UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO

Facultad de Ingeniería

PRÁCTICA DE CONTROL DEL PÉNDULO INVERTIDO

T E S I S

Que para obtener el título de :

INGENIERO MECANICO ELECTRICISTA

P r e s e n t a :

LUIS JULIÁN RAMOS CRUZ

Director: Dra. Ma. Cristina Verde R.
Codirector: M. I. Rolando Carrera M.

México, D. F. **Marzo de 1995**

DEDICATORIAS

A mi madre *Engracia Ramos Cruz*, que con su trabajo, apoyo y amor, he logrado dar este paso en mi vida.

AGRADECIMIENTOS

A la Dra. Cristina Verde Rodarte y al M. en I. Rolando Carrera Mendez y a mi familia en general por el apoyo brindado para la realizacion del trabajo, ademas a los compañeros de la cordinación de Automatización del Instituto de Ingeniería. Y finalmente a la Facultad de ingeniería de la Universidad Autonoma de México, por haberme formado y preparado para ser un profesionista.

PRÁCTICA DE CONTROL DEL PÉNDULO INVERTIDO

1.- INTRODUCCIÓN

Uno de los problemas básicos en la enseñanza de la teoría de control en las carreras de ingeniería, es el contar con experimentos que permitan validar y obtener experiencia en procesos reales de los diseños desarrollados dentro de los cursos. Esto, se debe principalmente a que cualquier aplicación de las técnicas de control digital, aún a nivel de plantas prototipo requiere tanto del conocimiento impartido en los cursos para ajustar las leyes de control, como de experiencia en el manejo de:

- equipo periférico no soportado por los sistemas operativos comunes en computadoras personales
- adquisidores de datos tanto de variables analógicas como discretas
- lenguajes como C que permitan implantar algoritmos en tiempo real
- sensores y actuadores a través de los cuales se recibe y envía información de y hacia el proceso.
- leyes físicas y químicas que gobiernan el comportamiento del proceso.

Por tal motivo, se propuso el siguiente proyecto el cual consistió en diseñar e implantar un paquete interactivo de experimentos para facilitar el dominio de las técnicas actuales de control digital de procesos dinámicos, tratando en lo posible que el alumno se concentre en los problemas del diseño de las leyes de control por computadora y su ajuste.

Debido a que los servomecanismos, es decir sistemas en donde se controla básicamente posición y velocidad, son ampliamente usados en la práctica, además no requieren de gran destreza para su operación, y permiten visualizar fácilmente el desempeño de una buena estrategia de control, se adoptó el servomecanismo del péndulo invertido como proceso a controlar. Otra de las ventajas de este servo es que su comportamiento dinámico es semejante al de un proyectil en el momento de su lanzamiento o al de una grúa de carga. Además actualmente este sistema es uno de las usados para comparar las bondades de los esquemas nuevos de control y detección de fallas.

El paquete junto con los experimentos se diseñaron con base en una computadora personal mínimo PC386, el péndulo invertido AMIRA y las tarjetas de adquisición DAC62214 para PC (AMIRA) del Laboratorio de Automatización del Instituto de Ingeniería. Por lo que respecta al software se requiere del paquete matemático MATLAB para Windows versión 4.1. El programa en tiempo real fue implantado con el C++ de Microsoft. La figura 1.1 muestra el diagrama esquemático del sistema usado en la práctica.

El conjunto de experimentos o Paquete de Control del Péndulo Invertido denotado por sus siglas PCPEIN, requiere de los conocimientos básicos de control digital a nivel de licenciatura y cubre los siguientes aspectos teóricos:
- modelación e identificación de la fricción seca del servomecanismo
- compensación por medios estáticos y dinámicos de la fricción seca
- simulación de sistemas dinámicos discretos
- leyes de control con retroalimentación de los estados para sistemas digitales
- observadores de orden completo y reducido con su dos variantes predictivo y corriente.

Una de las grandes ventaja del paquete interactivo PCPEIN es su amigabilidad por lo que respecta al manejo de sensores, actuadores, sistemas de protección del péndulo almacenamiento de datos y en general a todos las opciones de visualización de resultados en tiempo real. Es decir para el usuario es transparente el manejo de dato dentro de la PC y por tanto puede concentrarse en asimilar y dominar los aspectos de control de la práctica. La parte de análisis y síntesis fuera de línea se realiza también de manera iterativa y amigable a través de menús usando el paquete matemático MATLAB.

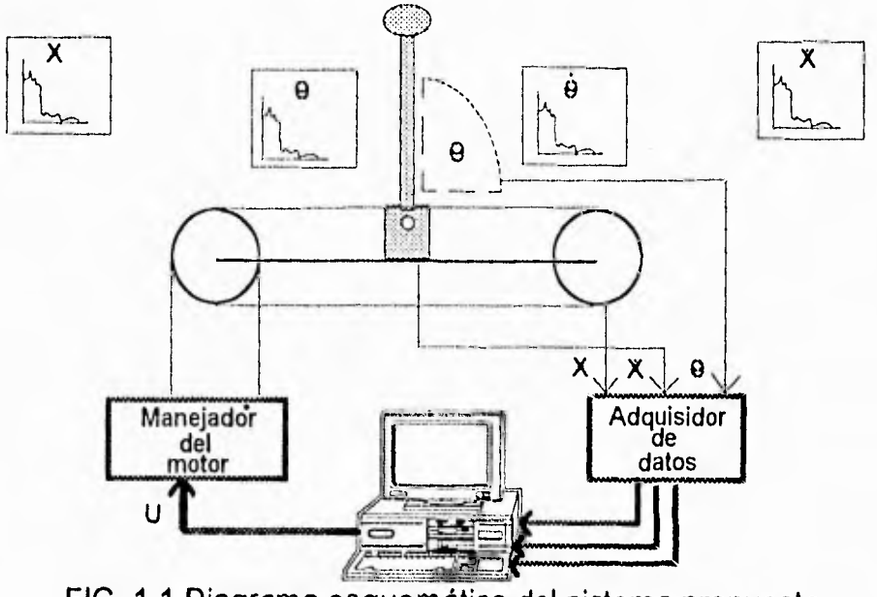


FIG. 1.1 Diagrama esquemático del sistema propuesto

Con objeto de facilitar la presentación del trabajo realizado, el documento se ha dividido en siete capítulos, dejando la información técnica complementaria en tres apéndices.

En el capítulo uno se presentan los objetivos y aspectos fundamentales del trabajo. El capítulo dos está dedicado a los aspectos teóricos de los sistemas lineales discretos que se requieren para realizar la práctica propuesta. En

particular, se trata el tema de asignación de polos por medio de una retroalimentación de estados y el de diseño de observadores dinámicos de orden completo y reducido.

En el tercer capítulo se deriva el modelo matemático tanto del carro como del péndulo invertido a partir de las leyes físicas del sistema. Además se incluye la linearización de ambos modelos alrededor de un punto de funcionamiento.

El cuarto capítulo trata el tema de la fricción seca del servomecanismo. En particular se presentan los mecanismos empleados para identificar la fricción cuando se asume que ésta, puede modelarse por medio de una no linealidad estática, y cuando se le asigna un valor constante desconocido el cual es estimado con base en la teoría de estimadores de estado.

En el capítulo cinco se analiza el comportamiento estático del carro bajo control cuando se compensa la fricción seca, usando los dos modelos identificados en el capítulo cuatro. La comparación de ambos modelos lleva a concluir que el estimador dinámico permite lograr desempeños más satisfactorios, ya que reduce apreciablemente el error en estado permanente del carro. En este capítulo, se incluye el diseño de la ley de control y del estimador de la velocidad angular del péndulo cuando se usa un observador de orden completo y uno reducido. La comparación entre ambos observadores valida los comentarios reportados en la literatura, en el sentido que el error de estimación con un observador de orden reducido se aproxima a cero generalmente más rápidamente que el error con uno observador de orden completo.

El capítulo seis presenta paso a paso con base en un ejemplo la descripción del paquete PCPEIN. Finalmente en el capítulo seis se dan algunas conclusiones y comentarios sobre el paquete y la experiencia obtenida con los primeros usuarios.

En el apéndice A se presentan los valores numéricos de los coeficientes de los modelos continuo y discretos del carro y del péndulo invertido. El apéndice B cubre los aspectos técnicos de los convertidores A/D y D/A empleados y fueron tomados de los datos reportados por el fabricante del equipo. Para terminar en el apéndice C se incluyen los listados de los programas principales escritos en C++ y los del intérprete MATLAB que conforman el paquete PCPEIN. Se anexa a este documento un disco con las rutinas de lectura y escritura de archivos, asi como las rutinas de generación de gráficos.

2 TEORÍA BÁSICA DE SISTEMAS DISCRETOS

2.1 Variables de estado

El comportamiento de cualquier sistema dinámico de paramentros concentrados se puede expresar por medio de un conjunto de ecuaciones diferenciales ordinarias de primer orden. A este tipo de modelo se le conoce como *representación en variables de estado.*

Así la ecuación general en variables de estado para un sistema dinámico lineal e invariante en el tiempo descrito por un conjunto de ecuaciónes diferenciales lineales de N-ésimo orden, con una sola entrada, está dado por

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \vdots \\ \dot{x}_n \end{bmatrix} = \begin{bmatrix} f_{11} & f_{12} & \cdots & f_{1n} \\ f_{21} & & & \vdots \\ \vdots & & & \vdots \\ f_{n1} & & & f_{nn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} + \begin{bmatrix} g_1 \\ g_2 \\ \vdots \\ g_n \end{bmatrix} u_g \qquad (2.1)$$

donde $x_1, x_2, \ldots x_n$ representa el conjunto de variables de estado, f_{ij} y g_i son coeficientes constantes y u_g es la señal de entrada; si se considera que la salida y del sistema depende linealmente de los estados, ésta puede expresarse como

$$y = \begin{bmatrix} h_1 & h_2 & \cdots & h_n \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} + J u_g \qquad (2.2)$$

donde los coeficientes h_i son constantes y J es el coeficiente que determina la relación directa entre la entrada u_g la salida y. En forma compacta la representación (2.1) y (2.2), puede escribirse como

$$\dot{X} = FX + Gu + G_1 w \qquad (2.3)$$

$$y = HX + Ju + J_1 w \qquad (2.4)$$

en donde la entrada del sistema u_g se ha dividido en dos entradas, una sobre la cual se tiene control, denotada u, y la otra considerada como una perturbación w, sobre la cual no se tiene control.

Las matrices F, G, G_1, H, J y J_1 son matrices de coeficientes constantes de dimensión nxn, nx1, nx1, 1xn, 1x1 y 1x1, respectivamente y dependen del sistema considerado en particular.

Aplicando la transformada de Laplace a las ecuaciónes (2.3) y (2.4), se obtiene la función de transferencia de la entrada u a la salida y

$$g(s) = \frac{y(s)}{u(s)} = H(sI - F)^{-1}G + J \tag{2.5}$$

la cual caracteriza la relación entrada/salida del sistema en el dominio de la frecuencia (w se supone nula). Los valores propios del sistema en lazo abierto se obtienen como solución de la ecuación característica

$$\det(\lambda_i I - F) = 0 \tag{2.6}$$

y corresponden con los polos de la función de transferencia g(s), cuando el sistema es controlable y observable.

Considerando que el sistema de control va a ser realizado por medio de una computadora digital, entonces, el comportamiento del sistema (2.3) y (2.4) en los instantes de muestreo KT se describe como

$$X_{(kT+T)} = A_D X_{(kT)} + B_D u_{(kT)} + B_{D1} w_{(kT)} \tag{2.7}$$

$$y_{(kT)} = C X_{(kT)} + D u_{(kT)} + D_1 w_{(kT)} \tag{2.8}$$

donde

$$A_D = e^{FT} \tag{2.9}$$

$$B_D = \int_0^T A_D(t)\, G\, dt \quad ; \quad B_{D1} = \int_0^T A_D(t)\, G_1\, dt \tag{2.10}$$

$$C = H \tag{2.11}$$

$$D = J \quad ; \quad D_1 = J_1 \quad ; \quad T = \text{periodo de muestreo} \tag{2.12}$$

Análogamente al sistema contínuo, los valores propios del sistema se calculan a partir de la ecuación característica

$$\det(zI - A_D) = 0 \tag{2.13}$$

En particular, si los valores s_i propios del sistema contínuo se conocen, los valores propios del sistema discretizado se pueden obtener por medio de la transformación

$$z_i = e^{Ts_i} \tag{2.14}$$

En la figura (2.1) se presenta el diagrama de bloques del sistema discreto en variables de estado, considerando el ruido $w_{(kT)}$ igual a cero.

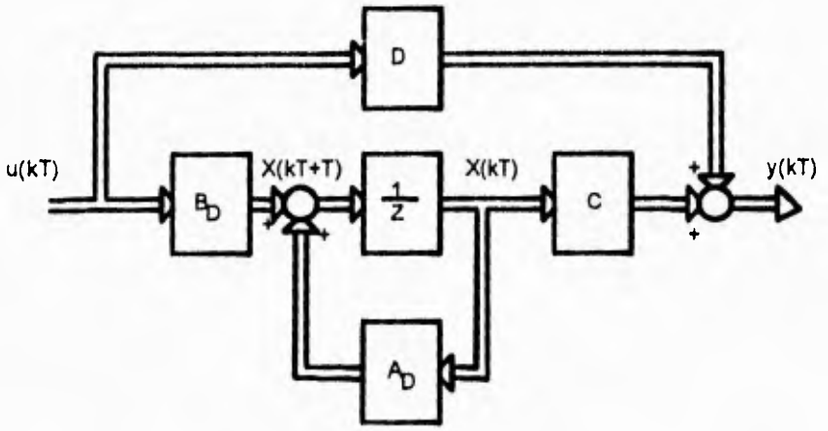


FIG. 2.1 Diagrama de bloques del sistema lineal

2.2 Regulación por asignación de polos

Una de las técnicas de control más comunmente empleadas para modificar la dinámica de un sistema, si se conoce todo el vector de estados **X**, es la de **asignación de polos** por medio de un esquema retroalimentado (para mayor información véase referencia 1). La representación en diagramas de bloques del sistema de control para la planta (2.7) y (2.8) se presenta en la figura (2.2), en donde r(k)T corresponde a la señal de referencia que debe seguir el sistema.

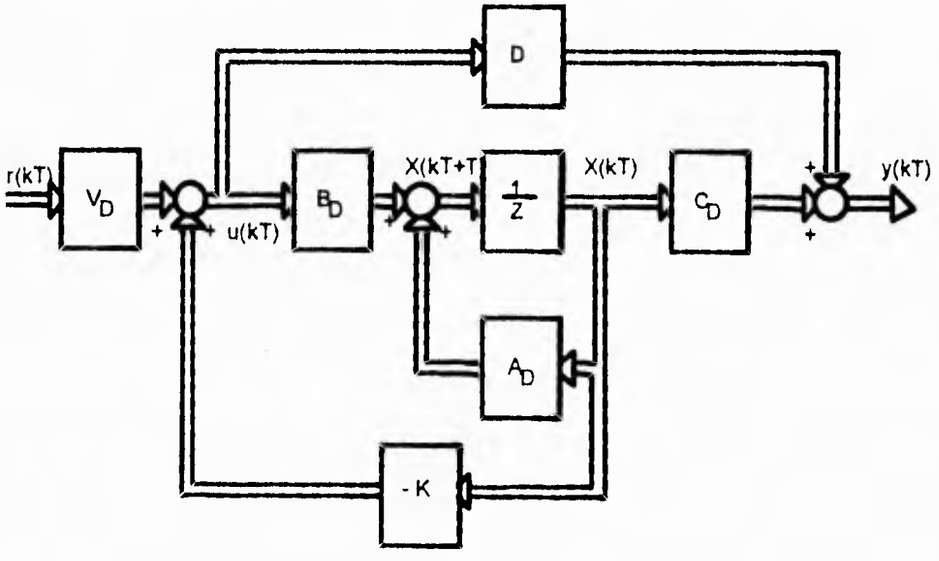


FIG. 2.2 Sistema con retroalimentación de estados

Del esquema de la figura (2.2), se observa que la ley de control u(k)T retroalimentada, es simplemente una combinación lineal de todos los estados; la ecuación que la representa es

$$u_{(kT)} = -K\,X_{(kT)} + V_D\,r_{(kT)} = -\begin{bmatrix} k_1 & k_2 & \cdots & k_n \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \\ \cdot \\ x_n \end{bmatrix} + V_D\,r_{(kT)} \qquad (2.15)$$

Por tanto, la ecuación dinámica del sistema en lazo cerrado se puede escribir como

$$X_{(kT+T)} = A_D\,X_{(kT)} - B_D\,K\,X_{(kT)} + B_{DI}V_D\,w_{(kT)} \qquad (2.16)$$

y la ecuación característica del sistema en lazo cerrado, está dada por

$$\det\left[zI - (A_D - B_D K)\right] = 0 \qquad (2.17)$$

y la cual corresponde a un polinomio de orden n, en z, que depende de las ganancias de la ley de control k_1, k_2, k_3, ...,k_n. Estas ganancias permiten asignar libremente la ubicación de los polos z_{d1}, z_{d2}, z_{d3},...,z_{dn} del sistema retroalimentado.

De aquí, que el diseño del control consiste en elegir la matriz de ganancia **K**, de tal manera que las raíces de la ecuación (2.17), se ubiquen en los lugares deseados (polos de sistema en lazo cerrado). Por tanto se debe satisfacer

$$\det\left[zI - (A_D - B_D K)\right] = (z - z_{d1})(z - z_{d2}).....(z - z_{dn}) \qquad (2.18)$$

para los valores de z_{d1}, z_{d2}, z_{dn} deseados. Los valores del vector **K** se pueden determinar, igualando coeficientes en el sistema (2.18); ya que se tienen n ecuaciónes y n incognitas.

Una alternativa para el cálculo directo de las ganancias de retroalimentación, k_1, k_2, ... k_n, sin tener que resolver directamente la ecuación (2.18) fue propuesta por Ackerman (referencia 1).

Se hace notar que existen diferentes mecanismos de diseño basados en una ley de control del tipo (2.15). Uno de los más eficientes es el regulador cuadrático lineal, sin embargo no se presenta en este trabajo por estar fuera del contexto (para mayor información consulte la referencia 2).

$$\hat{X}_{(kT+T)} = A_D \hat{X}_{(kT)} + B_D u_{g(kT)} + L\left[y_{(kT)} - C_D \hat{X}_{(kT)}\right] \tag{2.23}$$

Esta ecuación gobierna el comportamiento del *estimador de orden completo,* conocida como *predictor*, porque estima el estado en el instante (kT+T) con base en la salida de la planta en el instante (kT). Su representación en diagramas de bloques se muestra en la figura (2.3).

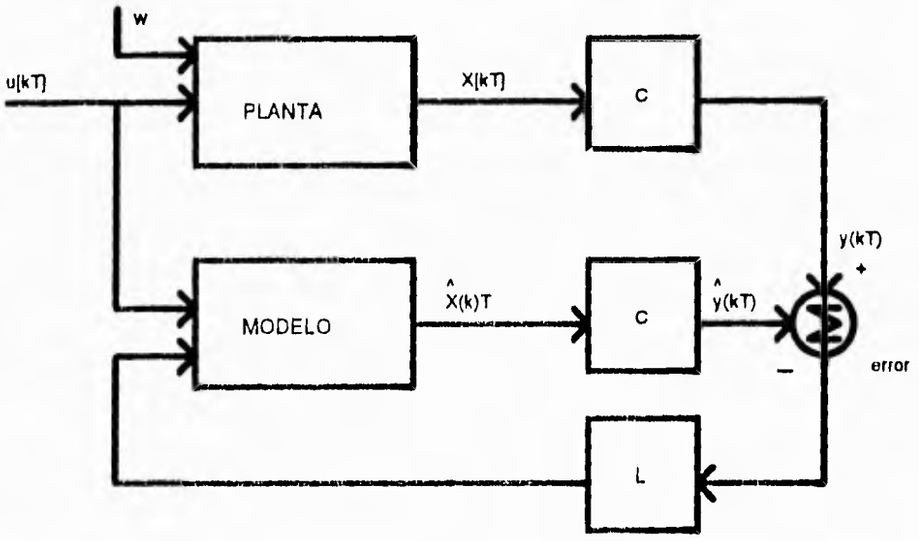


FIG. 2.3 Estimador de lazo cerrado

Para que el error $\tilde{X}$ tienda a cero, la ganancia del observador L

$$L = \left[l_1, l_2, ,,, l_n\right]^T \tag{2.24}$$

debe garantizar que los valores propios de la matriz (A_D - LC) se ubiquen dentro del círculo unitario. Este hecho está basado en que la dinámica del error está dada por

$$\tilde{X}_{(kT)} = \left[A_D - LC\right]\tilde{X}_{(kT)} + B_{DI}w \tag{2.25}$$

y por tanto la ecuación característica resultante es

$$\det\left[zI - (A_D - LC)\right] = 0 \tag{2.26}$$

Entonces, si se elige la matriz L, tal que (A_D - LC) tenga todos sus valores propios estables, el error tiende a cero mientras no exista una perturbación w, con lo cual, los estados estimados $\hat{X}$ convergerán hacia los estados reales X. Además, la dinámica de error se puede elegir libremente a través de L.

Por otro lado, si se observa la ecuación (2.25), se encuentra gobernada solamente por el término w. Por tanto si w = 0 el error convergerá a cero.

Para la selección de la matriz L se puede proceder de la misma forma que para la ley de control, simplemente transponiendo las matrices A_D, L y C.

Para mejorar la estimación de los estados observados, es común calcular $\hat{X}_{(kT)}$ en función de la salida $y_{(kT)}$ en el mismo instante. A este tipo, de estimador, se le conoce como estimador corriente (referencia 1).

Con objeto de facilitar la explicación se denotara $\hat{X}_{(kT/kT)}$ al valor estimado de X en el instante kT con base en información de la salida en el instante kT y $\hat{X}_{(kT/kT-T)}$ al valor estimado en el instante kT, obtenido a partir de informacion en el instante (k-1)T.

Reescribiendo la ecuación (2.23) del estimador predictivo considerando w=0, con la nueva notación, se tiene

$$\hat{X}_{(kT+T/kT)} = A_D \hat{X}_{(kT/kT-T)} + B_D u_{(kT)} + L(y_{(kT)} - C_D \hat{X}_{(kT/kT-T)}) \qquad (2.27)$$

Considerando que en el instante k se disponen de valores estimados de $\hat{y}_{(kT)}$, la ecuación (2.27), después de hacer un corrimiento de un periodo de muestreo, se transforma en

$$\hat{X}_{(kT/kT)} = A_D \hat{X}_{(kT-T/kT-T)} + B_D u_{(kT-T)} + L(y_{(kT)} - C_D \hat{X}_{(kT-T/kT-T)}) \qquad (2.28)$$

Ahora sustituyendo la ecuación (2.19) en (2.28), se obtiene

$$\hat{X}_{(kT/kT)} = A_D \hat{X}_{(kT-T/kT-T)} + B_D u_{(kT-T)} + L\left[y_{(kT)} - C_D (A_D \hat{X}_{(kT-T/kT-T)} + B_D u_{(kT-T)})\right] \qquad (2.29)$$

la cual se puede escribir como

$$\hat{X}_{(kT/kT)} = (I - LC)A_D \hat{X}_{(kT-T/kT-T)} + (I - LC)B_D u_{(kT-T)} + Ly_{(kT)} \qquad (2.30)$$

Es decir, el valor estimado de $\hat{X}_{(KT/kT)}$ depende de los valores anteriores tanto de u como de $\tilde{X}$ y del valor presente de la salida $y_{(KT)}$. Por tanto la expresión del estimador corriente de orden completo, se reduce a

$$\hat{X}_{(kT/kT)} = A_{Dx}\hat{X}_{(kT-T/kT-T)} + B_{Dx}u_{(kT-T)} + Ly_{(kT)} \tag{2.31}$$

donde.

$$A_{Dx} = (I - LC)A_D$$

$$B_{Dx} = (I - LC)B_D \tag{2.32}$$

2.3.2 Observador de orden reducido

Cuando no se requiere estimar u observar todo el vector de estados de un sistema, es posible disminuir la dimensión del observador, obteniéndose una reducción en el cálculo y mejorando la convergencia de los estados estimados.

El estimador de orden reducido parte de la idea de que en algunos casos, para un sistema de orden n, se puede medir un numero n_1 de estados, despejando directamente éstos a partir de la ecuación de la salidas; por tanto, únicamente se requiere diseñar un observador para los $m=n-n_1$ estados restantes. Bajo estas condiciones, el estado se puede subdividir y expresarlo en términos de las salidas $Y_{(k)T}$ (con n_1 términos) y las variables por observar.

$$\begin{bmatrix} Y_{(kT+T)} \\ \hat{X}_{B(kT+T)} \end{bmatrix} = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \begin{bmatrix} Y_{(kT)} \\ \hat{X}_{B(kT)} \end{bmatrix} + \begin{bmatrix} B_1 \\ B_2 \end{bmatrix} u_{(kT)} \tag{2.33}$$

Despejando el termino desconocido en la primera parte de la ecuacion (2.33) y considerando que éste es un valor estimado, se tiene

$$A_{12}\hat{X}_{B(kT)} = Y_{(kT+T)} - A_{11}Y_{(kT)} - B_1 u_{(kT)} \tag{2.34}$$

Por tanto, el error entre los estados reales $X_{B(KT)}$ y los estimados $\hat{X}_{B(KT)}$, se reduce a

$$e_{(KT)} = -A_{12}\hat{X}_{B(KT)} + Y_{(KT+T)} - A_{11}Y_{(KT)} - B_1 u_{(KT)} \tag{2.35}$$

Por otro lado, la ecuación dinámica del observador asociado a los estados no medibles , esta dada por

$$\hat{X}_{B(kT+T)} = A_{22}\hat{X}_{B(kT)} + A_{21}Y_{(kT)} + B_2 u_{(kT)} + L\left[Y_{(kT+T)} - A_{11}e_{(kT)} - B_1 u_{(kT)} - A_{12}\hat{X}_{B(kT)}\right] \tag{2.36}$$

Así que si se sustituye en (2.36) el valor del error $e_{(K)T}$ (ec. 2.35) y se aplica un retraso a (2.36) se obtiene el modelo del observador corriente de orden reducido

$$\hat{X}_{B(kT)} = (A_{22} - L A_{12})\hat{X}_{B(kT-T)} + (B_2 - LB_1)u_{(kT-T)} + (A_{21} - L A_{11})Y_{(kT-T)} + L Y_{(kT)} \quad (2.37)$$

Esta expresión se reduce a la ecuación general del estimador corriente de orden reducido

$$\hat{X}_{B(kT)} = A_B \hat{X}_{B(kT-T)} + B_B u_{(kT-T)} + F_B Y_{(kT-T)} + L Y_{(kT)} \quad (2.38)$$

en donde las matrices A_B, B_B y F_B, se definen como

$$A_B = \left[A_{22} - L A_{12}\right] \quad (2.39)$$

$$B_B = \left[B_2 - L B_1\right] \quad (2.40)$$

$$F_B = \left[A_{21} - L A_{11}\right] \quad (2.41)$$

Por tanto, con la matriz L y las ecuaciones (2.39) a (2.41), es posible implementar un estimador corriente de orden reducido. La figura (2.4) presenta el esquema del observador corriente de orden reducido.

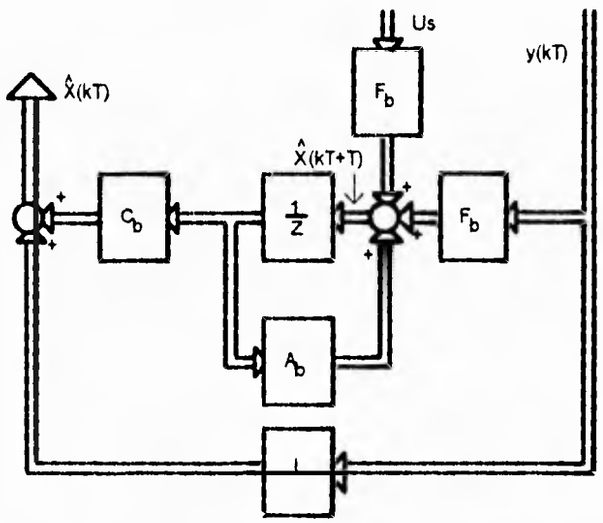


FIG. 2.4 Diagrama de bloques del estimador de orden reducido

Se hace notar, que el hecho de que el estimador corriente utilice los valores pasados y presentes para estimar el estado actual, permite reducir el error con mas rapidez en comparación con el estimador predictor, el cual no aprovecha la información actual para corregir el error.

Para mayor información sobre estimadores, véase referencia 2.

3 MODELO MATEMÁTICO DEL PÉNDULO INVERTIDO

El sistema del péndulo invertido está constituido por un carro y una varilla la cual en uno de sus extremos está fijada al carro por medio de un pivote y en el otro tiene una masa. El carro se mueve a lo largo de una barra guía, y el objetivo de control es mantener en equilibrio la varilla para cualquier posicion del carro.

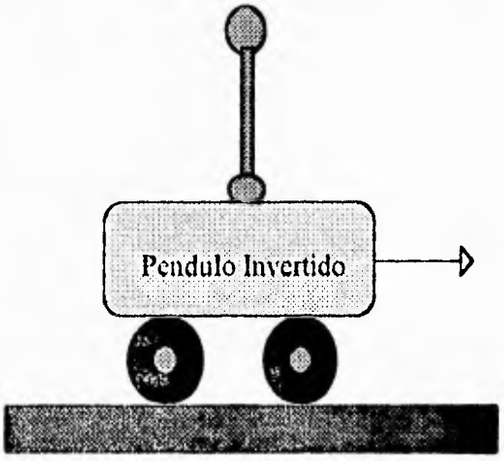


FIG. 3.1 Esquema simplificado del péndulo invertido

El carro se mueve a través de la barra, por medio de una banda de transmisión que a su vez está conectada a una polea accionada por medio de un motor de corriente directa.

Se dispone de un actuador el cual se considera que no tiene dinámica y que proporciona la potencia requerida para el motor.

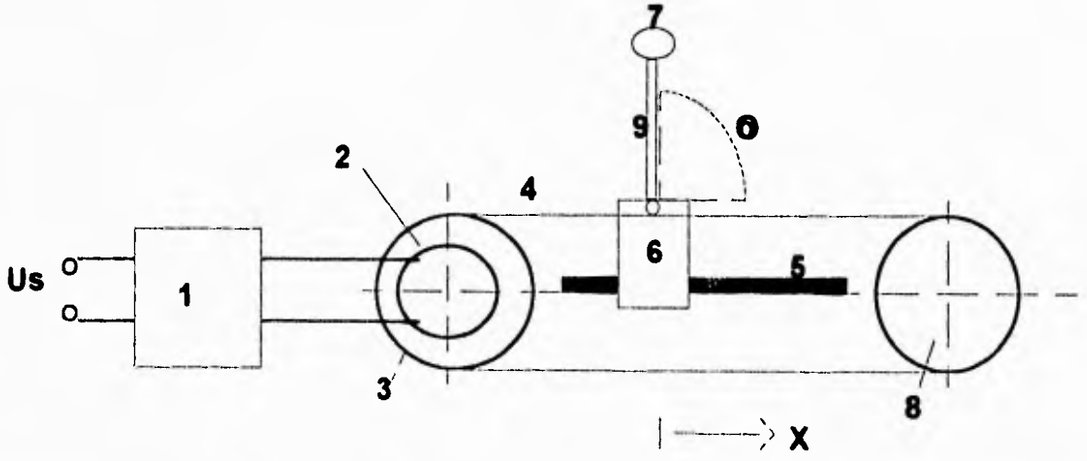


FIG. 3.2 Esquema principal del péndulo invertido

1.- Servo amplificador
2.- Motor
3.- Polea manejadora
4.- Banda de transmisión
5.- Barra guía de metal
6.- Carro
7.- Pesa del péndulo
8.- Polea guía
9.- Varilla del péndulo

3.1 Modelo del carro

Para obtener el modelo del carro, considérese el siguiente diagrama de cuerpo libre

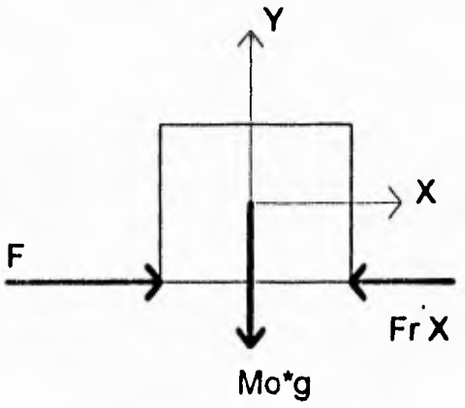


FIG. 3.3 Diagrama de cuerpo libre del carro

donde F denota la fuerza actuando sobre el carro (en el eje horizontal), y F_r es la constante de la fricción estática del carro, g es la aceleración de la gravedad. Entonces, la ecuación de balance de fuerzas en el eje horizontal está dada por

$$M_0\ddot{x} = \sum_{i=0}^{2} F_i = F - F_r\dot{x} \tag{3.1}$$

donde M_0 es la masa del carro. Es decir, el carro está gobernado por una ecuación diferencial de segundo orden, dado por

$$\ddot{x} + \frac{F_r}{M_0}\dot{x} = \frac{F}{M_0} \tag{3.2}$$

con constante de tiempo

$$\tau = \frac{M_0}{F_r} \tag{3.3}$$

Transformando (3.2) al dominio de la frecuencia considerando condiciones iniciales nulas, se obtiene la función de transferencia G(s)

$$G(s) = \frac{X(s)}{F(s)} = \frac{1/M_0}{s\left(s + F_r/M_0\right)} \tag{3.4}$$

Si la fuerza F y el voltaje aplicado al motor están relacionados por

$$F = K_f u \tag{3.5}$$

entonces se puede asociar el siguiente diagrama de bloques al comportamiento del carro, el cual está formado por un sistema de primer orden seguido de un integrador.

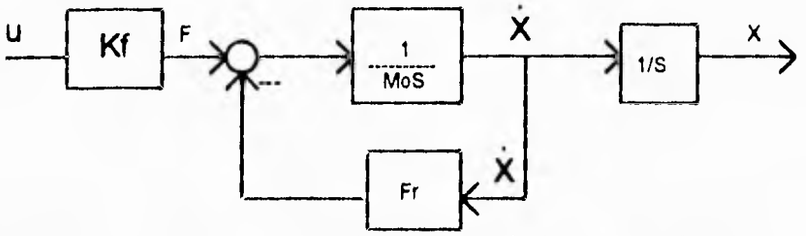


FIG. 3.4 Diagrama de bloques del sistema del carro

Seleccionando la posición y la velocidad del carro como variables de estado, la representación del sistema se reduce a

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & F_r/M_0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 0 \\ 1/M_0 \end{bmatrix} F = A_c X + B_c F \tag{3.6}$$

En este modelo la posición y la velocidad están dimensionados en metros y metros/segundo. Tomando en cuenta que los sensores introducen ganancias no unitarias, es necesario normalizar el modelo a variables eléctricas. La normalización se realiza transformando el vector **X** en un vector normalizado **Xn** por medio de **Xn=NX**, en donde

$$N = \begin{bmatrix} n_{11} & 0 \\ 0 & n_{22} \end{bmatrix} \tag{3.7}$$

es la matriz de normalización y los elementos n_{11} y n_{22} dependen de las ganancias de los sensores. Por tanto

$$A_n = N A_c N^{-1} \tag{3.8}$$

$$B_n = N B_c K_F \tag{3.9}$$

Empleando la expresión (3.7) a (3.9), se obtienen las matrices del sistema normalizado

$$A_n = \begin{bmatrix} 0 & k_3 \\ 0 & -k_1 k_2 \end{bmatrix} \tag{3.10}$$

$$B_n = \begin{bmatrix} 0 \\ k_1 \end{bmatrix} \tag{3.11}$$

con los valores de las constantes k_1, k_2 y k_3 dados por

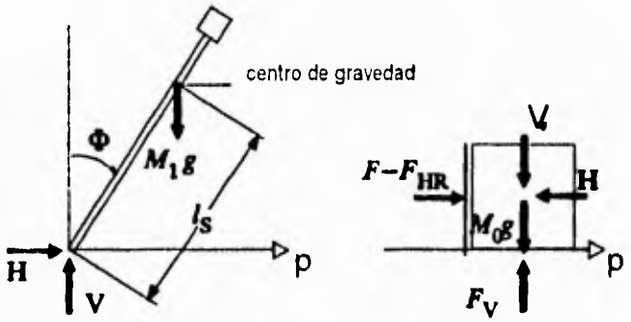
$$k_1 = \frac{K_F\ n_{33}}{M_0} \tag{3.12}$$

$$k_2 = \frac{F_r}{n_{33}\ K_F} \tag{3.13}$$

$$k_3 = \frac{n_{11}}{n_{33}} \tag{3.14}$$

3.2 Modelo del péndulo invertido

Con base en el diagrama de cuerpo libre de la barra, presentado en la figura (3.5), se puede derivar el comportamiento dinámico del péndulo invertido.



Péndulo Carro

FIG. 3.5 Diagrama de cuerpo libre del péndulo y del carro

Sea la masa del péndulo M_1 y la posición del carro p, entonces la fuerza horizontal que actua en el extremo inferior de la barra está dada por

$$H = M_1 \frac{\partial^2}{\partial t^2}(p + l_S \sin\Phi) \tag{3.15}$$

silmilarmente la componente vertical de la fuerza se reduce a

$$V = M_1 \frac{\partial^2}{\partial t^2}(l_S \cos\Phi) + M_1 g \tag{3.16}$$

Además, el momento angular de la varilla esta dado por

$$\Theta_S \frac{\partial^2 \Phi}{\partial t^2} = V l_S \sin\Phi - H l_s \cos\Phi - C\frac{\partial \Phi}{\partial t} \qquad (3.17)$$

donde Θ_S denota el momento de inercia del péndulo con respecto al centro de gravedad y C es la constante de fricción del péndulo. Por otro lado para el sistema del carro, la ecuación de movimiento se escribe como

$$M_0 \frac{\partial^2 p}{\partial t^2} = F - HF_r \frac{\partial p}{\partial t} \qquad (3.18)$$

donde M_0 es la masa del carro Fr es la constante de fricción y la fuerza actuando vía la banda de transmisión se representa por F.

Realizando las derivadas presentadas en las ecuaciones (3.15) y (3.16), estas se transforman en

$$H = M_1\left(\ddot{\Phi} + l_S\ddot{\Phi}\cos\Phi F - l_S\left(\dot{\Phi}^2\right)\sin\Phi\right) \qquad (3.19)$$

$$V = -M_1 l_S\left(\ddot{\Phi}\sin F + \left(\dot{\Phi}^2\right)\cos\Phi\right) + M_1 g \qquad (3.20)$$

Sustituyendo (3.19) y (3.20) en la ecuación (3.17) y (3.18), H y V son eliminadas, y después de algunas manipulaciones algebraicas se obtiene un juego de ecuaciones diferenciales no lineales.

$$\Theta\ddot{F} + C\dot{F} - M_1 l_S g \sin\Phi + M_1 l_S \ddot{\Phi}\cos\Phi = 0 \qquad (3.21)$$

$$M\ddot{\Phi} + F_r\Phi + M_1 l_S\left(\ddot{\Phi}\cos\Phi - \left(\dot{\Phi}^2\right)\sin\Phi\right) = F \qquad (3.22)$$

Se hace notar que las ecuaciones diferenciales no lineales (3.21) y (3.22) están acopladas y su solución describe el comportamiento del péndulo invertido en cualquier punto de funcionamiento. Estas ecuaciones son la base para obtener un modelo matemático linealizado para el péndulo invertido. y en ellas las siguientes abreviaciones fueron utilizadas.

$$\Theta = \Theta_S + M_1 l_S^2 \qquad (3.23)$$

$$M = M_0 + M_1 \qquad (3.24)$$

3.3 Linealización del sistema

Si se considera que el sistema actual se mueve en una pequeña región, cerca de un punto de funcionamiento, entonces es posible obtener un modelo lineal, en donde las desviaciones del estado con respecto al punto de operación definen el vector de estado.

Para ello, las ecuaciónes (3.21) y (3.22) se transforman, definiendo el vector de estado

$$X=\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix}=\begin{bmatrix} p \\ \Phi \\ V \\ w \end{bmatrix}=\begin{matrix} \text{posicion lineal} \\ \text{posicion angular} \\ \text{velocidad lineal} \\ \text{velocidad angular} \end{matrix} \tag{3.25}$$

con señal de entrada:

$$u=F \tag{3.26}$$

en

$$\dot{x}_1 = f_1(x,u) = x_3 \tag{3.27}$$

$$\dot{x}_2 = f_2(x,u) = x_4 \tag{3.28}$$

$$\dot{x}_3 = f_3(x,u) = b\,(x_2)\left(a_{32}\sin x_2 \cos x_2 + a_{33}x_3 + a_{34}\cos x_2 x_4 + a_{35}\sin x_2 (x_4)^2 + b_3 u\right) \tag{3.29}$$

$$\dot{x}_4 = f_4(x,u) = b\,(x_2)\left(a_{42}\sin x_2 + a_{43}\cos x_2 x_3 + a_{44}x_4 + a_{45}\cos x_2 \sin x_2 (x_4)^2 + b_4 \cos x_2 u\right) \tag{3.30}$$

en donde

$$b\,(x_2) = \left(1 + \frac{N^2}{N_{01}{}^2}\sin^2 x_2\right)^{-1} \tag{3.31}$$

$$N = M_1 l_s \tag{3.32}$$

$$N_{01}{}^2 = \Theta M - N^2 \tag{3.33}$$

$$a_{32} = -\frac{N^2 g}{N_{01}^{\ 2}} \quad a_{33} = -\frac{\Theta F_r}{N_{01}^{\ 2}} \quad a_{34} = \frac{NC}{N_{01}^{\ 2}} \quad a_{35} = \frac{\Theta N}{N_{01}^{\ 2}}$$

$$a_{42} = \frac{MNg}{N_{01}^{\ 2}} \quad a_{43} = \frac{NF_r}{N_{01}^{\ 2}} \quad a_{44} = \frac{MC}{N_{01}^{\ 2}} \quad a_{45} = -\frac{N^2}{N_{01}^{\ 2}} \tag{3.34}$$

$$b_3 = \frac{\Theta}{N_{01}^{\ 2}} \qquad b_4 = -\frac{N}{N_{01}^{\ 2}}$$

Si se elige el punto de operación alrededor del cual el carro se desplaza como k_1 con velocidad cero, y se denota como u_A a la desviación del control, el vector de estado en el punto de equilibrio X_A está dado por

$$X_A = \begin{bmatrix} x_1 = k_1 \\ x_2 = 0 \\ x_3 = 0 \\ x_4 = 0 \end{bmatrix} \qquad u_A = \Delta u \tag{3.35}$$

Posteriormente las ecuaciones (3.27) a (3.30) se pueden desarrollar alrededor del punto de equilibrio X_A por medio de una serie de Taylor, truncando está en la primera derivada. Es decir el modelo

$$\Delta \dot{x} \approx \frac{\partial f}{\partial x}\bigg|_{\substack{x=x_A \\ u=0}} \cdot \Delta \dot{x} + \frac{\partial f}{\partial u}\bigg|_{\substack{x=x_A \\ u=0}} \cdot \Delta u \tag{3.36}$$

permite describir las desviaciones $\Delta X = X - X_A$ del sistema, cerca del punto de equilibrio.

Derivando el vector f con respecto a X y u y evaluando el resultado en el punto de funcionamiento se tiene

$$\frac{\partial f}{\partial x}\bigg|_{\substack{x=x_A \\ u=0}} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & a_{32} & a_{33} & a_{34} \\ 0 & a_{42} & a_{43} & a_{44} \end{bmatrix} = A_A \tag{3.37}$$

y la matriz de entrada

$$\frac{\partial f}{\partial u}\bigg|_{\substack{x=x_A \\ u=0}} = \begin{bmatrix} 0 \\ 0 \\ b_3 \\ b_4 \end{bmatrix} = B_A \tag{3.38}$$

Por tanto, la desviación ΔX del punto de operación X_A define el vector de estado y Δu la señal de control, y el modelo linealizado se reduce a

$$\Delta\dot{x} = A_A \Delta x + b_A \Delta u \tag{3.39}$$

Para simplificar la notación, a partir de ahora se asume que X y u denotan desviaciones del punto de funcionamiento $x = \Delta x$ y $u = \Delta u$, por tanto

$$\dot{x} = A_A x + b_A u \tag{3.40}$$

describe el comportamiento del sistema alrededor del punto X_A y u_A

3.4 Normalización de las ecuaciones de estado

Como las ganancias del sensor de posición y de velocidad no son unitarios, de la misma manera que se hizo para el modelo del carro, es necesario normalizar el modelo con respecto a los valores electricos. Por tanto transformando

$$x_n = Nx \tag{3.41}$$

con la matriz diagonal

$$N = \begin{bmatrix} n_{11} & 0 & 0 & 0 \\ 0 & n_{22} & 0 & 0 \\ 0 & 0 & n_{33} & 0 \\ 0 & 0 & 0 & n_{44} \end{bmatrix} \tag{3.42}$$

$$u = -K_F u \tag{3.43}$$

donde el termino K_f normaliza el voltaje de entrada del servo amplificador con respecto al par, se obtiene

$$\dot{x}_n = NA_A N^{-1} x_n + NB_A K_F u \tag{3.44}$$

Finalmente definiendo

$$A_n = NA_A N^{-1} \tag{3.45}$$

$$B_n = NB_A K_F \tag{3.46}$$

se tiene

$$\dot{x}_n = A_N x_n + B_n u \tag{3.47}$$

Se hace notar que a los coeficientes n_{11}, n_{22}, n_{33}, se les asignó las ganancias de los sensores; pero como la velocidad angular, no es una variable medible se consideró arbitrariamente el valor de n_{22}.

4 EFECTO DE LA FRICCIÓN SECA EN EL SISTEMA

4.1 Introducción

El modelo del sistema péndulo-carro descrito en el capítulo anterior, desprecia la fricción seca; sin embargo el manual de las especificaciones del sistema, reporta que el efecto de la fricción seca es considerable en el comportamiento del sistema para cualquier régimen de operación. Por lo tanto, independientemente de la técnica de control que se desee implantar, se consideró adecuado caracterizar el comportamiento de esta fricción. A continuación se presenta el mecanismo empleado para la identificación de la no linealidad estática, producida por la fricción del carro. Se hace notar que debido a que las fuerzas de fricción solamente dependen de la velocidad del carro, es valido considerar para determinarla, únicamente al sistema del carro.

Primeramente con el objeto de comparar el comportamiento del modelo lineal ideal con el del sistema físico real, se propone excitar el sistema en lazo abierto con un escalón de 2.3 Volts. La figura (4.1) presenta la evolución de la velocidad V_s y posición del carro x_1, así como la señal de excitación Uo = 2.3.

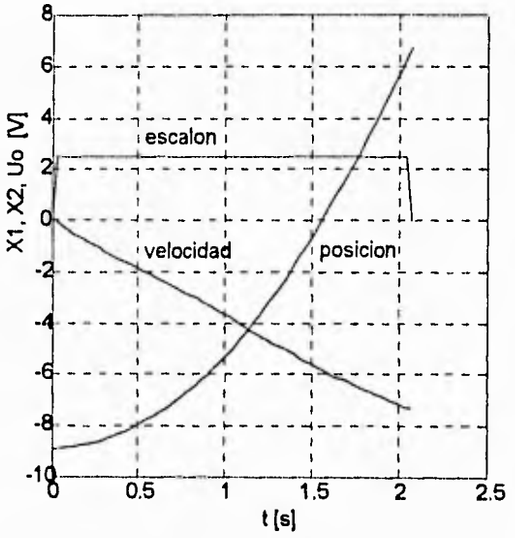


FIG. 4.1 Gráficas de posición, velocidad y escalón aplicado al carro

La figura (4.2) muestra la evolución de la velocidad $V_s(t)$ y $V_l(t)$ para una entrada escalón bajo las mismas condiciones de prueba, tanto del sistema real, como del modelo lineal

$$G(s) = \frac{k_1}{s + k_2} \qquad (4.1)$$

respectivamente. En la misma figura se traza la diferencia o error entre ambas respuestas, definido como

$$V_{nl}(t) = V_s(t) - V_l(t) \qquad \textbf{(4.2)}$$

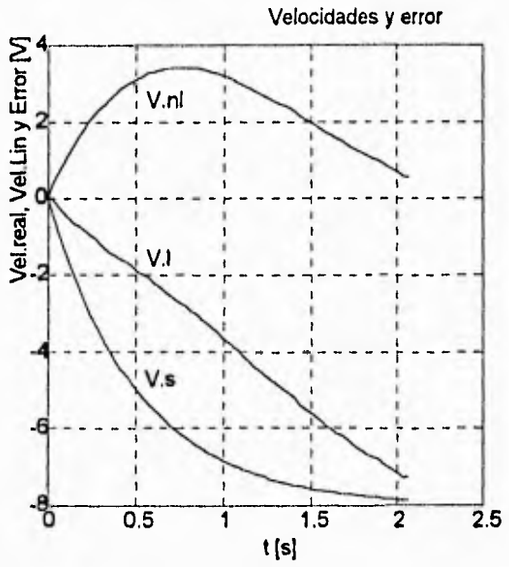


FIG. 4.2 Gráficas de la velocidad, lineal, real y error (no lineal)

De la diferencia entre ambas velocidades se concluye que el modelo lineal $V_l(t)$ está lejos de describir el comportamiento real del sistema $V_s(t)$. En particular el error $V_{nl}(t)$ puede verse como una fuerza externa $U_{so}(t)$ aplicada al sistema lineal G(s), la cual hay que determinar. En términos del diagrama de bloques de la figura (4.3) quiere decir que si $V_{nl}(t)$ es conocida es posible determinar $U_{so}(t)$ transformando el diagrama en un problema inverso, en donde el error $V_{nl}(t)$ es la respuesta, $U_{so}(t)$ es el efecto y $U_o(t)$ la causa. Bajo esta consideración, el diagrama de bloques resultante, se muestra en la figura (4.4).

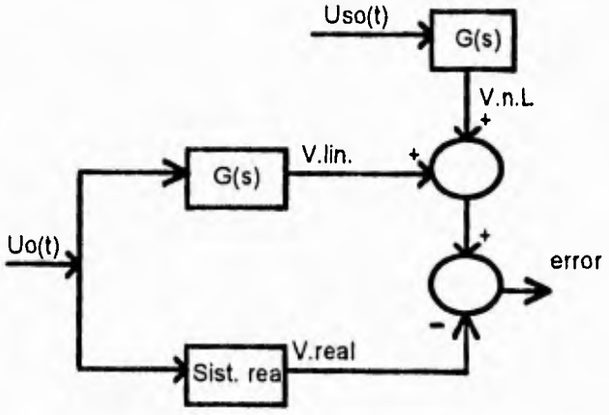


FIG. 4.3 Diagrama de bloques del error de velocidades

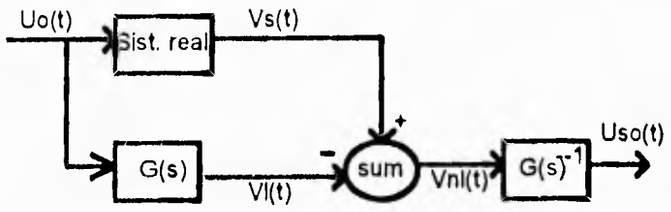


FIG. 4.4 Diagrama de bloques transformado del modelo y del proceso

4.2 Caracterización de la fricción seca

A partir del planteamiento arriba presentado, existen varias alternativas para estimar la función $U_{so}(t)$; en el dominio del tiempo se podrían hacer la convolución de los datos $V_{nl}(t)$ con la respuesta al impulso del sistema $g(t) = \mathcal{L}^{-1}(G(s)^{-1})$, sin embargo en el caso estudiado, la forma de la evolución del error $V_{nl}(t)$, sugirió el siguiente procedimiento.

Identificar el error $V_{nl}(t)$ con una función del tipo

$$\hat{V}_{n.l.}(t) = C_1 e^{-\lambda_1 t} + C_2 t e^{-\lambda_1 t} + C_3 \qquad (4.3)$$

la cual puede transformarse fácilmente al dominio de la frecuencia obteniéndose $\hat{V}_{nl}(s)$. Como resultado, el conocer $\hat{V}_{nl}(s)$ permite obtener la expresión de $U_{so}(s)$ simplemente multiplicando $\hat{V}_{nl}(s)$ por $G^{-1}(s) = (s + k2)/k1$. Finalmente $U_{so}(t)$ corresponde a la transformada inversa de Laplace de $U_{so}(s)$.

Empleando las bondades de la caja de herramientas de optimización del paquete MATLAB se estimaron los coeficientes de la expresión (4.3), con diferentes evoluciones del error $\hat{V}_{nl}(t)$ generados a partir de varias amplitudes del escalón $U_o(t)$. Como resultado, el valor del error de estimación

$$J = \frac{1}{n}\left(\sum_{i=1}^{n} V_{nl}(t_i) - \hat{V}_{nl}(t_i)\right)^2 \qquad (4.4)$$

se logró reducir a 0.389, obteniendo $\lambda_1 = 1.0$, $C_1 = 4.074$, $C_2 = 15.717$, $C_3 = -4.1$.

La figura (4.5) muestra los datos de $V_{nl}(t)$ junto con la función $\hat{V}_{nl}(t)$ estimada. De ésta se puede concluir que la estimación de $\hat{V}_{nl}$ es satisfactoria. Por tanto la transformada de Laplace de $U_{so}(t)$ se reduce a

$$U_{so}(s) = \hat{V}_{nl}(s) G^{-1}(s) \qquad (4.5)$$

o equivalentemente

$$U_{so}(s) = \frac{\hat{V}_{nl}(s)(s+k_2)}{k_1} \tag{4.6}$$

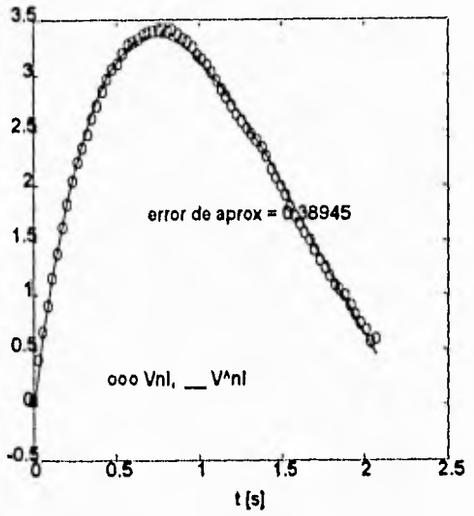


FIG. 4.5 Salida del sistema real y del modelo no lineal estimado

Si se sustituye la transformada de Laplace de la función estimada $\hat{V}_{nl}$ se obtiene la expresión

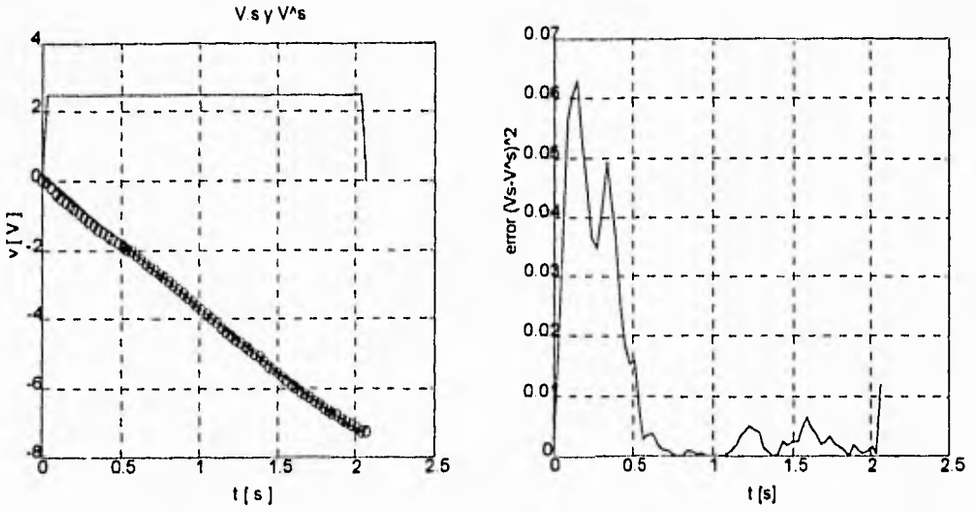
$$U_{so}(s) = \left[\frac{C_1}{k_1}\left(1+\frac{k_2-\lambda_1}{s+\lambda_1}\right)+\frac{C_2}{k_1}\left(\frac{s}{(s+\lambda_1)^2}+\frac{k_2}{(s+\lambda_1)^2}\right)+\frac{C_3}{k_1}\left(1+\frac{k_2}{s}\right)\right] \tag{4.7}$$

que corresponde a la transformada de Laplace de la función $U_{so}(t)$ desconocida. Por tanto en el dominio del tiempo la expresión de $U_{so}(t)$, se puede escribir como

$$U_{so}(t) = \delta(t)(C_1+C_3)\frac{1}{k_1}+\frac{C_3}{k_1}k_2\,u(t)+\left[\frac{C_2}{k_1}+(k_2-\lambda_1)\left(\frac{C_1+C_2\,t}{k_1}\right)\right]e^{-\lambda_1 t} \tag{4.8}$$

donde $\delta(t)$ y $u(t)$ son la función delta Dirac y el escalón respectivamente.

Para validar la caracterización de la fuerza de fricción dada por la ecuación (4.8), se simuló el sistema lineal considerando como excitación $U_s(t) = U_o(t) + U_{so}(t)$.

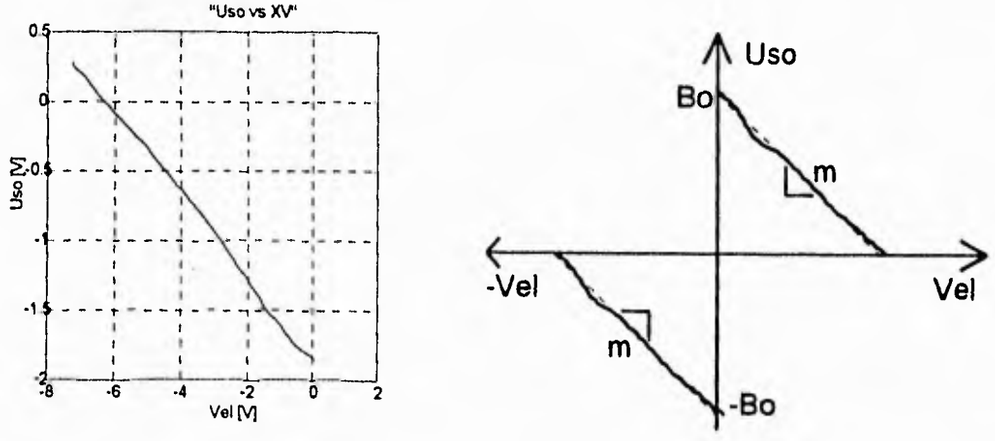


a) b)

FIG. 4.6 a) Respuestas del modelo no lineal, y del proceso real. b) Diferencia entre la salida real, y la simulada con el estimador diseñado.

La figura (4.6) a) muestra la gráfica de la velocidad del sistema simulado $\hat{V}_{nl}(t)$ junto con la velocidad real del sistema $V_s(t)$, en donde se puede apreciar la similitud entre ambas velocidades. La integral del error al cuadrado entre ambas respuestas fue de 0.8752 . La figura (4.6) b) muestra la diferencia entre la salida real y la simulada.

Sin embargo, el contar con una expresión de la fuerza $U_{so}(t)$, en función del tiempo, para una entrada escalón, no se puede decir que se disponga de una caracterización adecuada de la fuerza de fricción seca, en términos de la velocidad del sistema. Ya que se sabe que la fuerza de fricción seca, depende únicamente de la velocidad del sistema, se procedió a encontrar una relación entre la velocidad del carro y la función $\hat{U}_{so}(t)$ (4.8).



a) b)

FIG. 4.7 Gráfica de relación entre U_{so} y la velocidad del carro

La fig. (4.7) a) muestra la relación obtenida entre la velocidad del carro y la fuerza estimada de $\hat{U}_{so}(t)$, la figura (4.7) b) corresponde a la gráfica ideal de la fuerza de fricción seca contra la velocidad real del carro. Como puede verse de las gráficas la no linealidad identificada coincide con la forma ideal para la fricción en el rango de operación de los experimentos (3).

Con objeto de caracterizar la no linealidad presentada en la figura (4.7) a), se repitió el procedimiento para diferentes valores positivos y negativos de la excitación $U_0(t)$. En todos los casos se encontró que la no linealidad podía modelarse satisfactoriamente con la expresión.

$$U_{so}(t) = M_V\, V(t) + B_0\, \text{signo}(V(t)) \qquad (4.9)$$

en donde M_V y B_O son constantes que dependen de la excitación $U_0(t)$. En particular el valor de M_V resultó ser independiente de la entrada $U_0(t)$; el parámetro Bo se desvío de su valor real o Bo = -1.85 con ± 10% de tolerancia. Por tanto, el modelo del carro con la no linealidad estática se puede reducir al diagrama siguiente.

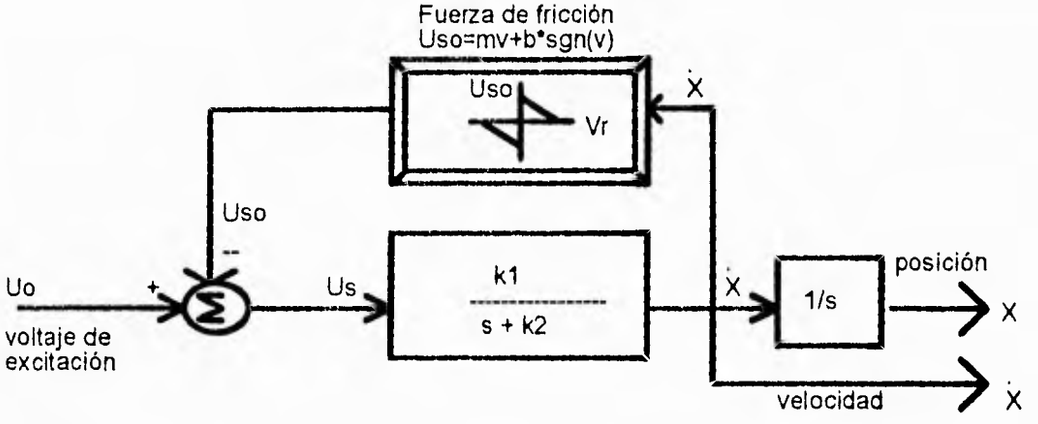


FIG. 4.8 Diagrama de bloques de la planta compensada

Como Bo es un parámetro del cual solo se conoce su media, cualquier controlador diseñado con base en el modelo de la figura (4.8), tendrá no solo que estabilizar y satisfacer los requerimientos de ancho de banda y atenuación de perturbaciones, sino que tendrá que cancelar los efectos de la incertidumbre de Bo, para poder lograr errores pequeños en estado permanente. En particular, como se verá en el capítulo cinco, cuando se estudie el controlador para el sistema completo (carro-péndulo), la incertidumbre juega un papel predominante en el desempeño del sistema de control, por lo que se propone mejorar la caracterización de la fuerza $U_{so}(t)$ por medio de un identificador en línea (un observador dinámico).

4.3 Observador dinámico para la fricción

Dado que la no linealidad de la fricción estimada se puede considerar como una función cuasi estática, con incertidumbre, se propone emplear un estimador dinámico, cuya función sea identificar en todo momento el valor actual de la fuerza de fricción. Esta técnica es ampliamente empleada en la comunidad de control y se conoce como filtro extendido de Kalman u observador (para más detalle ver referencia 2).

La idea básica del observador extendido consiste en asignar un sistema dinámico al parámetro por estimar, $U_{so}(t)$, cuya derivada es cero.

Es decir agregar al estado del sistema la ecuación

$$\frac{dU_{so}}{dt} = 0 \qquad (4.10)$$

y después diseñar un observador o filtro de Kalman en la forma convencional para el sistema aumentado.

En particular para el caso del carro, la ecuación de estado de la velocidad X_2, se reduce a.

$$\dot{X}_2 = -k_1 k_2 X_2 + k_1 U_{so} + k_1 U_0 \qquad (4.11)$$

donde U_{so} es la variable que se desea estimar.

Se hace notar que la velocidad X_2 no depende de la posición X_1 del carro, por lo que se propone emplear un observador de orden reducido, para estimar solamente la velocidad, y el parámetro de la fricción seca.

Bajo estas condiciones el modelo aumentado, para el carro con fricción seca, se reduce a

$$\begin{bmatrix} \dot{X}_2 \\ \dot{X}_3 \end{bmatrix} = \begin{bmatrix} -k_1 k_2 & k_1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} X_2 \\ X_3 \end{bmatrix} + \begin{bmatrix} k_1 \\ 0 \end{bmatrix} U_0 \qquad (4.12)$$

donde

$\dot{X}_2$ = velocidad del carro

$\dot{X}_3$ = fuerza de fricción

y la salida, corresponde a la velocidad, es decir

$$y = \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} X_2 \\ X_3 \end{bmatrix} \tag{4.13}$$

Así, el problema de estimar X_3 dinámicamente se reduce a encontrar un observador del sistema (4.12) a partir de y(t). Debido a que se desea implantar en una computadora digital, tanto el control como el estimador, antes de diseñar el estimador se debe discretizar el modelo (4.12). Para el caso especial del péndulo de laboratorio, se empleo un período de muestreo de 30 [ms].

5 APLICACIÓN DE LA LEY DE CONTROL Y ESTIMADOR EN EL PÉNDULO INVERTIDO

Antes de implantar el controlador y el estimador en el sistema completo, péndulo-carro, se propone analizar los efectos de los diferentes modelos de la fuerza de fricción en el subsistema del carro. La razón principal por la que se hizo este estudio es la de elegir el mejor estimador de la fricción seca y así poder compensarla adecuadamente.

5.1 Control del carro

Con base en la teoría y los modelos descritos en los capítulos anteriores, se diseñó el controlador para el carro considerando que se implantara en una computadora personal.

Por tanto, se tomaran como base las expresiones (2.13), (2.14) y (2.15), eligiendo los polos de lazo cerrado de acuerdo a la dinámica que se desee obtener para calcular e implantar el control.

5.1.1 Cálculo de la ley de control

El procedimiento consiste en asignar los polos para el modelo continuo, y transformarlos en el dominio de la variable z por medio de la ecuación (2.14), antes de calcular las ganancias de la ley de control, usando el método de Ackerman.

La función de MATLAB requiere para el cálculo de la ganancia K del vector de retroalimentación de estados, las matrices del modelo discreto del carro A_{dc} y B_{dc}, cuyos valores numéricos se encuentran en el apéndice A.

Un método alternativo para el cálculo del vector K, es el regulador cuadrático lineal discreto (DLQR), el cual se basa en la minimización de un índice de desempeño que pondera el control y los estados del sistema. Se hace notar que el programa desarrollado, está dotado de esta opción de diseño (referencia 2), a pesar de que ella está fuera del objetivo de esta tesis.

5.2 Modelos para compensar la fricción seca

En los análisis presentados anteriormente se hizo notar la no linealidad existente en el sistema debido a la fricción seca. Por tal motivo, una simple asignación de polos para el modelo lineal no garantiza un buen desempeño.

Para controlar la posición del carro es necesario entonces aplicar la ley de control y compensar la fuerza de fricción. Una manera simple y no necesariamente poco eficiente para lograrlo, consiste en retroalimentar la fuerza de fricción estimada, de tal manera que se logre una cancelación. A continuación se presentan los cuatro modelos de compensación implantados en la práctica desarrollada.

Por tanto el sistema de control junto con las compensaciones propuestas, se puede representar en forma de diagrama de bloques (figura 5.1).

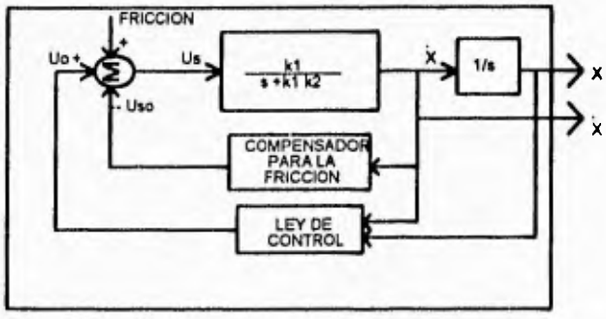


FIG. 5.1 Diagrama de bloques del sistema del carro, con retroalimentación de estados y con compensación

La primera opción consiste simplemente en no compensar la fricción, con lo que se verá qué tan crítico es el control sin compensación.

La segunda opción implantada considera que la fricción seca es constante, excepto que cambia de signo con la dirección del movimiento. Por tanto, se propondrá una compensación estática afectada por el signo de la velocidad (ecuación $U_{so}=B_o$ sign(V(t)), con B_o=cte).

La tercera opción consiste en asignarle a la función no lineal estática, la relación $\hat{U}_{so}(t)$ identificada en el capítulo 4 y descrita a continuación

$$\hat{U}_{so}(t) = M_V\, V(t) + B_o\ \text{signo}(V(t)) \qquad (5.1)$$

La cuarta y última técnica propuesta en la práctica consiste en una compensación a través de un observador dinámico ($U_{so}=\hat{X}_3$), basada en la estimación de la

fricción seca. El diseño del observador o estimador corriente de orden reducido se ha implantando de acuerdo a la metodología propuesta en el inciso (4.2). Es decir, se utilizó la ecuación (2.28), para estimar la fuerza de fricción.

$$\hat{X}_{(kT/k)T} = A_D\hat{X}_{(kT-T/kT-T)} + B_D u_{(kT-T)} + L(y_{(kT)} - C_D\hat{X}_{(kT-T/kT-T)}) \tag{5.2}$$

$$A_D = \begin{bmatrix} -k_1k_2 & k_1 \\ 0 & 0 \end{bmatrix} \tag{5.3}$$

$$B_D = \begin{bmatrix} k_1 \\ 0 \end{bmatrix} U_0 \tag{5.4}$$

$$C_D = [1\ 0] \tag{5.5}$$

donde $\hat{X}_2$ es la estimación de la velocidad del carro (V), $\hat{X}_3$ es la estimación de la fuerza de fricción (U_{SO}) y L es la ganancia del observador que se cálcula de acuerdo a una asignación de polos deseada ó empleando un filtro de Kalman. El diagrama de bloques del esquema de control con observador resultante se presenta en la figura (5.2).

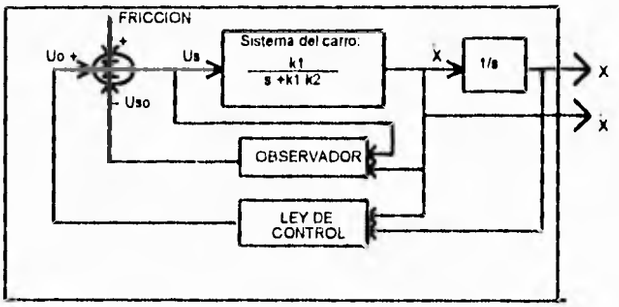


FIG. 5.2 Control del carro, con retroalimentación de estados y compensador dinámico

5.2.1 Comparación de los modelos

A continuación se presenta el experimento propuesto junto con sus resultados para comparar la regulación del carro, con y sin compensación.

Para estudiar el comportamiento del sistema a una respuesta escalón, el usuario debe asignar los polos del estimador y del sistema retroalimentado antes de ejecutar el programa en tiempo real.

El experimento consistió, primero en llevar el carro de una posición inicial de 30 a 0 [cm] sin compensación. La evolución del error de posición y de la variable de control se presentan en la figura (5.3).

Los resultados que a continuación se presentan fueron obtenidos con una asignación de polos de Psk = (-9, -9)

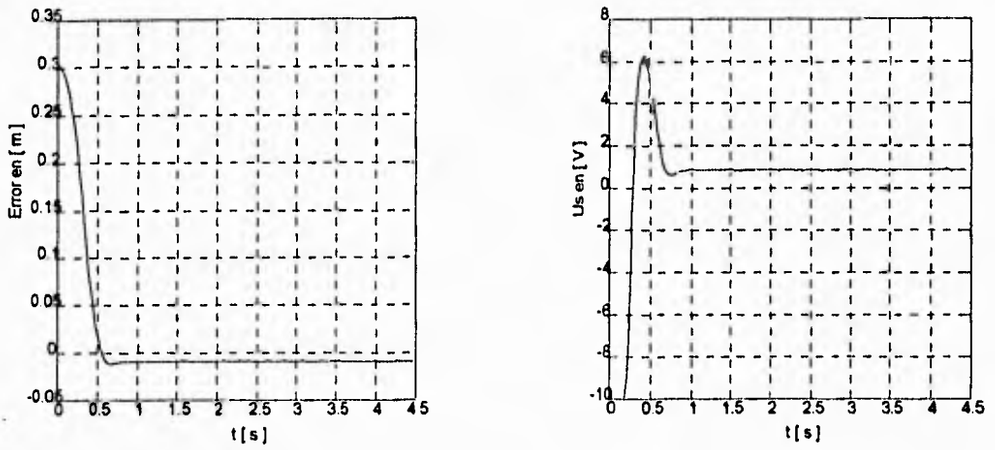


FIG. 5.3 Gráficas del error de posición (para U_{s0}=0) y del voltaje de control

Para evaluar el desempeño del controlador, se calculan las normas obtenidas a partir de la sumatoria del error

$$J_e = \frac{1}{200}\sum_{i=1}^{200}\left(x_{ref}(kT) - x_i(kT)\right)^2 \tag{5.6}$$

para la posición, y para el control a partir de

$$J_u = \frac{1}{200}\left(\sum_{i=1}^{200} u^2(kT)\right) \tag{5.7}$$

Esta última asociada con la energía que se le aplica al sistema para llevarlo de la posición inicial a la final.

De la figura (5.3) se observa la existencia de un error en estado permanente, el cual se justifica debido a la ausencia de compensación de la fricción. Los índices de desempeño J_e y J_u resultantes fueron 0.893 y 34.263.

Como segundo experimento se asigna un valor constante a la fricción y se repite la prueba anterior. La evolución del error y la variable de control se muestran en la figura (5.4). Los desempeños calculados en este caso fueron J_e=0.904 y J_U=35.26.

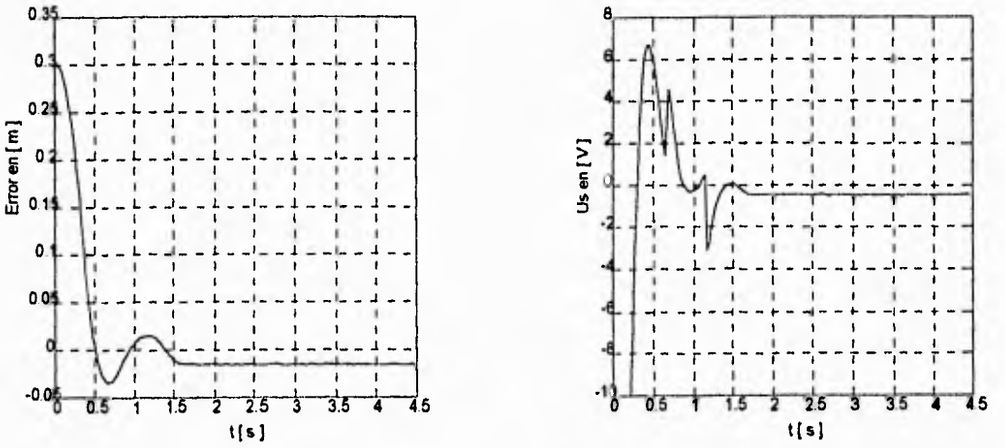


FIG. 5.4 Gráficas del error de posición (para U_{so}=cte), y del control

Para el tercer experimento, la compensación se realiza considerando la función lineal ecuación.(4.9). Los resultados del error de posición y la variable de control se muestran en la figura (5.5).

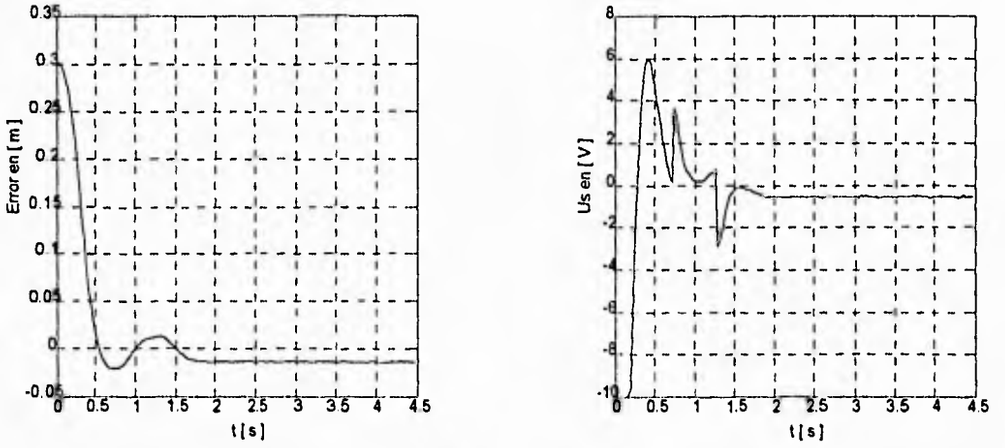


FIG. 5.5 Gráficas del error de posición (para U_{so}=MV(t)+Bo sign(V(t))), y del voltaje de control

Los valores de los índices de desempeño J_e=0.90 y J_u=33.7 se mantienen prácticamente iguales a los casos anteriores. Se hace notar que tampoco en este caso se logro reducir el error en estado permanente.

El ultimo experimento se realizo compensando la fricción por medio del estimador dinámico, con una asignación de polos del observador en PsL=(-9 y -10).

La figura (5.6) muestra la evolución del error y de la variable de control con este último experimento.

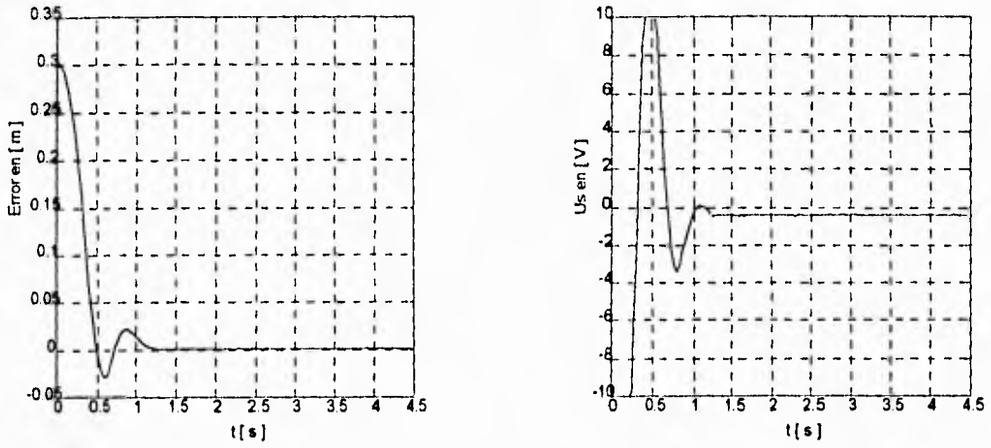


FIG. 5.6 Gráficas del error de posición (para $U_{so}=\hat{X}_3$), y del voltaje de control

Los desempeños resultantes en este caso fueron J_e=0.887 y J_u=42.0.

Es importante notar que con el estimador se reduce el valor del índice J_e, pero a costa de un incremento en la energía que se debe suministrar al sistema (J_u). Además se nota que el error en estado permanente logra eliminarse.

Al comparar las gráficas de la variable de control se observa que todas se saturan, lo cual implica que la energía demandada por el control es mayor a la que el actuador puede dar.

De la comparación de los índices J_e, se observa que el compensador con menor error de posición es el que utilizó un observador. Además, este compensador es el único que logra un error en estado permanente de cero.

De la comparación de la energía requerida para el control, se observa que el menor costo se logra con la compensación lineal, y el mayor con el compensador dinámico (observador).

Por tanto, si se esta buscando precisión en el control del carro, la mejor opción es la compensación por medio de un estimador.

Para disminuir la magnitud del voltaje aplicado al motor se pueden asignar valores más lentos para los polos tanto del estimador, como del control. Es decir, el emplear polos más lentos en la retroalimentación genera una menor demanda de energía a costa de hacer la respuesta del sistema más lenta.

5.3 Control y regulación del péndulo invertido

Debido a que la compensación de la fricción vía un estimador dinámico fue la que proporcionó mejores resultados para el carro, este tipo de compensación es la que se empleará para el sistema completo del péndulo invertido. Por tanto, a continuación se presenta el diseño del controlador y observador del sistema completo. En el diseño se estimara la velocidad angular ω y se compensará la fuerza de fricción U_{so}(fricción estática y seca).

5.3.1 Control del péndulo

Con base en la teoría y los modelos presentados en los capítulos anteriores, se diseñó el controlador para el péndulo, considerando que también se implantará en una computadora personal.

Por tanto, se tomaran como base las expresiones (2.13) , (2.14) y (2.15), eligiendo los polos de lazo cerrado, de acuerdo a la dinámica que se deseé obtener para calcular e implantar el control.

5.3.2 Cálculo de la ley de control

Semejante al caso del carro, en el péndulo invertido el procedimiento consiste en asignar los polos para el modelo continuo (s) y transformarlos en el dominio de la variable z por medio de la ecuación (2.12), antes de calcular las ganancias de la ley de control mediante el método de Ackerman.

La función de MATLAB (Ackerman), requiere para el cálculo de las ganancias K de la ley de control, las matrices del modelo discreto del péndulo A_D y B_D, cuyos valores numéricos se encuentran en el apéndice A.

Al igual que en el sistema con el carro, para el cálculo de la ley de control, también se puede utilizar el regulador cuadrático lineal discreto DLQR.

5.4 Diseño del estimador

Para controlar la posición y ángulo del péndulo invertido es necesario utilizar una retroalimentación de estados (ley de control) y además compensar la fuerza de fricción por medio de un estimador dinámico. Con objeto de comparar los modelos del estimador de orden completo y de orden reducido, se diseñaron e implantaron éstos para el modelo del péndulo invertido.

5.4.1 Estimador de orden completo

Partiendo de la necesidad de estimar la velocidad angular y la fuerza de fricción por medio de un observador dinámico, para el caso de un observador completo, se requieren estimar cinco estados con base en la información de tres variables medibles (posición lineal, velocidad lineal y posición angular).

Por lo que se agrega a los cuatro estados un quinto correspondiente a la fuerza de fricción seca U_{so}, y se considera U_s, como U_0+U_{so}, entonces el sistema de cinco variables queda representado por

$$\dot{X} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{bmatrix} = \begin{bmatrix} p \\ \Phi \\ V \\ \omega \\ U_{so} \end{bmatrix} = \begin{bmatrix} A_n & B_n \\ 0 & 0 \end{bmatrix} X + \begin{bmatrix} B_n \\ 0 \end{bmatrix} U_s \tag{5.8}$$

y el cual sirve como modelo base para diseñar el estimador de orden completo. En la ecuación (5.8) las matrices A_n y B_n son las matrices normalizadas del sistema continuo (ecuaciones. (3.45) y (3.46)). Los valores numéricos del modelo de cinco estados discreto asociado a (5.8), se presentan en el apéndice A.

El cálculo de las ganancias del estimador se realiza utilizando la matriz de salida del sistema y la matriz del sistema junto con los polos elegidos (discretizados para un periodo de muestreo de 30 ms).

Se presenta a continuación, la matriz que define las salidas del sistema

$$C_{d5} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{bmatrix} \tag{5.9}$$

El modelo del estimador corriente se encuentra representado por la ecuación (2.31) aquí reescrita

$$\hat{X}_{(kT/kT)} = A_{Dx}\hat{X}_{(kT-T/kT-T)} + B_{Dx}u_{(kT-T/kT-T)} + Ly_{(kT)} \tag{5.10}$$

donde

$$\begin{aligned} A_{Dx} &= (I - LC)A_D \\ B_{Dx} &= (I - LC)B_D \end{aligned} \tag{5.11}$$

5.4.2 Estimador de orden reducido

El diseño del observador o estimador de orden reducido se implanta de acuerdo a la metodología propuesta en el capitulo 2.3.2. Las salidas conocidas para el caso particular del péndulo invertido son los tres primeros estados y los estados desconocidos son la velocidad angular y la fuerza de fricción seca (U_{so})

$$\hat{X}_{B\,(kT+T)} = \begin{bmatrix} \hat{X}_4 \\ \hat{X}_5 \end{bmatrix} = \begin{bmatrix} \omega \\ U_{so} \end{bmatrix} \tag{5.12}$$

la ecuación (2.38) que se reescribe a continuación, es la expresión utilizada para calcular el estimador de orden reducido.

$$\hat{X}_{B(kT)} = A_B \hat{X}_{B(kT-T)} + B_B u_{s(kT-T)} + F_B Y_{(kT-T)} + L Y_{(kT)} \tag{5.13}$$

donde

$$A_B = \left[A_{22} - L A_{12}\right] \tag{5.14}$$

$$B_B = \left[B_2 - L B_1\right] \tag{5.15}$$

$$F_B = \left[A_{21} - L A_{11}\right] \tag{5.16}$$

El diagrama de bloques del esquema de control resultante con este observador se presenta en la figura (5.7).

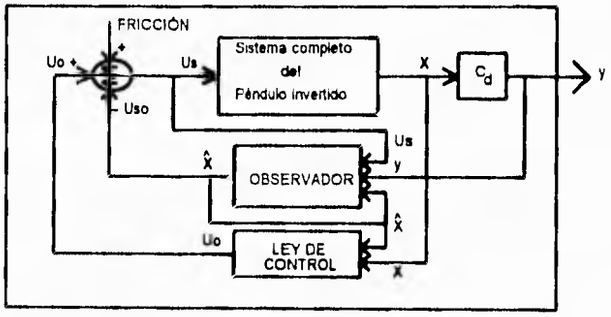


FIG. 5.7 Diagrama del sistema retroalimentado y compensado con observador.

Por tanto se han propuesto dos tipos de observadores, los cuales se comparan a continuación para concluir cual estima y compensa mejor la fricción del sistema.

5.5 Comparación de los estimadores diseñados

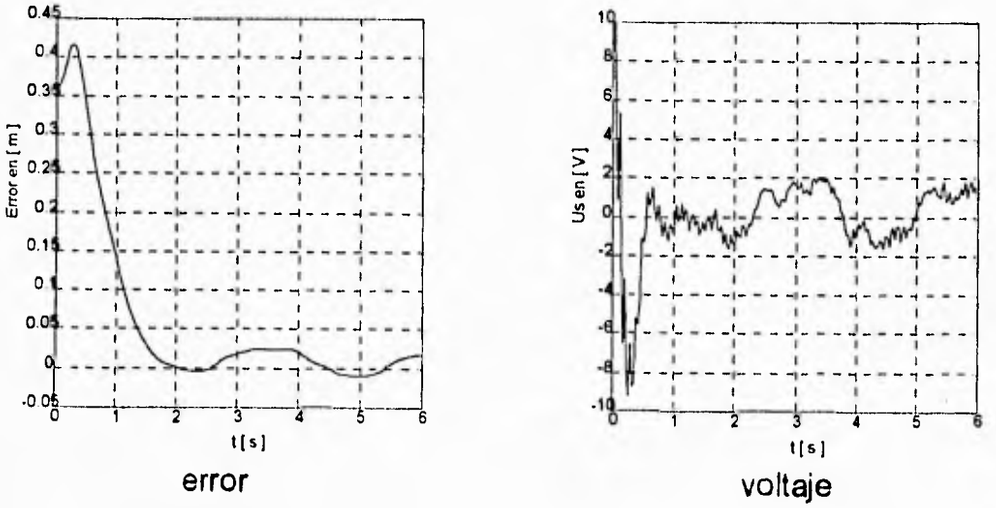
En este inciso se presenta el experimento propuesto para comparar la regulación del péndulo, junto con sus resultados utilizando los estimadores diseñados en el punto 5.3.3.

Para estudiar y simular el comportamiento del sistema a una respuesta escalón, el usuario debe asignar los polos del estimador y del sistema retroalimentado antes de ejecutar el programa en tiempo real, usando las rutinas disponibles en el paquete a través de MATLAB.

El experimento consiste, en primer lugar, en llevar el carro junto con el péndulo de una posición inicial de 35 cm a 0 cm, utilizando para ello cada uno de los estimadores. La evolución del error de posición y de la variable de control se presentan en la figura (5.8). Para evaluar el desempeño del controlador, se emplea tanto la sumatoria del error para la posición (J_e, ecuación. 5.6), como la sumatoria de la señal de control (J_u , ecuación. 5.7).

Los resultados que a continuación se presentan, fueron obtenidos con una asignación de polos para el control de Psk=(-4.3, -4.3, -4.3, -4.3), y una asignación de polos para el observador de orden completo de PsL=(-81, -82, -83, -84, -85)

De la figura (5.8) se observa la existencia de un error pequeño en estado permanente, el cual es debido a que no se logro compensar correctamente la fricción del sistema. Esto se debe a que el error de estimación no tiende a cero.



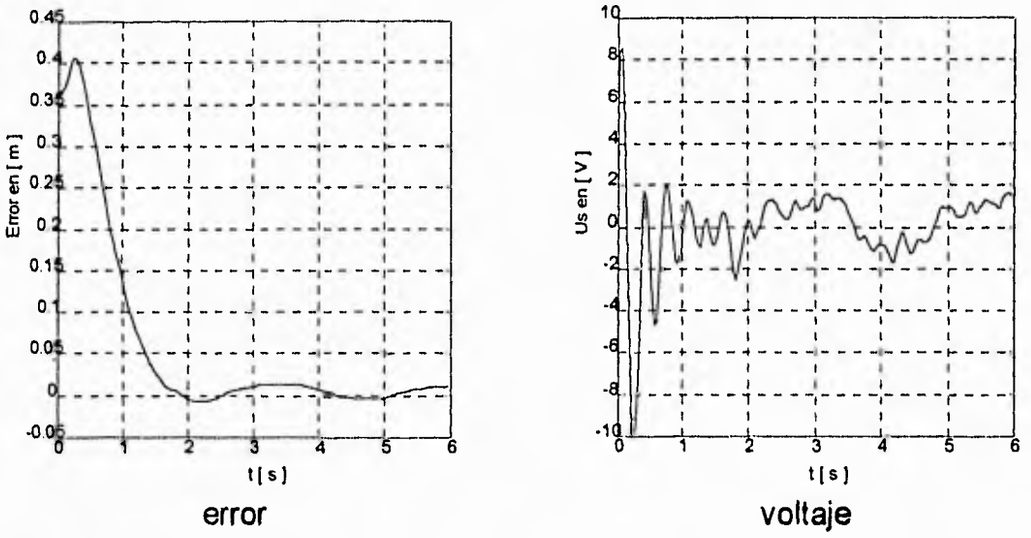
error voltaje

FIG. 5.8 Gráficas del error de posición y del voltaje de control para el péndulo invertido, utilizando un estimador de orden completo.

Los valores de los índices de desempeño J_e y J_u resultantes fueron 1.9063 y 30.320, respectivamente.

Como segundo experimento se utilizó el estimador de orden reducido. Los resultados que a continuación se presentan fueron obtenidos con una asignación de polos para el control de Psk=(-4.3, -4.3, -4.3, -4.3), y una asignación de polos para el observador, de PsL=(-99, -100).

La evolución del error y la variable de control se muestran en la figura (5.9).



error voltaje

FIG. 5.9 Gráficas del error de posición y del voltaje de control para el péndulo invertido, utilizando un estimador de orden reducido.

Los desempeños calculados en este caso fueron J_e=1.8585 y J_u=30.6083.

Al comparar las gráficas de error y del voltaje proporcional a la fuerza de control, puede notarse que el observador reducido logra un error de posición, menor que el observador de orden completo. Las oscilaciones alrededor del punto de operación (0 cm) en la figura (5.9) son menores que las obtenidas en la figura 5.8.

De la comparación del control, se observa que el menor costo se logra con el observador completo y el mayor con el de orden reducido. Por tanto, si se esta buscando precisión en el control del carro, la mejor opción es la compensación por medio de un estimador de orden reducido.

Para disminuir la magnitud del voltaje aplicado al motor, se pueden asignar valores más lentos para los polos tanto del estimador como del control. Es decir, el emplear polos más lentos en la retroalimentación genera una menor demanda de energía a costa de hacer la respuesta del sistema más lenta. Por tanto el

control con mejor regulación es el del **observador de orden reducido**, sin embargo hay que suministrarle mayor energía al sistema para obtener una buena regulación de la posición del péndulo.

Se puede concluir que para compensar la no linealidad del sistema, el uso de un estimador dinámico es la mejor opción, ya que como se vio para el control de posición del carro, este tuvo un mejor desempeño. Y ademas, el observador que logró un mejor desempeño desde un punto de control, fue él de orden reducido.

6 PAQUETE INTERACTIVO DE CONTROL PARA EL PÉNDULO INVERTIDO

6.1 Objetivo del paquete

La integración del paquete interactivo de control para el péndulo invertido denotado **PCPEIN** tiene como objeto facilitar al usuario el estudio de la técnica de control por retroalimentación de estado para sistemas lineales con no linealidades estáticas.

Como metas particulares se desea que el usuario pueda con base en el equipo de laboratorio (péndulo invertido):

-estudiar los efectos de la fricción seca en el comportamiento del servomecanismo
-probar técnicas de estimación de parámetros en linea
-obtener destreza en la técnica de diseño de controladores por asignación de polos
-validar las bondades de los observadores corrientes de orden reducido en un proceso real con incertidumbres.

6.2 Descripción del paquete

Para lograr el objetivo arriba presentado, se diseño la práctica usando el paquete matemático MATLAB para realizar las tareas de análisis y diseño, junto con rutinas en C que interactuan con el sistema operativo DOS para controlar el proceso en linea a través de convertidores analógicos/digitales y digitales/analógicos, y para presentar en línea los resultados.

El dotar al sistema de capacidad para guardar el comportamiento de las variables físicas facilita el análisis fuera de línea de los diferentes diseños de controladores y observadores de manera rápida y simple.

En otras palabras, el paquete esta estructurado con dos tipos de módulos, uno de análisis y diseño fuera de línea y otro, que opera en tiempo real con una serie de opciones interactivas que permite además de controlar el péndulo, conocer y registrar el estado del proceso en cualquier instante de tiempo. En el menú principal se incluye la opción ***Descripción general del programa*** para ayudar a los usuarios principiantes. Este prontuario se encuentra en el apéndice C.

El modulo de diseño y análisis estructurado en MATLAB esta dividido, en cuatro submodulos; dos para analizar los resultados de los experimentos y dos para validación de los diseños por medio de simulaciones.

El módulo de control en línea para la regulación del sistema en tiempo real se realizo en lenguaje C y esta formado por tres subprogramas. El primero llamado

Resp_esc.h, se emplea para registrar la respuesta del carro a una entrada escalón, los dos restantes **C_carro.h** y **C_pendul.h** ejecutan la tarea de control en línea, tanto para el carro como para el péndulo invertido respectivamente.

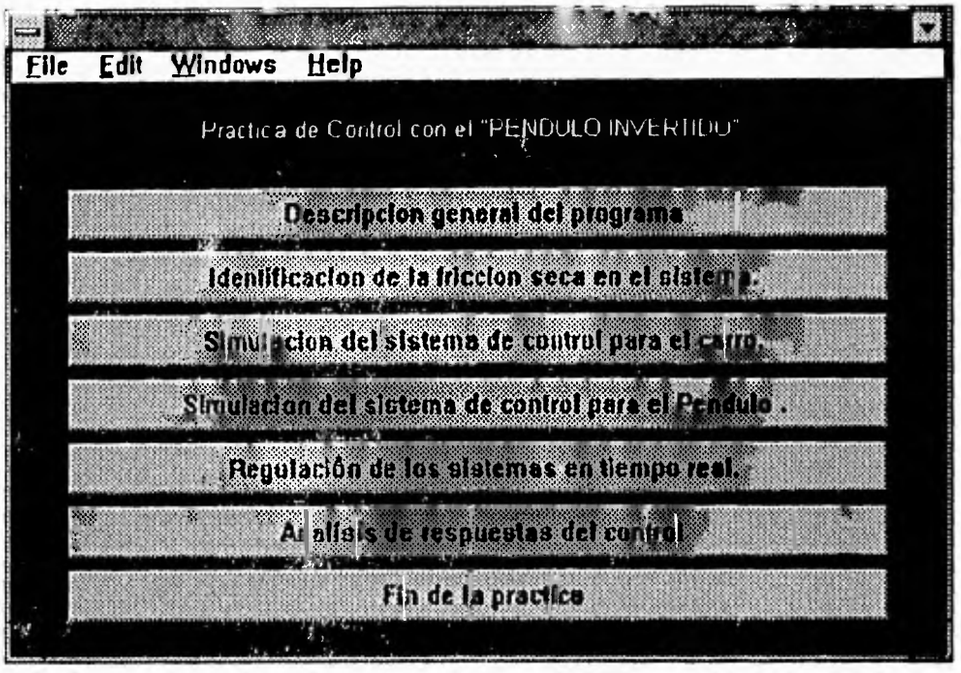


FIG. 6.1 Menú principal del paquete PCPEIN.

Al ejecutar el programa **PCPEIN**, desde Windows se muestra en la pantalla de la PC el menú de la figura (6.1).

Es decir, el usuario tiene posibilidad de elegir entre siete opciones. La primera **Descripción general del programa**, corresponde a la descripción del programa y se recomienda seleccionarla la primera vez que se usa el paquete. Las siguientes tres opciones **Identificación de la fricción seca en el sistema**, **Simulación del sistema de control para el carro** y **Simulación del sistema de control para el Péndulo**, ayudan al diseño y análisis del sistema, usando MATLAB. La quinta alternativa **Regulación de los sistemas en tiempo real**, está encargada de llamar al programa implantado en tiempo real, que realiza tareas de control en lazo abierto y lazo cerrado, de acuerdo con los atributos especificádos en archivos y suministrados por teclado. La penúltima opción **Análisis de la respuesta del control** ayuda a estudiar y validar con MATLAB el experimento con los datos que se hayan grabado durante la ejecución del control en línea. Finalmente la última elección sirve para dar por terminado el experimento.

6.3 Módulo de análisis y diseño fuera de línea

La rutina de **Identificación de la fricción seca del sistema**, fue diseñada con el objeto de analizar registros de la respuesta escalón del carro y caracterizar la no linealidad debida a la fricción del servomecanismo, basándose en el procedimiento descrito en el capítulo 4. Por tanto debe tenerse el registro del comportamiento real de la posición y de la velocidad del carro para una entrada escalón.

La rutina de identificación de la fricción seca en el sistema: ***respesc.m*** realiza básicamente las siguientes funciones:

-graficar la evolución de la posición y velocidad tanto del carro como del modelo lineal con base en el experimento descrito en el inciso 6.4.1.

-ajustar la función no lineal con base en el procedimiento descrito en el inciso 4.2.

-validar en simulación el modelo de la función calibrada para la fuerza de fricción seca.

La rutina de **Diseño y simulación del control del carro**, llamada ***simulcar.m*** calcula las ganancias de la ley de control y del estimador de la fuerza de fricción seca que serán utilizadas para el control del carro en tiempo real. Se cuenta con dos opciones de diseño, asignación de polos y DLQR. La rutina permite además simular el sistema diseñado con el controlador y estimador. Antes de terminar la rutina pregunta al usuario si desea salvar los valores de diseño para que éstos sean empleados posteriormente por la rutina de control en tiempo real del carro.

La rutina de **Diseño y simulación del sistema de control para el péndulo invertido**, llamada ***c_pendul.m*** calcula básicamente las ganancias de la ley de control y estimador para la fricción seca y la velocidad angular, que posteriormente serán utilizadas. Para el diseño se cuenta con dos opciones; asignación de polos y DLQR. El observador corriente es diseñado considerando dos opciones, observador de orden completo u observador de orden reducido. Después de realizar los cálculos, la rutina permite simular el comportamiento del sistema cuando se emplea un observador predictivo y corriente; ésto con el fin de analizar el efecto de la predicción en la dinámica del estimador. Al final de la simulación el usuario determina si desea salvar las ganancias encontradas, para posteriormente ser usadas en el control en tiempo real.

Se hace notar, que los archivos generados por las rutinas de diseño para el carro y el péndulo, están almacenados en código ascii, para que puedan ser usados por el programa en línea.

La rutina de **Análisis de respuestas del control**, llamada ***nv2.m*** ayuda a estudiar el comportamiento del control implantado, ya sea para el carro o el péndulo. Por tanto se debe tener registrado, al menos un experimento del control en lazo cerrado del carro y/o péndulo para ejecutarlo. En particular, la rutina gráfica el error de posición, así como la evolución del control y calcula la integral del error cuadrático de posición y la del control cuadrático (J_e y J_u, ecuación 5.1 y 5.2).

Debido a que el código de las rutinas en MATLAB es auto contenido y de fácil interpretación, no se consideró necesario hacer una descripción detallada de las rutinas, y unicamente se incluye el listado de los programas en el apéndice C; sin embargo a continuación se describen con detalle las rutinas escritas en C para el control y presentación de resultados en línea.

6.4 Módulo de tareas en línea

La ejecución de las tareas de este modulo se inicializa seleccionando en el menú principal, figura 6.1, la opción ***Regulación de los sistemas en tiempo real***; como resultado de ésta acción se activa la pantalla presentada en la figura 6.2, y el usuario está en posibilidad de ejecutar cualquiera de las tres rutinas de control en línea que se describen a continuación.

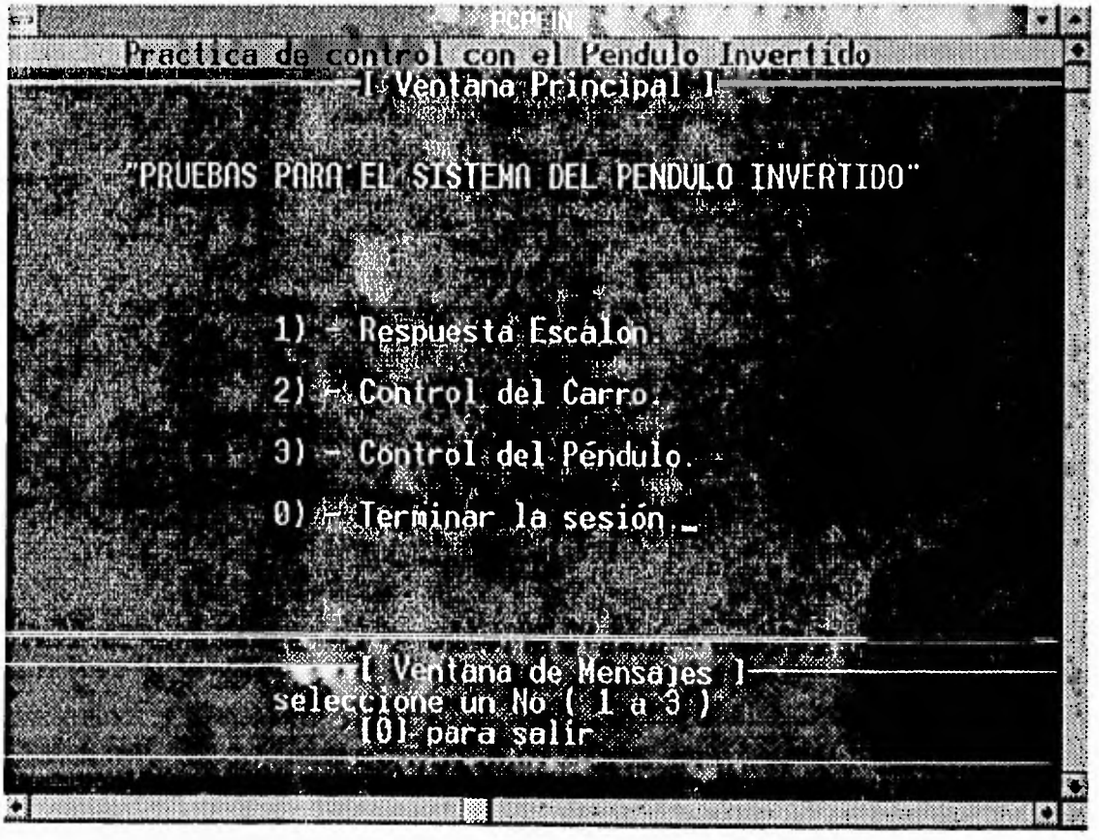


FIG. 6.2 Menú de las rutinas en tiempo real.

6.4.1 Respuesta escalón del sistema

El objetivo de esta rutina es excitar al sistema con una señal escalón, para obtener su respuesta y generar un archivo de datos con los valores de la velocidad, posición, tiempo y escalón aplicado, para poder estudiar y analizar su comportamiento en estado transitorio y permanente.

La primera tarea del modulo, es transparente para el usuario, y consiste en verificar que el equipo externo esta en condiciones de operación y dar un mensaje indicando que el carro debe colocarse en uno de los extremos de la barra.

A continuación se pregunta por la magnitud del escalón (Volts) que se desea aplicar, (menor a | 3.3 | [v]), el signo del escalón es asignado en función de la posición del carro. Por convención el carro se desplaza positivamente de izquierda a derecha y negativamente de derecha a izquierda, el sentido de la velocidad es contrario a esta convención.

Finalmente el programa envía el voltaje y registra el comportamiento del sistema con una frecuencia de muestreo de 30 ms. Posteriormente almacena las muestras del experimento, y solicita el nombre del archivo donde guarda la información con formato de programa de MATLAB.

6.4.2 Control del carro

Esta rutina se encarga de llevar a cabo el control de posición del carro en tiempo real, empleando los tres diferentes métodos de compensación propuestos en el inciso 5.2.

Primero el programa lee los archivos generados durante el diseño de la ley de control y observador, y después realiza el control del carro ya sea sin compensación o con cualquiera de los compensadores implantados.

Para poder ejecutar esta rutina se debe tener en el subdirectorio de trabajo el conjunto de archivos de donde se leerán los parámetros del controlador y de la caracterización de la fuerza de fricción.

El diagrama de flujo presentado en la figura (6.3), muestra la secuencia de la rutina de control, la cual esta formada por los siguientes submodulos:

1.- Lectura de archivos (ganancias y matrices del sistema).
2.- Selección de la compensación para la fuerza de fricción.
3.- Iniciación de variables y equipo.
4.- Atención a la interrupción y cálculo del control.
5.- Almacenamiento temporal de variables.
6.- Graficacion de resultados en tiempo real.
7.- Almacenamiento de datos en archivos (con formato para MATLAB).

y cuyas tareas se describen a continuación.

El submódulo de lectura de archivos se inicia automáticamente al seleccionar la rutina y es el encargado de adquirir de un archivo las matrices del controlador y del estimador que se desea probar. Además mediante el siguiente menú, se escoge el tipo de compensación para la fuerza de fricción bajo estudio.

-Comandos de selección para el tipo de compensación

[1]- Selección de compensación nula (Uso=0).
[2]- Selección de compensación constante (Uso=K(signo(velocidad)).
[3]- Selección de compensación estática.
(Uso=M*velocidad + Bo(signo(velocidad)).
[4]- Selección de compensación por observador dinámico.
(Uso=Fza.fricción = X5^).

Posteriormente, automáticamente se verifica que el equipo esté en condiciones correctas de operación. En caso positivo se pregunta por la posición deseada del carro, teniendo libertad para seleccionar si se desea almacenar en un archivo el resultado del experimento. Al responder estas preguntas se activa la rutina de interrupción que calcula el control y mantiene activado el actuador, iniciándose propiamente el control en línea. Esta acción se ejecuta de manera compartida con un menú de opciones para manejar el experimento y el tipo de presentación de resultados. Es decir se puede modificar en línea la presentación de resultados y el estado del experimento.

Dentro de la opción de alteración del experimento se cuenta con las siguientes posibilidades vía el teclado de la PC:

[D]- Selección de una nueva posición de referencia, para realizar el control y regulación del carro.
[A]- Almacenaje de muestras durante el control (200 muestras).
[W]- Cambio de posición y almacenar datos al mismo tiempo ([D]+[A]).
[+]- Incremento de la posición del carro (desplazarlo hacia la derecha 2 cm).
[-]- Decremento de la posición del carro (desplazarlo hacia la izquierda 2 cm).
[espacio]- Mandar el carro al origen (centro del riel).
[*]- Terminación del control en línea cuando se está en modo texto y regresar a modo texto cuando se esta en modo gráfico.

Para la presentación de resultados se cuenta con dos regímenes, el de modo texto y el gráfico. Al iniciarse el control está activado el modo texto y como resultado se presentan continuamente los valores instantáneos de las variables del proceso. Bajo este modo el oprimir la tecla [*] causa, como se mencionó anteriormente, que se suspenda el control en línea estando el usuario en posibilidad nuevamente de seleccionar entre las cuatro opciones del menú de las rutinas en tiempo real (figura 6.2).

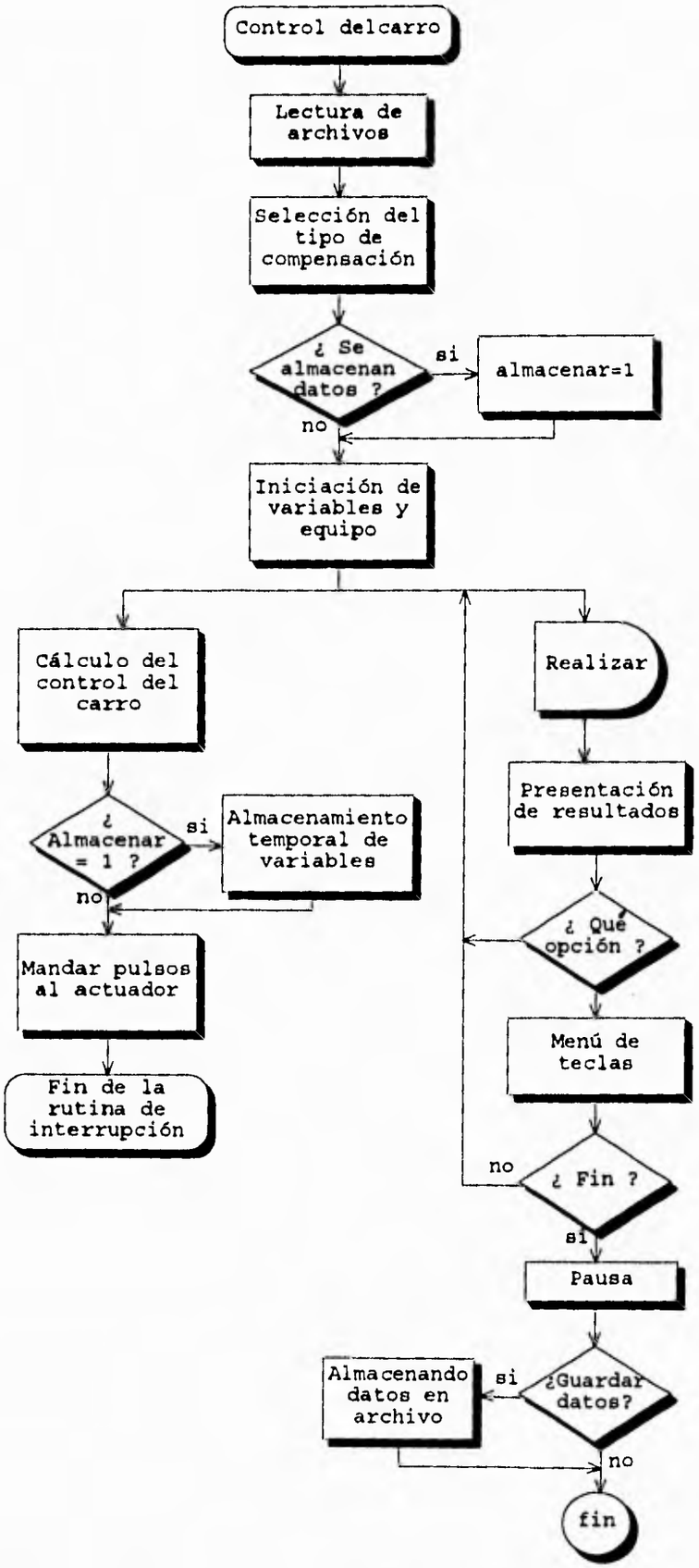


FIG. 6.3 Diagrama de flujo, del programa de control para el carro

La activación del modo gráfico se logra oprimiendo la tecla [G] iniciándose así la emulación de un osciloscopio por medio de la pantalla. Este modo permite graficar una variable del proceso en función del tiempo dentro del siguiente conjunto: posición, velocidad, control sin compensación, velocidad estimada, fuerza de fricción y control compensado. Además se presentan simultáneamente los valores instantáneos de las variables del sistema. La selección de la variable a graficar se logra oprimiendo la tecla asociada a la variable de acuerdo con la siguiente tabla.

-Teclas para selección de la variable a graficar:

[P]- posición (p).
[V]- velocidad del carro (V).
[B]- velocidad estimada (X3^).
[C]- ley de control (Uo).
[0]- fricción estática y seca (Uso).
[U]- fuerza de control del sistema (Us).

La tecla [H] estando en modo gráfico congela la gráfica en la pantalla. Para cambiar de modo gráfico al texto, es necesario oprimir la tecla [*]. Se hace notar que para suspender la tarea de control en línea, es necesario estar en modo texto. Por tanto si se esta en modo gráfico es necesario oprimir dos veces la tecla [*] para suspender el control en línea.

Por otro lado la rutina de atención a la interrupción que corre en modo compartido con las rutinas de presentación de resultados y de manejo del experimento, es la encargada de enviar un tren de pulsos cada 3 ms al actuador del equipo para mantenerlo activo durante el control, de manejar los convertidores Analógico/Digital y Digital/Analógico, los puertos digitales de entrada y salida, y de actualizar la acción de control a cada instante de muestreo (referencia 4).

Esta interacción compartida entre rutinas se logra programando el temporizador de la computadora para que genere una interrupción cada 3 ms.

La figura 6.4 presenta el diagrama de flujo de la rutina del cálculo del control, la cual es atendida cada 10 interrupciones, usando un contador. En otras palabras la rutina se ejecuta cada 30 ms y una vez que ésta se termina el procesador regresa a continuar con las tareas del módulo principal (figura 6.3).

Las actividades que realiza la rutina de atención a la interrupción son las siguientes

1).Leer los estados de los convertidores (posición y velocidad del carro).
2) Verificar que el carro esté dentro de los limites de +/- 50 [cm.], de no ser así termina el control.
3) Verificar la posición destino y estimar la fricción (Uso).
4) Calcular la ley de control Uo (descrita por la ecuación (2.15)).
5) Compensar (Us=Uo+Uso) la fuerza de fricción.

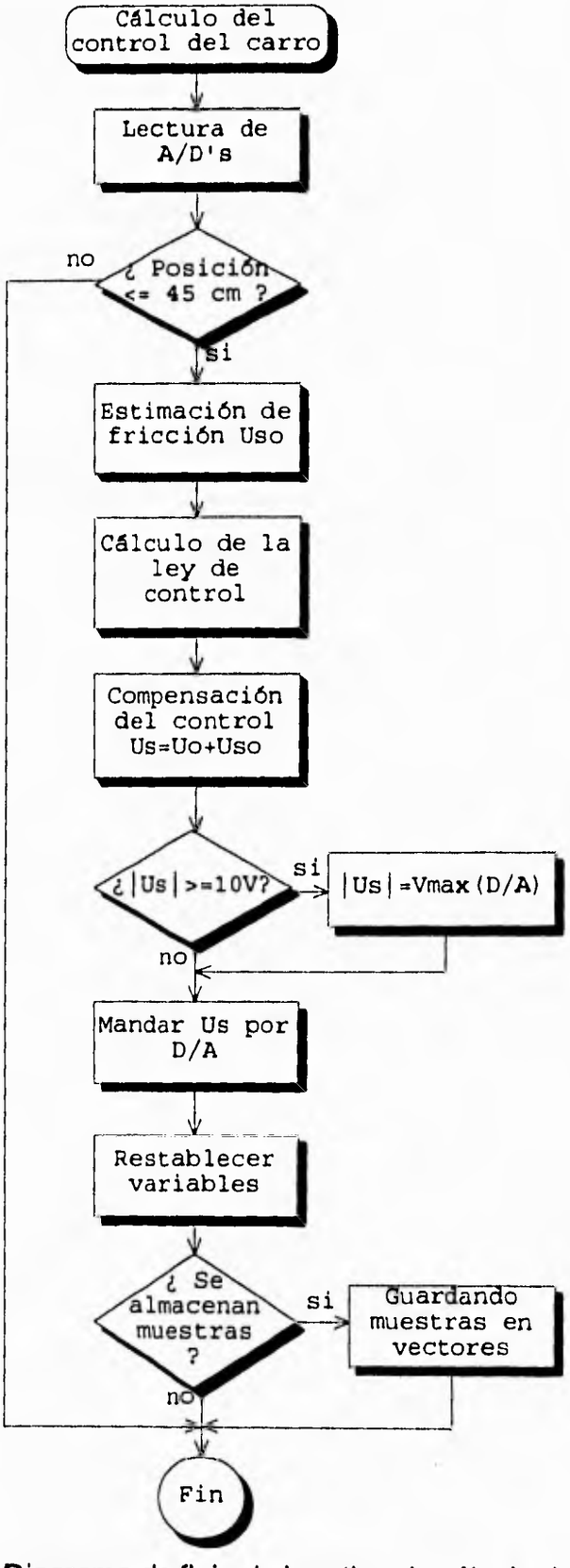


FIG. 6.4 Diagrama de flujo de la rutina de cálculo del control

6) Verificar que la magnitud del control no rebase los límites de los convertidores Digital-Analógico (± 10 [v]). En caso de saturación se sustituye por el voltaje máximo y envía una alarma acústica.
7) Enviar el voltaje por el convertidor uno.
8) Restablecer los estados de algunas variables, y sí está activada la opción de almacenaje, guardar los estados del sistema y regresar al programa principal.

6.4.3 Control del péndulo invertido

La rutina de **control del péndulo invertido** es la encargada de ejecutar, de manera muy similar a la rutina del control del carro, el control en línea del péndulo invertido a través de un conjunto de opciones que gobiernan el experimento y la presentación de los resultados.

Básicamente primero se leen los archivos de datos de la ley de control y del observador que se desea estudiar y después se ejecuta el control del sistema de acuerdo a las opciones seleccionadas.

Para poder ejecutar esta rutina primero se debe tener almacenado en un archivo del subdirectorio de trabajo, los datos del controlador y estimador bajo prueba. Se hace notar que se puede probar tanto un observador de orden reducido como uno completo. Sin embargo debe coincidir la opción que se desea probar con la rutina seleccionada cuando se diseñó el observador.

El diagrama de flujo presentado en la figura 6.5 muestra la secuencia de la rutina, la cual similarmente a la rutina del control del carro está formada por los siguientes 7 submódulos.

1.- Lectura de archivos (ganancias y matrices del sistema).
2.- Selección del observador (de orden completo ó de orden reducido).
3.- Iniciación de variables y equipo.
4.- Atención a la interrupción y cálculo del control.
5.- Almacenamiento temporal de variables.
6.- Graficación de resultados en tiempo real.
7.- Almacenamiento de datos en archivos (con formato para MATLAB).

Inicialmente el programa solicita por medio del menú de la figura 6.6, el tipo de observador con el cual se realizara el control. Después, el programa lee los archivos del observador y de la ley de control, y automáticamente se verifica que el equipo esté en condiciones correctas de operación. En caso positivo se pregunta por la posición deseada del carro, iniciandose así el control del péndulo, y teniendo libertad para seleccionar si se desea almacenar en un archivo el resultado del experimento.

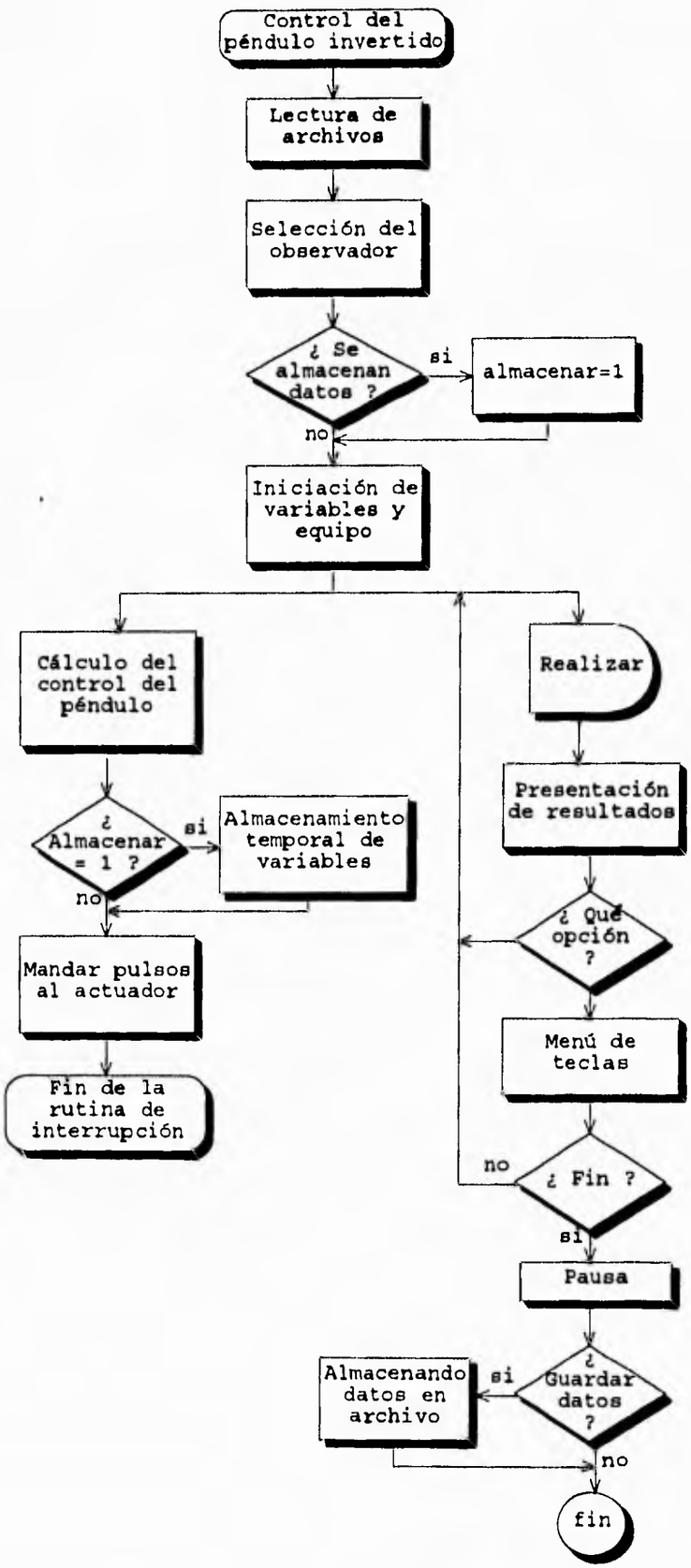


FIG. 6.5 Diagrama de flujo de la opción control del péndulo

Al responder estas preguntas se activa la rutina de interrupción que calcula el control y mantiene activado el actuador, iniciándose propiamente el control en línea. Esta acción se ejecuta de manera compartida con un menú de opciones para manejar el experimento y el tipo de presentación de resultados. Es decir se puede modificar en línea la presentación de resultados y el estado del experimento. De la misma manera que en la opción del carro.

Dentro de la opción de alteración del experimento se cuenta con las siguientes posibilidades vía el teclado de la PC:

[D]- Selección de una nueva posición de referencia, para realizar el control y regulación del péndulo.
[A]- Almacenamiento de muestras durante el control (200 muestras).
[W]- Cambio de posición y almacenar datos al mismo tiempo ([D]+[A]).
[+]- Incremento de la posición del carro (desplazarlo hacia la derecha 2 cm).
[-]- Decremento de la posición del carro (desplazarlo hacia la izquierda 2 cm).
[espacio]- Mandar el carro al origen (centro del riel).
[*]- Terminación del control en línea cuando se está en modo texto, y regresar a modo texto cuando se esta en modo gráfico.

Para la presentación de resultados se cuenta con dos regímenes, el de modo texto y el gráfico. De la misma manera que para el control del carro. Bajo el modo texto al oprimir la tecla [*] se suspende el control en línea estando el usuario en posibilidad nuevamente de seleccionar entre las dos opciones del menú de las rutinas en tiempo real (figura 6.6).

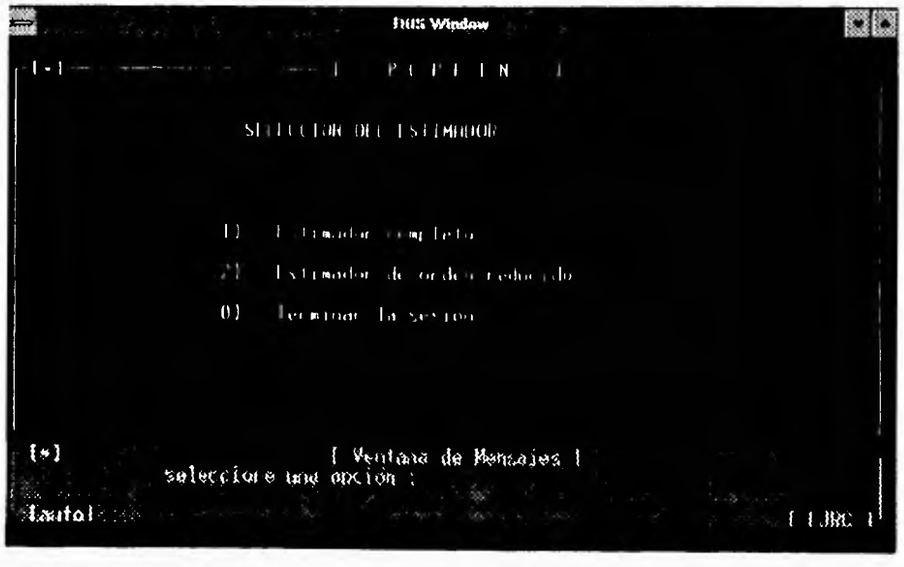


FIG. 6.6 Menú de selección del observador

La activación del modo gráfico se logra oprimiendo la tecla [G] con lo cual se grafica una variable en función del tiempo dentro del siguiente conjunto: posición, ángulo, velocidad, velocidad angular, control sin compensación, velocidad estimada, fuerza de fricción, y control compensado (en caso de usarse el observador de orden completo también se pueden graficar los estimados de la posición, velocidad y del ángulo). Además se presentan simultáneamente los valores instantáneos de las variables del sistema. La selección de la variable a graficar se logra oprimiendo la tecla asociada a la variable de acuerdo con la siguiente tabla.

[P]- posición del carro.
[E]- error de posición.
[V]- velocidad del carro.
[A]- ángulo del péndulo.

[C]- ley de control Uo.
[0]- fricción estática y seca Uso.
[U]- fuerza de control del sistema Us.

[1]- 'X^1- Estimación de la posición del carro.
[2]- 'X^2- Estimación del Ángulo del péndulo.
[3]- 'X^3- Estimación de la Velocidad del carro.
[4]- 'X^4- Estimación de la Velocidad angular.

-Teclas de control para el movimiento del carro.

[+]- Incrementar la posición del carro (desplazarlo hacia la derecha 5 cm).
[-]- Decrementar la posición del carro (desplazarlo hacia la izquierda 5 cm).
[espacio]- Mandar el carro al origen (centro del riel).

La tecla [H] estando en modo gráfico congela la gráfica de la pantalla. Para cambiar de modo gráfico al texto es necesario oprimir la tecla [*]. Por tanto, si se esta en modo gráfico es necesario oprimir dos veces la tecla [*] para suspender el control en linea. Se hace notar que para suspender la tarea de control en línea es necesario estar en modo texto.

Por otro lado la rutina de atención a la interrupción que corre en modo compartido con las rutinas de presentación de resultados y de manejo del experimento, es basicamente la misma que la del carro (figura 6.4) es la encargada de enviar un tren de pulsos cada 3ms al actuador del equipo para mantenerlo activo durante el control, de manejar los convertidores Analógico/Digital, y Digital/Analógico, los puertos digitales de entrada y salida, y de actualizar la acción de control para el péndulo invertido a cada instante de muestreo. La unica diferencia estriba en el cálculo del estimador y el control.

Debido a que las actividades que realiza la rutina de atención a la interrupción son similares a las del control del carro, presentadas anteriormente, se omite aquí la descripción.

6.5 Experimento Patrón

Con base en un experimento a continuación se ilustra la operación del paquete **PCPEIN**. El instructivo para la instalación de **PCPEIN** se encuentra en el apéndice C y los requerimientos mínimo para la ejecución son:

1) PCAT-386 con ambiente Windows
2) Tarjeta de convertidores PCL812 de AMIRA [4]
3) Péndulo invertido y actuador AMIRA [4]
4) Paquete MATLAB para windows [5]
5) Paquete **PCPEIN**

El experimiento cubre los aspectos presentado en los capítulos 4 y 5 de este trabajo. Es decir, modelado de una nolinealidad estática, que para el caso del péndulo es la fuerza de fricción, diseño de controladores con retroalimentación de estados, y síntesis de observadores predictivos y corrientes tanto de orden reducido como completo. Para simplificar la explicación del experimento, éste se ha divido en tres tareas independientes, la de modelado y calibración de la fuerza de fricción del carro, la de diseño del control del carro, y la del control de péndulo.

Una vez instalado el paquete **PCPEIN**, encienda el actuador del sistema y estando en el ambiente Windows, ejecute el programa MATLAB y corra la función ***pendulo.m*** que se encuentra en el subdirectorio x:\pendulo\ con x el disco donde se almacenó el paquete. Como resultado de esta serie de acciones se presenta en la pantalla el menú principal del paquete, figura 6.1. Se recomienda a los usuarios novatos seleccionar la opción **Descripción del programa** para familiarizarse con el paquete.

6.5.1 Modelado y Calibración de la fuerza de fricción

La iniciación de la tarea de modelación y calibración de la fuerza de fricción se realiza excitando al sistema con un escalón y obteniendo la posición y velocidad del carro sin péndulo. Para tener un mayor rango de operación del sistema, el carro debe estar posicionado en uno de los extremos de la barra. Bajo estas condiciones de operación el usuario debe activar la ventana **Regulación de los sistemas en tiempo real**, del menú principal, o el ícono **PCPEIN** de las aplicaciones de Windows. Esta acción genera el menú de la figura 6.7 iniciandose el diálogo que activa la rutina en tiempo real.

Despúes, al oprimir la tecla 1, la computadora pregunta por medio de la pantalla mostrada en la figura 6.8 el valor del escalón de entrada y el cual debe ser menor 3.3 V para evitar aceleraciones grandes. Para el ejemplo aquí tratado se eligio un valor de 2.5 V.

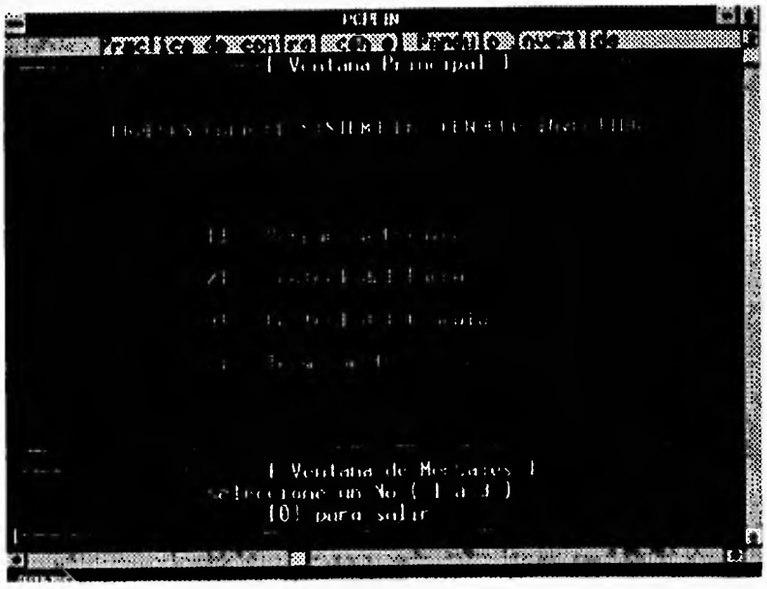


FIG. 6.7 Menú de las rutinas en tiempo real

Valor del escalón, de entrada ?
2.5[↵]

Escalón = 2.5 [V]

Posición inicial del carro = -8.96455 [V] => -0.601648 [m]

Mover el carro a cualquiera de los extremos [izq.] o [der]

"INICIA EL CONTROL"

FIG. 6.8 Condiciones para el escalón

La iniciación del control se percibe por un señal acústica y la duración de la excitación depende de tiempo en que tarde el carro en alcanzar el otro extremo de la barra. Al encontrarse el carro en el extremo de la barra se suspende el escalón y se muestra en una ventana (figura 6.9), los valores finales de la posición y velocidad del carro, y el número de muestras registradas con un intervalo de 30ms. El diálogo continua al oprimir cualquier tecla por medio de la ventana de la figura 6.10. En este momento se registra el nombre del archivo, y se pregunta si desea continuar generando respuestas escalón. En caso afirmativo la rutina presenta la ventana de la figura 6.8; por el contrario en caso negativo [N] el programa regresa al menú de las rutinas en tiempo real (figura 6.7).

Escalón para excitar el carro = 2.5 [V]

Posición inicial del carro = -8.96455 [V]=> -0.601648 [m]

Posición final = 6.68309 [V]=> 0.448529 [m]
Velocidad final = -7.27528 [V]=> 0.95226 [m/s]

activo = 0
No de muestras leídas = 70

FIN DEL CONTROL
[↵]

FIG. 6.9 Aplicación del escalón al sistema

Grabar los datos en un archivo [s/n] ? S

Dar el nombre del archivo con extension m, incluyendo
unidad y directorio

D:\pendulo\ejresc1.m[↵]

FIG. 6.10 Almacenando muestras en archivo

Dar el nombre del archivo para cargar los datos, el cual
debe tener extension m. "0" para salir

a:\ejresc1[↵]

Fig. 6.11 Datos del archivo

Para continuar la tarea de modelación e identificaión se debe desactivar la ventana de las rutinas en tiempo real y seleccionar la opción **identificaicón de la fricción seca en el sistema.** En la primera pantalla (figura. 6.11) se pregunta por el nombre del archivo de datos de la respuesta al escalón del carro. Con estos datos el programa realiza las siguientes tareas:

1) Ajustar, los parámetros C1, C2, C3 y lan1de la función de la ecuación 4.3 minimizando el error de velocidad, ecuación 4.4, por medio de la caja de herramienta **optimization.** Durante el ajuste se muestra la ventana de la figura 6.12, en donde se indica el tipo de la función que se aproxima.

2) Graficar la evolución de la función ajustada junto con los datos; para el ejemplo considerado se obtuvo la gráfica de la figura 6.13.

FUN_JU2 regresa el error entre los datos y los valores calculados, asumiendo una función de la forma:

```
FUN = C1*exp(-lan1*t) + C2*t*exp(-lan1*t) + C3;
        |     |     |      |     |
       X1    X2    X3     X2    X4
```

Fig. 6.12 Ajuste de los parametros de la función

3) Reportar a través de la ventana de la figura 6.14, los coeficientes que minimizan el error de velocidad.

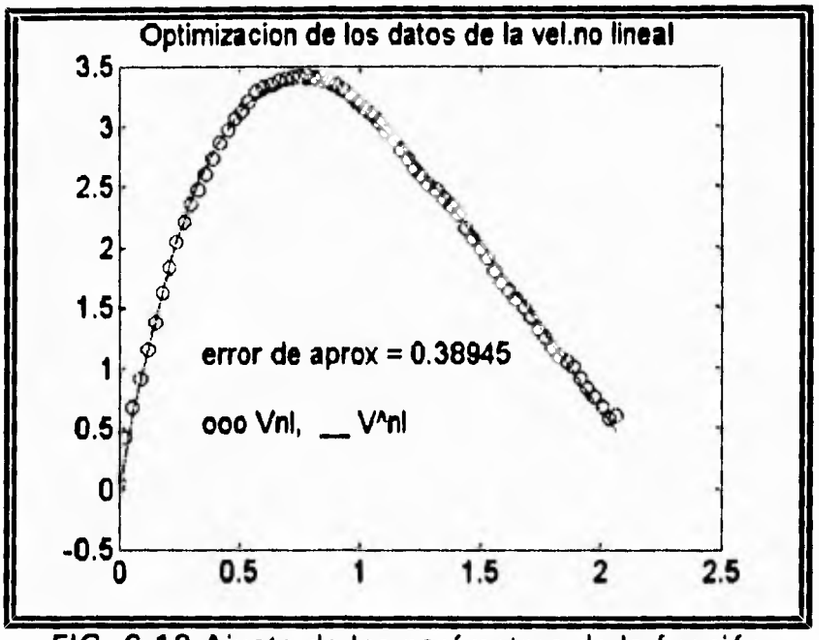


FIG. 6.13 Ajuste de los parámetros de la función

Los valores obtenidos para la función propuesta son

C1 = 4.074208
C2 = 15.717690
C3 = -4.105336
landa1 = 1.005221

FIG 6.14 Valores identificados para la función de la ecuación 4.3

4) Presentar 2 ventanas con los resultados asociados a la identificación de la fuerza de fricción .

La primera ventana (figura. 6.15) esta formada por 4 juegos de gráficas en función del tiempo: en la gráfica superior izquierda, se observa la evolución de la posición (rojo), velocidad (verde) y excitación (azul) del sistema real; en la grafica superior derecha se presenta la evolución de la velocidad del modelo

lineal V_l (verde), del proceso real V_s (rojo) y del error entre ellas V_{nl} (azul). En la parte inferior de la pantalla se gráfica en el lado izquierdo nuevamente la función identificada V_{nl} (azul) y en el lado derecho las fuerzas de excitación del modelo; U_{so} corresponde a la fuerza de fricción (rojo), U_o a la fuerza de excitación del motor (verde) y $U_{so}+U_o$ es la suma de ambas fuerzas (azul).

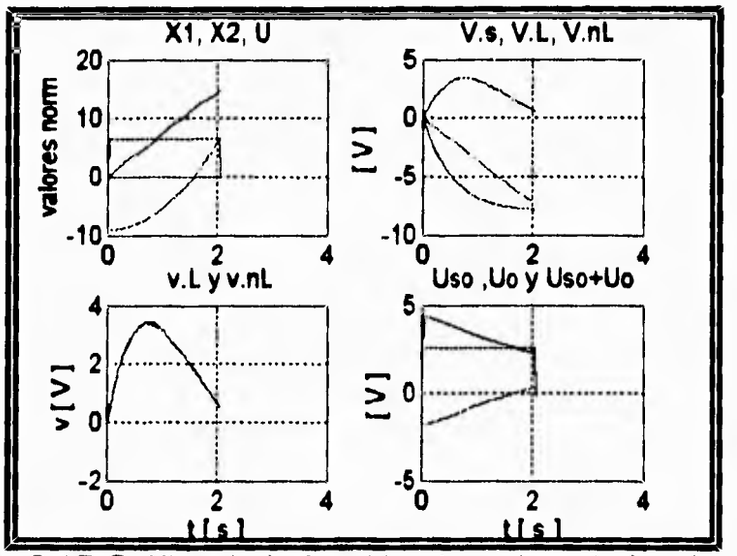


FIG. 6.15 Gráfica de la función ajustada y de los datos.

El segundo conjunto de graficas figura 6.16, permite validar la identificación. En este caso unicamente las tres gráficas de la parte superior izquierda corresponden a evoluciones en función del tiempo.

La gráfica en color rojo representa la excitación para el sistema simulado, y las gráficas sobrepuestas (verde y azul) corresponden a la velocidad real y a la simulada con la no linealidad identificada.

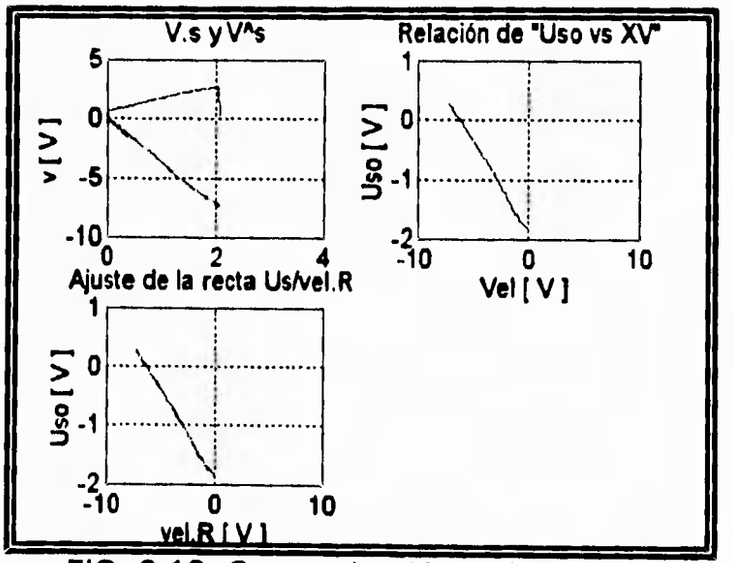


FIG. 6.16 Caracterización de la fricción

La gráfica superior derecha muestra la evolución de la fuerza Uso obtenida a partir de la figura **4.4**, con respecto a los datos reales de la velocidad del carro. La gráfica inferior izquierda corresponde a la evolución de la fuerza Uso con respecto a la velocidad obtenida con el modelo no lineal. De ambas pantallas se ve que practicamente no hay una diferencia notoria entre el comportamiento del modelo identificado y el proceso real.

A continuación el programa ajusta con base en la fuerza de fricción Uso y la velocidad real del carro, mostrada en el primer cuadrante de la figura **6.16**, la mejor recta para este conjunto de datos. Durante el ajuste se activa la ventana de la figura **6.17**. Una vez que se terminó el cálculo se presentan los parámetros de la recta por medio de la ventana de la figura **6.18**. Con esta pantalla se termina la identificación y el analisis de la respuesta escalón del sistema. Por tanto el programa regresa a la pantalla de la figura **6.11**. Al dar la opcion **0** estando en esta última pantalla, el programa regresa al menu principal (figura **6.1**).

FIG. 6.17 Aproximación de la curva a una recta

Valores de la recta ajustada :

Uso(v) = -0.299212*v + -1.877563*sgn(v) V

Dar el nombre del archivo para cargar los datos. "0" para salir

0[↵]

FIG. 6.18 Parámetros de la recta ajustada

6.5.2 Compensación de la fricción seca

La segunda parte del experimento tiene como objeto estudiar los efectos de diferentes modelos de la fricción seca en el control del carro cuando se implanta un esquema de linealización por cancelación. Para ello se diseña primeramente, fuera de línea, el controlador por retroalimentación de estados para el modelo lineal usando la opción **Simulación del sistema de control para el carro**. Después se propone analizar el comportamiento del carro en tiempo real con la ley de control, modificando unicamente el modelo no lineal empleado para su cancelación por medio de la opción **Regulación de los sistemas en tiempo real.** Para estudiar las bondades de la compensación con estimador dinámico se requiere diseñar el estimador, antes de activar la opción en tiempo real.

El experimento se inició seleccionando la opción **Simulación del sistema de control para el carro**. Como resultado de esta acción se activa la ventana de la figura 6.19 y se pregunta si se desea diseñar el control por asignación de polos o por medio del DLQR. Para el ejemplo aqui tratado, se propuso una asignación de polos repetidos en -4 para el modelo contínuo. La pantalla de la figura 6.20 muestra los polos asignados tanto para el caso contínuo como el discreto, y la ganancia K para el modelo discreto.

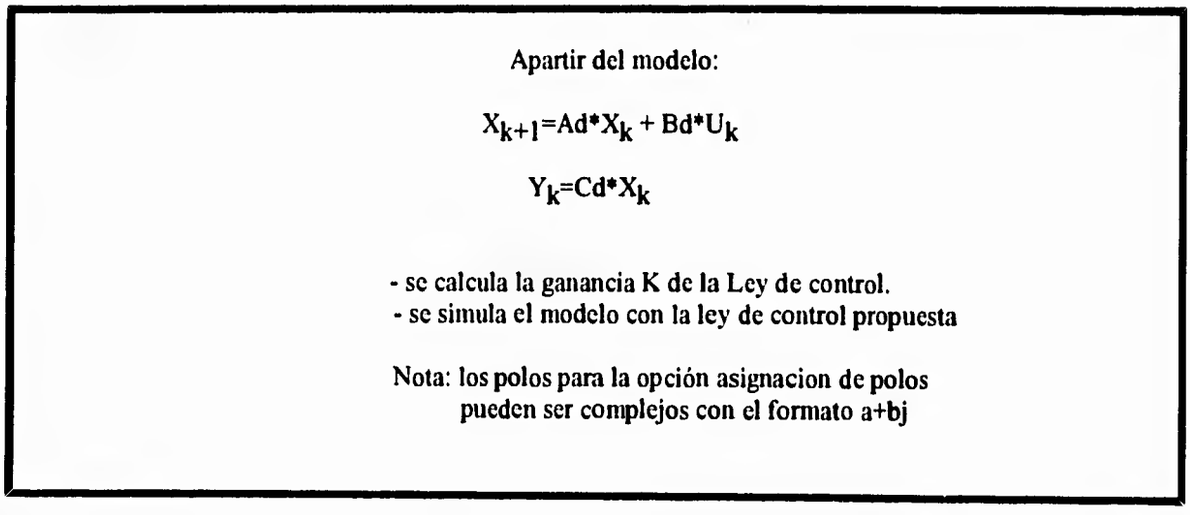


FIG. 6.19 Antecedentes para el cálculo del control

Como segundo paso para el estimador de la fricción, el programa automáticamente solicita después de haber diseñado el control, el tipo de algoritmo que se desea emplear para el estimador (asignación de polos o DLQR). En el caso aqui tratado se eligió la opción de asignación de polos con valores en -9 y -10 para el caso contínuo.

Diseño del controlador

Periodo de muestreo 0.03 s

Asignación del primer polo
Psk = **-4** ↵
Pzk(1) = 0.88692

Asignación del segundo polo
Psk = **-4** ↵
Pzk(1) = 0.88692

Ganancia del control
K1 = 1.20803
K2 = -0.902272

FIG 6.20 Polos y Ley de control

A continuación el programa presenta el resultado del observador y pregunta a través de la ventana de la figura 6.21, si se desea simular el sistema de control en lazo cerrado. En caso afirmativo, el programa presenta el comportamiento de la posición, velocidad y acción U_0 del sistema en lazo cerrado, como se muestra en la figura 6.22. Estando en esta ventana se está en posibilidad de salvar el diseño realizado al teclear un 1 seguido de un regreso de carro. Así se almacena el modelo discreto del carro, los valores de los polos deseados en continuo y las ganancias del controlador y del estimador en los archivos con nombre xx_car.res, y además el programa de diseño y simulación regresa al menú principal de la figura 6.1

Diseño del observador

Asignación del primer polo

PsL = **-9**↵
PzL(1) = 0.763379

Asignación del segundo polo
PsL = **-10**↵
PzL(2) = 0.740818

Ganancias del observador
L1 = 0.439334
L2 = -0.338984

Desea continuar con la simulación del sistema ([1]=si , [ENTER]=no)
continua = ? **1**↵

FIG 6.21 Cálculo del estimador predictor

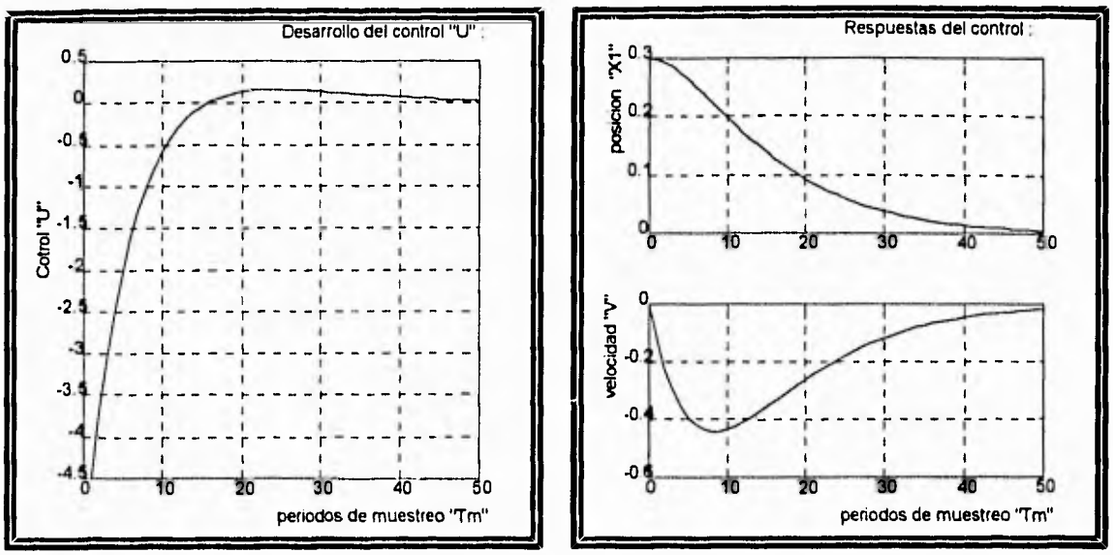


FIG 6.22 Simulación del control del carro

Una vez diseñado el controlador y el estimador, se seleccionó la opción **Control en tiempo real** del menú principal, figura 6.1 y como consecuencia el programa mostró la pantalla de la figura 6.2. En estas condiciones eligiendo la opción **control del carro** se inició el dialogo, a través de la figura 6.23, para seleccionar el modelo de la fricción; para el experimento aqui presentado se seleccionó la opción 1 primeramente (sin compensación).

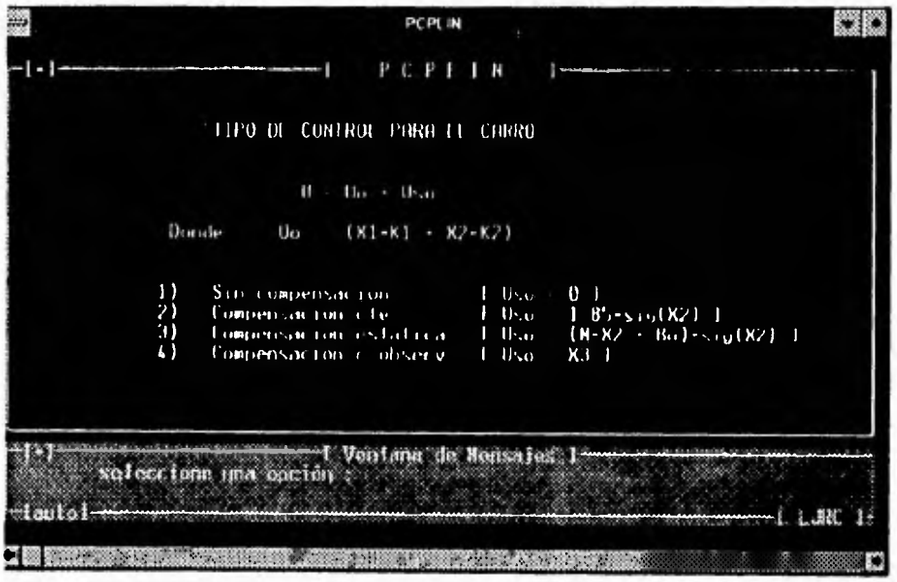


FIG. 6.23 Selección del tipo de compensación para la fricción

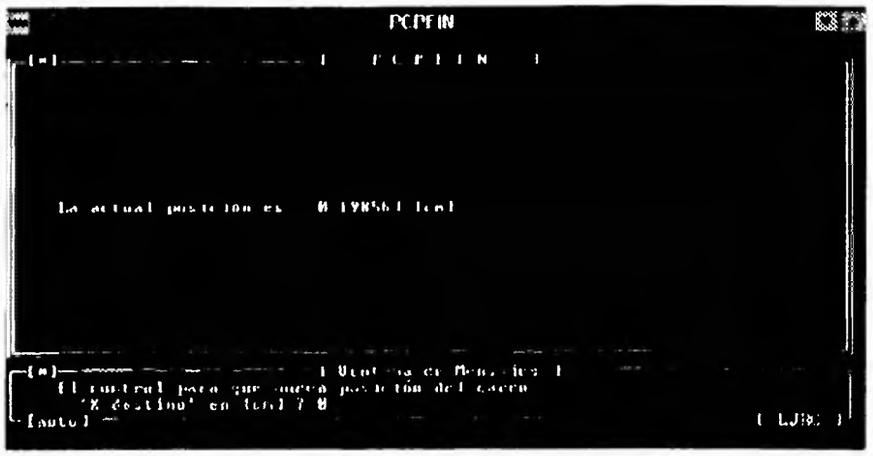


FIG. 6.24 Selección del referencia inicial

A continuación la pantalla, figura 6.24, mostró la posición actual del carro que es de 0.91 cm, estando el usuario en posibilidad de ordenar la posición deseada y sí se graba el experimento; el caso ejemplificado corresponde a 0 cm, sin grabacion del experimento. Al terminar este diálogo se inició el control en tiempo real, persiviendolo por medio de una señal acústica y la activación de la pantalla de resultados en modo texto de la figura 6.25. La opción Alm permite iniciar la grabación del experimento en línea con una ventana de 200 muestras, estando en modo texto.

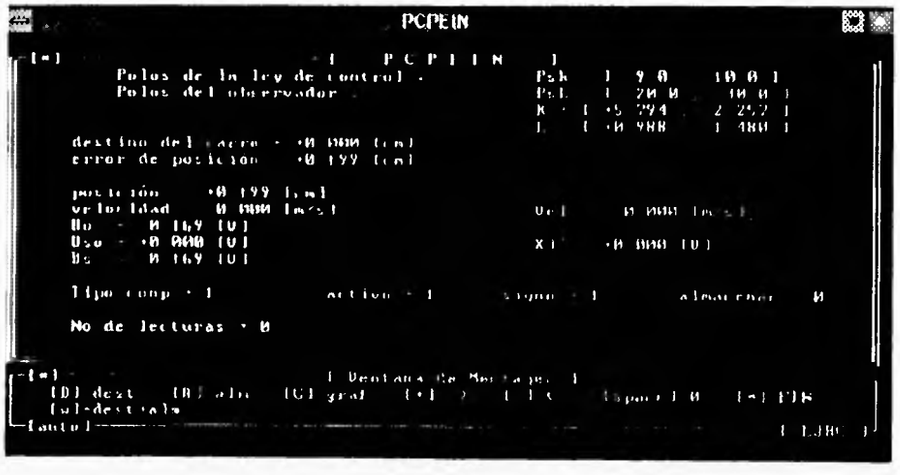


FIG. 6.25 Variables del sistema de control en tiempo real

Estando el carro en la posición de 35 cm al teclear una **G** se activó el modo gráfico y se inicio el barrido de la posición del carro, figura 6.26. Aproximadamente dos segundos después de iniciado el barrido se tecleó la barra espaciadora por lo que el carro regreso a la posición cero con la dinámica mostrada en la figura 6.26.

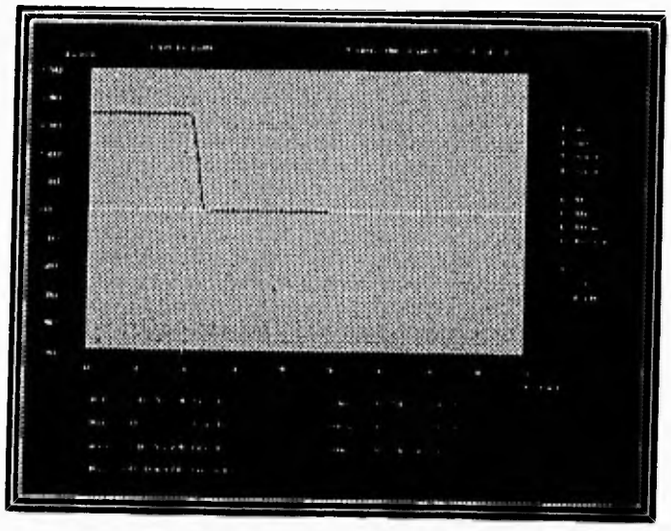


FIG. 6.26 Posición sin compensación de la fricción

Se hace notar que el usuario tiene la posibilidad en todo momento de congelar la imagen por medio de la tecla H. Y al teclear P, V el programa continúa con el barrido de la posición o velocidad respectivamente. Estando en la pantalla de la figura 6.26 se tecleó tres veces el * y por tanto el programa regresó a la pantalla de la figura 6.2.

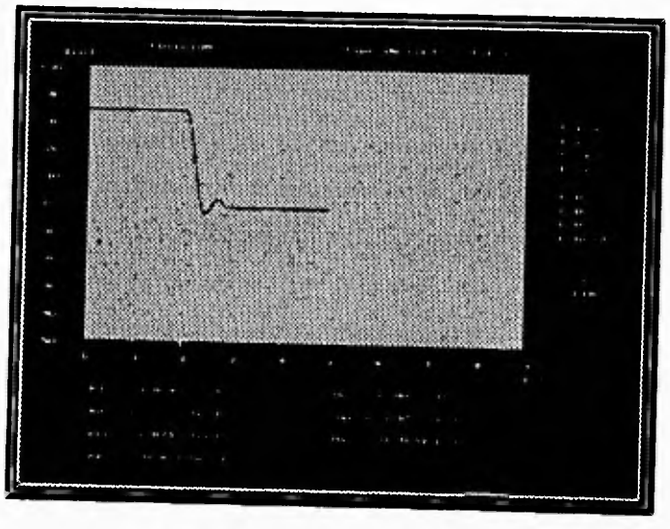


FIG. 6.27 Grafica de la posición con compensación constante

Las figuras 6.27, 6.28 y 6.29 corresponden a la posición del carro al repetir el experimento con las tres opciones diferentes restantes de compensación. Se puede observar en el ángulo superior derecho de las pantallas la indicación del tipo de modelo empleado para la fricción.

Comparando el error con los cuatro modelos considerados se concluye que el estimador dinámico logra el valór más pequeño en estado permanente siendo este de aproximadamente 0.13 cm.

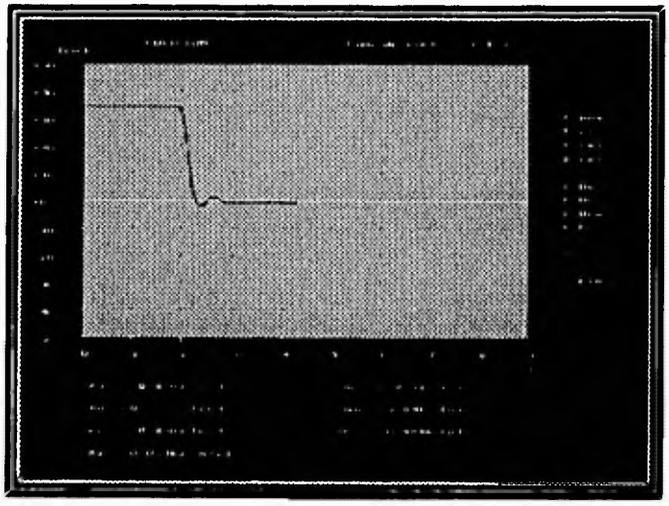


FIG. 6.28 Grafica de la posición con compensación estática

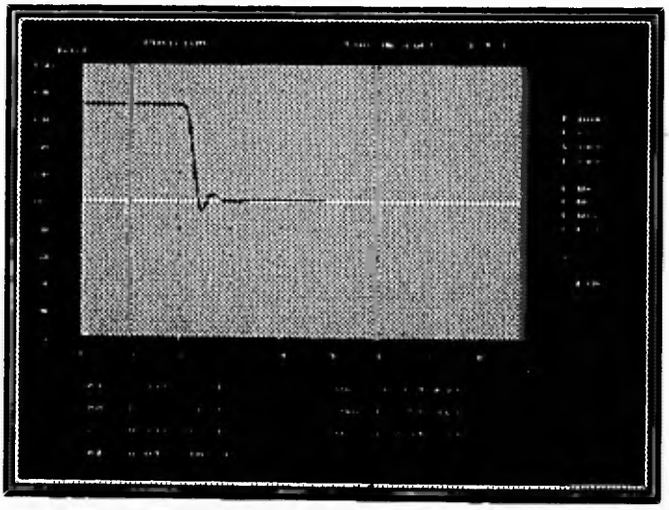


FIG. 6.29 Grafica de la posición con compensación por estimador dinámico

6.5.3 Control del péndulo invertido

La última parte del experimento tiene como objetivo analizar los efectos de la ley de control por retroalimentación de estados cuando se requiere implantar un observador para estimar algunas de las variables de estado. En particular la práctica permite estudiar el efecto de los observadores de orden completo y reducido.

De manera similar a la parte del experimento del carro, inciso 6.5.2, es necesario diseñar primero, fuera de línea, usando la opción ***Simulación del sistema de control para el péndulo invertido***, el controlador y el estimador de orden reducido o completo, esta última en función del interés del usuario. Posteriormente con base en el diseño realizado se implanta el control del péndulo.

El experimento se inicia seleccionando la opción ***Simulación del sistema de control para el péndulo***. Como resultado de esta acción se activa la ventana de la figura 6.30, en donde se pregunta al usuario si desea modificar los polos asignados al sistema en lazo cerrado, por omisión, que corresponden a -4.3 repetido 4 veces para el modelo contínuo.

C_PENDUL.M

Cálculo de la ley de control y estimador
con base en el modelo normalizado y discretizado
obtenido a partir de las matrices 'A' y 'B' con un periodo
de muestreo de 0.03 s

obteniendo así las ecuaciones

Xk+1=Ad*Xk + Bd*Uk
Yk=Cd*Xk

¿ Desea Modificar los polos del sistema de control por omisión ?
([1]=si , [ENTER]=no)

Pks = (-4.3 , -4.3 , -4.3 , -4.3)

FIG. 6.30 Mensajes para el lazo de control

En el caso aquí presentado se aceptaron los valores de los polos por omisión; sin embargo es posible modificar los polos vía la formula de Ackerman opción dos o diseñar la ley de control vía un regulador cuadrático lineal discreto (DLQR), opción uno.

Ley de control

K1 = -1.15669
K2 = 1.54334
K3 = 2.46129
K4 = 0.33832

Especifique el tipo de observador (2 ó 5 estados) [2/5]
= ? **2** ↵

FIG. 6.31 Ganancia de la ley de control y selección del tipo de observador

Una vez aceptados los polos por omisión (o definir la ley de control a verificar) el programa muestra la ganancia de retroalimentación K y pregunta si se desea diseñar un observador de dos estados o uno de cinco estados a través de la pantalla de la figura 6.31. En la pantalla mostrada aquí se seleccionó la opción dos de estimador de orden reducido. Como resultado de esta orden el programa pregunto que tipo de método de diseño se desea emplear para el estimador. De nueva cuenta el usuario tiene la posibilidad de seleccionar entre una asignación de polos o un filtro de Kalman. Bajo la opción de asignación de dinámica se propusieron valores de -99 y -100 para los polos del modelo contínuo del observador. La pantalla de la figura 6.32 muestra el diálogo entre la computadora y el diseñador. La última pregunta permite que el usuario simule el sistema diseñado antes de implementarlo en tiempo real.

Especifique el método que se desea emplear: (Kalman(1) o Ackerman(2)) ?
= ?2

Asigne los polos del observador

dar el valor del polo PsL(1)
PsL = **-99** ↵
PzL(1) = 0.0513033

dar es el valor del polo PsL(2)
PsL = **-100**↵
PzL(2) = 0.0497871

Ganancia del observador

L =

0 3.1764e+001 -7.1451e+000
0 -5.8336e-003 -5.3138e+000

Desea continuar con la simulación ([1]=si [ENTER]=No)
continua = ? **1** ↵

FIG. 6.32 Cálculo del estimador de orden reducido, mediante asignación de polos

La figura 6.33 muestra el resultado de la simulación para el observador predictivo y corriente. En particular se presentan los errores de simulación tanto para la velocidad angular como la fuerza de fricción con ambos observadores.

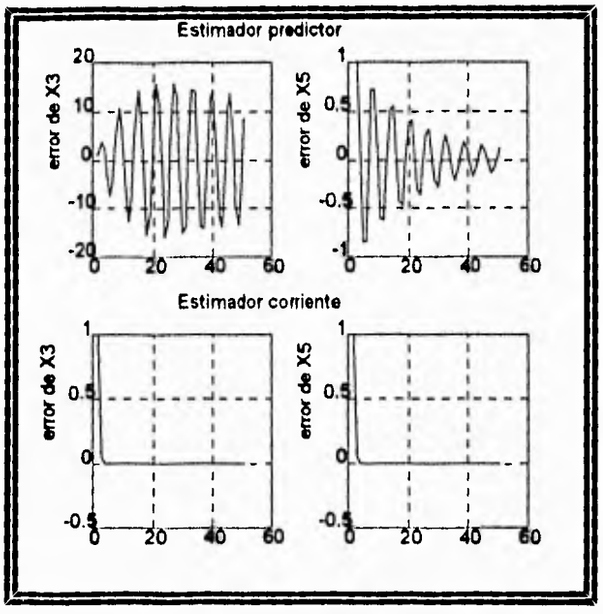


FIG. 6.33 Error de la velocidad angular y de la fuerza de control, empleando un estimador predictivo y uno corriente

Finalmente al oprimir cualquier tecla se inicia el diálogo de la figura 6.34, para especificar si se salva el diseño realizado, teniendo cuidado de guardarlo en el subdirectorio en donde se encuentra el programa de control en línea.

Desea salvar el diseño para usarse en tiempo real ([1]=si ó [ENTER]=no)
salvo = ? **1** ↵
Desea continuar con el diseño [1]=si o [ENTER]=no ?
continua = ? ↵

FIG. 6.34 Diálogo para salvar el diseño

A continuación de manera similar al caso del control del carro se seleccionó la opción **Control en tiempo real** del menú de la figura 6.1, y como consecuencia el programa inicia el diálogo a través de la figura 6.2. Bajo estas condiciones el seleccionar la opción **Control del péndulo invertido**, se activa la ventana de la figura 6.35 en donde se establece el tipo de estimador a probar.

En el ejemplo aquí presentado, se opto por el estimador reducido y se recuerda que sí se desea, se puede salvar el arranque del experimento. Las opciones aquí presentadas corresponden a 0 cm sin guardar datos. En este momento el péndulo debe estar en posición vertical.

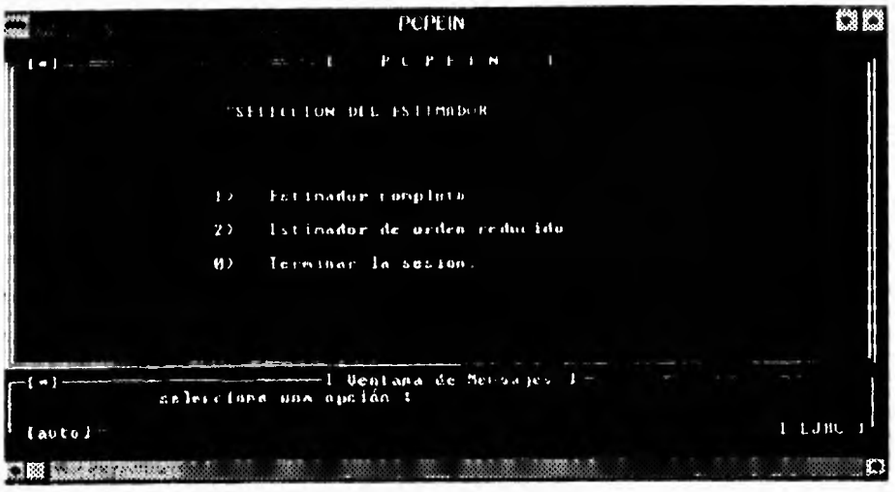


FIG. 6.35 Selección de tipo de estimador

La figura 6.36 presenta la pantalla de inicialización del control en línea, en donde se pregunta por la posición deseada del carro.

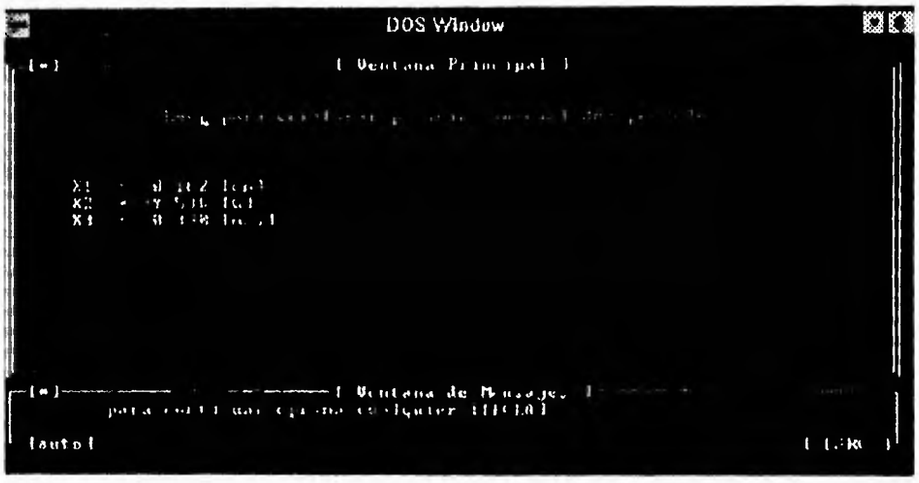


FIG. 6.36 Condiciones iniciales de las variables

Al terminar este diálogo se inició el control en tiempo real persiviendolo por medio de una señal acústica y la activación de la pantalla de resultados en modo texto de la figura 6.37.

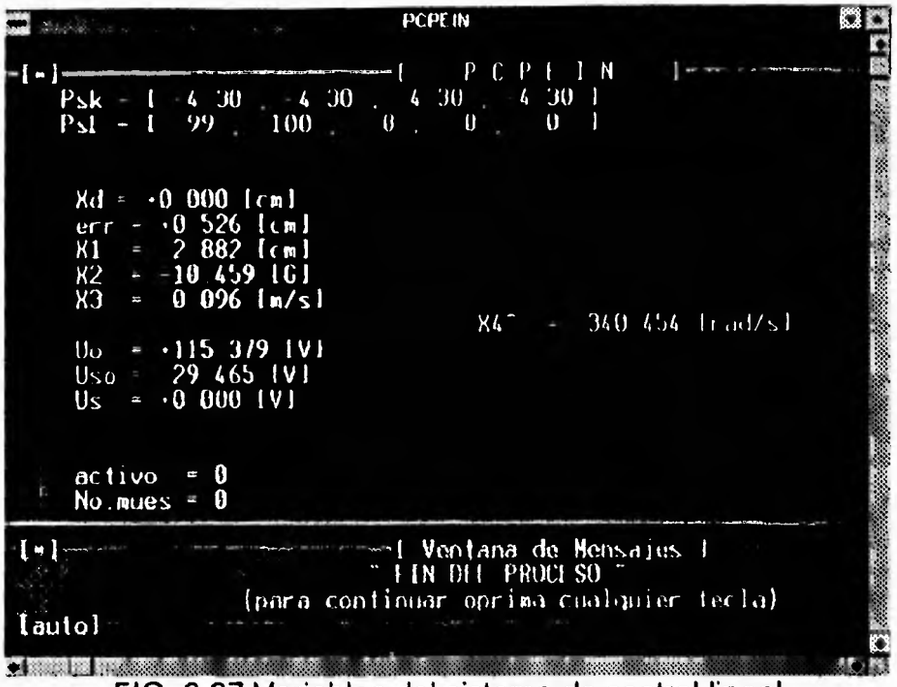


FIG. 6.37 Variables del sistema de control lineal

El experimento realizado de manera análoga al del control del carro consistió en trasladar la posición del carro de 35 cm o 0 cm con las opciones que brinda el modo gráfico del paquete.

La única diferencia estriba en el numero de variables que se muestran ya que el modelo del péndulo tiene 4 variables de estado. y dependiendo del tipo del observador implementado se tienen 2 estados observados o 5.

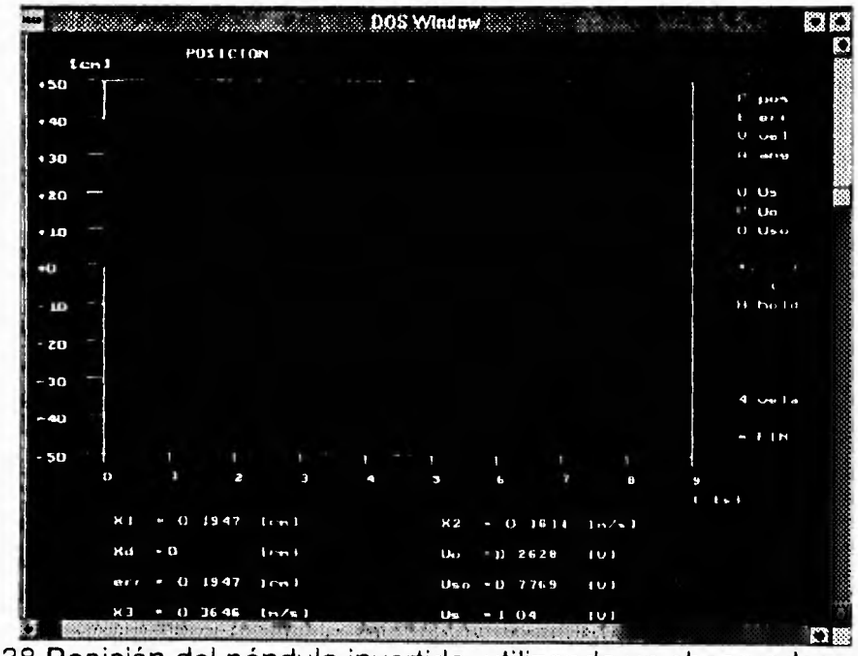


FIG. 6.38 Posición del péndulo invertido, utilizando un observador reducido.

En las figuras 6.38 y 6.39, se presentan las gráficas de los experimentos realizados con el observador de orden reducido y de orden completo, respectivamente.

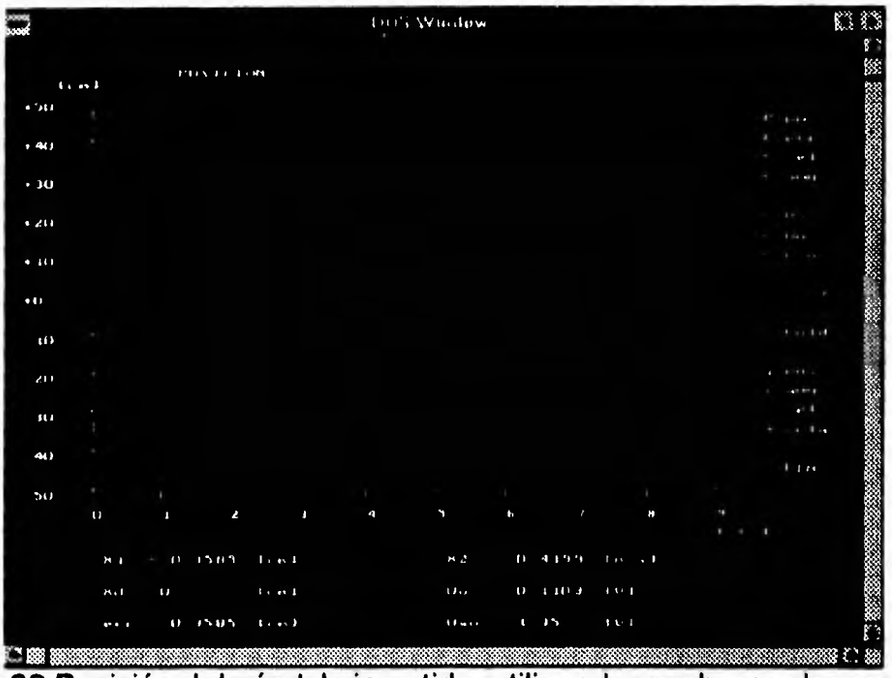


FIG. 6.39 Posición del péndulo invertido, utilizando un observador completo

Para salir del modo gráfico, similarmente al experimento del carro, estando en modo texto el teclear * desactiva el control en línea.

6.5.4 Análisis del experimento

Una vez hechos los experimentos en línea, con objeto de analizar el desempeño del control implantado para el péndulo o el carro se sugiere obtener por medio de la opción ***Análisis de respuestas de control*** del menú de la figura 6.1 o **nv2.m** en MATLAB, la función que muestra las gráficas del error y de las variables del sistema. Los resultados para el caso aquí tratado corresponden a las gráficas de la figura 5.9, por lo que se considero innecesario presentarlas nuevamente.

La secuencia en que se presentan las gráficas es; primeramente el error de posición, a continuación, después al ir desactivando el modo "pause" de MATLAB se muestra la acción de control o voltaje aplicado al motor y las normas cuadráticas del error de posición Je y del voltaje del control Ju.

7 CONCLUSIONES Y COMENTARIOS

El trabajo desarrollado consistió en la elaboración de un paquete formado por rutinas de análisis, diseño y regulación en tiempo real, implantado en una computadora personal que permite que estudiantes de nivel de licenciatura adquieran destreza en las técnicas más ampliamente usadas para el diseño de controladores con retroalimentación de estados y de cancelación de nolinealidades estáticas.

El paquete denotado **PCPEIN** se diseñó tomando como proceso especial a controlar el péndulo invertido de Laboratorio de Automatización del Instituto de Ingeniería y las tarjetas de convertidores **A/D** y **D/A** modeloe **DAC6214** de Amira. En particular, como se mostró en el capítulo **6** del ejemplo, el emular un osciloscopio en el monitor de la PC para mostrar las variables del proceso en línea permite visualizar e interpretar claramente los fenómenos de atenuación de perturbaciones y desempeño robusto producto de la retroalimentación de sistemas discretos.

La gran ventaja del paquete en tiempo real es su fácil manejo, ya que permite diseñar y analizar el esquema de control por medio del paquete matemático **MATLAB** con **SIMULINK** operando desde el sistema Windows para despúes automáticamente de manera amigable implantar y verificar los controladores diseñados al conectar el péndulo invertido a una computadora personal a través de la tarjeta de convertidores A/D y D/A y del actuador del proceso. De esta manera se logra aumentar la eficiencia en la enseñanza del diseño de controladores, ya que el usuario se concentra en los problemas del control digital eliminando el tener que conocer el hardware específico y programar los manejadores de los periféricos de la pc.

Como parte complementaria al paquete se diseñó una práctica que ayuda a usuarios novatos a explotar las bondades del paquete. Para diseñar la práctica se tomó en cuenta que el primer paso en toda tarea de control es el modelado del proceso. Como resultado del análisis de la validación del modelo del péndulo estudiado se llegó a la conclusión que la nolinealidad estática producto de la fricción seca no puede despreciarse en el diseño y juega un papel importante para lograr desempeños satisfactorios. Específicamente la no compensación de la fricción genera un error en estado permanente para un entrada escalón, que no puede explicarse para un servomecanismo lineal con un integrador, como es el caso del péndulo. Por lo tanto se propusó emplear una cancelación directa de la fricción seca. La cancelación de la fricción puede implantarse con tres tipos de aproximaciones estáticas y con un estimador dinámico.

Por lo que respecta a la parte del controlador lineal, éste se puede diseñar por medio de rutinas amigables desarrolladas en **MATLAB** usando el método de asignación de polos o del regulador óptimo cuadrático, ambos con observadores reducidos o completos del tipo corriente. Esta parte del paquete está dotada de opciones para simular el controlador y estimador antes de implantarlo en línea.

La parte de la rutina en línea que cálcula la acción, presenta los resultados, y maneja los equipos periféricos de manera amigable via menús fueron escritas en C de Microsoft versión 1.2. Con ello se evita que el usuario tenga que programar las interfaces entre el péndulo y la computadora personal.

En general, se puede decir que el paquete **PCPEIN** unido al problema abierto de control del péndulo invertido es una herramienta de trabajo para integrar aspectos de teoría y aplicación del control digital.

Se hace notar que el paquete **PCPEIN** deja una puerta abierta para desarrollar e implantar nuevos algoritmos de control, ya que unicamente se debe modificar la parte del programa que calcula la acción de control, dejando sin modificación el resto de la infraestructura básica del paquete. De hecho actualmente se utiliza el paquete para desarrollar trabajos de investigación en el campo de detección de fallas de sistemas dinámicos.

BIBLIOGRAFÍA

Referencía [1] : Franklin F. Gene, J. David Powell, Abbas Emami-Naeini, **Control de sistemas dinámicos con retroalimentción**, Addison Wesley, Iberoamericana, 1991.

Referencía [2] : *ASTRÖM*, J. Karl and Wittenmark Bjorn, Computer controlled systems, Theory and design, Englewood cliffs, Prentice-Hall, 1984.

Referencía [3] : OGATA, Katsuhiko, Ingeniería de control moderna, Prentice-Hall Hispanoamerica, 1982.

Referencía [4] : Amira Gmbh 1992 Laboratory Setup Inverted Pendulum (LIP100). Telefono: 0203/37809-50.

Referencía [5] : Manual de MATLAB, The mathworks, inc. Matlab 4.0 Simulink 1.2c, 1993.

Referencía [6] : Auslander, David M. Takahashi, Yasundo Rabins, Michael J. Introducción a sistemas y control. MacGraw-Hill.

Referencía [7] : Ackerman, Jurgen. Sampled Data Control Systems. Springer Verlag Berlín Heidelberg 1985.

Referencía [8] : Lewis, Frank L. Applied Optimal Control &Estimation: Digital design & Implementation/Frank L. Lewis Prentice-Hall 1992

APÉNDICE A

CARACTERÍSTICAS DEL PÉNDULO

A nivel de referencia, a continuación se presentan los datos proporcionados por el distribuidor del sistema (manual Laboratory Setup Inverted Pendulum). La **posición del carro**, se mide por medio de un potenciometro circular, el cual esta fijado al manejador en el motor. Para la **velocidad del carro**, se emplea un tacometro, el cual está fijo al motor de corriente directa. El **ángulo de la varilla o péndulo**, se sensa a través de un potenciometro de capa, fijado al pivote del péndulo.

Los parámetros del sistema, se encuentran en la siguiente tabla, en la cual, el único parámetro que puede variar es el del coeficiente de fricción estático.

Constante	Valor numérico	Unidades
K_F	2.6	N/V
n_{11}	14.9	V/m
n_{22}	-52.27	V/rad
n_{33}	-7.64	Vs/m
n_{44}	-52.27	Vs/rad
M_0	3.2	Kg
M_1	0.329	Kg
M	3.529	Kg
l_s	0.44	m
Θ	0.072	Kgm^2
N	0.1446	Kgm
N_{01}^2	0.23315	Kg^2m^2
N^2/N_{01}^2	0.0897	
F_r	6.2 + Δ	Kg/s
	0.0009	Kgm^2/s

Tabla. A.1 Cantidades físicas

Con base en la tabla A.1 y un periodo de muestreo de 0.03 segundos, los valores del modelo del sistema son las matrices del sistema continuo A_c, normalizado A_n y discreto A_d para el carro respectivamente

$$A_c = \begin{bmatrix} 0 & 1 \\ 0 & -1.9375 \end{bmatrix} \quad \text{(A.1)}$$

$$A_n = \begin{bmatrix} 0 & -1.9503 \\ 0 & -1.9375 \end{bmatrix} \quad \text{(A.2)}$$

$$A_d = \begin{bmatrix} 1 & -0.0568 \\ 0 & 0.9435 \end{bmatrix} \quad \text{(A.3)}$$

las matrices de entrada continua B_c, normalizada B_n y discreta B_d, para el carro respectivamente

$$B_c = \begin{bmatrix} 0 \\ 0.3125 \end{bmatrix} \quad \text{(A.4)}$$

$$B_n = \begin{bmatrix} 0 \\ -6.2075 \end{bmatrix} \quad \text{(A.5)}$$

$$B_d = \begin{bmatrix} 0.0053 \\ -0.1809 \end{bmatrix} \quad \text{(A.6)}$$

Y las matrices del sistema normalizado y discreto para el observador de orden reducido formado por los estados de la velocidad y la fricción, respectivamente

$$A_{n2} = \begin{bmatrix} -1.9375 & -6.2075 \\ 0 & 0 \end{bmatrix} \quad \text{(A.7)}$$

$$A_{d2} = \begin{bmatrix} 0.94353 & -0.18092 \\ 0 & 1 \end{bmatrix} \quad \text{(A.8)}$$

$$B_{n2} = \begin{bmatrix} -6.2075 \\ 0 \end{bmatrix} \quad \text{(A.9)}$$

$$B_{d2} = \begin{bmatrix} -0.18092 \\ 0 \end{bmatrix} \quad \text{(A.10)}$$

Usando los datos de la tabla (A.1), se obtienen los siguientes valores numéricos de las matrices del sistema para el pendulo A_A.

$$A_A = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & -0.88 & -1.915 & 0.0056 \\ 0 & 21.473 & 3.85 & -0.136 \end{bmatrix} \quad \text{(A.11)}$$

y para la matriz de entrada

$$B_A = \begin{bmatrix} 0 \\ 0 \\ 0.30882 \\ -0.62032 \end{bmatrix} \quad \text{(A.12)}$$

Ademas, los valores normalizados de ambas matrices son

$$A_n = \begin{bmatrix} 0 & 0 & -1.95 & 0 \\ 0 & 0 & 0 & 1.0 \\ 0 & -0.12864 & -1.915 & 0.00082 \\ 0 & 21.47 & 26.31 & -0.1362 \end{bmatrix} \quad \text{(A.13)}$$

$$B_n = \begin{bmatrix} 0 \\ 0 \\ -6.1343 \\ -84.303 \end{bmatrix} \quad \text{(A.14)}$$

Similarmente al carro, para un periodo de muestreo de 0.03 [s], las matrices matrices del modelo .discreto son

$$A_d = \begin{bmatrix} 1 & 1.108e-4 & -5.68e-2 & 4.1e-7 \\ 0 & 1.01 & 1.16e-2 & 3.0035e-2 \\ 0 & -3.755e-3 & 0.944 & -3.299 \\ 0 & 0.6435 & 0.768 & 1.006 \end{bmatrix} \quad \text{(A.15)}$$

$$B_d = \begin{bmatrix} 0.0053 \\ 0.0372 \\ -0.1789 \\ 2.461 \end{bmatrix} \quad \text{(A.16)}$$

Finalmente, para el sistema discretizado aumentado de cinco estados, las matrices están dadas por

$$A_{d5} = \begin{bmatrix} 1 & 1.1075E-4 & -5.68e-2 & 4.1e-7 & 5.282E-3 \\ 0 & 1.01 & 1.16e-2 & 3.0035e-2 & 3.722E-2 \\ 0 & -3.755e-3 & 0.944 & -3.299 & -1.788E-1 \\ 0 & 0.6435 & 0.768 & 1.006 & 2.4603 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \qquad \text{(A.17)}$$

$$B_{d5} = \begin{bmatrix} 5.282E-3 \\ 3.722E-2 \\ -1.788E-1 \\ 2.4603 \\ 0 \end{bmatrix} \qquad \text{(A.18)}$$

APÉNDICE B Descripción de la tarjeta interfaz DAC6214 y las utilerías que la manejan

B.1 CARACTERÍSTICAS GENERALES DE LA TARJETA

INFORMACIÓN GENERAL DEL DAC 6214

El DAC 6214 es una tarjeta de uso general para IBM PC/XT/AT y computadoras compatibles. La cual es usada para manejar entradas y salidas analógicas y digitales para diferentes aplicaciones y para el control.

CARACTERÍSTICAS

La tarjeta cuenta con los siguientes dispositivos y con las siguientes características.

* 6 entradas analógicas bipolares.
  - convertidor A/D 7572 con resolución de 12 bits c/u, ± 10 V.

* 2 salidas analógicas bipolares.
  - convertidor D/A 7542 con resolución de 12 bits c/u, ± 10 V.
  - también cuenta con 2 canales internos para observar las salidas analógicas.

* contador HCTL 2000 con resolución de 12 bits.
  - 1 entrada para medir rotación y posición.

* 4 entradas digitales (TTL compatibles).

* 4 salidas digitales (TTL compatibles).

B.2 DESCRIPCIÓN DEL PROGRAMA DE PRUEBAS PANTA2.C

La finalidad de este programa es que el usuario de este sistema pueda verificar el funcionamiento de la tarjeta interfaz DAC6214.

El programa PANTA2.C cuenta con dos menúes, uno para entradas y salidas analógicas y otro para entradas y salidas digitales; en general se pretende dar con el un ejemplo de las aplicaciones de cada servicio de la tarjeta. Dentro del menú analógico se incluye un inciso en el cual se pueden leer todos los canales de entrada, tanto analógicos como digitales, además de contar con un ejemplo en el que se utilizan interrupciones del temporizador de la computadora.

El programa está dividido en 3 subprogramas generales y a su vez cada uno contiene un conjunto de rutinas, las cuales se describen a continuación.

Las librerías o subprogramas que utiliza el programa PANTA2.C, son las siguientes.

```
#include "lee_pen2.h"
#include  "rutpl_da.h"
#include  "rut_ada1.h"
```

El conjunto de módulos que utiliza el programa, son los siguientes.

.Módulo principal.

.Módulo de lectura de datos digitales.

.Módulo de escritura de datos digitales.

.Módulo de lectura del convertidor A/D.

.Módulo de escritura al convertidor D/A.

.Módulo de lectura datos [D y A].

Módulo Principal:(main();)

Este módulo es el que forma al menú principal y se encarga de llamar a los demás módulos en el momento en que es seleccionado por el usuario.

Lectura de datos digitales:(DI();)

Esta rutina se encarga de leer el estado general de las entradas digitales y muestra en el monitor el valor de la entrada como un número decimal y además el estado de la entrada en cada bit (D0 a D3). También muestra el valor del contador de vueltas, que ésta incluido en la tarjeta, junto con las entradas digitales.

A la salida de esta rutina se pregunta si se desea borrar el valor del contador para poder iniciar una nueva cuenta de vueltas. Este puede contar vueltas positivas o negativas.

Escritura de datos digitales:(DO();)

Esta rutina se encarga de escribir un número binario en el puerto de salida de datos digitales y dado que es un número de 4 bits éste se puede representar como un número en hexadecimal y de esa manera se introduce.

Lectura del convertidor A/D:(AD();)

Esta rutina selecciona uno de seis convertidores A/D, ordena la conversión y lee el valor de su entrada. Dado que la resolución del convertidor es de 12 bits y que su entrada es bipolar, el convertidor es capaz de leer un voltaje mínimo de +/- 4.88 mV y un voltaje máximo de +/- 10 V.

A la salida de esta la rutina se pregunta si se desea leer otro convertidor.

Escritura al convertidor D/A:(DA();)

En esta rutina uno puede seleccionar cualquiera de los dos convertidores D/A. Se puede mandar un nivel de voltaje máximo de +/- 10 V y un voltaje mínimo de +/- 4.88 mV.

Lectura de datos [D y A]:(AD_DI();)

Esta rutina lee los seis convertidores A/D, los cuatro bits digitales y el contador de vueltas. De esta manera es posible mostrar en pantalla todas las entradas analógicas y digitales al mismo tiempo. Esta es una aplicación que emplea casi todas las rutinas desarrolladas.

Para ver otra aplicación de las rutinas, se ha realizado el programa ENT_SAL11.C en el cual se reduce el período de muestreo a un intervalo de 1 a 54 ms. Este programa lee del convertidor A/D_0 y dicha lectura se escribe en el convertidor D/A_1. Las rutinas que se utilizan pertenecen al programa original (PANTA2.C).

B.3 DATOS TÉCNICOS

INTRODUCCIÓN

La tarjeta ésta formada por dos registros principales :

1.- Registro de direccionamiento HWADR con dirección base de la tarjeta 300 Hex.

2.- Registro de datos DATR cuya dirección para cargar y leer los datos direccionados previamente en HWADR es 303 Hex.

Con base en estos dos registros se direccionan los puertos de los convertidores A/D y D/A, contador de revoluciones, 4 puertos digitales de entrada y salida. Para después leer o escribir algun dato en el registro de datos DATR.

Como se describió en las características de la tarjeta, la resolución de los convertidores A/D y D/A es de 12 bits. Para lo cual se definen los siguientes conceptos.

Si una palabra binaria está compuesta de dos BYTES o sea 16 bits, se define LOW-BYTE a los bits D0 al D7 y HIGH-BYTE a los bits D8 a D15. Además cada BYTE es dividido en dos niveles : LOW-NIBBLE: bits (D0 a D3) o (D8 a D11); y HIGH-NIBBLE: bits (D4 a D7) o (D12 a D15).

En nuestro caso se usan los bits D0 a D12. Por tanto se tienen palabras binarias de tres niveles y las lecturas o escrituras de éstas se realizan en dos pasos.

DIRECCIONES DE PUERTOS PARA LECTURA Y ESCRITURA (R/W).

PARA EL CONVERTIDOR A/D (ADC)

0x00 :Para leer el LOW-BYTE. bits (D0 a D7).

0x01 :Para leer el HIGH-BYTE.bits (D8 a D12).

0x60 :Para leer el bit D7. Indicación de fin de conversión.

0xA0 :Para selección del número del convertidor A/D.
Se usan los bits D4 a D6, direcciones 0 a 5.
Este es un puerto digital de salida de datos.

Nota: 0x .. indica qu e es un número hexadecimal.

PARA EL CONVERTIDOR D/A (DAC)

Para el DAC_0 se usan las direcciones 0x20 a 0x23 y para el DAC_1 se usan las direcciones 0x40 a 0x43. Las descripciones de uno corresponden con la misma secuencia que la del otro.

0X20 :Para escribir el LOW-BYTE LOW-NIBBLE.

0x21 :Para escribir el LOW-BYTE HIGH-NIBBLE.

0x22 :Para escribir el HIGH-BYTE LOW-NIBBLE.

0x23 :Para escribir el bit de inicio de conversión.

0xA0 :Para selección del número del convertidor D/A.
Se usan los bits D4 a D6, direcciones 6 y 7.
Este es un puerto digital de salida de datos.

PARA ENTRADAS DIGITALES:

0x60 :Para lectura de bits (D0 a D3).

PARA SALIDAS DIGITALES:

0xA0 :Para escritura de bits (D0 a D3).

PARA EL CONTADOR DE REVOLUCIONES.:

0x80 :Para inicializar el contador.

0x88 :Para leer el HIGH-BYTE.

0x89 :Para leer el LOW-BYTE.

COMO LEER DEL ADC

PASOS:

1.- :Para seleccionar el número del ADC escribir en HWADR 0xA0.

2.- Escribir en DATR el número del ADC seleccionado, utilizando los bits D4 a D6. Los canales posibles son 0,1,2,3,4 y 5.

3.- Realizar una lectura del LOW-BYTE para empezar la conversión del A/D. Esto se logra escribiendo en HWADR 0x00 y leyendo el dato en DATR.

4.- Monitorear el bit 7 de el puerto de entradas digitales escribiendo en HWADR 0x60 y leyendo en DATR el bit 7, el cual indica el fin de la conversión. Debe de ser leído constantemente para detectar su cambio de 0 a 1 (bit-7=>BUSY), con el bit 7 igual a 1 se indica que la conversión ha terminado.

5.- Leer los bits D0 a D12 en las direcciones 0x00 y 0x01.

6.- Realizar la conversión de nibble's a palabras completas de 12 bits realizando corrimientos y la operación OR.

7.- Realizar la conversión de un número de 12 bits a un nivel de voltaje de +/- 10 V. Recordar que el 0 está a la mitad del número de 12 bits y que las lecturas más bajas (00...00) corresponden a los voltajes negativos y que las mas altas (11...11) corresponden a los voltajes positivos (2^{12} = 4096 y voltaje total =20 V).

COMO ESCRIBIR AL DAC

1.- Convertir el nivel de voltaje correspondiente a un número de 12 bits, recordando las notas del inciso 7 de los pasos para el ADC.

2.- Dividir el número de 12 bits en NIBBLE'S.

3.- Seleccionar el DAC en el cual se desea mandar la información. Escribiendo la dirección 0xA0 en HWADR y en DATR el número del DAC seleccionado. Usando los bits D4=D5=1 y D6=(0/1) para el DAC correspondiente y D0 a D3=0.

4.- Escribir los NIBBLE'S en sus correspondientes direcciones y después mandar el bit de inicio de conversión OE=0 a la dirección 0x23.

COMO LEER DEL PUERTO DIGITAL

1.- Se escribe en HWADR la dirección 0x60 y se lee de DATR los bits D0 a D3.

COMO ESCRIBIR EN EL PUERTO DIGITAL

1.- Se escribe en HWADR la dirección 0xA0 y en DATR el número digital de 4-bits (D0 a D3).

COMO LEER EL CONTADOR DE REVOLUCIONES

1.- Escribir en HWADR 0x88 y leer de DATR el HIGH-BYTE de la misma forma leer el LOW-BYTE de 0x89

Para borrar el contador escribir en HWADR sólo 0x80.

NOTAS Y COMENTARIOS

El lenguaje de programación C requiere del uso de variables declaradas UNSIGNED CHAR e INT.

En la declaración de variables se declararon como variables globales a HWADR y DATR, para ser utilizadas en todas las rutinas.

Para el uso de las rutinas sólo se debe de incluir el nombre del archivo que contiene todas las rutinas y las declaraciones antes mencionadas. Por tal motivo sólo es necesario llamar a las rutinas como si fueran una función dentro del programa a realizar y la declaración de variables globales antes mencionadas se puede omitir, ya que dentro del archivo en el que fueron declaradas las funciones se declararon también las variables globales.

Además cada rutina está compuesta por varias subrutinas, así que no es necesario utilizar la misma estructura de programación, sino que uno puede formar los módulos a su gusto. Por tal razon se presentan a continuación los nombres y descripciones de las rutinas que utiliza cada módulo.

_void DI();

.int lee_DI():

Lectura de un número del puerto digital. Está compuesto por 4 bits.

.int lee_cont():

Lectura de un número del puerto digital. Consta de 12 bits.

.void limpia_cont():

Limpia el valor almacenado por el registro del contador.

_void DO();

.void write_dig():

Escribe en el puerto digital un número de 4 bits.

_void AD();

.unsined char selcan_A():

Pregunta por el número del ADC y devuelve el número correspondiente.

.int lee_AD(unsigned char canal):

Esta rutina recibe el número del ADC y lee de éste el número correspondiente de 12 bits y devuelve el valor leído.

.float dig_volt(int dato_AD):

Esta rutina recibe un número de 12 bits y realiza la conversión correspondiente al valor que esté entre +/-10 V. Devolviendo un número fraccional.

_void DA();

.float pide_volt():

Esta rutina pide un valor de voltaje que esté entre +/- 10 V. Devolviendo este valor en un número fraccional.

.int AN_dig(float voltaje):

Esta rutina convierte el valor del voltaje dado, (entre +/- 10 volts) a un número de 12-bits. Devolviendo su valor en un número entero.

.unsigned char seel_DAC():

Esta rutina solicita el número del DAC de salida, devolviéndo dicho valor (0/1).

.void escribe_DA(int valor_DA, unsigned char sal_DAC):

Esta rutina escribe un número de 12 bits en el DAC que se ha seleccionado (0/1).

_void AD_DI();

Esta rutina hace uso de las rutinas que leen de los ADC y de los puertos digitales.

B.4 INTERRUPCIONES

INTRODUCCIÓN

Para el uso de la tarjeta de conversion analogico digital, fue necesario programar el temporizador (timer) interno de la PC, que activa la interrupción 8. Se programa a éste para que genere una interrupción (por hardware) cada tiempo determinado.

Para lo anterior se desarrollaron rutinas que programan el temporizador, el cual al llegar a la cuenta con la cual fue programado provoca una interrupción, la que es atendida por una rutina de atención a la interrupción. Por tanto, cada que el timer termine su cuenta, se producirá una interrupción y se activará una rutina de atención a la interrupción.

Dado que esto es ejecutado automáticamente o internamente, se puede realizar cualquier otra tarea mientras esté puesta la interrupción, y para que el temporizador cuente de 1 ms a 54 ms, hay que cargar en los registros del temporizador el numero deseado para la interrupción.

Las rutinas que se encargan de realizar la interrupción son las siguientes:

* pide_time(); Pide el periodo de interrupción. para programar el temporizador de la PC.

* rut_INT(); Rutina de atención a la interrupción. En ésta se pone el procedimiento que uno desea hacer en cada interrupción.

* prog_timer();	Programa el temporizador de la PC.
* start_INT();	Inicia la interrupción.
* eof_INT();	Indica el fin de la rutina de interrupción.
* actualiza_time();	Calcula el tiempo final.
* reset_PC_timer();	Restablece el estado original del temporizador.
* stop_INT();	Indica el fin de la interrupción y por tanto se detiene la rutina de atención a la interrupción.

PROGRAMACIÓN DEL TEMPORIZADOR PARA GENERAR INTERRUPCIONES

Se dan los pasos necesarios para programar una interrupción por hardware.

Declaración de variables

Se declaran variables que sirven para guardar la hora del reloj y además una rutina o variable que almacena el vector de interrupciones. La sintaxis es:

```
void interrupt(*old_INT) (void);    /* para salvar el vector de interrupción-8 */
struct time timesist;               /*: guarda la hora del sistema        */
struct time timeendsist;            /*: guarda la hora_F del sistema      */
struct time timetotsist;            /*: actualiza la hora del sistema     */
```

El temporizador es un registro que es decrementado con una frecuencia de 1.194 Mhz, cuando la cuenta en el registro es cero entonces se genera una interrupción.

El siguiente paso es realizar la rutina que programa al temporizador, la cual recibe como parámetro de entrada el periodo de interrupción (en este caso de 1 a 54 ms). El cristal que controla al temporizador es de 1.194 MHZ, por tratarse de una maquina con procesador 386 o 486. El número con el que se programa al temporizador es un número divisor que divide la frecuencia del cristal en el número dado, para establecer el periodo de interrupción (por omision el temporizador está programado para interrumpir cada 54 ms; por tanto el número introducido es 54*1194, que equivale a 18.52 ticks por segundo (el termino tick se define como la frecuencia con la que el temporizador realiza las interrupciones), que es el estándard para cualquier PC).

La forma de programar al temporizador es cargando en la dirección 0x43 (dirección del temporizador) un número que contiene el modo de operación.

El formato es el siguiente:

Función	No.cont.	No.lect.	Modo	Tipo contador
Binario	0 0	1	1 0	1 0 0
	MSB			LSB

Que equivale al número hexadecimal 0x34

Este número se carga en dos partes: parte alta y parte baja. A continuación se presenta el ejemplo de la rutina que programa al temporizador.

```
/* **********  cargando e iniciando el temporizador  ************/

void prog_timer( unsigned int msec )
{
 unsigned int  t;

 /* 1194 es (1.194 MHZ)/1000 del clock de una (80-386) */
        /* t= 1194*1 = 1194     ;  1.194M/t = 1000 ticks/s  */
        /* t= 1194*54= 64480    ;  1.194M/t = 18.52 ticks/s */
 t = msec * 1194;
 outportb( 0x43, 0x34 );          /* cargando la configuración del contador

 MODO = 0x34 =     00              1              10       100
 !!!!!   número contador=0  2lecturas=LBS,MBS  modo=2  cont=BCD       !!!!!
                                 */
                                 /* ejemplo para t=1*1194          */
 outportb( 0x40, low(t) );        /* cargando LBS=94 en el contador       */
 outportb( 0x40, high(t) );       /* cargando MBS=11 en el contador       */
}
```

Una vez preparada la rutina que programa al temporizador, el siguiente paso es programar la interrupción, en este proceso se deshabilita a las interrupciones, se salva la hora del sistema, se salva el vector de interrupciones para el tiempo, se programa el temporizador, se carga un nuevo vector de interrupciones y se vuelve a habilitar las interrupciones. A continuación se presenta la rutina que programa la interrupción.

```
void start_INT1( unsigned int t_INT )
{
 DISABLE;                         /* deshabilitando interrupción de hardware   */
 gettime( &timesist );            /* guardando la hora del sistema en time_sist */
 old_INT = getvect(8);            /* salvando el vector de la interrupción(8)  */
 prog_timer( t_INT );             /* programando el temporizador con el T de muestreo */
                                  /* variable "T_INT", que contendra el tiempo de muestreo (1..54 en ms)
*/
 setvect( 8, Rut_INT1 );          /* modificando el vector de la interrupción-8 para cargar la rutina de
servicio a la INT ( en este caso Rut_INT1 ) */
 ENABLE;                          /* activando interrupción de Hadware                */
}
```

NOTA: al ser cargado el nuevo vector de interrupción, se está cargando también la rutina de atención a la interrupción.

B.5 USO DE INTERRUPCIONES PARA MINIMIZAR EL TIEMPO DE MUESTREO

Para ésto sólo es necesario utilizar las rutinas pide_time(), start_INT(), y stop_INT(), que se usan en el programa principal, ya que todas las rutinas se encuentran conectadas entre si y sólo hay que poner una nueva rutina de atención a la interrupción. Esta debe de contener el procedimiento a realizar cada vez que suceda la interrupción. Al final de ésta se incluye a eof_INT() que indica el final de la rutina de atención a la interrupción.

Los pasos a seguir son los siguientes:

1.-Declarar la rutina de atención a la interrupción, por ejemplo:

```
void interrupt Rut_INT1();              /* vector de  Interrupción                    */
void prog_timer( unsigned int msec );/* rutina para prog. el temporizador          */
void start_INT1( unsigned int t_INT );/* rutina que genera la interrupción          */
void eof_INT();                         /* fin de la rut de atencion a la interrupción */
void reset_pc_timer( void );            /* rutina para restablecer el temporizador    */
void stop_INT();                        /* rutina que detiene la interrupción         */
void actualiza_time();                  /* rutina que actualiza el temporizador       */
```

Para evitar el copiar las rutinás, sólo se debe de incluir el nombre del archivo que tiene contenidas las rutinas que anteriormente se declararon, para poder utilizar las interrupciones y poder programar el temporizador de la PC.

2.- Programar la rutina que establece la interrupción (en este caso start_INT1();).

3.- Programar la rutina de atención a la interrupción (en este caso Rut_INT1();).

Para declarar la rutina de atención a la interrupción, solo es necesario poner:

```
void interrupt Rut_INT()
{
 cuerpo de la rutina de atención a la interrupción
        .....
        .....
          .
          .
        .....
  eof_INT();      /*    Instrucción de fin de rutina de atención a la interrupción */
}
```

Como la rutina de atención a la interrupción no recibe ni entrega ningún parámetro, entonces es necesario declarar como variables globales a las variables utilizadas por la rutina del programa principal.

4.- Programar el módulo principal, en el cual se incluirán los siguientes comandos:

-Pedir el tiempo de muestreo (con pide_time()), este será el periodo de interrupción.

-Inicia la interrupción (con start_INT();).

-Entonces ya se puede realizar cualquier rutina en el módulo principal (ajena a la rutina que fue declarada como rutina de atención a la interrupción).

-Para terminar la rutina de atención a la interrupción, sólo se pondrá la declaración de fin de interrupción (se declara con stop_INT();), esta detendrá la interrupción y restablecerá el vector y el temporizador de la computadora.

-Fin del programa principal.

A continuación se presenta lo que sería la estructura del programa principal se presenta el programa rutpl_da() que llama a la rutina que inicia los parámetros de un pulso que es mandado continuamente al actuador. Hace uso de la rutina de atención a la interrupción y como cuerpo del programa se queda dentro de un ciclo, en el cual manda llamar a una rutina que manda diferentes niveles de voltaje al actuador del sistema péndulo-carro.

```
/*************************************************************************
                        PROGRAMA PRINCIPAL
*************************************************************************/
void rutpl_da()
{
 do
   {
    clrscr();
    inicia_pulso();              /* inicia caracteristicas del pulso de activ. del ACTUADOR*/
    t_INT=3;
    start_INT1(t_INT);           /* INICIO DE LA INTERRUPCIÓN                    */

    DA_actu();                   /* rutina que escribe al puerto analógico y al péndulo  */
    stop_INT();                  /* FIN DE LA INTERRUPCIÓN                       */
    printf(" para terminar oprima \"q\" si no oprima cualquier tecla ");
   } while (getche() != 'q');
}     /*  fin de main        */
```

Finalmente se presenta, una lista de librerías que contienen todas las rutinas para el uso de la tarjeta, además de rutinas de programación para el carro y péndulo invertido.

Lista de las rutinas, que deben incluirse en un un programa en C (incluidas en el paquete PCPEIN).

```
/* ============ Declaración de archivos incluidos para manejar ============== */
/* ============ las rutinas principales de control.  ============== */

#include "lee_pen2.h"     /* rutinas para leer y escribir al los convertidores A/D y D/A. */
#include "rutpl_da.h"     /* rutinas para gen. INT y para generar los pulsos que activan el actuador . */
#include "lect_mat.h"     /* para leer matrices de K y polos del sistema del carro. */
#include "lee_matp.h"     /* para leer matrices del sistema, vectores de la ley de control 'K' y del estimador
'L' y polos para el sistema del PÉNDULO . */
#include "write_ve.h"     /* declaración de variables y rutinas para grabar arreglos de vectores (de los
estados) en un archivo*/
#include "graf_c.h"       /* para trazar los estados del carro          */
#include "panta2.h"       /* para poner colores y marco       */

/* ******** RUTINAS PRINCIPALES DE CONTROL PARA EL SISTEMA ********** */

#include "resp_esc.h"     /* Programa para obtener respuesta escalón. */
#include "r_carr2.h"      /* Programa para control del carro. */
#include "r_pend1.h"      /* Programa para el control del "PÉNDULO". */

/* ================================================================== */
```

Los listados de los programas principales se encuentran en el ápendice C, y además el trabajo cuenta con un disquete extra que contiene los programas fuente.

APÉNDICE C

Para evitar el aumento del presente reporte se listan en este apéndice únicamente los dos programas principales que conforman el paquete PCPEIN, anexando un disquete con las rutinas de lectura y escritura de archivos, y de la generación de gráficos. Como información complementaria se indica la forma de instalar el paquete en disco duro de la computadora personal.

El paquete PCPEIN se proporciona en un disquete de 3½, el cual contiene los subdirectorios pendulo y fuentes. El primero contiene los programas escritos en MATLAB junto con el programa ejecutable C_PENDUL.EXE para el control en tiempo real. El subdirectorio fuentes como su nombre lo indica contiene el programa y rutinas fuente escritos en C++.

La ejecución del paquete puede hacerse desde el propio disquete o a través de un disco duro. En el caso de la última opción se debe generar un subdirectorio en el disco duro en el cual se copia el contenido del subdirectorio péndulo que contiene la versión ejecutable de c_pendul y los programas de MATLAB.

Para evitar saturar el disquete de información se recomienda los datos del experimento y de los parámetros de control y modelado en el subdirectorio péndulo del disco duro.

Para dar una descripción esquemática de la conformación del paquete PCPEIN a continuación en la Fig. 1.C se presenta el diagrama de bloques de sus rutinas, detallando por separado en la fig. C.2 el programa principal del control en línea c_pendul.c.

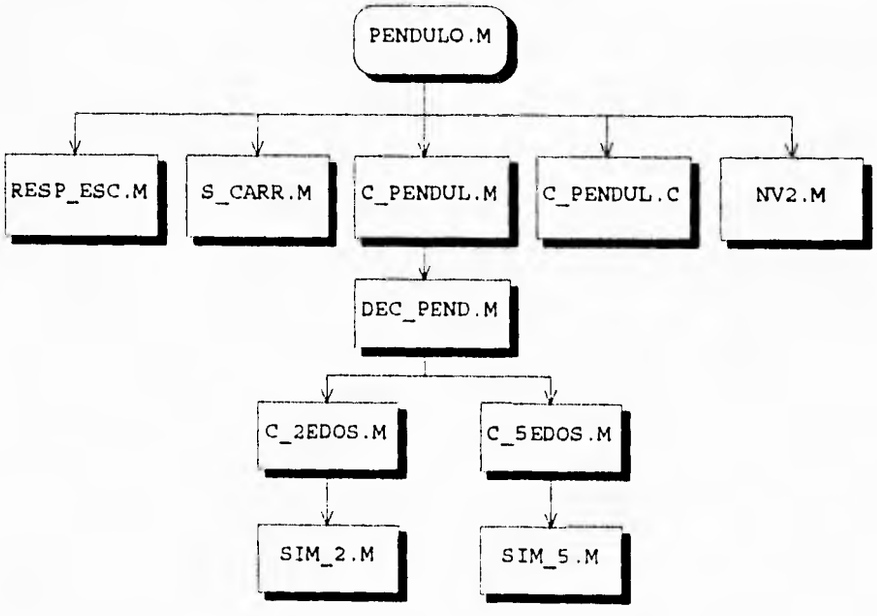


FIG. C1 Diagrama de bloques del paquete PCPEIN

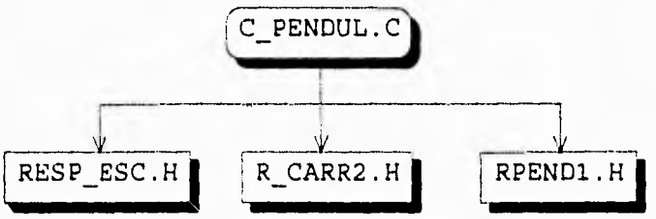


FIG. C2 Diagrama de bloques del programa de control en línea

LISTADO DEL PROGRAMA Y RUTINAS PRINCIPALES EN MATLAB

```
% PENDULO.M
%
%          El siguiente programa está constituido, por una serie de
%         rutinas que desempeñan cálculos de control y simulaciones.
%
%          Se encuentra compuesto por dos rutinas de análisis de resultados, y
%         por dos rutinas que realizan los cálculos para el control de los dos
%         sistemas (carro y péndulo), junto con la simulación, los cuales,
%         ademas generan archivos de datos, para poder controlarlos en tiempo
%         real.
%
% Las rutinas utilizadas son las siguientes:
%
%  - RESP_ESC.M : Es utilizada para realizar pruebas con el archivo de datos
%                        obtenido en el programa C_PENDUL.EXE (en "C"). De la
%                        opción de respuesta escalón para el carro.
%
%  - S_CARR.M : Es utilizado para simular el control del carro. Solicita
%                        los polos de la ley de control  y  del el observador de
%                        dos estados. Salva las ganancias de la ley de control (K),
%                        del observador (L) y las matrices de estados.
%
%  - C_PENDUL.M : Esta es la rutina principal que realiza una simulación.
%                        Ademas calcula la retro de estados y ganancias para el
%                        observador (ambos, por dos métodos: DLQR y Ackerman).
%                        Salva las ganancias de la ley de control (K), del observador
%                        (L) y las matrices de estados.
%
%  _C_PENDUL.EXE : Es el programa que realiza el control en tiempo real.
%
%  _NV2.M :   Realiza el análisis de error para el control de posición
%                        y calcula las normas de error y de energía usadas para el
%                        control (del carro y del péndulo).

%-------------------------------------------------------------------------
% clear
clc
while 1

        clc,clg
        men1=menu(' Practica de Control con el "PÉNDULO INVERTIDO" :',...
                'Descripción general del programa',...
                'Identificación de la fricción seca en el sistema.',...
                'Simulación del sistema de control para el carro.',...
                'Simulación del sistema de control para el Péndulo .',...
                'Regulación de los sistemas en tiempo real.',...
                'Análisis de respuestas del control',...
                'Fin de la practica');

        if men1==1,
                delete(1)
          clc, help péndulo, pause;
        elseif men1 == 2,
```

```
                delete(1)
            resp_esc;
        elseif men1 == 3,
                delete(1)
            s_carr;
        elseif men1 == 4,
                delete(1)
            c_pendul;
        elseif men1 == 5,
            clc, clg, delete(1)
            disp(' Para correr los programas de control en tiempo real, fuera del ambiente Windows.')
            disp(' Oprima el juego de teclas [ALT] y [ENTER] al mismo tiempo')
                disp(' ')
            disp(' y al terminar esta aplicación, para cerrar la ventana de trabajo,')
            disp(' oprima doble clic con el botón izquierdo del ratón, al cuadro que se encuentra ')
            disp(' en la esquina superior izquierda de la ventana de trabajo ')
                disp(' ')
            disp(' Para comenzar el programa de control en tiempo real oprima cualquier tecla ')
            pause
            !c_pendul;
        elseif men1 == 6,
                delete(1)
                nv2;
        elseif men1 == 7,    % opción para terminar.
                break;break, close
            % close, quit
            % clear, clc, clg, return,return,return;
        else  disp(' esta opción no existe "elija" correctamente una opción');
      disp('\n        --- oprima cualquier tecla ---      ');
          pause;
        end;

end;
delete(1)
clc;close,quit
```

%RESP_ESC.M

```
%                   Este programa permite analizar la respuesta escalón del carro
%             con base en un registro real del sistema y cálcula el error entre el
%             comportamiento del modelo Lineal y el proceso real.
%             Además aproxima el error del modelo lineal a una función.
%             Finalmente simula y compara resultados entre la respuesta real y la
%             del sistema identificado y caracteriza la fuerza Uso.
%
%             Para identificar el error se utiliza la función FUN_JU2.M
%             Para identificar la relación Uso/vel. se utiliza la función RECTA.M

 clc;
diary resp_esc.doc
%--------------------------------------------------------------------------
while 1      % para entrar a un menú dentro de un ciclo
 clc;
 disp('     Dar el nombre del archivo para cargar los datos, el cual');
 opcion=input('                    debe tener extensión .m  "0" para salir ');

 if (opción == 0)    % opción para terminar.
          return;
 end

 Ux= ESC(2);        % tomando el valor del escalón del arreglo.

 Kf=2.6;     % [N/V]
 Mo=3.2;     % [Kg]
 Fr=6.2;     % [Kg/s]
 n33=-7.64;  % [V*s/m]
 % ====== valores de G(s) ==========
```

```
K1=Kf*n33/Mo;
K2=Fr/Mo;

% =========== cálculo del error y de la velocidad lineal ================
 Ktx=(Kf*n33/Fr)*(1-exp(-Fr*t/Mo))*Ux;     % velocidad lineal del carro
 y=XV-Ktx;                                  % error o parte no lineal

%%% Optimización de los datos del error para aproximar una función a estos.
% ----------------------------------------------------------------------
Co = [1 1 1 1];       % condiciones iniciales para empezar a aproximar constantes.
op(1)=0;
[Cx,op] = leastsq('fun_ju2',Co,op,[],y,t);   % proceso de optimización

C1=Cx(1);
C2=Cx(3);
C3=Cx(4);
lan1=Cx(2);

            yerr=C1*exp(-lan1*t) + C2*t.*exp(-lan1*t) + C3;
            f=y-yerr;
            xt=max(t)/3;
            yt=max(yerr)/3;
            title('Optimización de los datos de la vel.no lineal  ');
            text(xt-xt/2,yt,['error de aprox = ' sprintf('%g ', norm(f))]);
            text(xt-xt/2,yt-yt/2,['ooo Vnl,  __ V^nl ']),
pause
clg, delete(1)

x2=[Cx op(8)];      % op(8) es el error de aproximasion

% ------------[ despliegue de resultados obtenidos ]--------------------
clc
help fun_ju2;

%  yerr=C1*exp(-lan1*t) + C2*t.*exp(-lan1*t) + C3;

disp(' Los valores obtenidos para la función propuesta son ');
fprintf('\n\t                C1 = %f ',C1);
fprintf('\n\t                C2 = %f ',C2);
fprintf('\n\t                C3 = %f ',C3);
fprintf('\n\t             landa1 = %f ',lan1);
fprintf('\n\n\n\n');
pause

% =================== generación de uso ================================

A=(C1+C3)/K1;                 % cte (impulso)
B=C3*K2/K1;                    % cte (escalón)
C=(K2-lan1)*(C1+C2*t)/K1 + C2/K1;    % vector

Uso = B + C.*exp(-lan1*t);
Uso(1) = Uso(1)+A;
Uso=Uso;
Us = ESC + Uso;
data=[XV Uso];

% ========= gráfica de los vectores posición , velocidad y escalón ========
ESC1=ESC*2.6;                          % [N]
X1=X*100/14.9/ESC1;                    % [m/N]
V=XV*100/(-7.64)/ESC1;        % [m/s/N]

subplot(221);
plot(t,X1,'r',t,V,'g',t,ESC1,'b')
title('X1, X2, U')
ylabel('valores norm')
Tt= num2str(Ux);
grid
```

```
% ============= gráfica de los vectores vel. error y vel.lin =============
subplot(222)
plot(t,XV,'r',t,Ktx,'g',t,y,'b')
title(' V.s, V.L, V.nL')
ylabel('[ V ] ')
grid
%============== gráfica del error y de la aproximación ===============
subplot(223)
plot(t,yerr,'r',t,y,'b')
title(' v.L y v.nL'),
xlabel('t [ s ]')
ylabel('v [ V ] ')
grid
% ============= gráfica de Uso , Uo y Us =============
Us2= ESC - Uso;     % Esto es, porque se le suma Uso al
subplot(224);       % escalón para linealizar al sistema.
plot(t,Uso,'r',t,ESC,'g',t,Us2,'b')
title('Uso ,Uo y Uso+Uo')
xlabel('t [ s ]')
ylabel('[ V ] ')
grid
pause
clg, delete(1)


%************** [ nuevo juego de gráficas ] ********************

% ========== simulación de los datos obtenidos ==========================
% === comprobación de la función generadora de la fricción en el sistema ==
num=K1;
den=[1 K2];

y2=lsim(num,den,Us,t);      % Us=ESC+Uso
y3=lsim(num,den,ESC,t);

% ============== graficás de simulaciones ==============================
subplot(221)
plot(t,y2,'-',t,Us,'g',t,XV,'-'),
title('V.s y V^s');
ylabel('v [ V ] ')
grid
errsim=y2-XV; nerrs=norm(errsim)

% ============= gráfica Uso vs XV=============
subplot(222);
plot(XV,Uso)
title('Relación de "Uso vs XV"')
xlabel('Vel [ V ] ')
ylabel('Uso [ V ] ')
grid

%-------------------------------------------------------------------
%-----------[ optimización de la ecuación de la recta ]-----------------
%              ( cuando obtengo Us/vel.real )
%
%  Us=Esc+Uso  ;  XV=velocidad real del sistema.
%  Vu=Mx*XV + Bo ==>Uso
%
uo=[1. 1];
op(1)=0;
[vu1 op]=leastsq('recta',uo,op,[],XV,Uso);
vu1=[vu1 op(8)];

Mx=vu1(1);
Bo=vu1(2);
Vu=Mx*XV + Bo;
disp(' Los valores de la recta ajustada, son :');
```

```
fprintf('\n\n\t Uso(v) = %f*v  +  %f*sgn(v)   [V] \n\n',Mx,Bo);

subplot(223);
plot(XV,Vu,'--',XV,Uso,'-')
title('Ajuste de la recta Us/vel.R ')
xlabel('vel.R [ V ]')
ylabel('Uso [ V ] ')

grid
pause
subplot(111);
clg, delete(1)

% ------------------------------------------------------------------------
diary off
end;
% ====== Salvando resultados en un archivo.MAT para procesarse después ====

clc
disp('Introduzca el comando "save name" entre apóstrofes');
salida=input('esto es para que la sesión se salve con este nombre : ');
fprintf('/n/n');

disp('Las variables con los resultados se están salvando con el nombre que diste');
disp('pero con extensión ".mat" ');
eval(salida);   % salvando los resultados obtenidos
```

%S_CARR.M

```
%
%     Apartir de las ecs.en var de edo.Disc.:
%
%           Xz=Ad*x + Bd*u
%           Yz=Cd*x
%
%     Se calcularan las ganancias: K's de la Ley de control
%
%     y el estimador de estados.
%
%    " Con lo cual se realizara una simulación del sistema. "
%           (solo para la ley de control)
%
%     Nota:  PK: Es el vector de polos para la ley de control.
%          (estos pueden contener #'s complejos [a+/-bj])
 clear,close,
 format short e;

 Kf=2.6;
 Mo=3.2;
 Fr=6.2;
 n11=14.9;      % n11 [V/m]. cte para cambiar de metros a volts
 n33=-7.64;

 K1=(Kf*n33)/Mo;
 K2=Fr/(n33*Kf);
 K3=n11/n33;

% matrices normalizadas del modelo del carro
 An=[0  K3    ;
    0 -K1*K2 ];

 Bn=[0 ;
    K1];

 Tm=0.03;
 [Ad,Bd]=c2d(An,Bn,Tm);
 Cd=[1  0];
```

```
clc, clg;
delete(1)
help s_carr,

j=sqrt(-1);
fprintf('\n El periodo de muestreo es = %g ',Tm);
fprintf('\n\n Introduzca los polos del controlador Ps \n');
for i=1:2,
        fprintf('\n cual es el valor del polo Psk( %g ) \n\t',i );
        Psk(i)=input('Psk = ');
        Pzk(i)=exp(Psk(i)*Tm);  fprintf(' Pzk( %g) = %g ',i,Pzk(i));
end

K=acker(Ad,Bd,Pzk);
fprintf('\n\n La ley de control obtenida fue:\n\t  K1 = %g \n\t  K2 = %g \n',K(1),K(2) );
%  eig(Ad-Bd*K);

fprintf('\n\n Introduzca los polos para el observador PsL \n');
for i=1:2,
        fprintf('\n cual es el valor del polo PsL( %g ) \n\t ',i );
        PsL(i)=input('PsL = ');
        PzL(i)=exp(PsL(i)*Tm);  fprintf(' PzL( %g) = %g ',i,PzL(i));
end

An2=[-K1*K2  K1 ; 0   0 ];
Bn2=[K1 ; 0];

[Ad2,Bd2]=c2d(An2,Bn2,Tm);
L=acker(Ad2',Cd',PzL);
fprintf('\n\n Las ganancias para el observador son:\n\t  L1 = %g \n\t  L2 = %g \n',L(1),L(2) );

eig(Ad2-L'*Cd);
pause;
clc;

XK1=[30*nl1/100  0];
clc;
N=50;

% ********** CONDICIONES INICIALES PARA ARREGLOS **********

        XK11=XK1;          % inicial izando el arreglo de estados. de entrada
        Xi(1)=0;          % inicial izando el arreglo del # de muestras
        XK1=XK1';
        mat_sist= Ad-Bd*K;   % Matriz representativa de la planta

% ************************************************************

        for k=1:N,
          Xi(k+1)=k;              % arreglo del # de muestras
          Tu(k)=k;

          % ecuación que me representa a la planta
          XXK=mat_sist*XK1;
          XK1=XXK;

          % almacenando las entradas en un arreglo
          XK11=[XK11;XK1'];

          U=-K*XXK;
          UU1(k)=U;
        end

diary off;

%----- gráficas de la respuesta del control, posición y velocidad ----------
```

```
 g_carr;

% ------- almacenando en archivos para controlar en tiempo real ----------
delete(1)
sx=0;
fprintf('salvamos archivos para usarse en tiempo real ( [1]=si , [ENTER]=no )');
sx=input('salvo = ?');

if sx == 1'
 save PK_carr.res Psk /ascii;
 save PL_carr.res PsL /ascii;
 save A2_carr.res Ad2 /ascii;
 save B2_carr.res Bd2 /ascii;
 save K_carr.res K /ascii;
 save L_carr.res L /ascii;
end;


%                          C_PENDUL.M
%
%%              Cálculo de la ley de control y estimador
%             con base en el modelo normalizado y discretizado
%             obtenido a partir de las matrices 'A' y 'B' con un
%                    periodo de muestreo de 0-035
%
%                                 obteniendo así las ecuaciones
%                                 Xk+1=Ad*Xk + Bd*Uk
%                                  Yk=Cd*Xk
%

%    Nota:  Nn: Matriz de normalización.
%       Cd: Matriz de salida.
%          Pk: Es el vector de polos para la ley de control.
%       Pl: Es el vector de polos para el observador.
%
%-------------------------------------------------------------------------

clear, close
clc;
dec_pend;     % declaración de constantes y matrices del péndulo invertido.

%-------------------------------------------------------------------------

%disp(' valores caracteristicos de la matriz Ad');
%Vc=eig(Aa);
%-------------[ selección de los polos para la ley de control ]------------
help c_pendul;
Pks=[-4.3;-4.3;-4.3;-4.3]';
fprintf('\nSe desean Modificar los polos de control por omisión ? \n\t\t( [1]=si , [ENTER]=no )');
fprintf('\n\n\n');
fprintf('\tPks = ( %g , %g , %g , %g )\n',Pks(1),Pks(2),Pks(3),Pks(4));

format short e;
que=input('  = ?');

if que == 1,
 ksist=0;
 fprintf('El controlador se diseña con el DLQR o Ackerman  [1/2]');
 ksist=input('controlador = ?');

 if ksist==1,
  QC=input('QC = '); r=1; [K,S,Pkz]=dlqr(Ad,Bd,QC,r);
  fprintf('\n\n La ley de control obtenida fue:\n');
  fprintf('\n\t  K1 = %g \n\t  K2 = %g ',K(1),K(2) );
  fprintf('\n\t  K3 = %g \n\t  K4 = %g \n\n',K(3),K(4) );

  Pks=log(Pkz)/0.03;
```

```
  fprintf('\n Y los polos de la ley de control : \n\n');
  fprintf('\tPks = ( %g , %g , %g , %g )\n',Pks(1),Pks(2),Pks(3),Pks(4));
  pause

 elseif ksist==2,
  fprintf('\n\n\n Introduzca los polos del controlador (a +/- bj) n n');
   for i=1:4,
            fprintf('cual es el valor del polo Psk( %g ) ',i );
            Pk=input('Psk = ');
            Pks(i)=Pk;
            Pkz(i)=exp(Pk*0.03);
            fprintf(' Pkz(%g) = %g \n',i,Pkz(i));
   end
  K=acker(Ad,Bd,Pkz);
  fprintf('\n\n Ley de control obtenida \n');
  fprintf('\n\t K1 = %g \n\t K2 = %g ',K(1),K(2) );
  fprintf('\n\t K3 = %g \n\t K4 = %g \n\n',K(3),K(4) );
 end
else
  Pks=[-4.3;-4.3;-4.3;-4.3]';
  Pkz=exp(Pks*0.03);
  K=acker(Ad,Bd,Pkz);
  fprintf('\n\n Ley de control obtenida\n');
  fprintf('\n\t K1 = %g \n\t K2 = %g ',K(1),K(2) );
  fprintf('\n\t K3 = %g \n\t K4 = %g \n\n',K(3),K(4) );
end

%=============================================================
qsist=0;
fprintf('Especifique el tipo de observador (2 ó 5 estados) [2 5]');
qsist=input(' = ?');

if qsist ==2,
 c_2edos;
end

if qsist ==5,
 c_5edos;
end

clc;
fprintf('Desea continuar con el diseño [1]=si ó [ENTER]=No ?');
que=input('continua = ?');
if que ==1,
 c_pendul;
end
```

%DEC_PEND.M Declaración de constantes y datos para el péndulo invertido:

```
%
% Normalización y discretización de las matrices del sistema de 4 estados
%
%----------------[ lista de constantes del modelo del péndulo ]-------------------

Kf=2.6;       %[N/V]
n11=14.9;     %[V/m]
n22=-52.27;   %[V/rad]
n33=-7.64;    %[Vs/m]
n44=-52.27;   %[Vs/rad]

Mo=3.2;       %[Kg]
M1=0.329;     %[Kg]
M=3.529;      %[Kg]
ls=0.44;      %[m]

Og=0.072;     %[Kg*m^2]
```

```
N=0.1446;      % [Kg*m]
N012=0.23315;  % [Kg^2*m^2]
N2N01=0.0897;

Fr=6.2;        % [Kg/s]
C=0.009;       % [Kg*m^2/s]
g=9.81;        % [m/s^2]

%-------------------------------------------------------------------------

a32=-N*N*g/N012;   a33=-Og*Fr/N012;   a34=N*C/N012;   a35=Og*N/N012;

a42=M*N*g/N012;    a43=N*Fr/N012;     a44=-M*C/N012;  a45=-N*N/N012;

b3=Og/N012;        b4=-N/N012;

%----------------[ declaración de matrices en t. continuo ]----------------

Aa=[0   0   1   0
   0   0   0   1
   0  a32 a33 a34
   0  a42 a43 a44];

Ba=[ 0
    0
    b3
    b4];

%---------------[ normalización de las matrices del sistema ]--------------

% matriz de normalización

Nn=[n11 0  0  0
   0  n22 0  0
   0   0 n33 0
   0   0  0 n44];

An=Nn*Aa*inv(Nn);       % normalización de A continua

Bn=Nn*Ba*Kf;            % normalización de B continua

%---------------[ discretización de las matrices del sistema ]-------------

Tm=0.03;       % declaración del periodo de muestreo para discretizar
                        % las matrices del sistema continuo.

[Ad,Bd]=c2d(An,Bn,Tm);

%--------[ matrices del sistema en el manual de datos ]---------------------

%Ad=[ 1  1.108e-4  -5.68e-2  4.1e-7
%    0  1.01       1.16e-2  3.0035e-2
%    0  -3.755e-3  0.944    -3.299e-5
%    0  0.6435     0.768    1.006    ];

%Bd=[ 0.0053
%    0.0372
%   -0.1789
%    2.461];

%------[ discretización de las matrices del sistema de 5 estados]-------------

 [Ax,Bx]=c2d([An Bn;0 0 0 0 0],[Bn;0],0.03);
 Cc=[1 0 0 0 0
    0 1 0 0 0
    0 0 1 0 0];
```

```
%C_5EDOS.M
%         Cálculo de ganancias "L" para un observador completo de 5 estados.
%        Incluye 2 métodos DLQR (Kalman) y asignación de polos.
%
clc;
%---------[ selección de los polos para el estimador de 5 estados ]-----------
qobs=0;
fprintf('Especifique el método que desea emplear (Kalman(1) o Ackerman(2)) ?');
qobs=input(' = ?');
if qobs ==1,

% - filtro de Kalman de orden 5 ----------------------------------------
 G=eye(5); q=input('q='); r=input('r=');
 L=Ax'dlqe(Ax,G,Cc,q,r), pzl=eig(Ax-L'Cc);
 PsL1=(log(pzl)/0.03),
 PsL=real(PsL1');

elseif qobs ==2,
% - asignación de polos -(5) -------------------------------------------
 fprintf('\n\n\n Asigne los polos del observador \n\n');
  for j=1:5,
            fprintf('dar el valor del polo PsL( %g ) ',j );
            PL=input('PsL = ');
            PsL(j)=PL;
            PzL(j)=exp(PL*0.03);
            fprintf(' PzL(%g) = %g \n',j,PzL(j));
  end
 L=place(Ax',Cc',PzL);
 L=L'
else
 c_5edos;
end;

clc;
% -------------[formando las matrices para el observador [----------------
%
%          X^(k) = [I-LC]Ax*X^(k-1) + [I-LC]Bx*Us(k-1) + L*Y(k)
%
%            X^(k) = Axy*X^(k-1) + Bxy*Us(k-1) + L*Y(k)
% ------------------------------------------------------------------------

 Tm=Tm*1000;
 Axy=[eye(5)-L*Cc]*Ax;
 Bxy=[eye(5)-L*Cc]*Bx;

% ------------------------------------------------------------------------
w=0;
fprintf('Desea continuar con la simulación No=[ENTER] continuar=[1] ');
w=input('continua = ?');
if w == 1,
 sim_5;      % simulación del sistema;
end;

% ----------[ salvando matrices y ganancias para tiempo real ]-------------
sx=0;
fprintf('Desea salvar el diseño para usarse en tiempo real [1]=si ó [ENTER]=no ');
sx=input('salvo = ?');

if sx == 1'
 save A2_pend.res Axy /ascii;
 save B2_pend.res Bxy /ascii;
 save Tm_pend.res Tm /ascii;
 save K_pend.res K /ascii;
 save L_pend.res L /ascii;
 save PL_pend.res PsL /ascii;
 save Pk_pend.res Pks /ascii;
 save datsal_p
```

```
end;


% simula 5 edos:

%- simulación del sistema  retroalimentado XK y del estimador
%- con esquemas de predicción XP y filtraje XC

%- condiciones iniciales
XK=[ 1  1  1  1  1]';
XP=[0 0 0 0 0]';
XC=[0 0 0 0 0]';
XK11=XK';  XP11=XP'; XC11=XC';

%- cálculo de las ecuaciones recursivas del sistema con observador
for k=1: 30
   % --------- almacenando vectores ------------
   j(k)=k;
   XK11=[XK11; XK'];XP11=[XP11; XP'];
   XC11=[XC11; XC'];

   % ------ simulación de la planta y cálculo de la ley de control -----

   UK=K*[XK(1);XK(2);XK(3);XC(4)];
   XK1= Ax*XK  -Bx*UK;

   %---------------- cálculo del estimador predictor ------------
   XP1= (Ax- L*Cc)*XP + L*Cc*XK - Bx*UK;

   %---------------- cálculo del estimador corriente ------------
   XC1= XP1 + L*Cc*(XK1- XP1);

   % --- salvando estados actuales en estados pasados ----------
   XK=XK1;  XP=XP1; XC=XC1;
end

%- cálculo de los errores máximos de predicción % errP & filtraje errC

errP=(XK11 - XP11); map=max(abs(errP));
errC=(XK11 - XC11); mac=max(abs(errC));
clg;

%- gráfica de los errores de estimación

subplot(221),plot(errP(:,1)), title('estimador predictor'),grid
ylabel('error de X1')
subplot(222),plot(errP(:,2)), title('estimador predictor'),grid
ylabel('error de X2')
subplot(223),plot(errP(:,3)), title('estimador predictor'),grid
ylabel('error de X3')
subplot(224),plot(errP(:,4)), title('estimador predictor'),grid
ylabel('error de X4')
pause, delete(1)

plot(errP(:,5)), title('estimador predictor'),grid;
ylabel('error del control'), pause , delete(1)

clg,
subplot(221),plot(errC(:,1)), title('estimador corriente'),grid
ylabel('error de X1')
subplot(222),plot(errC(:,2)), title('estimador corriente'),grid
ylabel('error de X2')
subplot(223),plot(errC(:,3)), title('estimador corriente'),grid
ylabel('error de X3')
subplot(224),plot(errC(:,4)), title('estimador corriente'),grid
ylabel('error de X5')
pause,delete(1)
```

```
plot(errC(:,5)), title('error de fuerza de control'),grid;
ylabel('error del control') , pause
delete(1)
```

%c_2edos.m

```
%        Cálculo de ganancias "L" para un observador reducido de 2 estados.
%       Incluye 2 métodos DLQR (Kalman) y asignación de polos.
%
clc;
%--------------[ fraccionalización de la matriz de 2 estados ]----------------

A11=Ax(1:3,1:3); A12=Ax(1:3,4:5);

A21=Ax(4:5,1:3); A22=Ax(4:5,4:5);

B1=Bx(1:3);     B2=Bx(4:5);

%---------[ selección de los polos para el estimador de 2 estados ]-------------
qobs=0;
fprintf('Especifique el mÉtodo que desea emplear (Kalman(1) o Ackerman(2)) ?');
qobs=input(' = ?');

if qobs ==1,

%---------- filtro de Kalman de orden 2 ---------------------------------
 G=eye(2); q=input('q='); r=input('r=');
 L=A22*dlqe(A22,G,A12,q,r), pz1=eig(A22-L*A12);
 PsL1 =(log(pz1)/0.03);
 PsL=real(PsL1')
 % pause;

elseif qobs ==2,

% - asignacion de polos -(2) ----------------------------------------------
 fprintf('\n\n\n Asigne los polos del observador \n\n');
  for j=1:2,
            fprintf('dar el valor del polo PsL( %g ) ',j );
            PL=input('PsL = ');
            PsL(j)=PL;
            PzL(j)=exp(PL*0.03);
            fprintf(' PzL(%g) = %g \n',j,PzL(j));
  end
 % asignación de polos de con place(Ad',Cd',PzL);
 L=place(A22',A12',PzL);
 L=L'
 eig(A22-L*A12);
else
 c_2edos;
end;
clc;
%-------[ determinación de las variables para obtener el estimador ]---------
%
%       X^(k) = Ab*X^(k-1) + Bb*Us(k-1) + L*Y(k) + Fb*Y(k-1)
%-------------------------------------------------------------------------

Ab=A22-L*A12;

Fb=A21-L*A11;

Bb=B2-L*B1;

%-------------------------------------------------------------------------
w=0;
fprintf('Desea continuar con la simulación ( [1]=si ó [ENTER]=No ) ');
w=input('continua = ?');
```

```
if w == 1,
  sim_2;  % simulación del sistema;
end;

%-------------------------------------------------------------------
sx=0;
fprintf('Desea salvar el diseño para usarse en tiempo real ( [1]=si ó [ENTER]=no )');
sx=input('salvo = ?');

if sx == 1'
 save A22_pend.res Ab /ascii;
 save B22_pend.res Bb /ascii;
 save FB2_pend.res Fb /ascii;
 save K2_pend.res K /ascii;
 save L2_pend.res L /ascii;
 save PL2_pend.res PsL /ascii;
 save Pk2_pend.res Pks /ascii;
 save dat3pend
end;
```

% SIM_2EDO.M

```
% ------- inicialización de variables ------------
 XK=[1 1 1 1 1]';
 XP=[1 1 1 0 0]'; XC=[0 0]';
 XK11=XK'; XP11=XP'; XC11=XC';
 LT=[eye(3);L];

%-------- simulación del sistema y del estimador --------
 for k=1:50
 % ---------- almacenando arreglos y vectores -----------
  j(k)=k;
  XK11=[XK11; XK'];
  XP11=[XP11; XP'];
  XC11=[XC11; XC'];
  UK=K*[XK(1);XK(2);XK(3);XC(1)];

  % ------- simulación de la planta -----------
  XK1=Ax*XK -Bx*UK;

  % ------- simulación del estimador predictivo ---------
  XP1=(Ax - LT*Cc)*XP -Bx*UK + LT*Cc*XK;

  % ------- simulación del estimador corriente ---------
  XC1=Ab*XC - Bb*UK + (A21-L*A11)*XK(1:3,1) + L*XK1(1:3,1);

  % ------ salvando estados actuales en estados pasados -------
  XK=XK1;  XP=XP1; XC=XC1;
 end

errP=(XK11(:,4:5)-XP11(:,4:5)); map=max(abs(errP));
errC=(XK11(:,4:5)-XC11(:,1:2)); mac=max(abs(errC));
subplot(221),plot(errP(:,1)),title('Estimador predictor'),grid,
ylabel('error de X3')
subplot(222),plot(errP(:,2)),grid % title('errP de la fuerza de control')
ylabel('error de X5')
subplot(223),plot(errC(:,1)),title('Estimador corriente'),grid
ylabel('error de X3')
subplot(224),plot(errC(:,2)),grid, % title('errC de la fuerza de control')
ylabel('error de X5')
pause
clg, delete(1)
```

% NV2.M

```
% este programa debe correrse después de haber almacenado datos de una
% practica control, del carro o del péndulo

while 1       % para entrar a un menú dentro de un ciclo
 clc;
 clear,
 disp(' De el nombre del archivo de muestras, para analizar. "0" para salir');
 opcion=input('debe de ser de extensión ".m" (omitirla) : ');

 if (opción == 0)     % opción para terminar.
          return;
 end

          err=X-Xd;
          plot(t,err,'b'); title('Error de la posición:       ');
          xlabel('t [ s ]'),
          ylabel('Error en [ m ] ');
          grid, pause

          plot(t,ESC,'r'); title('voltaje aplicado al motor ');
          xlabel('t [ s ]'),
          ylabel('Us en [ V ] ');
          grid, pause

          clg;delete(1)
          nx=norm(err);
          fprintf('\n norma del error para el control de posición:   nx = %g',nx);

          nu=norm(ESC);
          fprintf('\n norma de la energía utilizada para el control: nu = %g',nu);
          pause

end, break
```

LISTADO DEL PROGRAMA PRINCIPAL EN LENGUAJE C++

```
/* ----------------------------[ C_PENDUL.C ]----------------------------- */
/* programa que llama a los programas del Resp_esc.h,R_car2.h y r_pend1.h   */
/* Programa principal que llama a todas las rutinas.h  */

  char op_men1;        /* variable para identificar la opción seleccionada del menú principal :resp.escalon, control del carro, control del
péndulo */

/* ==========  Declaración de archivos incluidos para manejar ============== */
/* ==========  las rutinas principales de control.          ============== */

#include "lee_pen2.h"   /* rutinas para leer y escribir al los convertidores A/D y D/A. */
#include "rutpl_da.h"            /* rutinas para gen.INT y para generar los pulsos que activan el Actuador . */
#include "lect_mat.h" /* para leer matrices de K y polos del sistema del carro. */
#include "lee_matp.h"            /* para leer matrices del sistema, vectores de la ley de control 'K' y del estimador 'L' y polos para el
sistema del PÉNDULO . */
#include "write_ve.h"   /* declaración de variables y rutinas para grabar arreglos de vectores(de los estados) en un archivo*/
#include "graf_c.h"    /* para graficar estados del carro */
#include "panta2.h"    /* para poner colores y marco    */

/* ******** RUTINAS PRINCIPALES DE CONTROL PARA EL SISTEMA ********************* */

#include "resp_esc.h"   /* Programa para obtener respuesta escalón. */
#include "r_carr2.h"   /* Programa para control del carro. */
#include "r_pend1.h"  /* Programa para el control del "PÉNDULO". */

/* ============================================================================== */
void main()
{
  int op;

  do
```

```
{
 textcolor(15);
 clrscr();
 vent_fondo();    /* desplegar ventana principal con textos      */
 textbackground(1);
 textcolor(15);
 gotoxy(15,5); printf(" \"PRUEBAS PARA EL SISTEMA DEL PÉNDULO INVERTIDO\" ");
 gotoxy(25,10); printf("1) - Respuesta Escalón. ");
 gotoxy(25,12); printf("2) - Control del Carro. ");
 gotoxy(25,14); printf("3) - Control del Péndulo. ");
 gotoxy(25,16); printf("0) - Terminar la sesión.");
 /* gotoxy(10,21); printf("seleccione una opción :  ");*/

 do {
        op_men1 = getch();
        op = op_men1;
        op -= 48;
        } while (op<0 || op>3);

 if (op==1) resp_esc();
 if (op==2) r_carr2();
 if (op==3) r_pend1();
 } while (op!=0);
 clrscr();
}


/* ------------------------{ RESP_ESC.H }------------------------------ */
/* programa que aplica un escalón para poder estudiar y analizar el sistema NO lineal del carro */

#include <math.h>
#include <bios.h>
#include <conio.h>
#include <stdio.h>

void resp_esc();          /*    programa principal = main()          */

/* ***************** DECLARACIÓN DE RUTINAS DEL PROGRAMA ******************** */

void verif_pos();                 /* verifica que el carro este en los extremos*/
void escalón();                          /* pide el valor del escalón a aplicarle al c*/
void control_r();          /* programa que realiza el control del péndulo  */
void interrupt Rut_INT_R( );   /* rutina de atención a la interrupción    */
void start_INT_R( unsigned int t_INT ); /* rutina para inicializar la INT   */
void mensaje1_R();                /* mensajes de posición velocidad etc          */
void mensajes_R();                /* mensajes de posición velocidad etc          */

/*    ***************** PROGRAMA PRINCIPAL ******************       */
void resp_esc()
{
 unsigned int tiempo;
 char que;

 do
  {
  clrscr();
  escalon();          /* pide el valor del escalón para excitar el carro */
  verif_pos();        /* verificar la posición inicial del carro         */
  clrscr();
  mensaje1_R();       /* despliegue de la posición inicial del carro */
  inicializando_var(); /* inicializando variables y vectores globales*/
  tiempo=1;           /* Tint=1 porque la cuenta es intema c/cont2  */
  ESC[i]=0;

/* ==========inicio de la interrupción y del control========== */
  start_INT_R(tiempo); /* inicializando interrupciones              */
```

```
    verif_actu();       /* rutina para verificar que el actuador se active */
    printf("\a\a\a");
    activo=1;           /* bandera. que avisa que el proceso esta activo  */
    do
            {
             mensajes_R();
             if ( activo == 0 )
               {
                gotoxy(20,23);  printf(" \" FIN DEL PROCESO \" ");
                stop_INT();     /* Deteniendo la interrupción */
                goto fin_m;
               }
            } while (!kbhit());

/* ==================== final del escalón e INT ========================== */
    fin_m:
    tam=i-1;                    /* tamaño del arreglo a almacenar          */
    stop_INT();
    mensajes_R();   /* mandando el ultimo valor a pantalla  */
    getch();                    /* pausa para continuar y grabar datos     */

    if (tam > 20)
           {
            clrscr();
            gotoxy(12,6); printf(" Deseas grabar los datos en archivo [s/n] ? ");
            do
             {
              que=getch();
             } while (que!= 'n' && que!= 'N' && que!= 's' && que!= 'S' );

            if (que == 's' || que == 'S')
             {
              lee_name();
              escr_arch(name);
             }
           }
    clrscr();
    gotoxy(18,6); printf(" Deseas realizar otro experimento [s/n] ? ");
    do
            {
             que=getch();
            } while (que!= 'n' && que!= 'N' && que!= 's' && que!= 'S' );

    } while (que == 's' || que == 'S');

  gotoxy(20,24); printf(" \" FIN DEL PROGRAMA \" ");
/*  getch();            */
  clrscr();
}
/* ************************** fin de main ***************************** */

void escalon()
{
            char *escalo_n;

            printf("\n\n Que valor del escalón para impulsar el carro [v] \n\n");
            printf("\n\n \t intervalo de (0 . a . 3) [v]  \n Uo = ");
            gets(escalo_n);
            fza=atof(escalo_n);
            clrscr();
}


void verif_pos()
{
   do
    {
            lect_in();  /* lectura inicial para conocer posición inicial */
```

```
            X11= fabs(X1/n11);  /* valor absoluto de la posición en [m]  */
            mensaje1_R();

            if ( X11 < 0.5 )
             {
              gotoxy(6,9);
              printf(" Mover el carro a cualquiera de los extremos [izq] o [der]");
             }
     } while (X11 < 0.5 );
     printf("\a");

     if (X1 > 0 )
            fza *= -1;
     delay(3000);
     printf("\a\a");
     lect_in();          /* 2a lect para verificar la posición inicial    */
     Xo=X1;                      /* salvando la posición inicial del carro */
     if (Xo > 0.0)
            signo=-1;/* signo contrario ala posición inicial para poder */
     if (Xo< 0.0)       /* distinguir la dirección ala que se dirige.     */
            signo=1;
}


void mensaje1_R()
{
   gotoxy(6,2);  printf("Escalón para excitar el carro = %g [V]",fza);
   gotoxy(37,6); printf("                                    ");
   gotoxy(6,6);
   printf(" posición inicial del carro = %g [V]=> %f [m]",X1,X1/n11);
}


void mensajes_R()
{
   gotoxy(29,8); printf("                          ");
   gotoxy(6,8); printf(" posición del carro = %g [V]=> %f [m]",X1,X1/n11);

   gotoxy(29,9); printf("                          ");
   gotoxy(6,9);
   printf(" velocidad del carro = %g [V]=> %f [m/s]",X21,X21/n33);

   gotoxy(23,11); printf("    ");
   gotoxy(15,11); printf("activo = %d ",activo);

   gotoxy(40,12); printf("    ");
   gotoxy(15,12); printf("No de muestras leídas = %d ",i);
}

/* ================================================================== */
/* ************** rutina que utiliza a las lecturas ******************** */

void control_r()
{
          /* lectura del A/D_0    */
  X1 = (dig_volt(lee_AD('0'))) + 0.145;
  X13[i]=X1;

          /* lectura del A/D_2     */
  X21 = (dig_volt(lee_AD('2'))) + 0.088;
  X2[i++]=X21;              /* almacenando la lectura en un vector */

  limite = 0.4*signo;
  X11=X1/n11;

/* ***************** condiciones para una Xo negativa ********************** */
  if (signo == 1)
    {
```

```
            if ( X11 >= limite )
             {
              fza=0.0;
              activo=0;
             }
            else
              activo=1;
    }

/* ***************** condiciones para una Xo positiva ********************** */
  if (signo == -1)
    {
            if ( X11 <= limite )
             {
              fza=0.0;
              activo=0;
             }
            else
              activo=1;
    }

/* ****************** Salida del ESCALÓN por el D/A_0 ********************** */

                    /* 0.014 cte. de error del DAC_0 de salida */
  ESC[i]=fza;
  escribe_DA(AN_dig(fza+0.014),'0');        /*  mandando el escalón   */
}


/* ------------------------------------------------------------------- */
          /* === Inicializando interrupciones === */

void start_INT_R( unsigned int t_INT )
{
 DISABLE;              /* deshabilitando INT de Hardware          */

 gettime( &timesist );     /* guardando la hora del sistema en time_sist */
 old_INT = getvect(8);     /* salvando el vector de la INT(8)         */
 prog_timer( t_INT );      /* programando el timer con el T de muestreo  */

 /* variable "t_INT", que contendra el tiempo de muestreo [1..54] en ms */
 setvect( 8, Rut_INT_R );     /* modificando el vector de la INT-8 para
                                        cargar la rutina de servicio a la INT   */
 ENABLE;               /* activando INT de Hardware             */
}

          /* === Rutina de atención ala interrupción (R/W) === */

void interrupt Rut_INT_R( )
{

 cont3++;                        /* 1er contador para los pulsos para el actuador */
 if (actu2 == 1 )        /* comienzo del contador si el actuador esta activo. */
            cont2++;     /* contador para el periodo de muestreo  */

 if (cont2 == Tm)      /* Tm : es el periodo de muestreo              */
  {
   cont2=0;
   control_r();
  }
/* ================== ========== ========== ================== */

 if(cont3 >= 3)
 {
  cont3=0;
  cont++;          /* incrementando el 2o contador para pulsos  */
  if (cont==10)     /* ( cada 10 veces se repite ) = T         */
    cont=0;
```

```
 pulsos();          /* manda pulsos por D1 y D2 en 10 tiempos   */
 }
/* ========== ========== ========== ========== ========== ========== */
 eof_INT();
}



/* ----------[ R_CARR2.H ]---------- */
#include <math.h>
#include <bios.h>
#include <conio.h>
#include <stdio.h>

void r_carr2();                    /* rutina principal = main                    */

/* ********** DECLARACIÓN DE RUTINAS DEL PROGRAMA ********** */
void carga_vectores();             /* carga archivos de vectores y matrices   */
void verif_pos_carr();             /* verifica posición inicial del carro     */
void destino_c();                  /* filtrado para las muestras leidas       */
void tipo_control();               /* selección del tipo de control.          */
void tipo_de_conpt();              /* cácula el tipo de compensación          */
void grabamos();                   /* pregunta si se graban los datos en archivo */
void controla();                   /* programa que realiza el control del péndulo */
void ver_signo();                  /* verificando el signo de la velocidad    */
void resta_mat();                  /* minimiza cálculos con matrices constantes */
void interrupt Rut_INT3( );        /* rutina de atención a la interrupción    */
void start_INT3( unsigned int t_INT );  /* rutina para inicializar la INT     */
void mensaje_p();                  /* despliegue de X1, X2, X3, U ,Uso  y error */
void mensajes();                   /* despliegue de polos y vectores de K y L  */

/*     ********** PROGRAMA PRINCIPAL **********      */
void r_carr2()
{
 unsigned int tiempo;
 char que,tecla=' ';

 clrscr();
 gotoxy(15,10); printf(" \"C O N T R O L   D E L   C A R R O\" ");
 gotoxy(15,14); printf("UTILIZANDO DIFERENTES MÉTODOS DE COMPENSACIÓN ");
 delay(1500);

 /* ========== lectura de archivos con datos  calculados extemamente ====== */

 carga_vectores();          /* carga archivos de vectores y matrices   */

 /* ========== Inicio del programa de control ========== */
 do
  {
   tipo_control();                  /* seleccionando tipo de compensación del carro  */
   verif_pos_carr();                /* verifica. y almacena. posición inicial del carro  */
   inicia_vark();                   /* inicializa variables para almacenar estados K=K-1  */
   destino_c();                     /* pide una nueva posición para realizar el control. */
   tiempo=1;                        /* Tint=1 porque la cuenta es interna con cont2  */
   archiva=0;                       /* bandera que indica que no se graban los datos  */
   grabamos();                      /* pregunta. si se almacenan los datos en archivo  */
   inicializando_var();             /* inicializando variables y vectores globales  */

   /* ========== inicializando INTerrupciones ========== */
   start_INT3(tiempo);              /* inicializando interrupciones     */
   verif_actu();     /* rutina para verificar que el actuador se active */
   mensaje_p();         /* despliegue de polos y vectores de K y L    */
   activo=1;
   tecla=Tip_c;
   do
    {
     switch (tecla)
           {
```

```
        case '+': if (destino<0.49)
                    destino+=0.01;
                  else printf("\a\a");      /* indicación limites de posición */
                  break;
        case '-': if (destino>-0.49)
                    destino-=0.01;
                  else printf("\a\a");   /* indicación limites de posición */
                  break;
        case ' ': destino=0.0;
                  break;
        case 'd':
        case 'D': clrscr();
                  destino_c(); /* pide una nueva posición para el control.  */
                  mensaje_p();          /* despliegue de polos y vectores de K y L */
                  break;
        case 'g':
        case 'G': graf_c(); mensaje_p();  /* llama a graf_c();    */
                  break;
        case '1':
        case '2':
        case '3':
        case '4': Tip_c=tecla;
                  break;
        case 'a':
        case 'A': archiva=1;        /* activa modo de almacenar arreglos*/
                  break;
        case 'w':
        case 'W': clrscr();
                  destino_c(); /* pide una nueva posición para el control.  */
                  archiva=1;                        /* activa modo de almacenar arreglos */
                  mensaje_p();/* despliegue de polos y vectores de K y L */
                  break;
        }

 do
        {
         mensajes();
         if ( activo == 0 )
           {
            stop_INT();       /* Deteniendo la interrupción */
            goto FIN_M;
           }
        } while (!kbhit());
 tecla=getch();
 } while(tecla!='*');

FIN_M:
printf("\a\a\a");
gotoxy(5,22); printf("                                    \" FIN DEL PROCESO \"
gotoxy(5,23); printf("          (para continuar oprima cualquier tecla)             ");
stop_INT();
getch();

if (tam>20)
 if (archiva==1)
        {
         lee_name();                    /* pide nombre para grabar datos en archivo */
         escr_arch(name);     /* escribiendo datos en un archivo              */
        }

clrscr();
gotoxy(12,6); printf(" Deseas realizar otro experimento [s/n] ? ");
 do
  {
        que=getch();
  } while (que!= 'n' && que!= 'N' && que!= 's' && que!= 'S' );

} while (que == 's' || que == 'S');
```

```
gotoxy(20,23); printf(" \" FIN DEL PROGRAMA DE COMPENSACIÓN\"    ");
delay(1000);
clrscr();
}
/* ************************** fin de main ************************** */

void carga_vectores()
{
carga_vect_K();         /* carga el vector de ganancias K          */
carga_vect_L();         /* carga el vector de estimación L         */
carga_pol_c();          /* carga polos del sistema                 */
carga_pol_L();          /* carga polos del sistema                 */
carga_AB();                     /* carga matrices del subsistema           */
resta_mat();                    /* minimizando operaciones al estimar estados */
pend=10.0/200.0;
}


void tipo_control()
{
 int op_1;

   clrscr();
   vent_P();
   gotoxy(15,5); printf("   \"TIPO DE CONTROL PARA EL CARRO\" ");
   gotoxy(15,8); printf("          U = Uo + Uso  ");
   gotoxy(15,10); printf(" Donde :  Uo = -(X1*K1 + X2*K2) ");
   gotoxy(15,13); printf("1) - Sin compensación ..... [ Uso = 0 ]");
   gotoxy(15,14); printf("2) - Compensación cte ..... [ Uso = 1.85*sig(X2) ]");
   gotoxy(15,15); printf("3) - Compensación estatica . [ Uso = (M*X2 + Bo)*sig(X2) ]");
   gotoxy(15,16); printf("4) - Compensación e.observ . [ Uso = X3 ]");
   gotoxy(10,22); printf("seleccione una opción : ");

   do {
           op_1 = getch();
           Tip_c = op_1;           /* variable que indica el tipo de compensación */
           op_1 -= 48;
           } while (op_1<1 || op_1>4);
}


void verif_pos_carr()
{
  clrscr();
  vent_P();
  do
   {
          lect_in();  /* lectura inicial para conoser posición inicial  */
          X11 = fabs(X1/n11);  /* valor absoluto de la posición en [m]  */
          gotoxy(37,6); printf("                          ");
          gotoxy(6,6);printf(" posición inicial del carro = %g [cm]",X1*100.0/n11);
          if ( X11 > 0.5 )
            {
            gotoxy(6,9);
            printf(" Mover el carro dentro de los limites (+/- 50) [cm] ");
            }
   } while (X11 > 0.5 );

/*    printf("\a");*/   /* sonido de indicación de que esta en posición */
/*    delay(3000); */ /* retraso de tiempo para acomodar el carro    */
/*    printf("\a\a"); */              /* 2o sonido para indicar inicio */
   lect_in();          /* 2a lect para verificar la posición inicial   */
   Xo=X1;                          /* salvando la posición inicial del carro */
}


void destino_c()
{
```

```
 char *posi;

 vent_P();
 gotoxy(6,11);printf("La actual posición es = %g [cm]",X1*100.0/n11);
 gotoxy(5,22);  printf(" El control para que nueva posición del carro ");
 gotoxy(5,23);  printf("  'X_destino' en [cm] ? ");
 do
  {
   gets(posi);
   destino=atof(posi);
   if (fabs(destino) >50) printf("\n\a Posición NO permitida \" fuera del limite \" \a");
  } while(fabs(destino) > 50);    /* para que este dentro de +/- 50 */
 destino /= 100.0;                                  *   pasando de cm a metros       */
}


void grabamos()
{
  char que;

            clrscr();
            vent_P();
            gotoxy(5,22); printf(" Deseas grabar los datos en archivo [s/n] ? ");
            do
             {
             que=getch();
             } while (que!= 'n' && que!= 'N' && que!= 's' && que!= 'S' );
            archiva=0;
            if (que == 's' || que == 'S')
             {
             archiva=1;
             i=0;          /* inicializando subindice para muestras     */
             tam=0;       /* inicializando variable que almacena la longitud de las mustras*/
             gotoxy(10,10);
             printf(" \"NOTA\" : Se tienen unicamente 10 segundos de control");
             gotoxy(21,11); printf("El sonido indicara el inicio y el final ");
             }
}


void mensaje_p()
{
  clrscr();
  vent_P();
  gotoxy(5,22);
  printf("[D]=dest  [A]=alm  [G]=graf  [+]=->  [-]=<-  [space]=0  [*]=FIN");
  gotoxy(5,23); printf("[w]=dest+alm");
  gotoxy(10,3); printf(" Polos de la ley de control :        Psk = [ %+2.1f , %+2.1f ] ",Pol_C[1],Pol_C[2]);
  gotoxy(10,4); printf(" Polos del observador :              PsL = [ %+2.1f , %+2.1f ] ",Pol_L[1],Pol_L[2]);
  gotoxy(40,5); printf("       K = [ %+5.3f , %+5.3f ] ",K[1],K[2]);
  gotoxy(40,6); printf("       L = [ %+5.3f , %+5.3f ] ",L[1],L[2]);
}


void mensajes()
{
   if ( (fabs(UU)) >= 9.9 )
            printf("\a");
   gotoxy(27,7); printf("                ");
   gotoxy(6,7); printf(" destino del carro = %+5.3f [cm]",destino*100.0);

   gotoxy(27,8); printf("                ");
   gotoxy(6,8); printf(" error de posición = %+5.3f [cm]",error*100.0/n11);

   gotoxy(19,10); printf("                ");
   gotoxy(6,10); printf(" posición  = %+5.3f [cm] ",X1*100.0/n11);

   gotoxy(19,11); printf("                                                                ");
```

```
   gotoxy(6,11); printf(" velocidad = %+5.3f [m/s]",X21/n33);
   gotoxy(48,11); printf(" Vel^ = %+5.3f [m/s]",XX2k/n33);

   gotoxy(13,12); printf("              ");
   gotoxy(6,12);  printf(" Uo  = %+5.3f [V]",UU2);
   gotoxy(13,13); printf("                                                                    ");
   gotoxy(6,13);  printf(" Uso = %+5.3f [V] ",Uso);
   gotoxy(48,13); printf(" X3^ = %+5.3f [V]",X3k);

   gotoxy(13,14); printf("                 ");
   gotoxy(6,14);  printf(" Us  = %+5.3f [V]",UU);
   gotoxy(6,16);  printf("                                                                                    ");
   gotoxy(6,16);  printf(" Tipo.conp.= %c \t activo = %d \t signo = %d \t almacenar = %d ",Tip_c,activo,signo,archiva);
   gotoxy(6,18);  printf(" No de lecturas = %d",i);
}


void ver_signo()       /* verificando el signo de la velocidad                    */
{
  if ( X21 >= delta)
     signo = 1;

  if ( X21 <= (-delta))
     signo = -1;

/*  if ( X21 < delta && X21 > (-delta) )      */
/*    if ( fabs(X21) < delta )                */
/*     signo = 0;                   */
}


void resta_mat()
{
 AL1=A2[1][1]-L[1];
 AL2=A2[2][1]-L[2];
 A12=A2[1][2];
 A22=A2[2][2];
}


void tipo_de_conpt()
{
  ver_signo();                    /* verificando el signo de la velocidad   */

  switch (Tip_c)
            {
             case '1' : Uso=0;
                        break;
             case '2' : Uso=-1.85*signo;
                        break;
             case '3' : Uso=(0.26*X21 - 1.86*signo);
                        break;
             case '4' : XX2k1= AL1*XX2k + A12*X3k + B2[1]*UU3 + L[1]*X21;
                        X3k1 = AL2*XX2k + A22*X3k + B2[2]*UU3 + L[2]*X21;
                        XX2k = XX2k1;  /* salvando estados estimados */
                        X3k  = X3k1;       /* salvando estados estimados */
                        Uso  = -X3k1;
                        break;
            }
            if(Tip_c != '4')
              {
               XX2k=0;
               X3k=0;
               UU3=0;
              }
}

/* ************** rutina que utiliza a las matrices ********************** */
```

```
void controla()
{
 float XX1;

          /* lectura del A/D_0     posición */
 X1 = (dig_volt(lee_AD('0'))) + 0.054;                    /*          -7.55; = 50 cm          */

          /* lectura del A/D_2     velocidad              */
 X21 = (dig_volt(lee_AD('2'))) + 0.09371;

 error=(X1 - (destino)*n11);   /* error de voltajes prop. a la posición */

 XX1=(X1/n11);                    /* condición de seguridad para el Péndulo  */
 if ( XX1 < 0.5 && XX1 > (-0.5) )
  {
   /* ****  calculo de la Fuerza de control ******* */
   /*          Uk = -K*X(k);            */
   /*          X = [ Posicion; Veloc]     */
   /* ****************************************** */

   UU2 = - (K[1]*error + K[2]*X21);  /* calculo de la Fuerza de control Uo */
   tipo_de_conpt();              /* cácula. el tipo de compensación          */
   UU = UU2 + Uso;               /* cácula. el control total del carro       */
   activo=1;
  }
 else
  {
  UU=0.0;
  activo = 0;
  }

 if ( (fabs(UU)) >= 10 )     UU=9.99*UU/fabs(UU);

MANDA:
 /* 0.014 constante de error del DAC_0 de salida */
 escribe_DA(AN_dig(UU),'0');  /* mandando señal de control   */

 rampa=pend*equis++;
 escribe_DA(AN_dig(rampa),'1');  /* mandando una rampa por el D/A_1   */
/*  escribe_DA(AN_dig(Uso),'1');            */

 if (equis>=200)                  /* son casi 6 segundos ya que 200*0.03=5.97*/
   equis=0;

 if (archiva == 1)
          {
          X13[i]=X1/n11;
          X2[i]=X21/n33;
          ESC[i]=UU;
          US0[i]=Uso;
          X2e[i]=XX2k1/n33;
          i++;
          if (i==150)
            {
            activo = 0;
            tam=i-1;
            }
          }

 UU3=UU;                          /* salvando Uk para usarla en el tiempo K+1          */
}

          /* === Inicializando interrupciones === */

void start_INT3( unsigned int t_INT )
{
 DISABLE;              /* deshabilitando INT de Hardware          */
```

```
 gettime( &timesist );      /* guardando la hora del sistema en time_sist */
 old_INT = getvect(8);      /* salvando el vector de la INT(8)            */
 prog_timer( t_INT );       /* programando el timer con el T de muestreo  */
 /* variable " t_INT ", que contendra el  tiempo de muestreo [1..54] en ms */
 setvect( 8, Rut_INT3 );    /* modificando el vector de la INT-8 para
                                          cargar la rutina de servicio a la INT   */
 ENABLE;                    /* activando INT de Hardware                  */
}

          /* === Rutina de atención ala interrupción (R/W) === */

void interrupt Rut_INT3( )
{
 cont3++;      /* 1er contador para los pulsos para el actuador */
 if (actu2 == 1 )        /* comienzo del contador si el actuador esta activo. */
            cont2++;     /* contador para el periodo de muestreo  */
 if (cont2 == Tm)
  {
   cont2=0;
   controla();
  }
/* ========================== ============ ============== ================== */
 if(cont3 >= 3)
  {
  cont3=0;
  cont++;           /* incrementando el 2o contador para pulsos  */
  if (cont==10)     /* ( cada 10 veces se repite ) = T           */
    cont=0;
  pulsos();         /* manda pulsos por D1 y D2 en 10 tiempos    */
  }
/* ========================== ============ ============== ================== */
 eof_INT();
}



/* ------------------------------[ RPEND1.H ]------------------------------ */
/* === programa que estima cinco variables pero con modificaciones de error === */
#include <math.h>
#include <bios.h>
#include <conio.h>
#include <stdio.h>

#include "graf_pe2.h"   /* para graficar estados del péndulo                  */
#include "pend2.h"      /* rutinas. para el estimador reducido de 2 variables.  */

/* ***************** DECLARACIÓN DE RUTINAS DEL PROGRAMA ******************** */

void r_pend1();                         /* rutina principal = main                    */
void controla_Pend();           /* programa que realiza el control del péndulo */
void interrupt Rut_INT_P( );            /* rutina de atención a la interrupción    */
void start_INT_P( unsigned int t_INT );  /* rutina para inicializar la INT  */
void mensaje_p_Pend();                  /* despliegue de X1, X2, X3, X4 U ,Uso     */
void mensajes_Pend();                   /* despliegue de polos y vectores de K y L */
void estima_edos();                     /* rutinaa. que elige el estimador de 2 o 5 estados */
void estima_edos_p5();                  /* rutinaa. que estima estados y cácula ley de control */
void limp_arr_edos();           /* rutinaa. que limpia arreglos para estimar estados*/
void limp_arr_edos_p5();                /* rutinaa. que limpia arreglos para 5 o 2 edos*/
void imp_edos_Pend();                   /* despliega los estados reales del péndulo  */
void lect_edos();                       /* para leer de los conv.A/D  X1,X2,X3    */
void verif_edos_Pend();                 /* para verificar la posición inicial del p*/
void imp_estimados();                   /* despliega los estados estimados del péndulo*/
void dif_estados();                     /* despliega el error entre estados reales */
                                        /* y estados estimados                */
void almacena_T_edos();                  /* rutina para almacenar estados, reales y estimados, */

/* ********************* PROGRAMA PRINCIPAL ******************************** */
void r_pend1()
```

```
{
unsigned int tiempo;
char que,tecla=' ';
double dest;

clrscr();
gotoxy(15,10); printf(" \" C O N T R O L  D E L  P E N D U L O \" ");
gotoxy(15,12); printf("       \" I N V E R T I D O \" ");
gotoxy(15,15); printf("PROGRAMADOR: Luis Julian Ramos Cruz ");
gotoxy(15,17); printf("CARRERA: Ing. Mec. Elect.(Área:  de Electrónica )");
gotoxy(15,19); printf("N.C.: 8638341-9              en Comunicaciones");
gotoxy(15,21); printf("                        [ LJRC ]");
 delay(3000);

menu_pendulo();     /* selección del tipo de observador (de 5 o 2 estados) */
/* op_men2='2';      */
/* ========= lectura de archivos con datos  calculados externamente ====== */
if (op_men2 == '0') goto FIN_P;
lee_mat_sist();          /* lectura de vectores y matrices del péndulo*/

/* ================== Inicio del programa de control ===================== */
do
 {
  verif_edos_Pend(); /* para verificar edos X1,X2,X3 del péndulo*/
  limp_arr_edos();   /* inicializa variables para almacenar estados K=K-1     */
  destino_c();       /* pide una nueva posición para realizar el control. */
  tiempo=1;          /* Tint=1 porque la cuenta es interna con cont2  */
  archiva=0;           /* bandera que indica que no se graban los datos  */
  grabamos();          /* pregunta. si se almacenan los datos en archivo    */

  dest=destino;
  /* ================ inicializando INTerrupciones ================== */
  inicializando_var(); /* inicializando variables y vectores globales   */
  start_INT_P(tiempo);              /* inicializando interrupciones      */
  verif_actu();      /* rutina para verificar que el actuador se active */
  destino=dest;

  /* ------------- inicia secuencia usando interrupciones---------- */
  mensaje_p_Pend();                 /* despliegue de polos y vectores de K     */
  activo=1;
  tecla='0';
  do
   {
    switch (tecla)
          {
          case '+': if (destino<0.48)
                       destino+=0.02;
                    else printf("\a\a");      /* indicación limites de posición */
                    break;
          case '-': if (destino>-0.48)
                       destino-=0.02;
                    else printf("\a\a");   /* indicación limites de posición */
                    break;
          case ' ': destino=0.0;
                    break;
          case 'd':
          case 'D': clrscr();
                    destino_c(); /* pide una nueva posición para el control.   */
                    mensaje_p_Pend();/* despliegue de polos y vectores de K y L */
                    break;
          case 'g':
          case 'G': graf_Pend(); mensaje_p_Pend();   /* llama a graf_c();    */
                    break;
          case 'a':
          case 'A': archiva=1;                 /* activa modo de almacenar arreglos*/
                    break;
          case 'w':
          case 'W': clrscr();
```

```
                    destino_c(); /* pide una nueva posición para el control.  */
                    archiva=1;                          /* activa modo de almacenar arreglos*/
                    mensaje_p_Pend();/* despliegue de polos y vectores de K y L */
                    break;
            }

   do
            {
            mensajes_Pend();
            if ( activo == 0 )
              {
              stop_INT();       /* Deteniendo la interrupción */
              goto FIN_M;
              }
            } while (!kbhit());
    tecla=getch();
   } while(tecla != '*');

  FIN_M:
  printf("\a\a\a");
  gotoxy(5,22);
  printf("                          \" FIN DEL PROCESO \"                                    ");
  gotoxy(5,23);
  printf("              (para continuar oprima cualquier tecla)       ");
  stop_INT();
  getch();

  if (tam>20)
   if (archiva==1)
            {
            lee_name();                         /* pide nombre para grabar datos en archivo */
            escr_arch(name);    /* escribiendo datos en un archivo              */
            }

  clrscr();
  gotoxy(12,6); printf(" Deseas realizar otro experimento [s/n] ? ");
   do
    {
          que=getch();
    } while (que!= 'n' && que!= 'N' && que!= 's' && que!= 'S' );

  } while (que == 's' || que == 'S');
 FIN_P:
 gotoxy(20,23); printf(" \" FIN DEL PROGRAMA DE COMPENSACIÓN\" ");
 delay(1500);
 clrscr();
 }
 /* ************************** fin de main ****************************** */


 void mensaje_p_Pend()
 {
   clrscr();
   vent_P();
   gotoxy(5,22);
   printf("[D]=dest  [A]=alm  [G]=graf  [+]=->  [-]=<-  [space]=0  [*]=FIN");
   gotoxy(5,23); printf("[w]=dest+alm");
   gotoxy(5,3);
   printf(" Psk = [ %2.2f , %2.2f , %2.2f , %2.2f ] ",Pol_C[1],Pol_C[2],Pol_C[3],Pol_C[4]);
   gotoxy(5,4);
   printf(" PsL = [ %3.0f , %3.0f , %3.0f , %3.0f , %3.0f ] ",Pol_L[1],Pol_L[2],Pol_L[3],Pol_L[4],Pol_L[5]);
   printf(" K = [ %4.2E , %4.2E , %4.2E , %4.2E ] ",KP[1],KP[2],KP[3],KP[4]);
 }


 void imp_edos_Pend()
 {
```

```
if ( activo == 1)
  {
  gotoxy(11,7); printf("              ");
  gotoxy(6,7); printf(" Xd = %+5.3f [cm]",destino*100.0);
  gotoxy(11,8); printf("              ");
  gotoxy(6,8); printf(" err = %+5.3f [cm]",error*100.0/n11);
  }

  gotoxy(11,9); printf("              ");
  gotoxy(6,9);  printf(" X1  = %+5.3f [cm] ",X1*100.0/n11);
  gotoxy(11,10); printf("             ");
  gotoxy(6,10);  printf(" X2  = %+5.3f [G] ",X31);
  gotoxy(11,11); printf("             ");
  gotoxy(6,11);  printf(" X3  = %+5.3f [m/s]",X21/n33);
}


void verif_edos_Pend()
{
 clrscr();
 vent_P();
 activo=0;
 do
  {
   gotoxy(15,5);
   printf("Loop para verificar posición inicial del péndulo ");
   lect_edos();          /* lectura de los estados de los A/D          */
   imp_edos_Pend(); /* despliegue de los estados                       */
   gotoxy(10,22);
   printf("para continuar oprima cualquier [TECLA] ");
  } while (!kbhit());
 clrscr();
}


void mensajes_Pend()
{
   if ( (fabs(UU)) >= 9.9 )
            printf("\a");
   imp_edos_Pend(); /* despliega el valor de X1,X2,X3 del péndulo      */
   imp_estimados(); /* desp. edos X1^,X2^,X3^,X4^,X5^ del péndulo      */
   gotoxy(13,13); printf("                          ");
   gotoxy(6,13);  printf(" Uo  = %+5.3f [V]\t",UU2);
   gotoxy(13,14); printf("                          ");
   gotoxy(6,14);  printf(" Uso = %+5.3f ]V] ",Uso);
   gotoxy(13,15); printf("                          ");
   gotoxy(6,15);  printf(" Us  = %+5.3f [V]",UU);
   gotoxy(6,18);  printf(" activo  = %d ",activo);
   gotoxy(6,19);  printf(" No.mues = %d",i);
}


void imp_estimados()
{

if (op_men2 == '1')
  {
  gotoxy(35,9); printf("                          ");
  gotoxy(35,9);  printf(" X1^ = %+5.3f [cm] ",XX_k1[1]*100.0/n11);
  gotoxy(35,10); printf("                          ");
  gotoxy(35,10); printf(" X2^ = %+5.3f [G] ",XX_k1[2]);
  gotoxy(35,11); printf("                          ");
  gotoxy(35,11); printf(" X3^ = %+5.3f [m/s]",XX_k1[3]/n33);
  }
  gotoxy(35,12); printf("                          ");
  gotoxy(35,12); printf(" X4^ = %+5.3f [rad/s]",XX_k1[4]);
   gotoxy(35,13); printf(" X5^ = %+5.3f [V]",XX_k1[5]);
}
```

```
void dif_estados()     /* imprime la diferencia entre estados y estados. estimados        */
{
  gotoxy(35,16); printf("                                    ");
  gotoxy(35,16); printf("X1-X1^= %+3.2f [cm] ",(X1-XX_k1[1])*100.0/n11);
  gotoxy(35,17); printf("                                    ");
  gotoxy(35,17); printf("X2-X2^= %+3.2f [G] ",X31-XX_k1[2]);
  gotoxy(35,18); printf("                                    ");
  gotoxy(35,18); printf("X3-X3^= %+3.2f [m/s]",(X21-XX_k1[3])/n33);
}


void lect_edos()
{
/* ============== lectura de estados de los convertidores A/D ============== */
          /* lectura del A/D_0     posición lineal        */
 X1 = (dig_volt(lee_AD('0'))) + 0.054;                    /* -7.55; = 50 cm  */

          /* lectura del A/D_1     posición angular       */
 X31 = (dig_volt(lee_AD('1'))) - 0.412+0.0848-0.1366; /* 0.4 : error de lectura */

          /* lectura del A/D_2     velocidad              */
 X21 = (dig_volt(lee_AD('2'))) + 0.09371+0.0007;

/*   almacenando las lecturas en un vector para poder estimar estados          */
 XX[1]=X1;     /* posición      */
 XX[2]=X31;          /* ángulo */
 XX[3]=X21;          /* velocidad angular.             */
}


void limp_arr_edos_p5()
{
 for (i=1;i<=5;i++)
    XX_k[i]=0.0;

 for (i=1;i<=3;i++)
    XX[i]=0.0;

 XX2[1]=X1;          /* inicializando estados Xk para poder estimar          */
 XX2[2]=X31;         /* esto es antes de empezar el control                  */
 XX2[3]=X21;

 UU3=0.0;                    /* inicializando Uk+1 =0                        */
 pend=10.0/200.0;    /* pendiente para generar la rampa          */
 UU2=0.0;                    /* inicializando Uk =0                                  */
}


void limp_arr_edos()
{
 if (op_men2 == '1') limp_arr_edos_p5();
 if (op_men2 == '2') limp_arr_edos_p2();
}


void estima_edos()
{
 if (op_men2 == '1')
    estima_edos_p5();

 if (op_men2 == '2')
    estima_edos_p2();
}


void estima_edos_p5()
```

```
{
int ii;
 /* ***********  Calculando el estimador.  ************ */
                    /* A*X^k + B*Uo + L(Xk-X^k) */
                    /* ------------------------ */
                    /* XX_k1(1)=posición lin.  */
                    /* XX_k1(2)=ángulo;        */
                    /* XX_k1(3)=velocidad lin. */
                    /* XX_k1(4)=vel.angular.   */
                    /* XX_k1(5)=Fricción est.  */
                    /* ----------------------- */

for (ii=1; ii<=5; ii++)
{
 XX_k1[ii] = Ad[ii][1]*XX_k[1] + Ad[ii][2]*XX_k[2] + Ad[ii][3]*XX_k[3]
           + Ad[ii][4]*XX_k[4] + Ad[ii][5]*XX_k[5] + Bd[ii]*UU3
           + LL[ii][1]*XX[1] + LL[ii][2]*XX[2] + LL[ii][3]*XX[3];
}

Uso=-XX_k1[5];                  /* identificando el estado X5 como Uso     */

 /* ***********  calculando la ley de control ************ */
                        /* U=-K*Xk         */
error=(XX[1] - (destino)*n11);   /* error de voltajes prop. a la posición */
UU2=-(KP[1]*error + KP[2]*XX[2] + KP[3]*XX[3] + KP[4]*XX_k1[4]);

UU = UU2 + Uso;  /*  Uso; cácula. el control total del carro   */

for (ii=1;ii<=5;ii++)
   XX_k[ii]=XX_k1[ii];          /* salvando estados de k+1 en estados de k */

UU3=UU;                         /* salvando Uk+1 en Uk          */
}

/* *************** rutina que utiliza a las matrices ********************** */

void controla_Pend()
{
 float XX1;                     /* para posición lineal en [cm]             */

/* ******************* inicio del algoritmo de control ************************** */
 lect_edos();                   /* leyendo estados X1,X2,X3 de los A/D     */

/* ============================================================================== */
 /* condición de seguridad para el Péndulo ( posición y ángulo) */
 XX1=(X1/n11);                  /* convirtiendo la posición de [V] a [m]*/
 if (fabs(XX1) < 0.51  &&  fabs(X31) < 9.99 )
  {
          /* ***********       calculando el estimador          ************      */
   estima_edos();  /* estimando estados y calculando la ley de control  */
   activo=1;
  }
 else
  {
   UU=0.0;
   activo = 0;
  }

 if ( (fabs(UU)) >= 10 )     UU=9.99*UU/fabs(UU);

MANDA:
 /* 0.014 constante de error del DAC_0 de salida */
  escribe_DA(AN_dig(UU),'0');  /* mandando señal de control   */

 /* ****************** fin del algoritmo de control ********************** */

 /* ========== sección para almacenar estados en un arreglo ============== */
  if (archiva == 1)
```

```
            {
            almacena_T_edos();
            if (i<=200)
              i++;
            if (i==200)
              {
               tam=i-1;
              }
            }

/* ========== sección para mandar una rampa por el D/A_1 ========== */
 rampa=pend*equis++;
 escribe_DA(AN_dig(rampa),'1');  /* mandando una rampa por el D/A_1   */
 if (equis>=200)                 /* son casi 6 segundos ya que 200*0.03=5.97*/
   equis=0;
}


void almacena_T_edos()
{
/* ************** arreglos para almacenar estados en archivo ************* */
   X13[i]=X1/n11;     /* vector de la posición */
   X2[i]=X21/n33;     /* vector de la velocidad del carro  */
   ÁNGULO[i]=X31;     /* vector para el ángulo del péndulo  */
   ESC[i]=UU;              /* U de control para el carro */
/*  ************** arreglos para almacenar estados estimados ************** */

 if (op_men2 == '1')
   US0[i]=XX_k1[5];              /* para almacenar Uso estimada =X^5          */

 if (op_men2 == '2')
   US0[i]=XX_k[5];   /* para almacenar Uso estimada =X^5          */

 if (op_men2 == '1')
  {
   X2e[i]=XX_k1[3];   /* para almacenar velocidad estimada         */

   X1e[i]=XX_k1[1];  /* para almacenar posición estimada          */
   X3e[i]=XX_k1[3];  /* para almacenar ángulo estimado            */
   X4e[i]=XX_k1[4];  /* para almacenar vel.ang.estimada           */
  }
}

          /* === Inicializando interrupciones === */

void start_INT_P( unsigned int t_INT )
{
 DISABLE;               /* deshabilitando INT de Hardware           */
 gettime( &timesist );     /* guardando la hora del sistema en time_sist */
 old_INT = getvect(8);     /* salvando el vector de la INT(8)          */
 prog_timer( t_INT );      /* programando el timer con el T de muestreo  */
 /* variable " t_INT ", que contendra el  tiempo de muestreo [1..54] en ms */
 setvect( 8, Rut_INT_P );      /* modificando el vector de la INT-8 para
                                          cargar la rutina de servicio a la INT   */
 ENABLE;                /* activando INT de Hardware                */
}


          /* === Rutina de atención ala interrupción (R/W) === */
void interrupt Rut_INT_P( )
{
 cont3++;      /*  1er contador para los pulsos para el actuador */
 if (actu2 == 1 )       /* comienzo del contador si el actuador esta activo. */
            cont2++;     /* contador para el periodo de muestreo  */

 if (cont2 == Tm)
  {
   cont2=0;
```

```
    controla_Pend();
   }
/* ================== ========== ========== ========== */

 if(cont3 >= 3)
  {
  cont3 =0;
  cont++;            /* incrementando el 2o contador para pulsos  */
  if (cont==10)      /* ( cada 10 veces se repite ) = T          */
    cont=0;
  pulsos();          /* manda pulsos por D1 y D2 en 10 tiempos    */
  }
/* ================== ========== ========== ========== */
 eof_INT();
}
```

Prototipo de laboratorio

Péndulo Invertido

Sistema de control para el péndulo Invertido mediante el uso del paquete PCPEIN

Objetivo: Desarrollar algoritmos inteligentes para el control de posición y la detección de fallas de un servomecanismo, el cual tiene un comportamiento dinámico semejante al de una grúa de carga y al de un proyectil durante la etapa de lanzamiento.

Instituto de Ingeniería