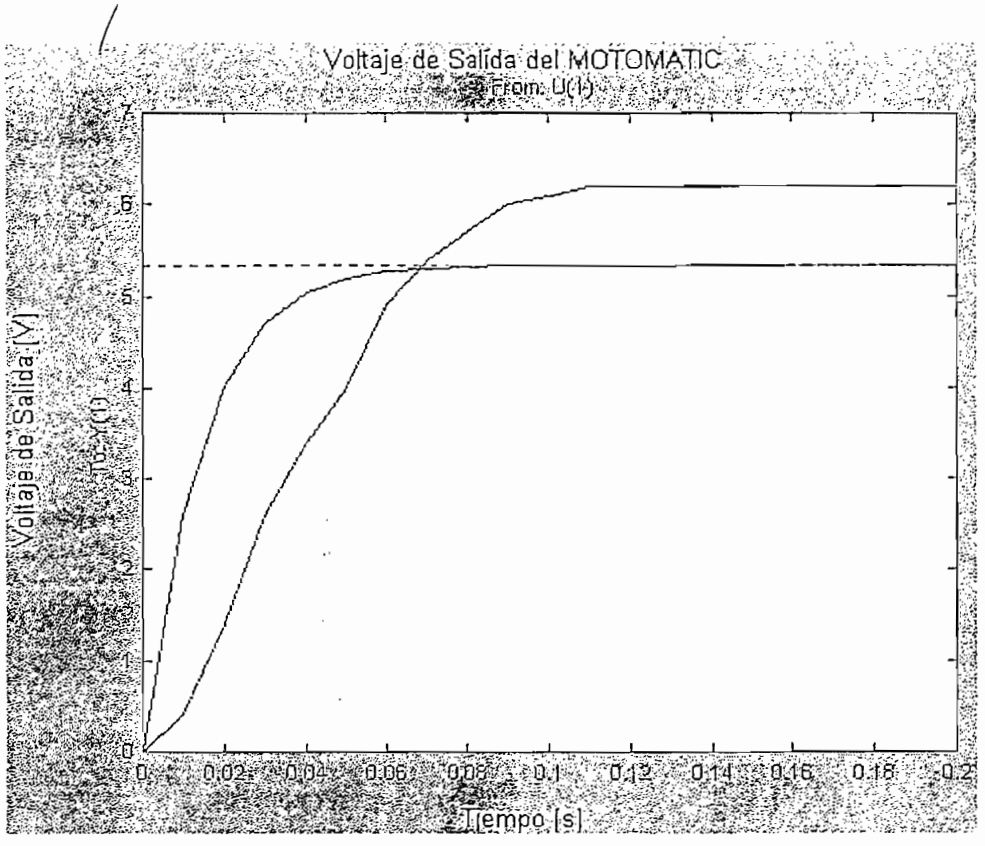


Figura 5.2

Simulación del diagrama de bloques del sistema MCSL – 100 con realimentación de velocidad

5.1.2 SISTEMA MCSL – 100 CON REALIMENTACIÓN DE POSICIÓN

En este caso se procede de igual forma que para el sistema anterior, con la diferencia que el diagrama de bloques es un poco más complejo (figura 5.3) ya que posee una etapa de integración y otra de reducción de velocidad.

Este sistema es oscilatorio amortiguado, posee un sobre pico pronunciado y un tiempo de establecimiento pequeño.

En la gráfica 5.4 se aprecia la comparación de la simulación del diagrama de bloques (color azul) con los datos medidos (color rojo).

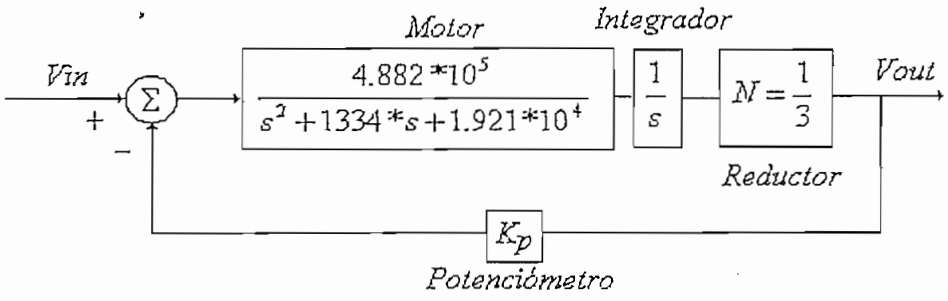


Figura 5.3

Diagrama de bloques del sistema MCSL – 100 con realimentación de posición

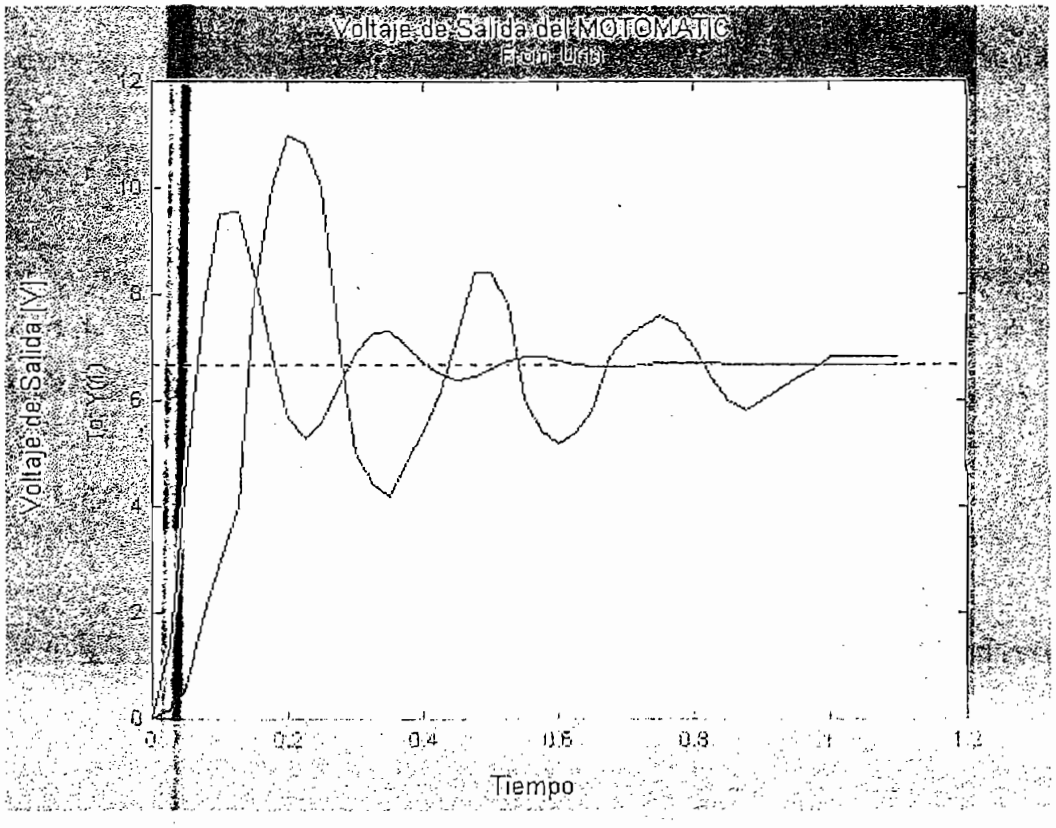


Figura 5.4

Simulación de diagrama de bloques del sistema MCSL – 100 con realimentación de posición

De la figura 5.4 se aprecia, que existe un retardo del sistema real (color rojo) con respecto al sistema simulado (color azul), debido a las no linealidades del motor y a la interfase de los elementos de transmisión de movimiento [13], en el

reductor de velocidad debido a la banda de transmisión y a los acoples entre el motor y el reductor de velocidad.

En la banda de transmisión hay un rozamiento debido al seguimiento a bajas velocidades rozamiento stip – slip, que se debe al efecto Stribeck [14]. Esto provoca un retardo en el tiempo.

Para los acoples, del modelo integrado de fricción de las propiedades dependientes del tiempo nos interesa la fricción estática creciente con aumento del tiempo dwell (tiempo que gasta una juntura en la condición de pegado) [14].

Todo esto se debe tomar en consideración ya que sé esta analizando un modelo real, y de lo expuesto anteriormente se tiene que se debe introducir no linealidades al modelo del sistema.

La frecuencia de oscilación también cambia como se ve, debido a las no linealidades, pero como éste sistema se atenúa no ponemos atención a la frecuencia de oscilación.

La no-linealidad del motor a causa del amplificador no es muy sensible ya que no afecta al sistema MCSL – 100 con realimentación de velocidad.

Existen formas de compensar y disminuir el efecto stip – slip [14]. Pero en este trabajo no se lo realizará ya que es un tema muy extenso.

5.2 RESULTADOS DE LA IDENTIFICACIÓN

5.2.1 EN TIEMPO

5.2.1.1 Sistema MCSL – 100 con realimentación de velocidad

Al identificar en el tiempo el sistema propuesto dió como resultado:

$$C(t) = 6{,}2 - 53{,}8534056 * e^{\frac{t}{0{,}01624837}} + 51{,}3820632 * e^{\frac{t}{0{,}01231533}}$$

Donde C(t) representa el voltaje de salida, y si realizamos la gráfica del modelo real vs. el modelo identificado se tiene la figura 5.5.

Figura 5.5

Identificación en tiempo del sistema MCSL – 100 con realimentación de velocidad

En la figura 5.5 se observa que existe una región en donde las dos gráficas (modelo real e identificado) no coinciden, esto se debe a que los datos tomados para él Motomatic no son muy buenos. Sin embargo esta aproximación es aceptable.

5.2.1.2 Sistema MCSL – 100 con realimentación de posición

En éste sistema, al ser oscilatorio amortiguado no se puede aplicar el mismo método de identificación anterior (método de determinación de las constantes de tiempo). Para este caso al sistema se lo debe aproximar a un sistema de segundo grado oscilatorio amortiguado:

$$G(s)=\frac{5600}{s^2+8s+849.7}$$

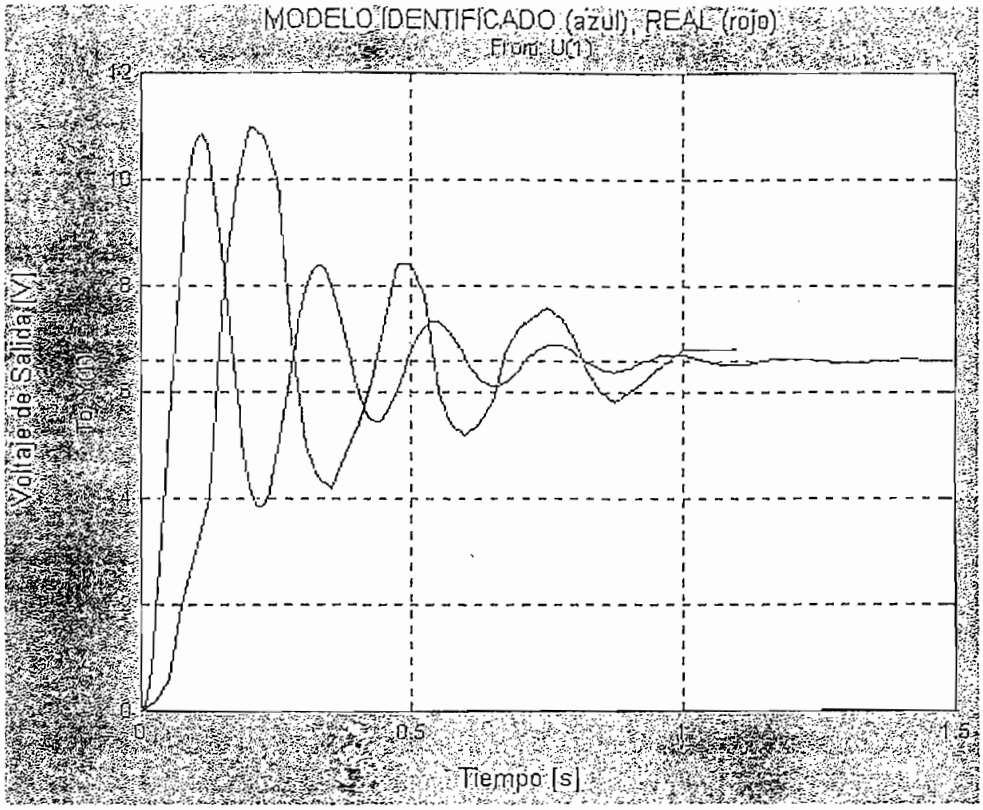


Figura 5.6

Identificación en tiempo del sistema MCSL – 100 con realimentación de posición

La gráfica muestra una aproximación aceptable del modelo identificado con el modelo real, pero el modelo identificado está retrasado un tiempo Td con respecto al modelo real por las no linealidades existentes, entonces es conveniente utilizar la aproximación de segundo orden de Padé [4], determinando un retardo promedio.

Para una aproximación de Padé de segundo orden:

$$e^{-s*Td} = \frac{1 - \frac{Td*s}{2} + \frac{(Td*s)^2}{8}}{1 + \frac{Td*s}{2} + \frac{(Td*s)^2}{8}}$$

Para Td=0.1 se tiene:

$$e^{-s*0.1} = \frac{1 - \frac{0.1*s}{2} + \frac{(0.1*s)^2}{8}}{1 + \frac{0.1*s}{2} + \frac{(0.1*s)^2}{8}} = \frac{0.01*s^2 - 0.4*s + 8}{0.01*s^2 + 0.4*s + 8}$$

Entonces el modelo final es:

$$G(s) = \frac{5600}{s^2 + 8s + 849.7} * e^{-s*Td}$$

$$G(s) = \frac{5600}{s^2 + 8s + 849.7} * \frac{0.01*s^2 - 0.4*s + 8}{0.01*s^2 + 0.4*s + 8}$$

Del cual sí se realiza la gráfica se tiene:

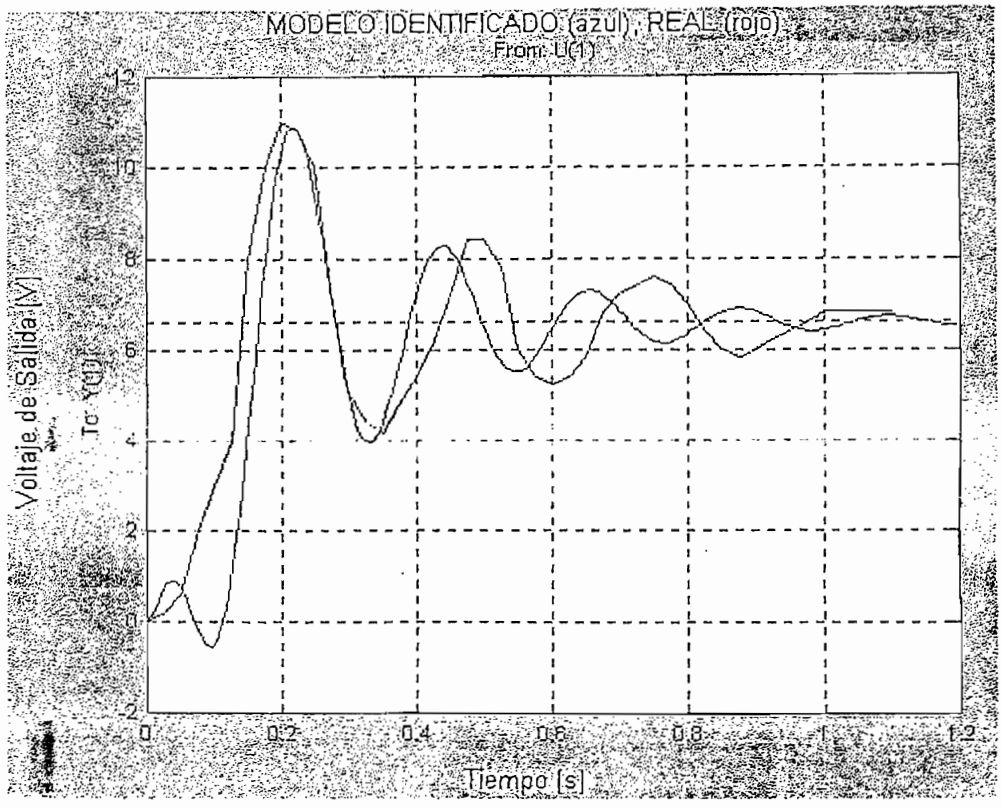


Figura 5.7

Identificación en tiempo del sistema MCSL – 100 con realimentación de posición y aproximación de segundo grado de Padé.

5.2.2 MINIMOS CUADRADOS

5.2.2.1 Sistema MCSL – 100 con realimentación de velocidad

Para este método solo es necesario tener el vector de información, el vector de salida y conocer de antemano el orden del sistema para obtener el modelo discreto con la aplicación de la rutina minimos_cuadrados. Con todos estos parámetros se tiene que el modelo discreto es:

$$y(k) = -0.92y(k-1) + 0.13y(k-2) + 0.05\mu(k-1) + 0.02\mu(k-2)$$

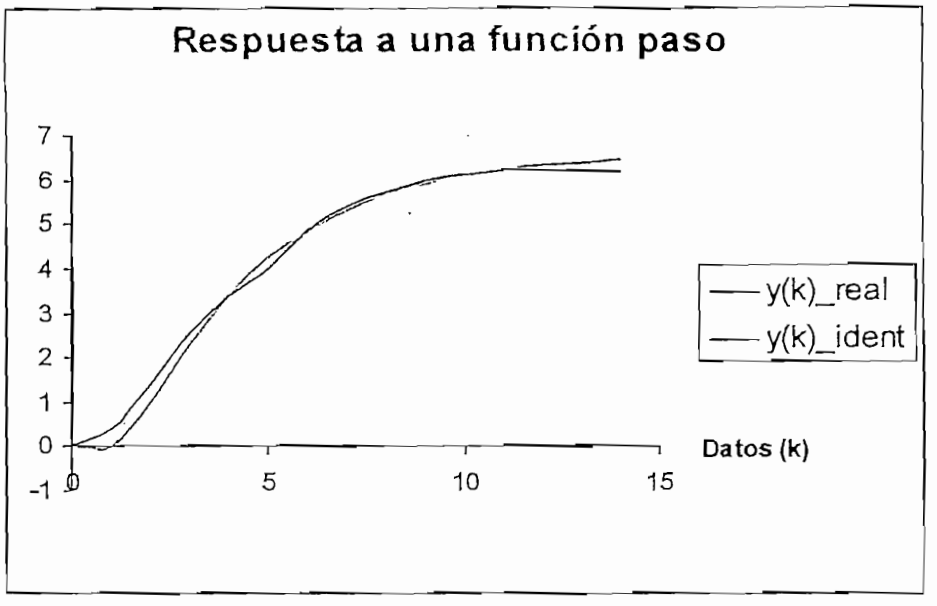


Figura 5.8

Modelo discreto del sistema MCSL – 100

El modelo discreto (color marrón) se observa que es bueno para el rango de valores establecido previamente, pero si se comienza a salir de dicho rango la curva se va deformando un poco. Otra acotación es que el modelo discreto tarda un pequeño tiempo en ajustarse a la gráfica de los datos medidos ya que como el sistema es de segundo orden debe esperar que existan los valores de y(0) y y(1), para comenzar a calcular los siguientes valores de la salida.

5.2.2.2 Sistema MCSL – 100 con realimentación de posición

En este caso se procede como en el caso anterior y se tiene el siguiente modelo discreto:

$$y(k) = -1.19y(k-1) + 0.32y(k-2) + 0.24y(k-3) + 0.034\mu(k-1) + 0.034\mu(k-2) + 0.052\mu(k-3)$$

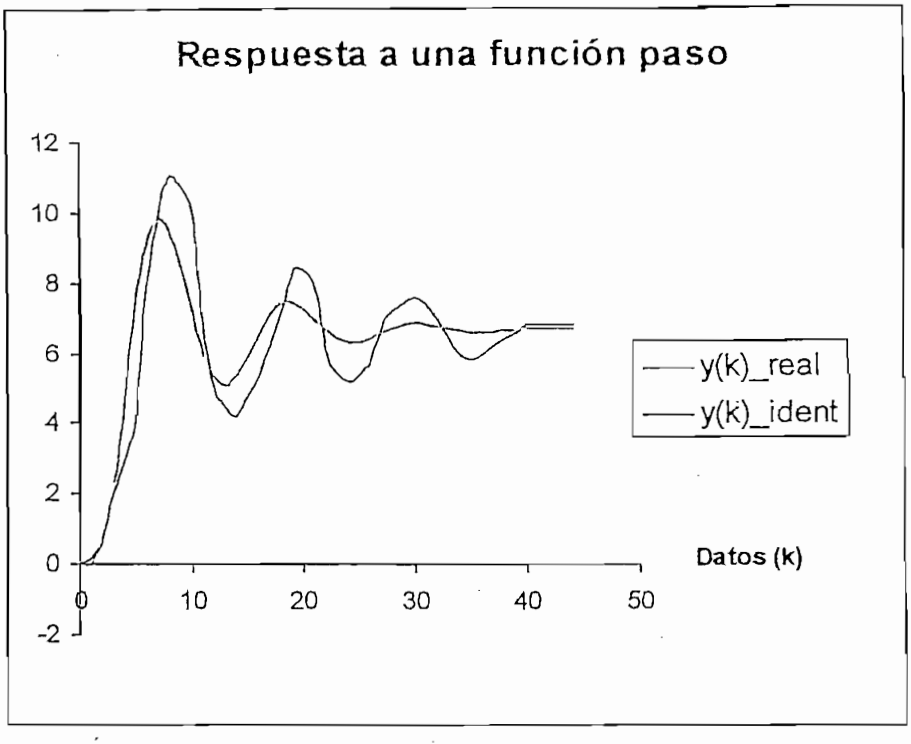


Figura 5.9

Modelo discreto del sistema MCSL – 100

En la comparación del modelo real (color azul) y el modelo identificado (color marrón) se tiene que el modelo identificado trata de seguir al modelo real pero como para este sistema los datos medidos no son muy buenos y es un sistema con no – linealidades la aproximación es aceptable.

El gráfico de los modelos discretos se los realiza sobre la base de la tabla de datos, aplicando el respectivo modelo de cada sistema, para valores discretos de k (Anexo 1).

5.3 RESULTADOS DE LA SIMULACIÓN CON REDES NEURONALES

5.3.1 SISTEMA MCSL – 100 CON REALIMENTACIÓN DE VELOCIDAD

La simulación mediante la red neuronal se presenta a continuación:

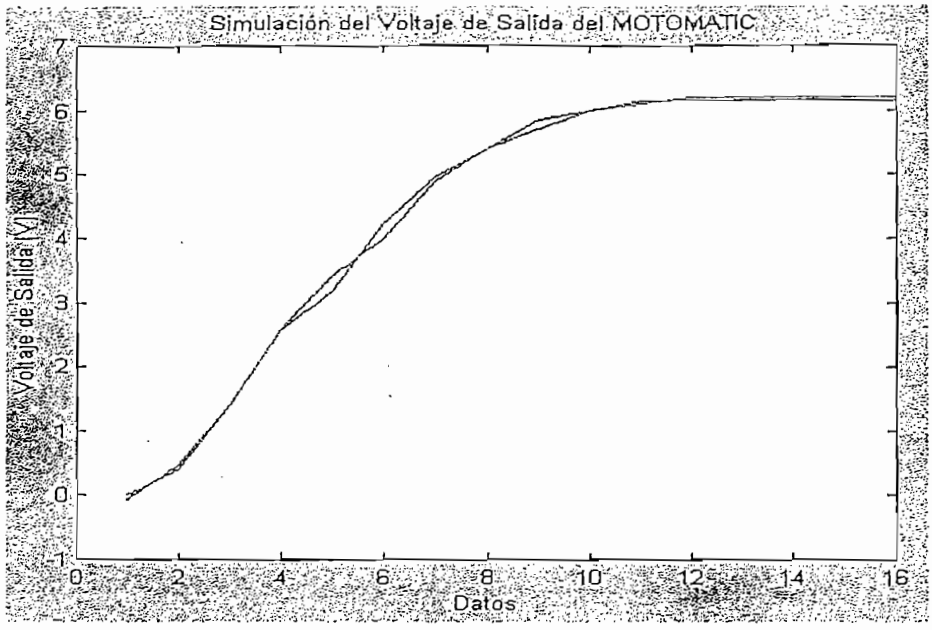


Figura 5.10

Simulación mediante redes neuronales del sistema MCSL – 100

Donde el modelo real (color rojo) y modelo simulado (color azul) son prácticamente idénticos.

5.3.2 SISTEMA MCSL – 100 CON REALIMENTACIÓN DE POSICIÓN

Al aplicar la misma estructura de red neuronal y simular este sistema se tiene la figura 5.11

De donde se ve que la red neuronal en el comienzo trata de seguir la señal de referencia o señal objetivo (color rojo), pero como la señal no es muy buena ya

que tiene cambios bruscos a causa de que los datos de medición no son muy buenos, la red neuronal no tiene un buen desempeño.

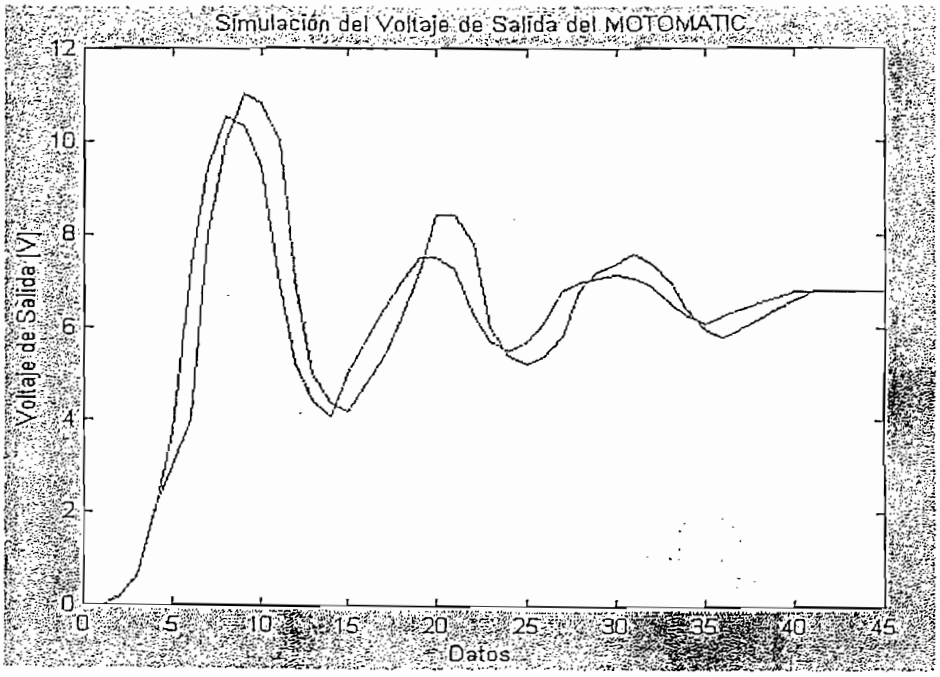


Figura 5.11

Simulación mediante redes neuronales del sistema MCSL – 100

CAPITULO 6

CONCLUSIONES

6.1 CONCLUSIONES

6.1.1 GENERALES

La simulación es de gran importancia en el campo de Control Automático ya que si se tiene el modelo de cualquier planta, se puede implementar diferentes técnicas de control, para de esta manera tener alternativas de control en cualquier proceso.

La simulación es un medio de verificación tanto de modelos como de técnicas de análisis y diseño. Generalmente se realizan comparaciones entre experimentos reales y simulación para validar un modelo.

En este caso se implementó tres tipos de simulación: por modelación, identificación y redes neuronales. Dando como resultado que para los sistemas propuestos la técnica de redes neuronales es la que mejor se aproxima al modelo real; aunque por identificación se tiene modelos aceptables.

Los objetivos del proyecto de titulación se cumplieron a cabalidad ya que se empleo modelación, identificación y redes neuronales para la máquina de corriente continua regulada por voltaje de armadura de los cuales se realizó la comparación de los diferentes métodos de simulación, para posteriormente realizar un software implementando lo descrito anteriormente.

6.1.2 PARTICULARES

El modelo obtenido mediante principios de ingeniería (modelación) es bien elaborado porque toma en cuenta la dinámica de la máquina.

Para la simulación mediante modelación se tiene que este método es muy laborioso, ya que se necesita conocer todos los parámetros del sistema a analizar.

Las técnicas de identificación dan resultados bastante buenos, aunque su exactitud se deteriora al trabajar con un modelo más complejo.

En el método de identificación se podría decir que no es tan complicado su implementación, sin embargo se debe tener una buena herramienta para tomar los datos del sistema a simular ya que de esto depende el correcto desempeño de la identificación desembocando en un buen modelo del sistema.

El método de redes neuronales, es el mejor de todos los mencionados anteriormente ya que su implementación no es muy complicada y los resultados que se obtienen son de muy buena calidad, no necesita de largos procesos de experimentación.

Las redes neuronales se comportan como una función de aproximación universal no – lineal, es por tal razón que este método da mejores resultados que los demás, ya que se está trabajando con no – linealidades y métodos como identificación, modelación son técnicas lineales.

Para un mejor resultado al utilizar redes neuronales, identificación o modelación se debe mejorar la calidad en las mediciones con alguna técnica de procesamiento previo como por ejemplo: mejorando los instrumentos de medición, colocando un filtro para eliminar ruido, etc.

6.2 RECOMENDACIONES

Una sugerencia para las mediciones experimentales es que se debe tener los instrumentos adecuados dependiendo de la cada caso.

Se debería realizar un estudio de no – linealidades para el motor y la interfase de los elementos de transmisión de movimiento si se quiere mejorar en cuanto a modelación del sistema.

Para él método de redes neuronales se debería profundizar en el manejo de las estructuras para de esta manera ampliar el rango de aplicaciones de dicho método de simulación.

Cuando se trabaja con procesadores (computador), se debe trabajar con datos de entrada libres de impurezas ya que de lo contrario a la salida del procesador se tendrá las mismas impurezas o mega impurezas.

BIBLIOGRAFIA

BIBLIOGRAFÍA

- [1] Chapman S. J., "Máquinas Eléctricas", Tercera edición, McGraw Hill.

- [2] Rivera P., "Control de Máquinas Eléctricas", Folleto, E.P.N.

- [3] "Motomatic Control System Laboratory", Electro – Craft Corporation, 1968.

- [4] Ogata K.,"Ingeniería de Control Moderna", Tercera edición, Prentice Hall, México 1998.

- [5] Dorf R.,"Sistemas Modernos de Control Teoría y Práctica", Wesley Iberoamericana, 1989.

- [6] Bastidas J., "Estudio Teórico Experimental de un Servomecanismo de Velocidad y Posición", Tesis de Grado, E.P.N., Quito 1981.

- [7] Chapra S., Canale R., "Métodos Numéricos para Ingenieros con Aplicaciones en Computadoras Personales", McGraw Hill, México 1988.

- [8] Burbano P., "Apuntes de Modelación y Simulación", E.P.N., Quito 2001.

- [9] Freeman James A., Skapura David M., "Redes Neuronales Algoritmos, aplicaciones y técnicas de programación", Addison Wesley/Diaz de Santos, 1991.

- [10] Becerra A., "Simulación de la Máquina de Inducción Mediante Redes Neuronales", Tesis de Grado, E.P.N., Quito 2001.

- [11] Demuth H., Beale M., "Neural Network toolbox ", PDF, Version 3.0,1998.

- [12] Romero A., "Desarrollo de un Módulo para la Enseñanza de Control Inteligente", Tesis de Grado, E.P.N., Quito, Febrero 2002.

- [13] Elgerd O., "Control System Theory", McGraw Hill, Kogakusha, página 334.

- [14] Bombon M., "Estudio Comparativo de Métodos de Compensación de Fricción en Servomecanismos", Tesis de Grado, E.P.N., Quito 1999.

- [15] Cunachi A., "Estudio del Sistema de Control de la Estación de Bombeo Lago Agrio del Oleoducto Trans – Ecuatoriano", Tesis de Grado, E.P.N., Quito 1996.

ANEXOS

ANEXO 1

Datos obtenidos de los modelos discretos del Motomatic con realimentación de posición (sistema 1) y del Motomatic con realimentación de velocidad (sistema 2)

SISTEMA 1

k	y(k)_real	y(k)_ident
0	0	0
1	0.4	0
2	1.4	1
3	2.6	2.32
4	3.4	3.4
5	4	4.28
6	4.9	4.85
7	5.4	5.31
8	5.7	5.66
9	6	5.91
10	6.1	6.1
11	6.2	6.24
12	6.2	6.35
13	6.2	6.4
14	6.2	6.49

SISTEMA 2

k	y(k)_real	y(k)_ident
0	0	0
1	0.2	0
2	0.6	0.68
3	2	2
4	3	5
5	4	7.29
6	8	9.1
7	10	10
8	11	9.53
9	10.8	8.51
10	10	7
11	7	6.04
12	5	5
13	4.4	5.1
14	4.2	5.36
15	4.8	5.93
16	5.4	6.58
17	6.2	7
18	7.2	7.4
19	8.4	7.43
20	8.4	7
21	7.8	6.94
22	6	7
23	5.4	6
24	5.2	6
25	5.4	6
26	5.8	6
27	6.8	7
28	7.2	7
29	7.4	7
30	7.6	7
31	7.4	7
32	7	7
33	6.4	7
34	6	7
35	5.8	7
36	6	7
37	6.2	7
38	6.4	7
39	6.6	7
40	6.8	7
41	6.8	7
42	6.8	7
43	6.8	7
44	6.8	7