Proyecto Integrador Profesional

Diseño y Control del Péndulo Invertido

Alumno: Alexey Sidenko
Ingeniería Electrónica
Legajo Nº 100233
soffter2000@yahoo.com.ar

Tutor: Dr. Ing. Ruben H. Milocco
rmilocco@yahoo.com.ar

Facultad de Ingeniería
Universidad Nacional del Comahue
Neuquén – 2011

Resumen

El modelo del péndulo invertido es un ejemplo clásico en la literatura de control ya que permite exponer de forma clara y didáctica una de las aplicaciones más importantes del control automático: el control de sistemas mecánicos inestables. El mismo tiene múltiples aplicaciones, como la estabilización de misiles, cargas transportadas en grúas, vehículos y robots. El sistema consiste en un carro sobre el cual se encuentra un péndulo invertido sujetado mediante un pivote sin fricción. El carro es movido por un motor que ejerce una fuerza en sentido horizontal, siendo esta la acción de control. El presente trabajo describe el diseño, construcción y ensayo de un péndulo invertido apto para ser utilizado en prácticas de laboratorio de la cátedra de Sistemas Controlados por Computadora. El mismo permitió experimentar con diversos tipos de control, así como también, aplicar diversas técnicas de identificación y modelado de sistemas.

Palabras Clave:

Péndulo invertido, control, sistemas controlados por computadora, variables de estado, control polinomial, filtro de Kalman, control por energía, MATLAB, USB, PIC, encoder

Abstract

The inverted pendulum model is a classic example in control literature as it allows a clear teaching one of the most important applications of automatic control: the control of unstable mechanical systems. It has many applications, such as missile, crane, vehicle and robot stabilization. The system consists of a cart on which is attached an inverted pendulum by a frictionless pivot. The cart is powered by a motor that exerts a force in the horizontal direction, this being the control action. This paper describes the design, construction and testing of an inverted pendulum suitable for use in the laboratory practices of the department of Computer Controlled Systems. It has aloud to experiment with different types of control, and also to apply different identification and system modelling techniques.

Key words:

Inverted pendulum, control, computer controlled systems, state variables, polynomial control, Kalman filter, energy control, MATLAB, USB, PIC, encoder

Agradecimientos

Este trabajo esta dedicado a todas las personas que hicieron posible esta realidad. A mi familia, que me dio su apoyo durante toda la carrera. A mi novia Victoria, por soportarme y estar siempre a mi lado. A mi tutor, Dr. Rubén Milocco por su dirección. A la Facultad de Ingeniería y a todos los amigos que encontré adentro. Gracias a todos.

Índice

Capítulo 1 - Introducción y Objetivos

1.1. Introducción

El péndulo invertido es uno de los problemas más clásicos del control ya que permite exponer de forma clara y didáctica una de las aplicaciones más importantes del control automático: el control de sistemas mecánicos inestables. El mismo consiste de péndulo, libremente articulado, que se encuentre montado sobre un carro capaz de desplazarse horizontalmente a voluntad mediante un motor eléctrico.

El control de un péndulo invertido posee múltiples aplicaciones, como por ejemplo en la construcción de dispositivos de transporte auto-balanceados (Segway [1], Honda U3-X [2]), en la estabilización de cohetes [3], estabilización de robots bípedos [4], en la estabilización de oscilaciones de cargas transportadas en grúas [5], en movilidad sobre sillas de rueda [6], etc.

1.2. Objetivos

El objetivo del presente trabajo es el diseño y desarrollo de un equipamiento didáctico, del tipo péndulo invertido, apto para ser utilizado en prácticas de laboratorio de la cátedra de Sistemas Controlados por Computadora, así como también, la implementación y ensayo de diferentes técnicas de control e identificación de sistemas.

La idea es que desde la computadora puedan experimentarse distintas estructuras de control tanto de la inclinación angular del péndulo, como de su posición sobre el eje horizontal, controlando la corriente del motor de comando del péndulo. De esta forma se busca que el estudiante pueda

familiarizarse con distintas estrategias de control programando en alto nivel, pudiendo graficar las variables importantes para su análisis y poder hacer procesamiento posterior de la información.

1.3. Fundamentación

El interés para desarrollar un equipo didáctico de experimentación surgió de la necesidad, dentro del Área de Control Automático de la Facultad de Ingeniería de la Universidad Nacional del Comahue, de disponer de dicho material para realizar las prácticas de laboratorio de las materias del área. El mismo permite experimentar con diversos tipos de controles como PID, realimentación de estados, control polinomial, control lineal cuadrático (LQR) y Gaussiano (LQG), redes neuronales, control no lineal y muchas otras variantes, así como también, aplicar diversas técnicas de identificación y modelado de sistemas. Se espera que este equipamiento sea de gran utilidad para los alumnos durante el cursado de la materia Sistemas Controlados por Computadora.

1.4. Estructura del Trabajo

El presente trabajo se divide en cinco capítulos: en el capítulo 2 se presenta la descripción detallada del diseño del sistema. En el capítulo 3 se realiza el modelado y la identificación de los parámetros del sistema, que serán luego utilizados en el capítulo 4 para el diseño de los distintos controladores. Finalmente, en el capítulo 5, se presentan las conclusiones del trabajo.

Capítulo 2 - Diseño del Péndulo Invertido

El propósito del equipo que se pretende diseñar es la implementación y ensayo físico de diferentes modelos matemáticos y estructuras de control, así como también, la evaluación de sus desempeños. Para esto es necesario poder controlar, desde una computadora, el motor que actúa sobre el péndulo y, al mismo tiempo, poder leer la información de la posición y el ángulo del mismo. La función de los circuitos electrónicos del equipo es solamente la de crear un vínculo entre la PC y la parte mecánica, a través de los sensores y un puente H. De esta forma es posible implementar cualquier algoritmo de control en la PC para hacer experimentos con el mecanismo.

En la figura 2.1 puede observarse un diagrama general de las partes que componen el sistema.

El mismo puede dividirse en tres grandes bloques: la parte mecánica, la parte electrónica y la interfaz de MATLAB [7] en la computadora. En el presente capítulo se explicará en detalle, el funcionamiento de cada uno de dichos bloques y las partes que los componen.

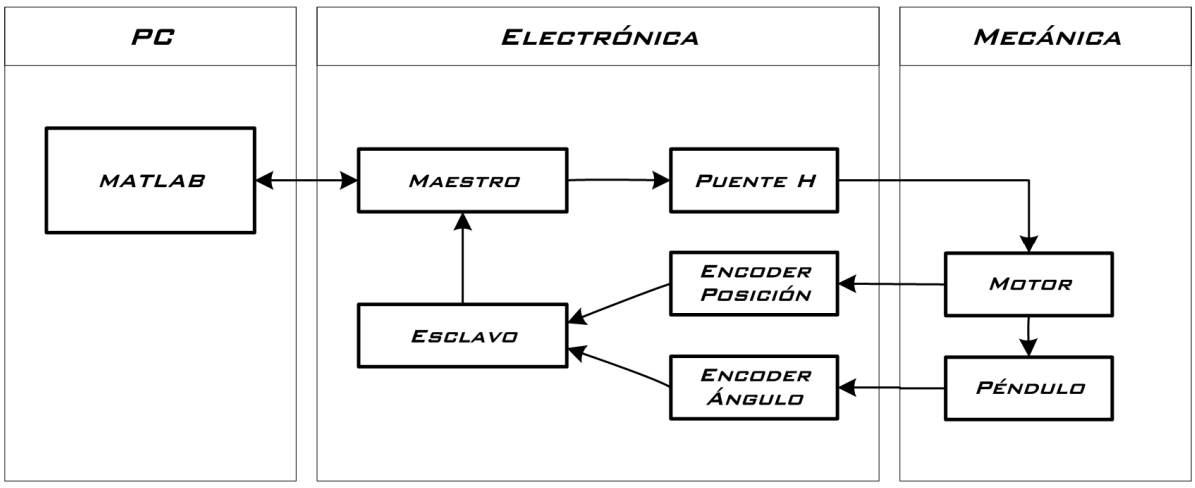


Figura 2.1: Esquema general del sistema

2.1. Diseño de la Mecánica

La primera etapa del trabajo consistió en el diseño, fabricación y ensamblaje de la parte mecánica del equipo. El paradigma del péndulo invertido exige la confección de un mecanismo tal que el péndulo, libremente articulado, se encuentre montado sobre un carro capaz de desplazarse horizontalmente a voluntad. Se requiere, además, la capacidad de medir con gran exactitud, tanto el ángulo de desviación del péndulo, como la posición del carro.

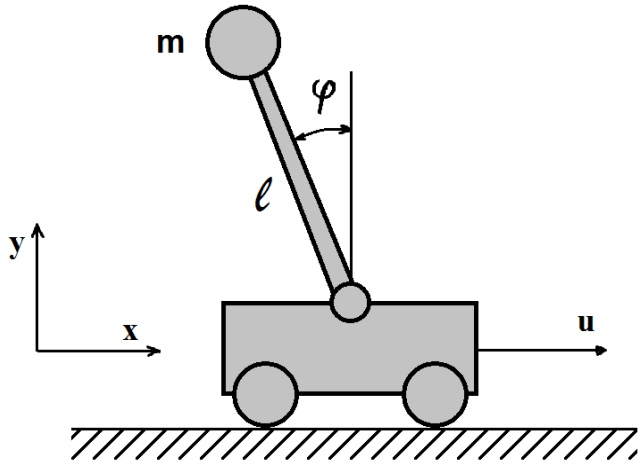


Figura 2.2: Diagrama esquemático del Sistema Mecánico

2.1.1. Péndulo

Para que el péndulo pueda rotar libremente es necesario montarlo sobre una articulación con muy bajo rozamiento. Para resolver este inconveniente se decidió utilizar el rodamiento de un disco rígido de PC. Este rodamiento resulta ideal debido a su pequeño tamaño y rozamiento casi despreciable.

El péndulo en si mismo no es más que una masa cuyo centro de gravedad se encuentra a cierta distancia de su eje de rotación. La implementación del mismo consistió simplemente de una planchuela de aluminio de 15x3.5 *cm* y 3 *mm* de espesor. El corte y mecanizado del mismo se realizó sin dificultad con herramientas manuales.

El montaje entre el péndulo y el rodamiento se realizo exitosamente mediante tres pequeños tornillos.

2.1.2. Rieles y Carro

Para efectuar el desplazamiento horizontal del eje del péndulo, el mecanismo del carro de una impresora resulta ideal. Se desensambló una impresora EPSON antigua para extraer los rieles, el carro y la correa, y se monto todo el conjunto sobre una base de MDF. Se montó el péndulo sobre el carro y se reemplazó el motor paso a paso de la impresora por un motor de corriente continua de imán permanente. El motor de corriente continua es más adecuado para los requerimientos del paradigma del péndulo invertido.

2.1.3. Gabinete

Para proteger los mecanismos y circuitos del sistema, de los factores del ambiente, se decidió construir un gabinete especial que los contenga. Se decidió utilizar Fibrofácil en la construcción del gabinete debido a la facilidad de mecanizado y ensamblaje. Este material además, añade peso al gabinete, lo que resulta ideal para absorber las fuerzas y vibraciones generadas por el mecanismo.

Se decidió dividir el gabinete en dos compartimientos:

- un compartimiento frontal para alojar todos los mecanismos del péndulo
- un compartimiento posterior para alojar todo el cableado y la electrónica

El compartimiento frontal, de 64x30x15 *cm*, se diseño con una compuerta superior para acceder a los mecanismos del péndulo y un panel frontal de vidrio para poder apreciar su funcionamiento desde el exterior. El compartimiento posterior también posee una compuerta para acceder a los circuitos en caso de necesitar efectuar reparaciones. Las dimensiones totales del gabinete son 67x33x30 *cm*.

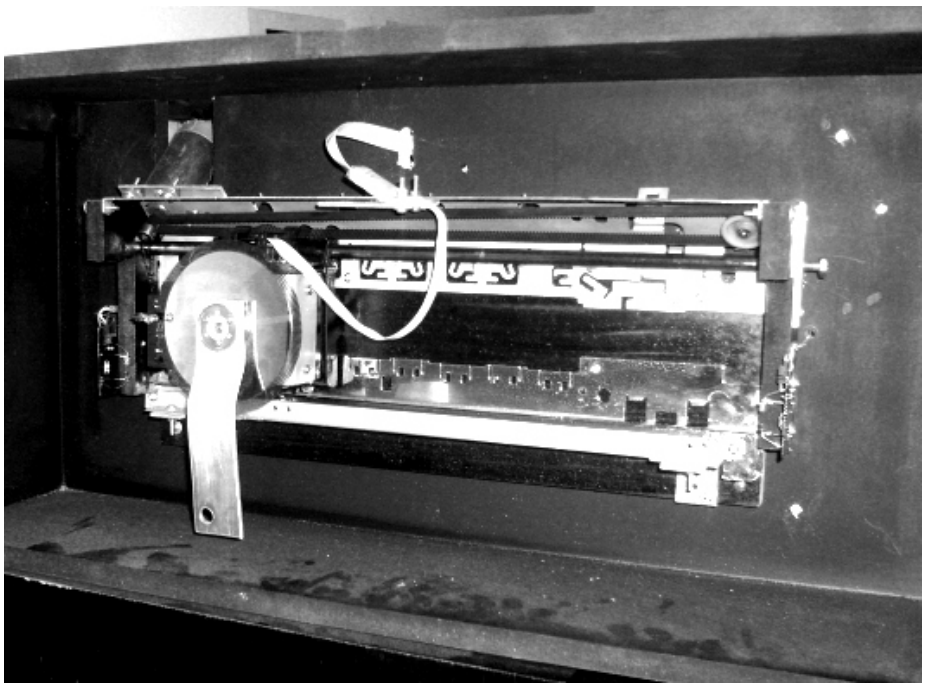


Figura 2.3: Fotografía del Conjunto Mecánico

2.2. Diseño de la Electrónica

Como se dijo anteriormente, la función de los circuitos electrónicos del equipo es solamente la de crear un vinculo entre la PC y la parte mecánica, a través de los sensores, el puente H y el motor.

En la figura 2.4 puede apreciarse un esquema general de la arquitectura electrónica elegida.

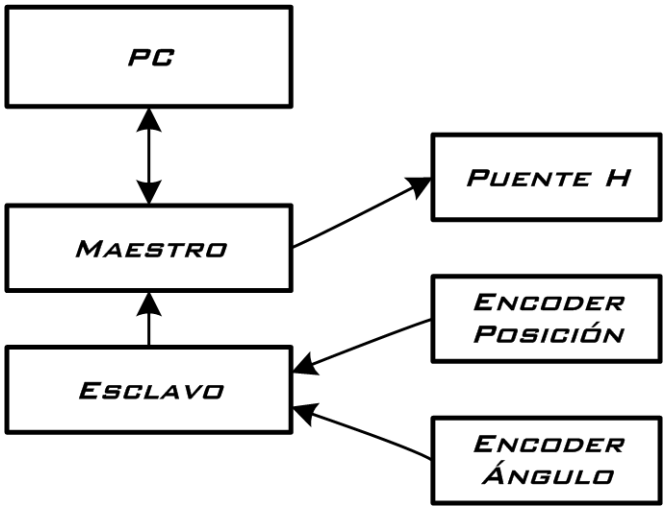


Figura 2.4: Esquema general de la arquitectura electrónica

El esquema utiliza dos microcontroladores: el microcontrolador Maestro se encarga de realizar las comunicaciones USB [8] con la PC y controlar el motor mediante el puente H, mientras que la tarea de administrar los encoders de posición y de ángulo se delega a un microcontrolador Esclavo por razones que se explican en la Sección 2.2.3.

2.2.1. Microcontroladores

Para la implementación del microcontrolador maestro, se decidió utilizar el microcontrolador PIC18F4550 [9] principalmente por su capacidad para la conexión USB. El mismo pertenece a la familia de microcontroladores de alta gama PIC18F y posee una serie de características que lo hacen ideal para esta aplicación:

- Conexión al puerto USB
- Modulo PWM
- Terminales para la programación in-circuit (ICSP)
- 32KB de memoria de programa Flash
- 2KB de memoria RAM
- Reloj de hasta 48 MHz (12 MIPS)
- 35 pines de entradas y salidas, distribuidas en 5 puertos (A, B, C, D y E)

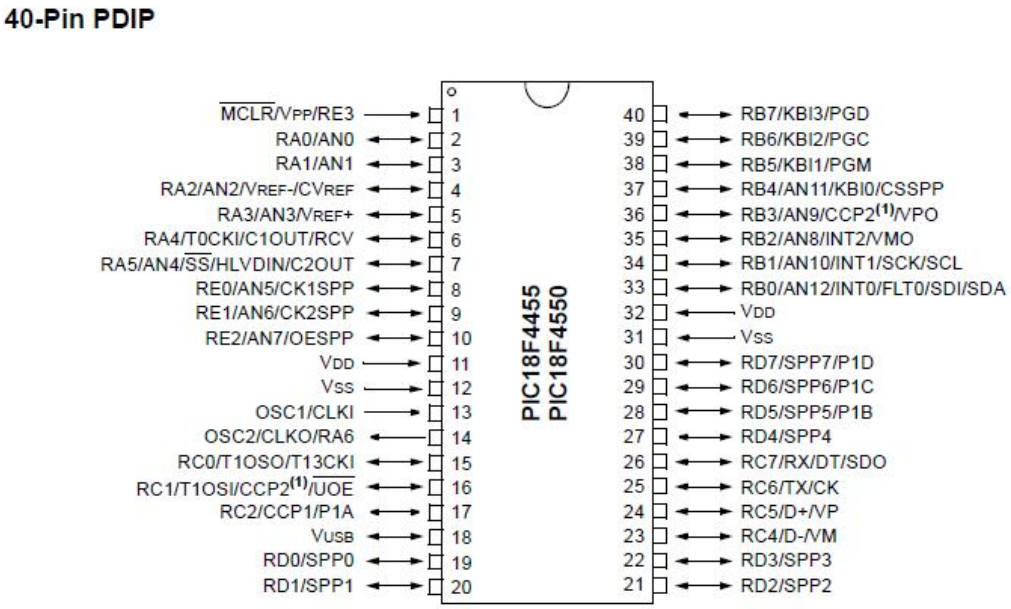


Figura 2.5: Pinout del microcontrolador utilizado [9]

Para la implementación del microcontrolador esclavo, se decidió utilizar el mismo microcontrolador, por cuestiones de simplicidad. Esto hizo más fácil la tarea de programación ya que los programas de maestro y esclavo podían compartir las mismas librerías y configuraciones. Además, si se desea disponer de un stock de repuestos para el equipo solo hace falta comprar un microcontrolador y no dos diferentes.

En la Tabla 5 del ANEXO V se detalla la función de cada uno de los puertos y pines utilizados del microcontrolador Maestro y Esclavo.

2.2.2. Encoders

Los encoders angulares disponibles en el mercado, además de costosos y voluminosos, tienen una resolución muy baja, llegando hasta 100 niveles por revolución como máximo. Por esta razón se decidió la fabricación de un encoder casero hecho a medida.

La configuración elegida fue la de encoder óptico incremental en cuadratura, por su simplicidad, precisión y bajo rozamiento. El mismo consiste de un disco de material transparente sobre el cual se encuentra impreso un patrón de franjas equiespaciadas. Dicho disco, solidario al péndulo, interrumpe alternativamente un haz de luz infrarroja, que emite un diodo LED sobre un fotodiodo receptor doble. Con un diámetro de 114 *mm* se pueden disponer 256 franjas sobre el perímetro del disco (512 flancos) lo que, con los sensores en cuadratura, permite discriminar hasta 1024 niveles por revolución. Los sensores en cuadratura, además de duplicar la resolución, permiten discernir el sentido de giro del disco.

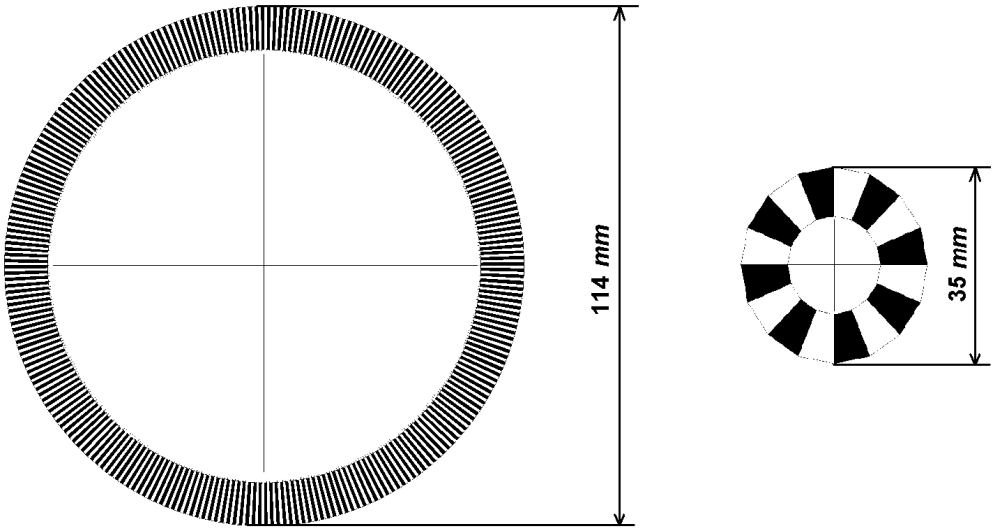


Figura 2.6: Discos transparentes con franjas equiespaciadas

Se utilizó un fotodiodo doble, que incluye los dos fotodiodos en un mismo encapsulado. Con esta disposición los fotodiodos se encuentran muy cerca uno del otro, lo que permite discernir patrones de líneas muy finas, del orden de 0,5 *mm*.

Como la fotocorriente de los fotodiodos es muy baja, del orden de los 50*uA*, es necesario amplificarla. Esto se logra mediante el circuito de la figura 2.7, el cual normaliza las señales de los fotodiodos a los valores lógicos de 0 y 5 *V*

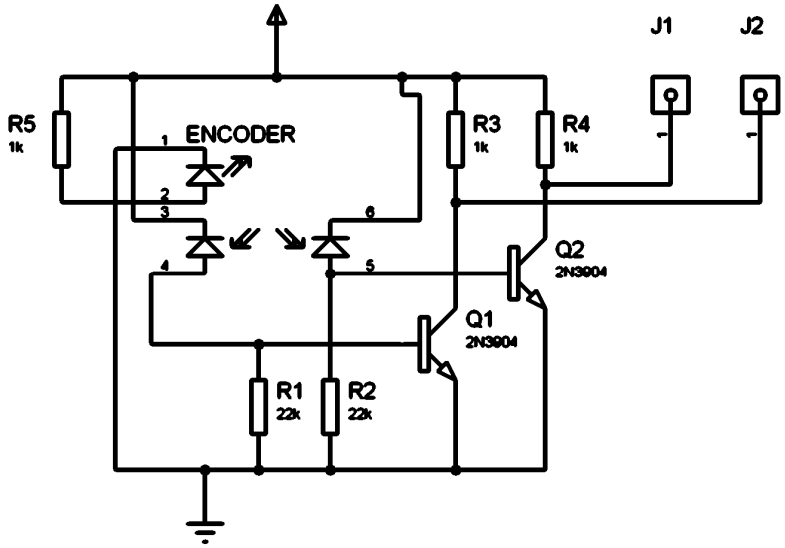


Figura 2.7: Circuito amplificador de los fotodiodos de los encoders

A medida que el disco gira, se generan dos señales de onda cuadrada a la salida del circuito. Estas señales se leen por el microcontrolador como palabras de dos dígitos. Conociendo la palabra presente y la anterior, se puede determinar si hubo cambios en la posición del disco usando la Tabla 1. Usando esta tabla, el microcontrolador incrementa o decrementa un contador interno, de esta forma se puede conocer en todo momento la posición del disco.

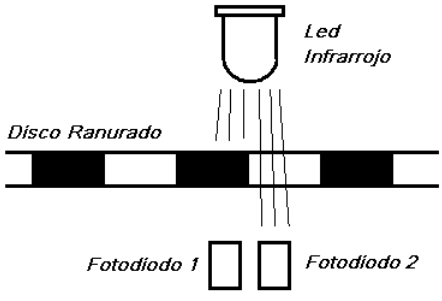


Figura 2.8: Disposición esquemática del disco ranurado y los fotodiodos

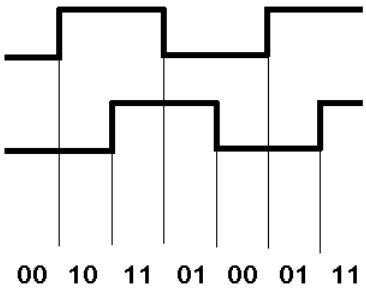


Figura 2.9: Señales generadas a la salida del circuito amplificador

Anterior	Presente	Resultado		
00	00	0		0 – No hubo cambios +1 – incremento -1 – decremento E - error
	01	+1		
	11	E		
	10	-1		
01	00	-1		
	01	0		
	11	+1		
	10	E		
11	00	E		
	01	-1		
	11	0		
	10	+1		
10	00	+1		
	01	E		
	11	-1		
	10	0		

Tabla 1: Tabla de decisión en base a las señales de los fotodiodos

Debido a la simplicidad y gran desempeño del encoder angular fabricado para la medición del ángulo, se decidió utilizar la misma técnica para la medición de la posición, instalando un encoder angular en el eje del motor. En este caso la precisión angular no es un factor tan importante, como si lo es, la gran velocidad de rotación. Es entonces necesaria una reducción de la resolución angular para evitar que se supere el ancho de banda del sensor. Al tener una resolución angular mas baja, se hace posible también la reducción del diámetro del disco. Es por esto que se utilizó un disco de 35 *mm* de diámetro y 8 franjas, con lo que se obtiene una resolución de 32 niveles por revolución, lo que equivale aproximadamente a una resolución de 1 *mm* en la posición.

Se realizaron algunos ensayos preliminares para verificar el correcto funcionamiento de los encoders. Se midieron las señales de salida con un osciloscopio, bajo diferentes condiciones de funcionamiento. Los resultados fueron completamente satisfactorios, algunos de ellos se muestran en las figuras 2.10 y 2.11.

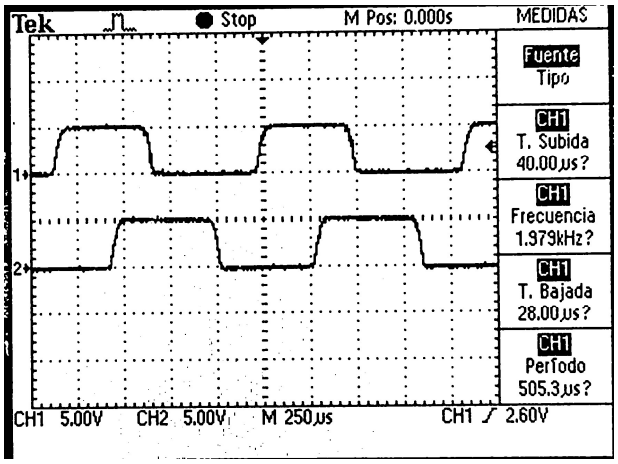


Figura 2.10: Péndulo girando a 232RPM

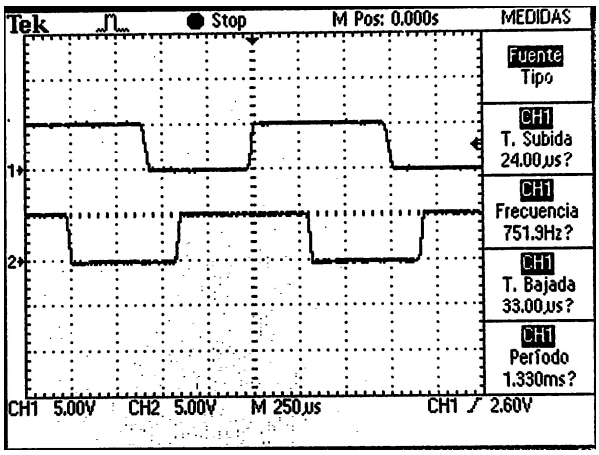


Figura 2.11: Motor girando a 5640 RPM

2.2.3. Arquitectura Maestro-Esclavo

Debido a que los encoders usados son del tipo incremental, estos requieren constante atención por parte del microcontrolador. Es necesario que las señales provenientes del encoder sean actualizadas por lo menos una vez cada 10us para evitar que el disco avance dos niveles y perder así el valor de referencia. El microcontrolador maneja la comunicación USB mediante interrupciones, es decir que puede ser retirado de sus tareas habituales para atender tareas correspondientes a la comunicación. Se hace imposible entonces garantizar la adecuada atención a los encoders usando un solo microcontrolador.

Para resolver este inconveniente se decidió utilizar dos microcontroladores en configuración Maestro-Esclavo. El microcontrolador Maestro se encarga de todas las tareas necesarias para el funcionamiento del sistema excepto la de atender periódicamente a los encoders. Esta función se delega al microcontrolador Esclavo el cual se encarga de leer periódicamente las señales de los encoders, actualizar los contadores según la Tabla 1 y de entregarle la información correspondiente al microcontrolador Maestro cuando éste la solicite.

2.2.4. Protocolo Paralelo

Para establecer una comunicación rápida y segura entre los microcontroladores maestro y esclavo, se decidió diseñar un protocolo paralelo especial, interconectando los pines de los puertos B y D de ambos microcontroladores (ver ANEXO V). De esta forma el microcontrolador maestro puede leer los datos desde los puertos, simulando ser un registro de memoria interno.

Se incluyeron dos pines (RC6 y RC7) para actuar como banderas en la comunicación. Una bandera solicita el envió del dato *x* (posición del carro), mientras la otra solicita el envío del dato *f* (ángulo de desviación del péndulo). En caso de que las dos banderas se encuentren bajas, el esclavo no escribe datos nuevos en los puertos, lo que permite una lectura segura por parte del maestro.

En la Figura 2.12 se observan las conexiones físicas entre los pines de ambos microcontroladores. Los pines de datos se encuentran cruzados para evitar puentes en el circuito impreso (PCB). La reinversión de los datos se realiza mediante software en el microcontrolador maestro.

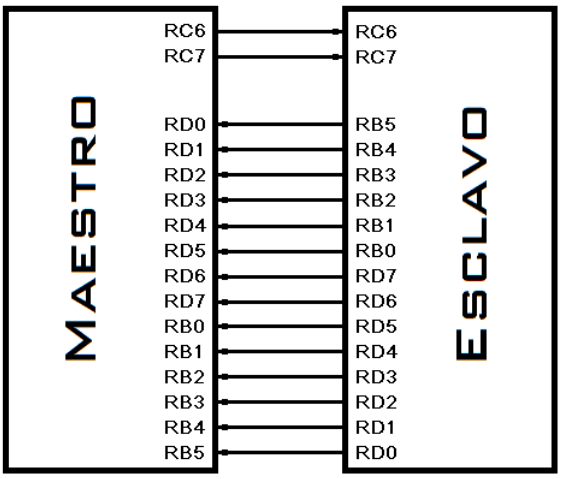


Figura 2.12: Conexiones físicas entre los pines de los microcontroladores Maestro y Esclavo

La secuencia de eventos que ocurren durante una transferencia de datos se observa en la figura 2.13. Inicialmente el microcontrolador maestro solicita el envió del dato *x* levantando la bandera 1. El microcontrolador esclavo responde en un tiempo de hasta 5 *us*, escribiendo los datos en el puerto. Luego de 8 *us* de levantada la bandera 1 el microcontrolador maestro la baja y lee los datos del puerto. La secuencia se repite idénticamente para leer el dato *f* con la Bandera 2.

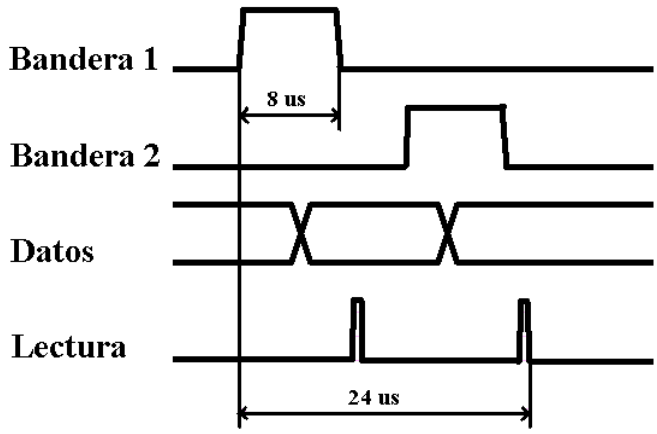


Figura 2.13: Secuencia de eventos durante una trasferencia de datos entre el microcontrolador Maestro y Esclavo

La lectura de ambos datos se completa 24 *us* luego de iniciada la transferencia.

2.2.5. Límites de Carrera

Para evitar que el carro, en su movimiento horizontal, colisione con alguno de los extremos del riel, se añadieron límites de carrera ópticos a ambos lados del mismo. Los mismos consisten simplemente de un diodo LED infrarrojo enfrentado con un fototransistor y la señal del mismo se conecta a dos pines (RB6 y RB7) del microcontrolador.

Una subrutina especial dentro del microcontrolador Maestro se encarga de revisar periódicamente el estado de los límites de carrera y desactivar el motor en caso de que sea necesario.

Estos límites de carrera se usan, además, para determinar la posición absoluta del carro al momento de encender el equipo *(ver sección 2.2.11.)*.

2.2.6. Comunicación USB

Se decidió utilizar el Bus USB para la comunicación entre el microcontrolador maestro y la PC debido a su gran aceptación y masivo uso. De esta forma se puede controlar el péndulo y realizar experimentos desde cualquier PC de escritorio o notebook.

Este protocolo permite el envío de paquetes de datos desde la PC, y permite, a continuación, recibir una respuesta desde el microcontrolador. De esta forma se puede enviar, en un paquete, la señal de control en un instante dado, y recibir luego las lecturas de los sensores. Desde la PC, todo el conjunto electrónico-mecánico que conforma el péndulo, es visto como un sistema dinámico de una entrada y dos salidas.

El modo de operación escogido es el de BULK TRANSFER debido a su capacidad para transmitir paquetes de datos a altas velocidades. Para la programación del microcontrolador se utilizaron las librerías proveídas por CCS (Custom Computer Services Inc.) [10] para trabajar en conjunto con la librería *mpusbapi.dll* de MICROCHIP [11]. Estas librerías fueron modificadas de acuerdo a las necesidades de la aplicación, cambiando los tamaños de los buffer y los descriptores del dispositivo (VID 0x04D8 PID 0x1011). Para la programación de la interfaz de PC se utilizó la librería *_mpusbapi.h* provista por MICROCHIP, la cual permite acceder al puerto USB desde el código de programación en MATLAB[12].

En la figura 2.14 se observa un esquema de la trama de 8 bytes utilizada en la comunicación USB.

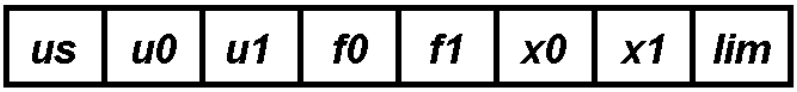


Figura 2.14: Trama USB

Variable	Tamaño	Descripción	Valores
us	1 byte	Signo de la señal de control	de 0 a 1
u1-u0	2 bytes	Amplitud de señal de control	de 0 a 1024
f1-f0	2 bytes	Ángulo de desviación del péndulo	de 0 a 1024
x1-x0	2 bytes	Posición del carro	de 0 a 1024
lim	1 byte	Estado de los limites de carrera	de 0 a 2

Tabla 2: Descripción de los Bytes de la trama USB

2.2.7. Puente H

Una vez recibida la señal de control desde la PC, es necesario aplicarla al motor. Una forma de hacer esto es a través de un puente H, utilizando una señal PWM. La figura 2.15 muestra la disposición adoptada. La misma consiste de 4 MOSFET canal-N (IRF540) [13] (Q1 Q2 Q3 Q4) y 2 MOSFET canal-P (IRF9530) [14] (Q5 Q6).

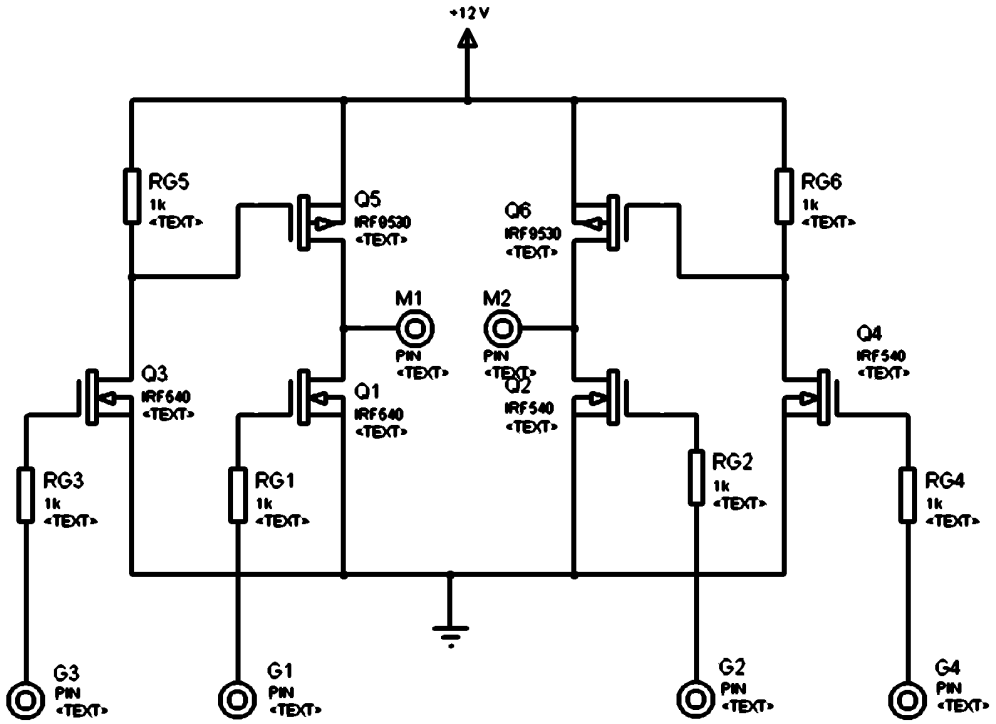


Figura 2.15: Circuito esquemático del Puente H

El sentido de giro positivo se habilita aplicando 5V sobre el borne G4 y la señal PWM sobre el borne G1. El sentido de giro negativo se habilita aplicando 5V sobre el borne G3 y la señal PWM sobre el borne G2. En ambos casos, los bornes restantes se ponen a masa para evitar cortocircuitos.

La señal PWM se genera utilizando los módulos especiales que incorpora el microcontrolador a tal fin. La frecuencia base escogida es de 11,7 khz y el ciclo de trabajo se establece a partir del valor absoluto de la señal de control.

2.2.8. Fuente de alimentación

Por razones de simplicidad y conveniencia, se decidió utilizar una fuente de PC estándar. La misma provee los 5V necesarios para la alimentación de toda la electrónica y los 12V para la alimentación del motor.

2.2.9. Placa de Circuito Impreso

Para la implementación física del circuito se fabricó una placa de circuito impreso (PCB), en la cual se incluyen los dos microcontroladores, el puente H, los conectores de alimentación y el Puerto USB. La misma puede verse en la figura 2.16. Se añadieron dos conectores ICSP para poder programar los microcontroladores. Por una cuestión de reducción de ruido, los circuitos amplificadores de los encoders se ubican en placas separadas, junto con los fotodiodos, y se conectan a la placa principal por medio de dos conectores IDC-10.

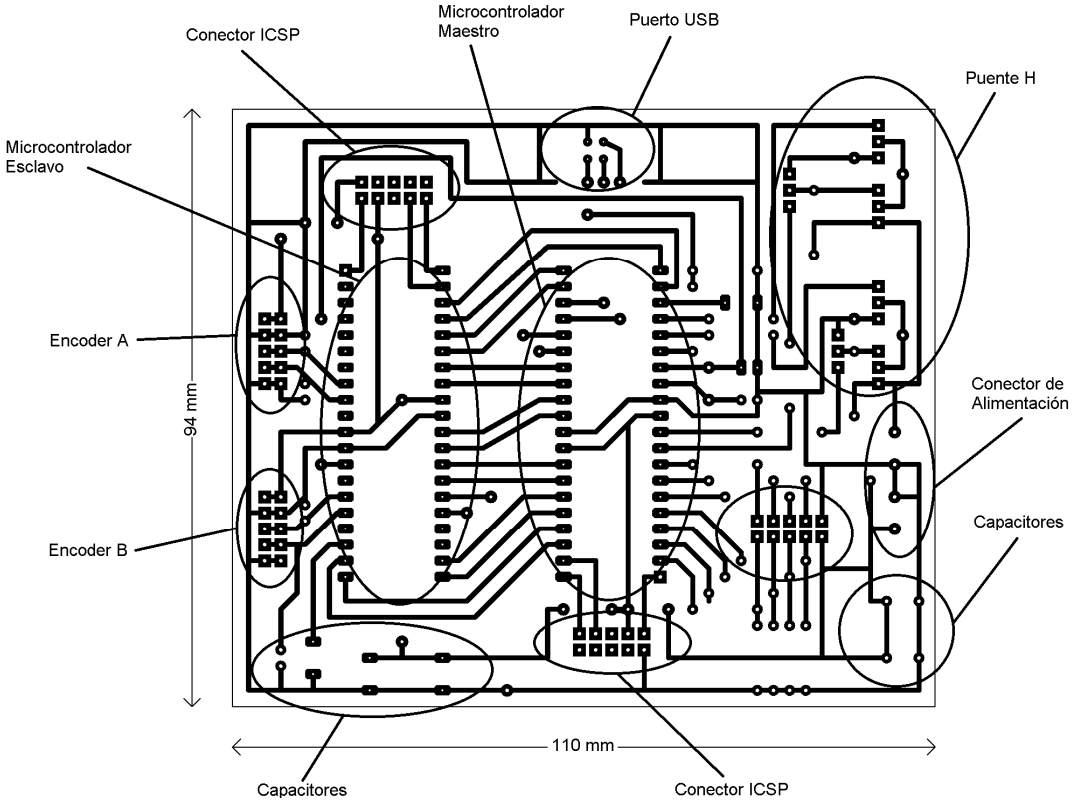


Figura 2.16: Placa de Circuito Impreso

2.2.10. Software

La estructura del software implementado en el microcontrolador se muestra en la figura 2.17.

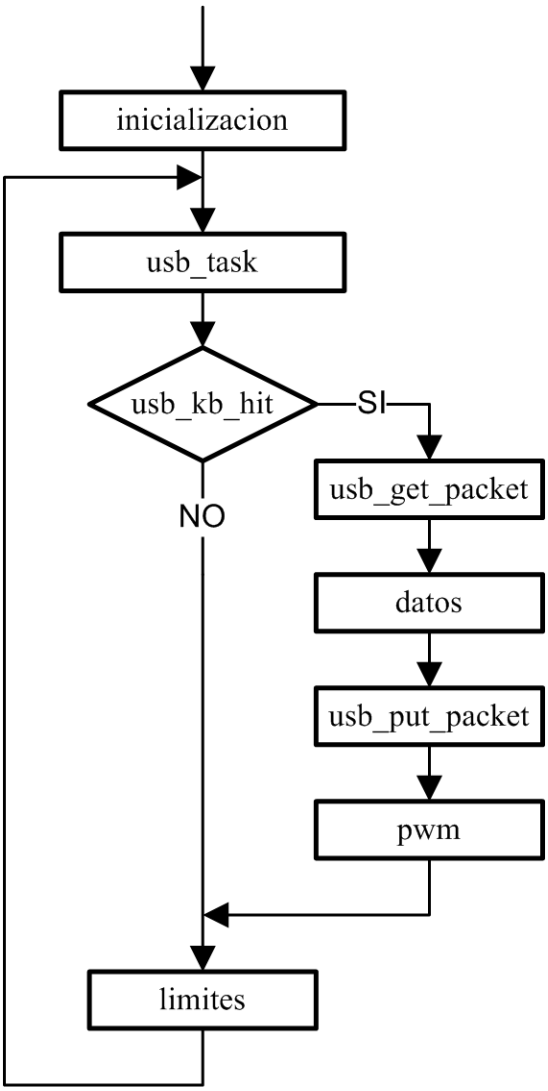


Figura 2.17: Estructura del software implementado en el microcontrolador Maestro

A continuación se describen brevemente cada una de las funciones y subrutinas que aparecen en la figura anterior:

inicializacion: se encarga de configurar, los puertos, los pines, los contadores y todas las variables necesarias para el funcionamiento del equipo.

usb_task: es una rutina que debe ejecutarse periódicamente, incluye tareas necesarias para la operación del puerto USB.

usb_kbhit: es una función que verifica si se envió un paquete de datos desde la PC.

usb_get_packet: es una función que recibe la trama de datos desde la PC y la almacena en variables internas del microcontrolador.

datos: es una subrutina que se encarga de realizar la comunicación con el microcontrolador esclavo, obtiene las lecturas de los encoders y las almacena en variables internas del microcontrolador

usb_put_packet: es una subrutina que envía los datos leídos de los encoders por el puesto USB, contestando el mensaje transmitido por la PC.

pwm: es una subrutina que se encarga de aplicar la señal de control sobre el motor, actualizando el ciclo de trabajo de los módulos PWM.

limites: es una subrutina que se encarga de revisar el estado de los límites de carrera y detener el motor en caso de que sea necesario, para evitar una colisión del carro con los límites del riel.

El código fuente del bucle principal y de las subrutinas puede verse en el ANEXO III.

2.2.11. Rutinas de inicialización

Debido a la configuración incremental de los encoders es necesario conocer el valor inicial de los parámetros del ángulo y la posición.

En el caso del ángulo, si el equipo se pone en funcionamiento mientras se encuentra detenido, es factible suponer que el péndulo se encuentra en la posición vertical inferior. El valor inicial escogido será entonces de *pi* radianes, pudiendo determinar luego el valor del ángulo en cualquier momento a partir de los incrementos detectados.

En el caso de la posición del carro, en cambio, es imposible conocer directamente la posición inicial del mismo en el momento de la puesta en funcionamiento del equipo. Para resolver este problema, se incluyó una rutina especial a tal fin en las rutinas de inicialización. La misma aplica una señal de

control constante sobre el motor haciendo que el mismo se desplace hacia uno de los extremos de su recorrido. El carro continúa este movimiento hasta accionar el límite de carrera correspondiente, en ese momento el carro se encontrará en uno de los extremos de su recorrido y se asigna entonces el valor inicial -1 en la posición.

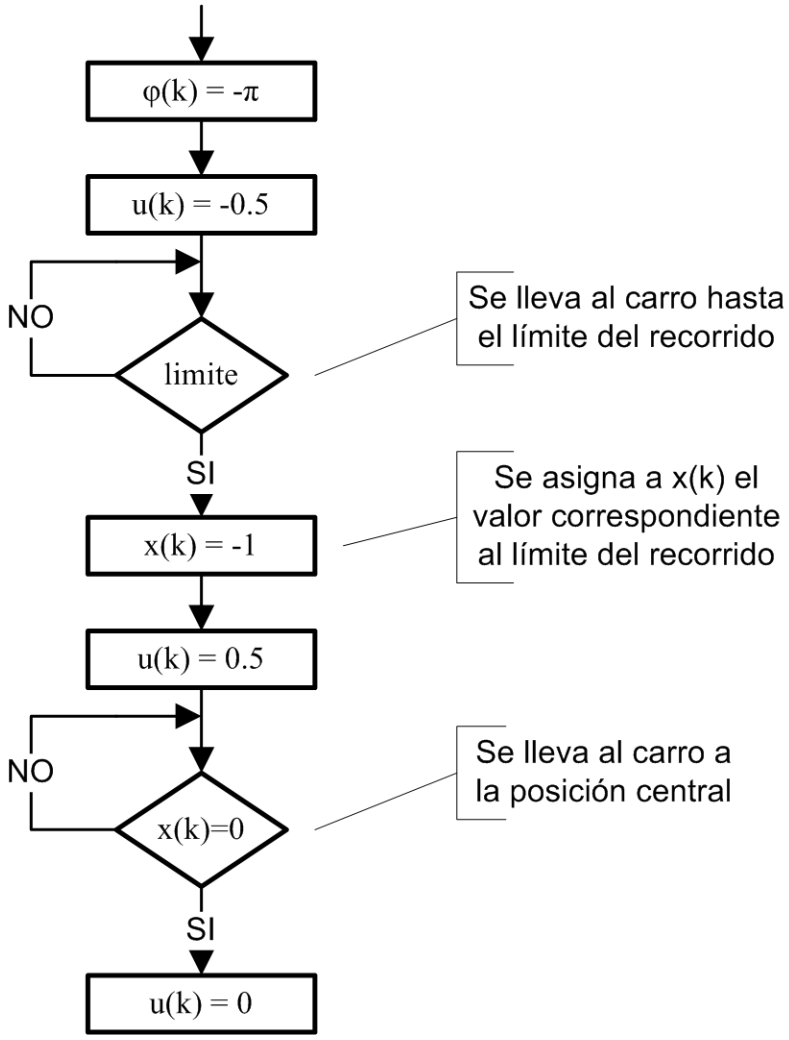


Figura 2.18: Rutina de inicialización para determinar la posición inicial del carro

2.3. Interfaz en MATLAB

La interfaz de PC consiste simplemente de un programa principal y una serie de subrutinas, implementadas en el editor de MATLAB. El mismo, como se mencionó anteriormente, permite acceder al puerto USB mediante el uso de la librería *_mpusbapi.h*, provista por MICROCHIP [11].

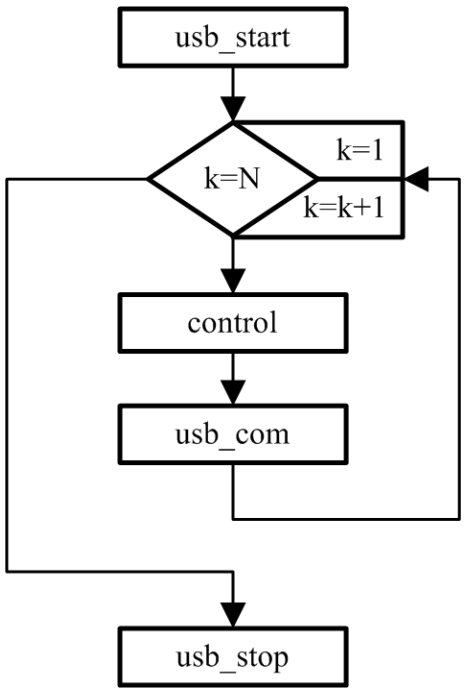
2.3.1. Subrutinas

Para simplificar la programación y operación de la interfaz se agrupó el código necesario en tres subrutinas.

- ***usb_start.m*** incluye todo el código necesario para cargar las librerías, declarar e inicializar las variables, e iniciar la comunicación USB,
- ***usb_com.m*** incluye todo el código para enviar la señal de control al microcontrolador, recibir su respuesta y normalizar las variables. También se encarga de realizar una pausa de h segundos.
- ***usb_stop.m*** incluye todo el código para finalizar la comunicación, descargar las librerías y liberar la memoria.

2.3.2. Estructura

Con las subrutinas antes descriptas, se armó una estructura de código, simple y efectiva, en la cual se puede insertar cualquier código de control.




```
h=0.02;
N=400;

usb_start;

    for (i=2:N)
     control;
     usb_com;

    end

usb_stop;
```

Figura 2.19: Diagrama de Flujo y Código Fuente de la estructura principal

2.3.3. Normalización de las Variables

Para trabajar de forma más cómoda, se normalizaron las variables del sistema a valores adimensionales dentro de los siguientes rangos:

$$-1 \leq u(k) \leq 1$$

$$-1 \leq x(k) \leq 1$$

$$-\pi \leq \varphi(k) \leq \pi$$

2.3.4. Período de muestreo

Para averiguar la frecuencia de muestreo máxima admisible, se realizaron ensayos para determinar los tiempos demora en la comunicación USB, es decir, cuanto tiempo se tarda en enviar una trama de 8 bytes al microcontrolador y recibir su respuesta. Los resultados se muestran en la Figura 2.20, en la cual se observa un histograma de los tiempos de demora para 1000 iteraciones.

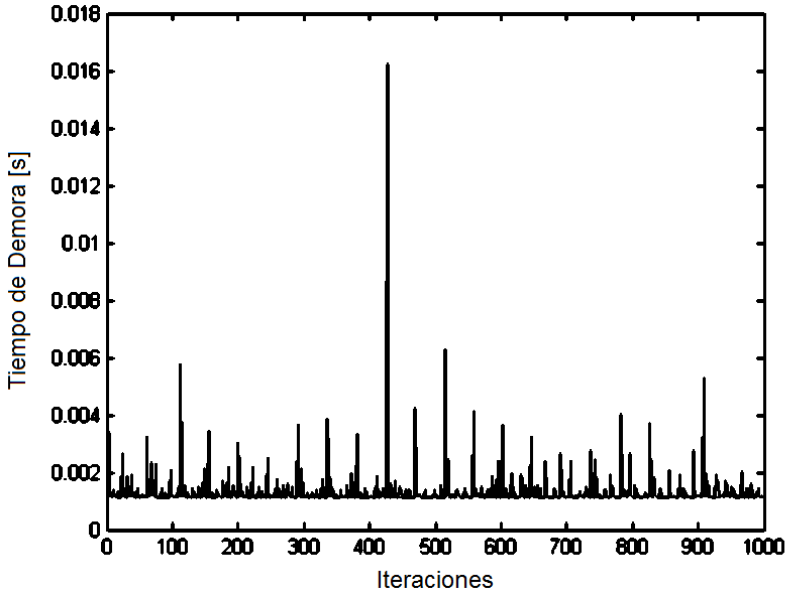


Figura 2.20: Histograma de Tiempos de demora en la comunicación USB

Se realizó un análisis estadístico que reveló una media de 1.2ms y una desviación estándar de 0.6ms, aunque con valores pico de hasta 16ms.

Para que la frecuencia de muestreo sea estable, es necesario incluir una demora en el bucle principal del programa, de forma que dicho bucle se ejecute una vez por cada período de muestreo. Para esto se hizo uso de los comandos ***tic*** y ***toc***. El comando ***toc*** simplemente devuelve el tiempo transcurrido a partir del último comando ***tic***. En la figura 2.21 se muestra el diagrama de flujo del segmento de código utilizado para realizar la demora. Esta demora se incluyó en la subrutina ***usb_com.m***. De esta forma, siempre transcurre un período de tiempo *h* entre cada comunicación con el microcontrolador.

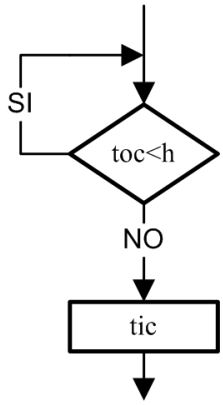


Figura 2.21: Diagrama de flujo del código de demora

Para que los tiempos de demora de la comunicación USB no influyan negativamente en la estabilidad de la frecuencia de muestreo, el tiempo *h* debe ser mayor que el valor pico de dichos tiempos de demora. Es por esto que se eligió un tiempo *h* de 0.02 s. Este es el período de muestreo utilizado siempre, de ahora en adelante.

2.3.5. Linealización del motor

Se detectó un comportamiento no lineal del motor, presentando una zona muerta para valores bajos de la señal de control. Se realizaron ensayos adicionales para determinar la respuesta del motor en función de la señal de control. Para esto se inyectaron pulsos de 200 ms de duración, y amplitud variable, a la entrada del sistema, y se midió el desplazamiento total del carro para cada pulso.

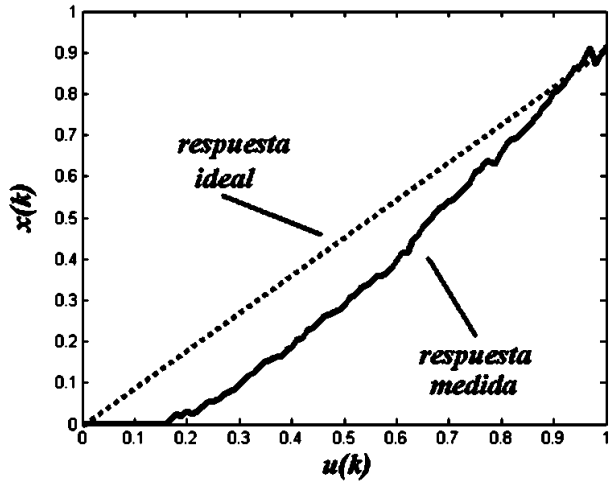


Figura 2.22: Respuesta no lineal del motor

Luego se elaboró un algoritmo que reduce la alinealidad, aplicando una función lineal a trozos especialmente definida. La función tiene tres tramos y diferentes pendientes para cada uno de ellos. El código implementado es el siguiente

```
u1=u0*18.3333;
if(u0>0.012)
    u1=(u0*1.0448)+0.2075;
end
if(u0>0.28)
    u1=(u0*0.7813)+0.2813;
end
```

donde u0 es la señal deseada, proveniente del controlador y u1 es la señal linealizada que se habrá de aplicar al motor. En la figura 2.23 se observa una gráfica de dicha función y en la figura 2.24 se muestra la respuesta que se obtiene luego de aplicar la función linealizadora a la señal de control. Se ve claramente una respuesta más lineal.

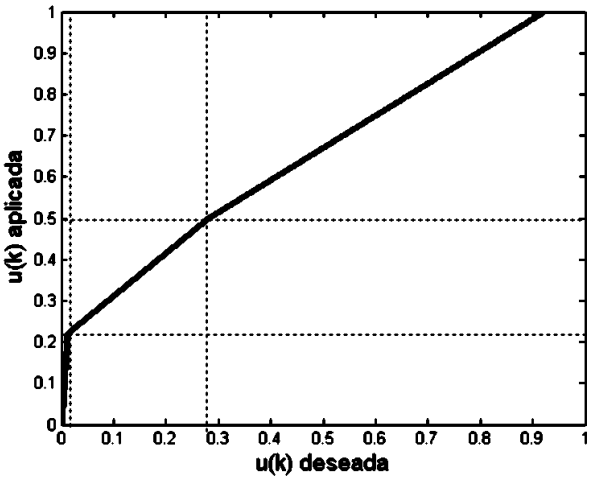


Figura 2.23: Función linealizadora a trozos

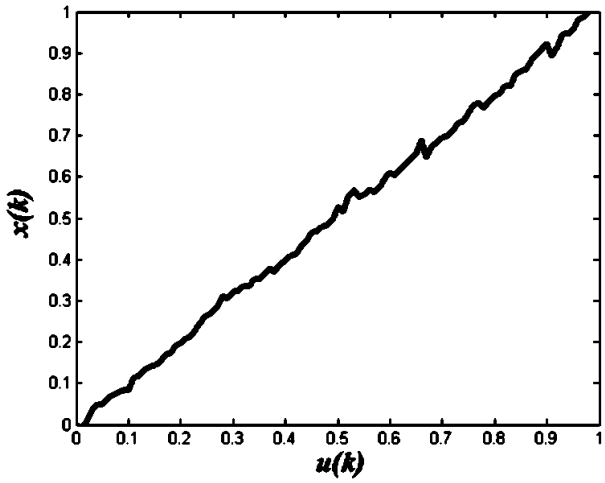


Figura 2.24: Respuesta linealizada

Capítulo 3 - Modelado e Identificación

Este capitulo estará dedicado a la determinación de un modelo matemático para el péndulo invertido, y a la determinación de sus parámetros, los cuales serán luego necesarios para el diseño del controlador en el Capítulo 4.

3.1. Modelado

Se considerará para el análisis, un modelo simplificado en el cual, el péndulo es representado por un cuerpo rígido, montado sobre una articulación sin rozamiento. Se desprecia también el rozamiento del péndulo con el aire.

Debido a que la masa del motor y del carro es mucho mayor que la masa del péndulo, consideraremos despreciables los efectos del movimiento del péndulo sobre el carro. De esta forma se simplifica el trabajo de modelado, ya que podemos modelar el péndulo y el carro por separado, para luego acoplarlos como dos sistemas en serie.

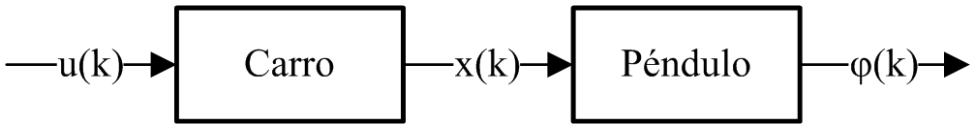


Figura 3.1: Modelo en Cascada

Esta simplificación se verifica de una forma sencilla, ya que si le damos un impulso al péndulo y lo hacemos girar libremente, no se observa movimiento en el carro.

3.1.1. Modelado del péndulo invertido

En la figura 3.2 vemos el modelo del péndulo simplificado en donde las únicas fuerzas que actúan sobre el mismo son las correspondientes a la gravedad y a la aceleración horizontal del carro.

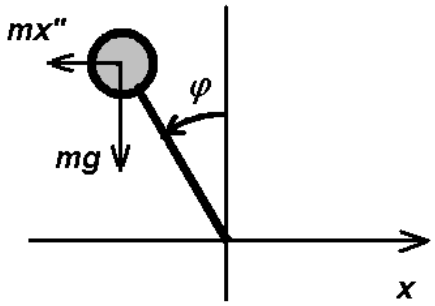


Figura 3.2: Modelo Simplificado del Péndulo Invertido

Aplicando las leyes de la mecánica clásica [15] obtenemos

$$J\varphi'' = \ell mg\ sen\varphi + \ell mx''\cos\varphi$$

Haciendo las aproximaciones $sen\varphi \approx \varphi$ y $\cos\varphi \approx 1$ llegamos a

$$J\varphi'' = \ell mg\ \varphi + \ell mx''$$

Estas últimas aproximaciones son validas únicamente cuando el péndulo se encuentra muy cerca de su posición vertical superior. Luego, aplicando transformada de Laplace, puede obtenerse la función de transferencia entre la aceleración horizontal y el ángulo de desviación del péndulo.

$$s^2 J\varphi = \ell mg\varphi + s^2 \ell mx$$

$$\frac{\varphi}{x} = \frac{s^2 \frac{\ell m}{J}}{\left(s^2 - \frac{\ell mg}{J}\right)}$$

Escrito en forma compacta

$$\frac{\varphi}{x} = \frac{s^2 dc}{\left(s^2 - c\right)}$$

Donde c y d son parámetros a determinar mediante la identificación.

3.1.2. Modelado del péndulo no invertido

Cuando el péndulo se encuentra en la posición vertical inferior, ya no son validas las aproximaciones hechas anteriormente. Es necesario entonces recalcular el modelo, cambiando el origen de medición del ángulo φ. La nueva variable cuyo origen es el eje vertical inferior, la llamaremos φ*.

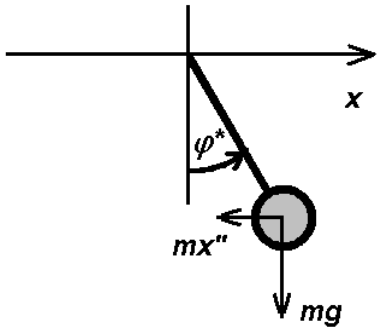


Figura 3.3: Modelo Simplificado del Péndulo No Invertido

Aplicando nuevamente las leyes de la mecánica clásica, obtenemos

$$J\varphi^{*\prime\prime} = -\ell mg\ sen\varphi^{*} - \ell mx^{\prime\prime}\cos\varphi^{*}$$

Se puede demostrar que la función de transferencia resulta

$$\frac{\varphi^{*}}{x} = \frac{-s^{2}dc}{\left(s^{2}+c\right)}$$

La diferencia entre ambos sistemas corresponde a una inversión en el signo del parámetro c. Esto es equivalente a una rotación de 90º en la ubicación de los polos y una inversión del signo en la ganancia.

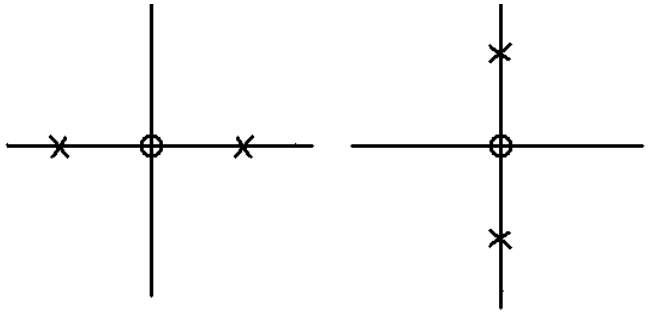


Figura 3.4: Ubicación de polos y ceros para el péndulo invertido (izq.) y el péndulo no invertido (der.)

3.1.3. Modelado del Motor

El circuito equivalente de un motor de corriente continua consiste de una fuente electromotriz, proporcional a la velocidad de giro del motor, en serie con la resistencia interna del devanado.

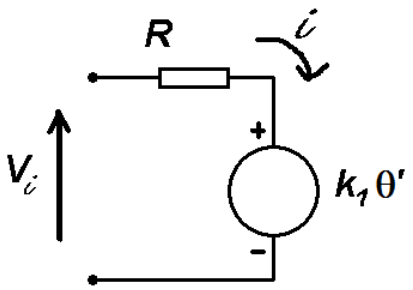


Figura 3.5: Modelo Simplificado del motor de CC

De aquí tenemos que

$$i = \frac{V_i - k_1\theta'}{R}$$

También sabemos que la cupla motora es proporcional a la corriente y que la aceleración angular es proporcional a la cupla motora menos el rozamiento (que se considera proporcional a la velocidad angular).

$$\tau_m = k_2 i \qquad \tau_r = k_3\theta' \qquad J\theta'' = \tau_m - \tau_r$$

remplazando obtenemos

$$J\theta'' = \frac{k_2}{R}V_i - \frac{k_2 k_1}{R}\theta' - k_3\theta'$$

aplicando transformada de Laplace

$$\frac{R}{k_2}\left(s^2 J + s\frac{k_2 k_1}{R} + sk_3\right)\theta = V_i$$

$$\frac{\theta}{V_i} = \frac{k_2}{Rs\left(sJ + \frac{k_2 k_1}{R} + k_3\right)}$$

Como el ángulo del motor es proporcional a la posición del carro y la tensión de entrada es proporcional a la señal de control, resulta

$$\frac{x}{u} = \frac{k_4 k_2}{Rs\left(sJ + \frac{k_2 k_1}{R} + k_3\right)}$$

o en forma compacta

$$\frac{x}{u} = \frac{ba}{s(s+a)}$$

Donde a y b son parámetros a determinar mediante la identificación.

3.1.4. Modelo Completo

A partir de estas funciones de transferencia podemos armar un sistema multi-variable, con una entrada y dos salidas. En la figura 3.6 se observan dos posibles formas de representación del sistema en diagramas de bloques.

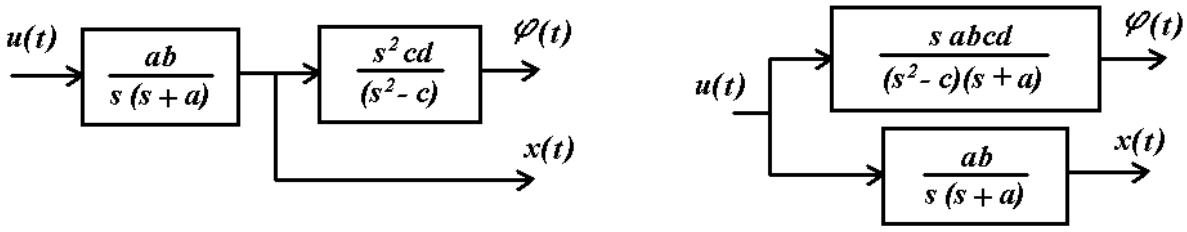


Figura 3.6: Posibles representaciones del sistema en diagramas de bloques

También es posible expresar el sistema como un sistema de cuarto orden en variables de estado

$$\begin{bmatrix} x''(t) \\ x'(t) \\ \phi''(t) \\ \phi'(t) \end{bmatrix} = \begin{bmatrix} -a & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ -dca & 0 & 0 & c \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x'(t) \\ x(t) \\ \phi'(t) \\ \phi(t) \end{bmatrix} + \begin{bmatrix} ba \\ 0 \\ badc \\ 0 \end{bmatrix} u(t)$$

$$y(t) = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x'(t) \\ x(t) \\ \varphi'(t) \\ \varphi(t) \end{bmatrix}$$

o en forma compacta

$$X'(t) = A\,X(t) + B\,u(t)$$

$$y(t) = C\,X(t)$$

donde $y(t)$ representa el vector de salidas $\begin{bmatrix} x(t) \\ \varphi(t) \end{bmatrix}$

3.1.5. Discretización

Visto desde la computadora, debido a la forma discreta de medir y aplicar la señal de control, el sistema se ve como un sistema discreto. Existen varias técnicas para derivar un modelo discreto del sistema a partir del modelo continuo de forma analítica [16], pero debido al alto grado de complejidad, inherente a un sistema de cuarto orden, la discretización se realizó en forma numérica mediante la función **c2d** de MATLAB.

El tipo de discretización utilizado es la discretización con mantenedor de orden cero, la cual consiste en suponer que la entrada se mantiene constante durante todo el período de muestreo (lo cual es cierto en esta aplicación). La misma es exacta en el sentido de que todas las variables de estado de ambos sistemas (continuo y discreto) coinciden en el instante de muestreo. De esta forma se obtiene un sistema discreto equivalente al continuo.

$$\begin{bmatrix} x'(k+1) \\ x(k+1) \\ \varphi'(k+1) \\ \varphi(k+1) \end{bmatrix} = \begin{bmatrix} \phi_{11} & \phi_{12} & \phi_{13} & \phi_{14} \\ \phi_{21} & \phi_{22} & \phi_{23} & \phi_{24} \\ \phi_{31} & \phi_{32} & \phi_{33} & \phi_{34} \\ \phi_{41} & \phi_{42} & \phi_{43} & \phi_{44} \end{bmatrix} \begin{bmatrix} x'(k) \\ x(k) \\ \varphi'(k) \\ \varphi(k) \end{bmatrix} + \begin{bmatrix} \gamma_1 \\ \gamma_2 \\ \gamma_3 \\ \gamma_4 \end{bmatrix} u(k)$$

$$y(k) = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x'(k) \\ x(k) \\ \varphi'(k) \\ \varphi(k) \end{bmatrix}$$

o en forma compacta

$$X(k+1) = \Phi\,X(k) + \Gamma\,u(k)$$

$$y(k) = C\,X(k)$$

3.2. Identificación Mediante Simulación

Para la identificación de los parámetros del sistema se implementó un algoritmo en MATLAB para simular respuestas de sistemas y compararlas, mediante una función de costo cuadrático, con la respuesta del sistema real. En la Figura 3.7 se puede ver un diagrama en bloques del algoritmo.

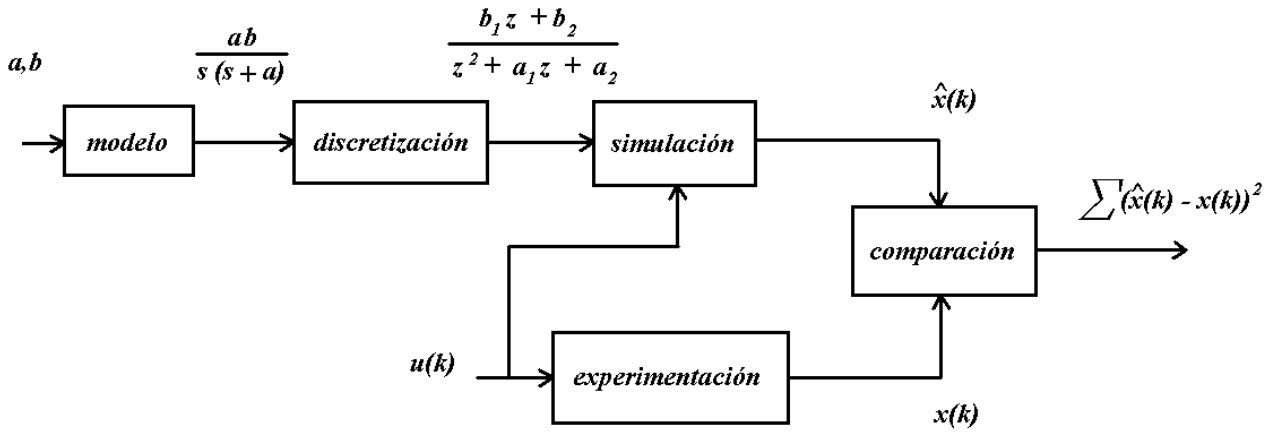


Figura 3.7: Diagrama en bloques del algoritmo de Identificación por Simulación

Existen muchos algoritmos de minimización basados en este criterio, como por ejemplo los que se describen en [17]. En este trabajo se utilizó el método de la grilla que, si bien tiene un alto costo computacional, nos garantiza encontrar la solución con una precisión arbitraria. El mismo se describe a continuación.

3.2.1. Identificación de la Función de Transferencia del Motor

Para empezar, se definió una entrada arbitraria $u(k)$ y se ejecutó el equipo para obtener la respuesta real del sistema a dicha entrada.

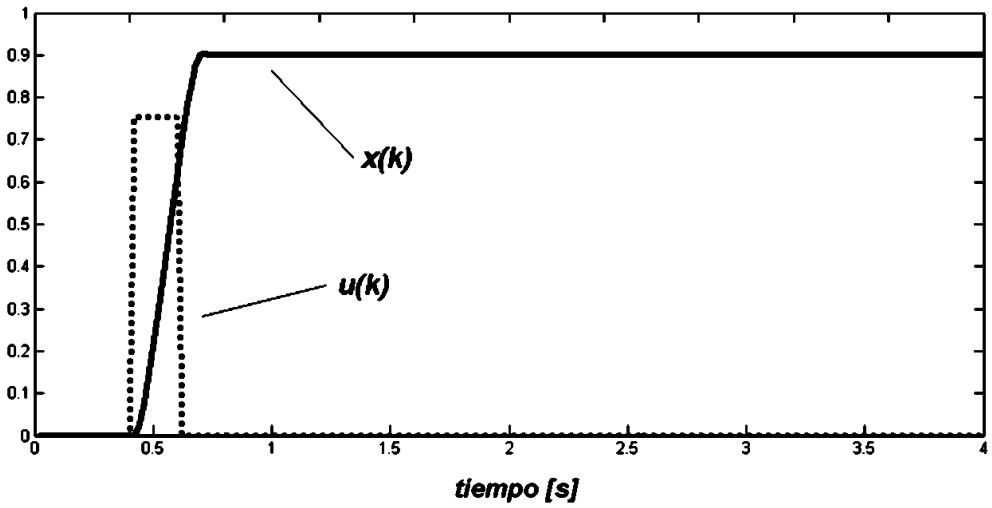


Figura 3.8: Respuesta real del sistema ante una entrada pulso

Luego, se definió arbitrariamente una grilla inicial de valores para los parámetros a y b, cubriendo una región de interés

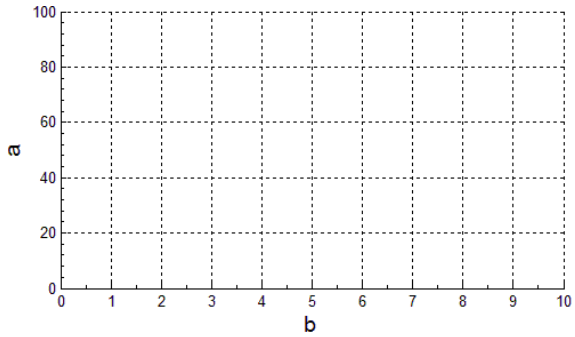


Figura 3.9: Grilla de valores para los parámetros a y b

Posteriormente se utilizó la función filter de MATLAB para simular un conjunto de respuestas $\hat{x}(k)$ de sistemas del tipo $\frac{ba}{s(s+a)}$ para cada par de valores (a, b) dentro de la grilla.

A continuación se calculó, para cada par de valores (a, b), el valor de la sumatoria del error cuadrático entre la respuesta real y la respuesta simulada.

$$e^2(a,b) = \sum \left(x(k) - \hat{x}(k)\right)^2$$

Finalmente de se seleccionó el par (a, b) que produce el error mínimo utilizando la función ***min*** de MATAB.

Debido a que este es un problema cuadrático lineal, está garantizado que es convexo y por lo tanto tiene un solo mínimo. Consecuentemente, si el error mínimo se encuentra en el interior de la grilla, esta asegurado que el mismo es la solución del problema. El algoritmo puede luego repetirse reduciendo el rango de variación de la grilla, para obtener estimaciones de mayor exactitud.

Por otro lado, si el mínimo se encuentra sobre el borde de la grilla, significa que la solución puede encontrarse fuera de la misma, por lo tanto habrá que redefinir la grilla con rangos de variación más amplios para repetir los cálculos.

Algunos de los resultados obtenidos pueden verse en las Figuras 3.10 y 3.11

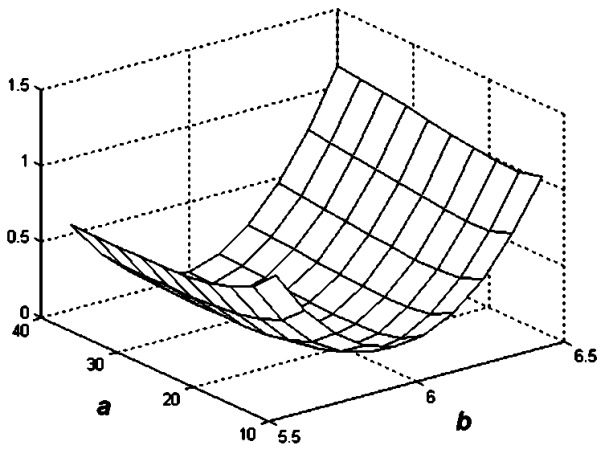


Figura 3.10: Error cuadrático en función de los parámetros a y b

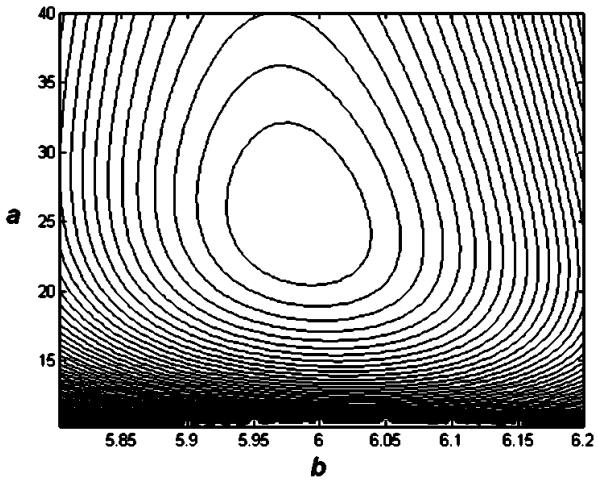


Figura 3.11: Contornos de la función de error cuadrático alrededor del punto mínimo

El par de parámetros que produjeron el error mas bajo fueron $a = 25.3$ y $b = 5.98$

3.2.2. Identificación de la Función de Transferencia del péndulo

Una vez identificados los parámetros del motor, se procedió con la identificación de los parámetros del péndulo. Debido a que resulta demasiado complicada la identificación del péndulo en la posición superior (ya que es un sistema inestable), se realizó la identificación del péndulo en la posición inferior (péndulo no invertido). Luego, ya que los parámetros son los mismos, se puede obtener la función de transferencia del péndulo invertido realizando los correspondientes cambios de signo. Debido a que

$$\frac{\phi^*}{x} = \frac{-s^2 dc}{\left(s^2 + c\right)} \qquad \text{y} \qquad \frac{x}{u} = \frac{ba}{s\left(s + a\right)}$$

la función de transferencia entere la señal de control y el ángulo de desviación del péndulo resulta

$$\frac{\phi^*}{u} = \frac{-s\,abcd}{\left(s^2 + c\right)\left(s + a\right)}$$

Luego, se procedió de la misma forma que en el caso anterior, esta vez para determinar los parámetros c y d, con una función de transferencia de tercer orden

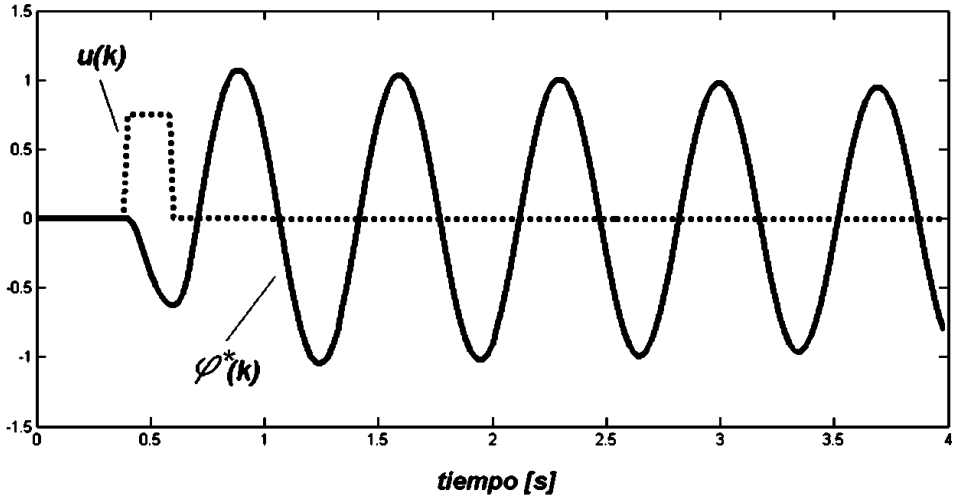


Figura 3.12: Respuesta real del Péndulo ante una entrada Pulso

En este caso, los parámetros obtenidos fueron $c = 80.2$ y $d = 0.017$

3.3. Identificación Mediante Cálculo Matricial

Otra forma de aplicar el método de mínimos cuadrados, a la identificación de los parámetros del sistema, es mediante el cálculo matricial [16] [17].

Supongamos que queremos identificar los parámetros de una función de transferencia discreta de segundo orden

$$x = \frac{b_1 z + b_0}{z^2 + a_1 z + a_0} u$$

A partir de una secuencia de entradas conocida u(k) y una secuencia de mediciones x(k) afectadas por ruido.

La función de transferencia puede reescribirse en forma de ecuación en diferencias, para calcular la predicción de x(k+1) dados los datos hasta el instante k

$$\hat{x}(k+1) = -a_1 x(k) - a_0 x(k-1) + b_1 u(k) + b_0 u(k-1)$$

O en forma matricial

$$\hat{x}(k+1) = \begin{bmatrix} x(k) & x(k-1) & u(k) & u(k-1) \end{bmatrix} \begin{bmatrix} -a_1 \\ -a_0 \\ b_1 \\ b_0 \end{bmatrix}$$

Repitiendo recursivamente, obtenemos

$$\begin{bmatrix} \hat{x}(2) \\ \hat{x}(3) \\ \hat{x}(4) \\ \vdots \end{bmatrix} = \begin{bmatrix} x(1) & x(0) & u(1) & u(0) \\ x(2) & x(1) & u(2) & u(1) \\ x(3) & x(2) & u(3) & u(2) \\ \vdots & \vdots & \vdots & \vdots \end{bmatrix} \begin{bmatrix} -a_1 \\ -a_0 \\ b_1 \\ b_0 \end{bmatrix}$$

$$\hat{Y} = \Psi \theta$$

El conjunto de parámetros θ que mejor se ajusta a los datos, es el que minimiza la sumatoria del cuadrado del error de predicción

$$e^2(\theta) = \sum_k (x(k) - \hat{x}(k))^2 = (Y - \hat{Y})^T (Y - \hat{Y}) = (Y - \Psi\theta)^T (Y - \Psi\theta)$$

El mínimo se obtiene igualando a cero la expresión de la derivada del error

$$\frac{\partial e^2(\hat{\theta})}{\partial \theta} = -\Psi^T (Y - \Psi\hat{\theta}) - (Y - \Psi\hat{\theta})^T \Psi = 0$$

$$\Psi^T (Y - \Psi\hat{\theta}) = 0$$

$$\hat{\theta} = (\Psi^T \Psi)^{-1} \Psi^T Y$$

Usando esta expresión es posible calcular una estimación de los parámetros del sistema discreto a partir de los datos experimentales u(k) y x(k).

3.3.1. Identificación de la Función de Transferencia del motor

El proceso de identificación de la función de transferencia del motor comienza inyectando una señal de prueba al equipo, en este caso una señal binaria pseudo-aleatoria, y registrándose los datos de salida.

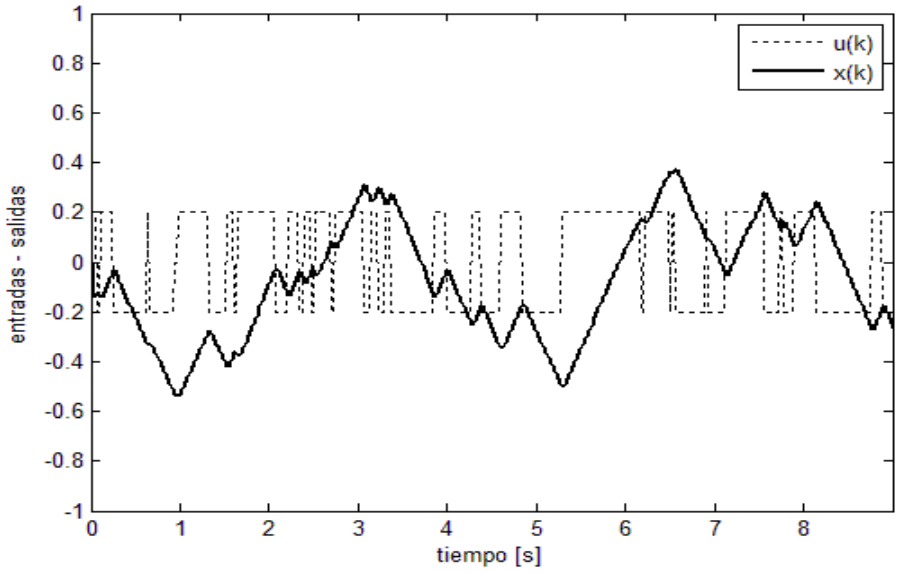


Figura 3.13: Entradas u(k) y salidas x(k) para una señal de prueba binaria pseudo-aleatoria

A continuación se ensamblan las matrices Y *y* Ψ siguiendo la forma

$$Y = \begin{bmatrix} x(3) \\ x(4) \\ x(5) \\ \vdots \end{bmatrix} \quad \Psi = \begin{bmatrix} x(2) & x(1) & u(2) & u(1) \\ x(3) & x(2) & u(3) & u(2) \\ x(4) & x(3) & u(4) & u(3) \\ \vdots & \vdots & \vdots & \vdots \end{bmatrix}$$

El vector de parámetros $\hat{\theta}$ se calcula como

$$\hat{\theta} = \left(\Psi^T \Psi\right)^{-1} \Psi^T Y = \begin{bmatrix} 1.2024 \\ -0.2024 \\ 0.0391 \\ 0.0299 \end{bmatrix} \qquad \Rightarrow \qquad x = \frac{0.0391z + 0.0299}{z^2 - 1.202z + 0.202} u$$

A partir de estos resultados es posible calcular el modelo continuo equivalente utilizando la función ***d2c*** de MATLAB, con lo cual se obtiene

$$x = \frac{-0.423\left(s - 816.6\right)}{\left(s + 79.89\right)\left(s - 0.003\right)} u$$

Haciendo algunas simplificaciones llegamos a

$$x = \frac{345.4}{s\left(s + 79.89\right)} u \qquad \Rightarrow \qquad a \cong 79.9 \qquad b \cong 4.32$$

3.3.2. Identificación de la Función de Transferencia del Péndulo

El mismo procedimiento puede aplicarse para la identificación de la función de transferencia del péndulo, esta vez con una función de transferencia de tercer orden. Al igual que en la sección 3.2.2. la identificación se realiza con el péndulo en la posición inferior, por lo que se trabajará con la variable de salida $\varphi^*(k)$.

Se inicia el experimento con el péndulo en la posición inferior y se aplica la señal de prueba binaria pseudo-aleatoria. En la Figura 3.14 vemos el comportamiento de $\varphi^*(k)$ al aplicar la señal de prueba.

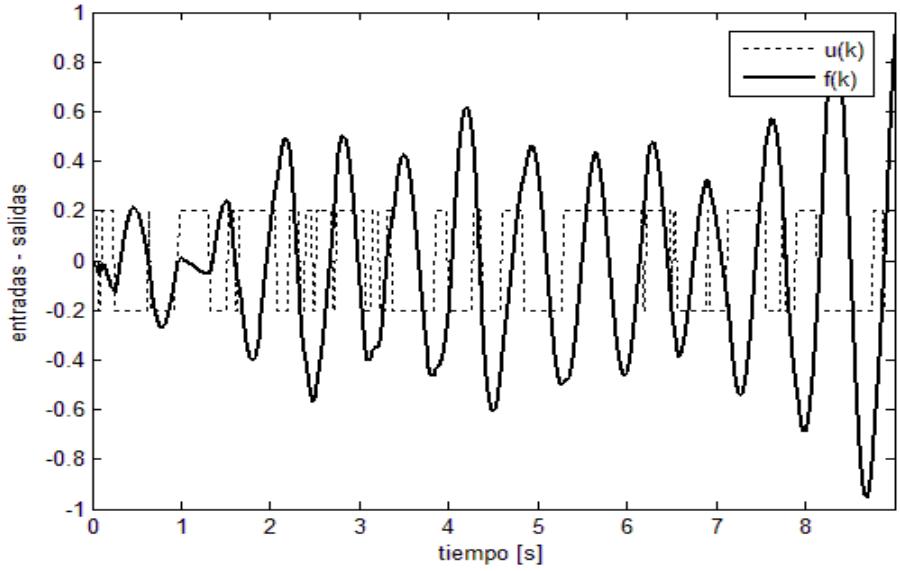


Figura 3.14: Entradas u(k) y salidas φ*(k) para una señal de prueba binaria pseudo-aleatoria

A continuación se ensamblan las matrices Y y Ψ siguiendo la forma

$$Y = \begin{bmatrix} \varphi^*(4) \\ \varphi^*(5) \\ \varphi^*(6) \\ \vdots \end{bmatrix} \quad \Psi = \begin{bmatrix} \varphi^*(3) & \varphi^*(2) & \varphi^*(1) & u(3) & u(2) & u(1) \\ \varphi^*(4) & \varphi^*(3) & \varphi^*(2) & u(4) & u(3) & u(2) \\ \varphi^*(5) & \varphi^*(4) & \varphi^*(3) & u(5) & u(4) & u(3) \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \end{bmatrix}$$

Y se calcula vector de parámetros como

$$\hat{\theta} = \left(\Psi^T \Psi\right)^{-1} \Psi^T Y$$

Obteniendo así

$$\theta = \begin{bmatrix} 2.099 \\ -1.266 \\ 0.138 \\ -0.051 \\ 0.0005 \\ 0.0471 \end{bmatrix} \qquad \varphi^* = \frac{-0.051z^2 + 0.0005z + 0.047}{z^3 - 2.099z^2 + 1.267z - 0.138} u$$

A partir de estos resultados se calcula el modelo continuo equivalente utilizando la función ***d2c*** de MATLAB, con lo cual se obtiene

$$\varphi^* = \frac{1.58\left(s - 355.5\right)\left(s + 1.716\right)}{\left(s + 98.6\right)\left(s^2 + 0.296s + 85.03\right)} u$$

Despreciando algunos términos de menor valor, llegamos a

$$\varphi^* \cong \frac{-561.7s}{(s+98.6)(s^2+85.03)}u$$

De donde pueden extraerse los valores de los parámetros

$$a \cong 98.6 \qquad c \cong 85.03 \qquad d \cong 0.016$$

3.3.3. Identificación del Sistema en Variables de Estado

El método de Mínimos Cuadrados mediante Cálculo Matricial puede extenderse a la identificación de sistemas en variables de estado.

A partir del modelo del sistema en Variables de Estado

$$X(k+1) \;=\; \Phi\, X(k) \;+\; \Gamma\, u(k)$$

Podemos expresar la predicción de x(k+1) a partir de los datos hasta el instante k, de la siguiente forma

$$\hat{X}^T(k+1) \;=\; \begin{bmatrix} X^T(k) & u^T(k) \end{bmatrix} \begin{bmatrix} \Phi^T \\ \Gamma^T \end{bmatrix}$$

Luego, repitiendo recursivamente, tenemos que

$$\begin{bmatrix} \hat{X}^T(1) \\ \hat{X}^T(2) \\ \hat{X}^T(3) \\ \vdots \end{bmatrix} = \begin{bmatrix} X^T(0) & u^T(0) \\ X^T(1) & u^T(1) \\ X^T(2) & u^T(2) \\ \vdots & \vdots \end{bmatrix} \begin{bmatrix} \Phi^T \\ \Gamma^T \end{bmatrix}$$

$$\hat{Y} \;=\; \Psi\theta$$

A partir de ahora, es posible aplicar el método de Mínimos Cuadrados, de la misma forma que en los casos anteriores. Suponiendo que todos los estados y las entradas son conocidos, la matriz de parámetros θ puede estimarse mediante

$$\hat{\theta} = \left(\Psi^T \Psi\right)^{-1} \Psi^T Y$$

De donde pueden extraerse la estimación de la matriz de transición Φ y del vector de entadas Γ del sistema bajo identificación.

$$\hat{\theta} \quad = \quad \begin{bmatrix} \hat{\Phi}^T \\ \hat{\Gamma}^T \end{bmatrix}$$

Las variables de estado $x(k)$ y $\varphi(k)$ son directamente medibles y sus valores son conocidos. Por otro lado $x'(k)$ y $\varphi'(k)$ no lo son, pero pueden ser estimadas mediante un operador de derivación, por ejemplo:

$$x'(k) = \frac{x(k-1) - x(k+1)}{2h} \qquad \varphi'(k) = \frac{\varphi(k-1) - \varphi(k+1)}{2h}$$

Al igual que en los casos anteriores, la identificación se realiza con el péndulo en la posición inferior, por lo que se trabajará con la variable $\varphi^*(k)$ en lugar de $\varphi(k)$. Se comienza inyectando una señal de prueba pseudo-aleatoria u(k) al sistema, y registrando las salidas $x(k)$ y $\varphi^*(k)$. Luego se estiman los valores de $x'(k)$ y $\varphi^{*\prime}(k)$ utilizando los operadores de derivación. En la Figura 3.15 se muestran algunos de los valores registrados para $u(k), x(k), x'(k), \varphi^*(k)$ y $\varphi^{*\prime}(k)$ durante un ensayo de identificación.

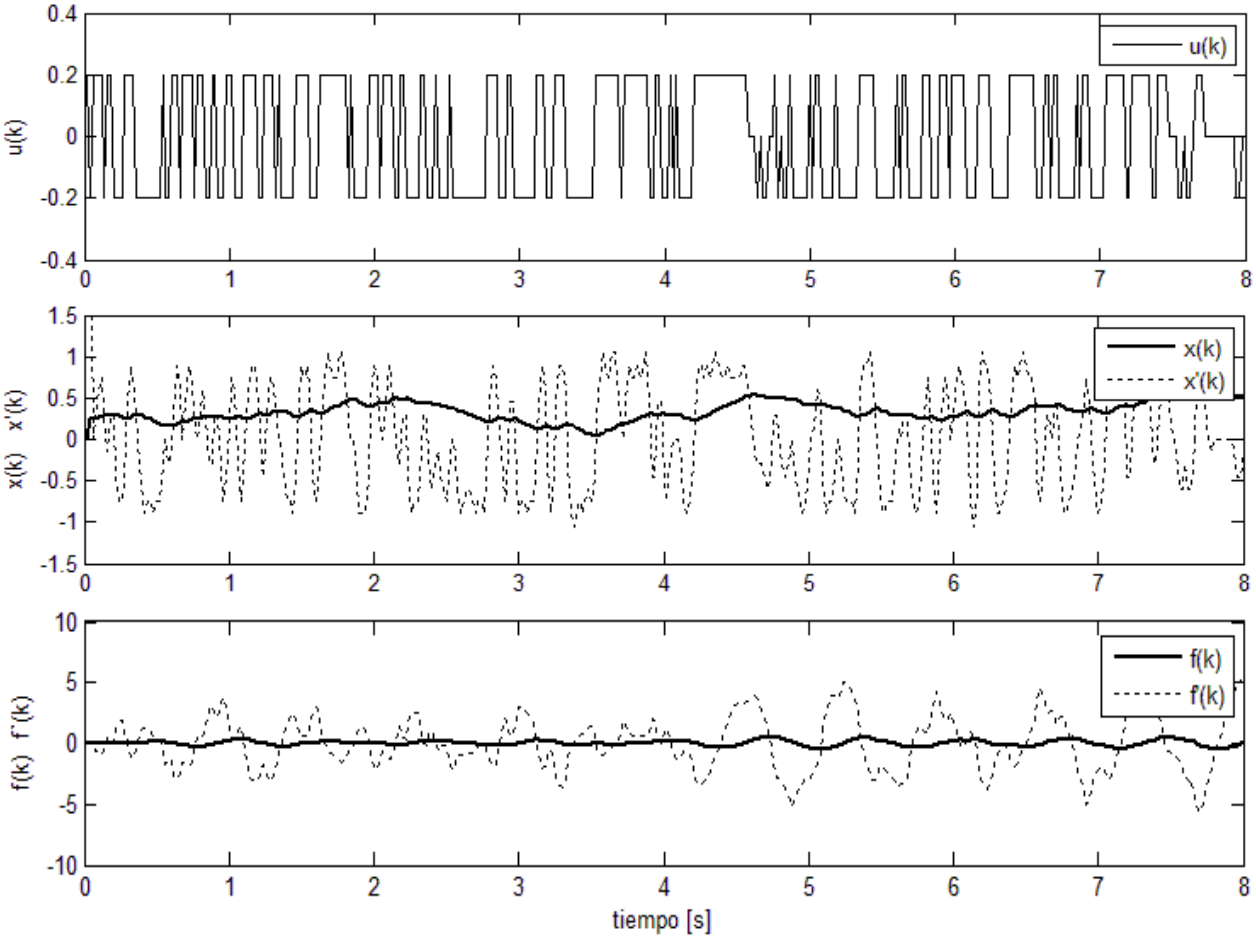


Figura 3.15: Entradas y salidas del sistema para una señal de prueba binaria pseudo-aleatoria

A continuación se ensamblan las matrices Y *y* Ψ siguiendo la forma

$$Y = \begin{bmatrix} X^T(1) \\ X^T(2) \\ X^T(3) \\ \vdots \end{bmatrix} \qquad \Psi = \begin{bmatrix} X^T(0) & u^T(0) \\ X^T(1) & u^T(1) \\ X^T(2) & u^T(2) \\ \vdots & \vdots \end{bmatrix}$$

Y se calcula la matriz de parámetros como

$$\hat{\theta} = \left(\Psi^T \Psi\right)^{-1} \Psi^T Y$$

Obteniendo así

$$\theta = \begin{bmatrix} 0.375 & 0.014 & 0.846 & 0.007 \\ 0.013 & 1.000 & -0.008 & 0.000 \\ 0.000 & 0.000 & 0.981 & 0.020 \\ 0.013 & 0.000 & -1.659 & 0.983 \\ 1.978 & 0.027 & -2.809 & -0.037 \end{bmatrix}$$

Luego

$$\Phi = \begin{bmatrix} 0.375 & 0.013 & 0.000 & 0.013 \\ 0.014 & 1.000 & 0.000 & 0.000 \\ 0.846 & 0.008 & 0.981 & -1.659 \\ 0.007 & 0.000 & 0.020 & 0.983 \end{bmatrix} \qquad \Gamma = \begin{bmatrix} 1.978 \\ 0.027 \\ -2.808 \\ -0.037 \end{bmatrix}$$

A partir de estos resultados se calcula el modelo continuo equivalente utilizando la función ***d2c*** de MATLAB, con lo cual se obtiene

$$A = \begin{bmatrix} -49.01 & 0.985 & -0.058 & 0.963 \\ 1.126 & 0.006 & 0.000 & 0.008 \\ 66.69 & -0.918 & -0.062 & -83.99 \\ -0.204 & -0.007 & 1.006 & -0.012 \end{bmatrix} \qquad B = \begin{bmatrix} 155.07 \\ 0.078 \\ -218.2 \\ 0.019 \end{bmatrix}$$

Luego de hacer algunas simplificaciones, obtenemos

$$A \cong \begin{bmatrix} -49.01 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 66.69 & 0 & 0 & -83.99 \\ 0 & 0 & 1 & 0 \end{bmatrix} \qquad B \cong \begin{bmatrix} 155.07 \\ 0 \\ -218.2 \\ 0 \end{bmatrix}$$

De donde, por comparación con el modelo teórico

$$\begin{bmatrix} x''(t) \\ x'(t) \\ \varphi^{*\prime\prime}(t) \\ \varphi^{*\prime}(t) \end{bmatrix} = \begin{bmatrix} -a & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ dca & 0 & 0 & -c \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x'(t) \\ x(t) \\ \varphi^{*\prime}(t) \\ \varphi^{*}(t) \end{bmatrix} + \begin{bmatrix} ba \\ 0 \\ -badc \\ 0 \end{bmatrix} u(t)$$

Podemos extraer

$$a \cong 49 \qquad b \cong 3.16 \qquad c \cong 84 \qquad d \cong 0.016$$

3.4. Identificación Mediante Ensayo

Una vez conocidos los valores aproximados de los parámetros, es posible realizar una identificación más fina realizando ensayos con el equipo.

Para esto, se asigna un valor al parámetro que se quiere determinar, se calcula el modelo y el controlador (*ver Capítulo 4*) en base a dicho valor, y luego se realiza el ensayo. Calculando la varianza de x(k), es posible obtener una medida de en cuanto se ajusta el modelo al sistema real. Repitiendo el ensayo para valores sucesivos del parámetro, es posible trazar una curva. El punto de varianza mínima corresponderá entonces al valor verdadero del parámetro en cuestión.

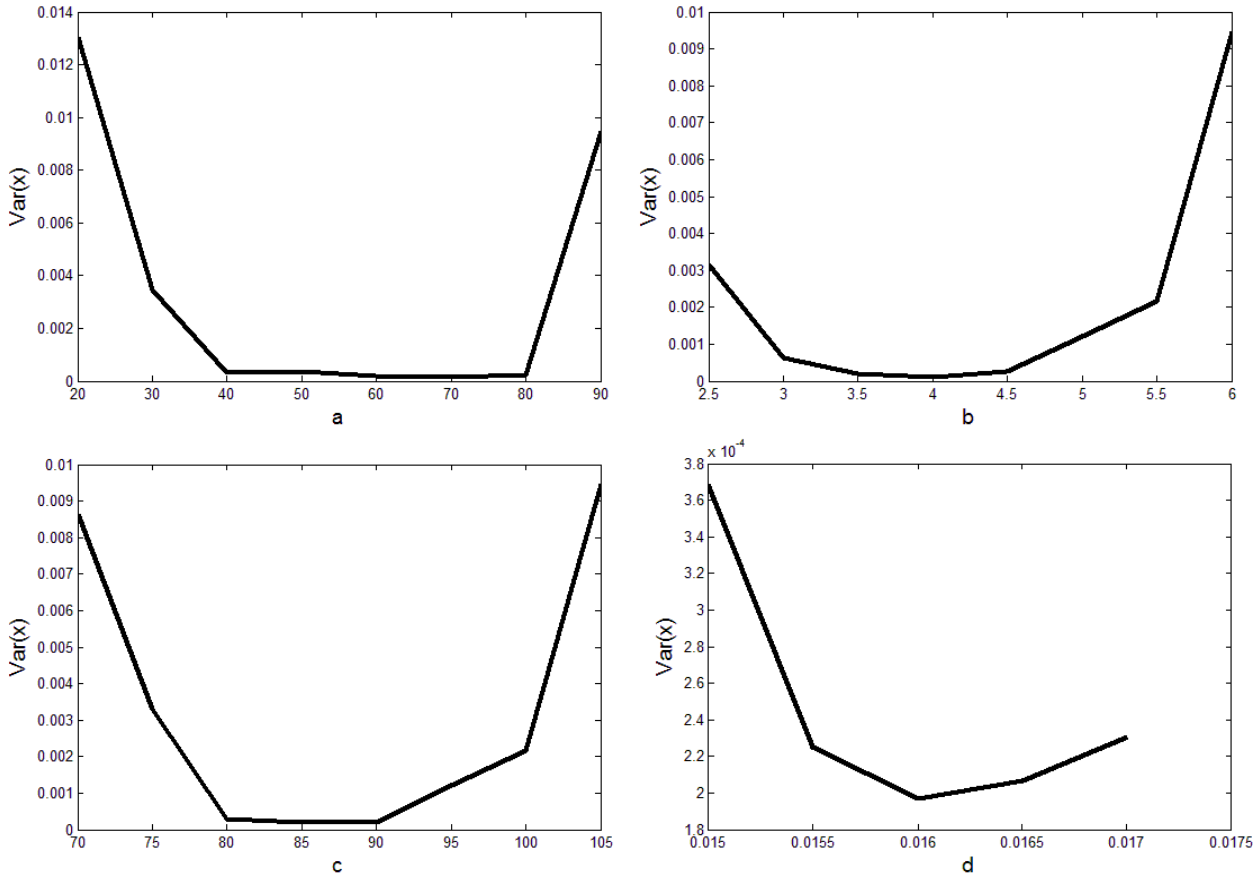


Figura 3.16: Varianza de x(k) en función de los parámetros a, b, c y d

Los parámetros obtenidos con este método fueron

$$a = 60 \qquad b = 4 \qquad c = 85 \quad \text{y} \quad d = 0.016$$

3.5. Comparación de los Métodos de Identificación

En esta sección se presentan los resultados obtenidos por los diferentes métodos de identificación y se realiza una breve comparación en base a los errores obtenidos con cada método.

Si bien es posible calcular una cota de error para los parámetros discretos obtenidos mediante los métodos de identificación por mínimos cuadrados [16], debido a la gran cantidad de cálculos involucrados, resulta prácticamente imposible realizar la propagación de dicho error a los valores de los parámetros continuos (a, b, c y d). Las cotas de error de dichos parámetros resultan entonces ser desconocidas.

Por esta razón, se elige uno de los métodos de identificación (el de mejor desempeño) como método de referencia. Luego, la comparación entre los diferentes métodos de identificación, se realiza contrastando los resultados obtenidos contra los valores del método de referencia.

El método de Identificación Mediante Ensayo (*Sección 3.4.*) fue el que mostró el mejor desempeño (minima varianza), por lo que los valores obtenidos con este método se toman como valores de referencia y se presumen como verdaderos.

A continuación se muestran resumidos en la Tabla 3, los resultados obtenidos con los diferentes métodos de identificación. Al lado del valor de cada parámetro se presenta el error porcentual con respecto al valor verdadero presunto.

	a		b		c		d	
	valor	e%	valor	e%	valor	e%	valor	e%
Identificación Mediante Simulación	25.3	-58%	5.98	+49%	80.2	-4%	0.017	+6%
Cálculo Matricial en Función de Transferencia	79.9	+33%	4.32	+8%	79.9	-4%	0.016	0%
Cálculo Matricial en Variables de Estado	49	-18%	3.16	-21%	84	-1%	0.016	0%
Valor Verdadero Presunto	60		4		85		0.016	

Tabla 3: Comparación de los métodos de identificación

Podemos observar que el método de Cálculo Matricial en Variables de Estado presenta los menores errores porcentuales, frente a los otros dos métodos.

Vemos que los valores de los parámetros c y d fueron identificados prácticamente sin error por los tres métodos. Por otro lado, existe gran incertidumbre con respecto a los valores de los parámetros a y b. Ensayos posteriores demostraron que existe una fuerte dependencia de estos dos parámetros con el tipo de señal de prueba utilizada, lo que evidencia la persistencia de un comportamiento no lineal en el motor. Se propone para trabajos futuros, la implementación de un lazo de realimentación local, para reducir los efectos no lineales indeseados del motor.

Capítulo 4 - Control del Péndulo Invertido

Este capítulo estará dedicado al diseño e implementación de diferentes controladores para el péndulo invertido, así como también, del análisis de su desempeño.

4.1. Control en Variables de Estado

4.1.1. Vector de Realimentación

Dado un sistema discreto del tipo

$$X(k+1) = \Phi X(k) + \Gamma u(k)$$

si aplicamos una realimentación de la forma

$$u(k) = -L X(k)$$

donde L es un vector, obtenemos un sistema nuevo sistema, de lazo cerrado, sin entradas

$$X(k+1) = (\Phi - \Gamma L) X(k)$$

La matriz de transición de este nuevo sistema es $(\Phi - \Gamma L)$. Escogiendo adecuadamente el vector L se pueden ubicar arbitrariamente los autovalores de $(\Phi - \Gamma L)$ y por lo tanto los polos del sistema de lazo cerrado. Para esto es necesario que el sistema sea controlable.

Si los polos de lazo cerrado se ubican dentro del círculo unitario, el vector de estados convergerá a cero, lo que implica que el péndulo tenderá a moverse hacia su posición de equilibrio vertical superior.

Existen muchos métodos para calcular el vector de realimentación L, algunos de ellos se describen en la bibliografía citada [16]. Debido al alto grado de complejidad, inherente a un sistema de cuarto orden, el calculo del vector L se realizó en forma numérica mediante la función ***place*** de MATLAB.

Para la implementación de la realimentación en variables de estado, primero es necesario conocer, en tiempo real, el vector de estados del sistema. Como no todos los estados del sistema son medibles, se hace necesaria la estimación de los mismos mediante un Observador de Estados.

4.1.2 Observador de Estados

Un Observador de Estados es un algoritmo que estima los valores del vector de estados a partir de los valores de las entradas y las mediciones de las salidas del sistema.

Un observador dinámico es un algoritmo que genera un sistema simulado, idéntico al sistema real. Si el sistema real y su simulación son sometidos a la misma entrada, y tienen las mismas condiciones iniciales, se puede esperar que, en todo momento, sus variables de estado sean las mismas. De este modo, el estado interno de la simulación se puede usar como una aproximación del estado interno del sistema real.

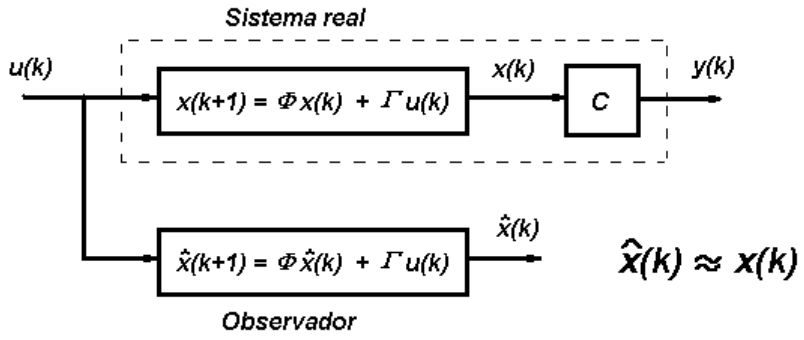


Figura 4.1: Observador de Estados Libre

En caso de que los dos sistemas no tengan las mismas condiciones iniciales (ya que estas generalmente son desconocidas) si el sistema es estable, la influencia de las condiciones iniciales irá disminuyendo a medida que pase el tiempo. Por lo tanto, si bien al principio tendrán valores diferentes, las variables de estado de ambos sistemas terminarán convergiendo asintóticamente a medida que transcurra el tiempo.

Para acelerar la convergencia del estado del sistema simulado al estado del sistema real, se puede estimular a la simulación con una realimentación del error entre la salida de los dos sistemas. De este modo, se logra modificar la dinámica del error de modo que converja más rápido. Otra ventaja

de esta configuración es que hace posible estimaciones de estados en sistemas inestables, como es el caso del péndulo invertido.

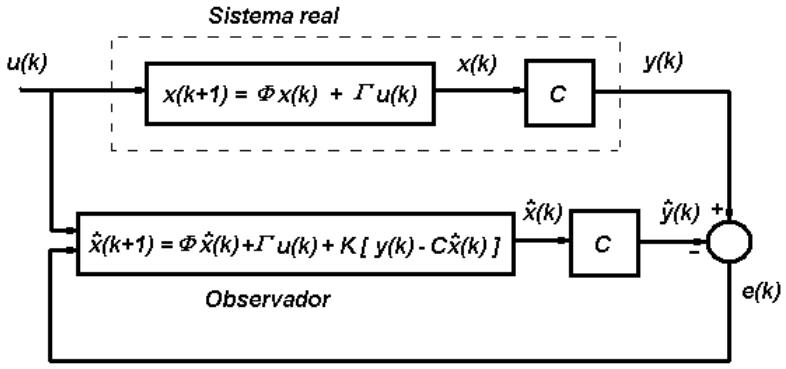


Figura 4.2: Observador de Estados Realimentado

Supongamos que se tiene un observador de la forma:

$$\hat{X}(k+1) \;=\; \Phi\,\hat{X}(k) \;+\; \Gamma\,u(k) \;+\; K\left[\; y(k) - C\,\hat{X}(k)\;\right]$$

$\hat{X}(k)$ es la estimación de $X(k)$ y su diferencia es el error de estimación:

$$\tilde{X}(k) = X(k) - \hat{X}(k)$$

Restando la ecuación del observador de la ecuación del sistema

$$X(k+1) - \hat{X}(k+1) \;=\; \Phi\,X(k) + \Gamma\,u(k) \;-\; \Phi\,\hat{X}(k) - \Gamma\,u(k) \;-\; K\left[\; y(k) - C\,\hat{X}(k)\right]$$

Obtenemos

$$\tilde{X}(k+1) \;=\; \left[\Phi - KC\right]\;\tilde{X}(k)$$

Vemos que si se escoge K de forma que los polos de $\left[\Phi - KC\right]$ sean estables, el error de estimación siempre convergerá a cero sin importar sus condiciones iniciales. Para esto es necesario que el sistema sea observable.

Los métodos para el cálculo del vector K son similares a los utilizados para el cálculo del vector L, algunos de ellos se describen también en la bibliografía citada [16]. Al igual que en el caso anterior, el cálculo del vector K se realizó en forma numérica mediante la función ***place*** de MATLAB.

Si en la ley de realimentación, ahora se reemplaza los valores del vector de estados por su estimación, tenemos:

$$u(k) = -L\,\hat{X}(k)$$

$$u(k) = -L\left(X(k) - \tilde{X}(k)\right)$$

El sistema ahora incluye la dinámica del error. El sistema resultante puede expresarse como:

$$\begin{bmatrix} X(k+1) \\ \tilde{X}(k+1) \end{bmatrix} = \begin{bmatrix} \Phi - \Gamma L & \Gamma L \\ 0 & \Phi - KC \end{bmatrix} \begin{bmatrix} X(k) \\ \tilde{X}(k) \end{bmatrix}$$

4.1.3. Controlabilidad y observabilidad del sistema

La controlabilidad del sistema se verificó calculando el rango de la matriz de controlabilidad.

$$W_C = \begin{bmatrix} \Gamma & \Phi\Gamma & \Phi^2\Gamma & \Phi^3\Gamma \end{bmatrix}$$

$$rango(W_C) = 4$$

Ya que el rango de dicha matriz es igual al orden del sistema, significa que el sistema es completamente controlable, lo que permite ubicar arbitrariamente los polos de lazo cerrado.

De forma similar, se verificó la observabilidad del sistema, calculando el rango de la matriz de observabilidad.

$$W_O = \begin{bmatrix} C \\ C\Phi \\ C\Phi^2 \\ C\Phi^3 \end{bmatrix}$$

$$rango(W_O) = 4$$

Ya que el rango de dicha matriz es igual al orden del sistema, significa que el sistema es completamente observable, lo que permite ubicar arbitrariamente los polos del observador.

4.1.4. Valor deseado

Si en vez de alimentar el observador con los valores medidos de la posición, se lo alimenta con una diferencia entre la misma y un valor de referencia, se obtiene un sistema como el de la Figura 4.3.

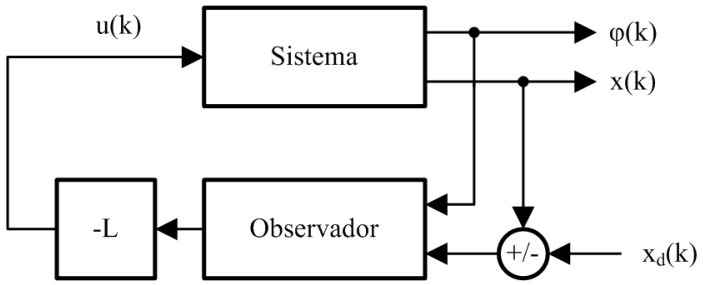


Figura 4.3: Esquema de Control en Variables de Estado con Regulación de la Posición Deseada

Esto resulta equivalente a desplazar el centro de coordenadas de la posición del carro, y por lo tanto, el esquema de realimentación tratará de llevar a cero la diferencia entre la posición y la posición deseada

$$\left(x(k) - x_d\right) \rightarrow 0 \qquad\qquad x(k) \rightarrow x_d$$

Es decir, que llevará la posición del carro, a su valor deseado. Ya que la dinámica del sistema no se ve afectada, frente a variaciones en la posición deseada, el sistema responderá entonces, de la misma forma que ante la imposición de una condición inicial.

4.1.5. Algoritmo de Diseño

Para facilitar el proceso de diseño del controlador, se implementó un programa en MATLAB. El mismo ensambla el sistema, realiza la discretización y calcula los vectores K y L de forma automática a partir de los parámetros a, b, c, y d. (ver ***modelado2.m*** en el Anexo IV)

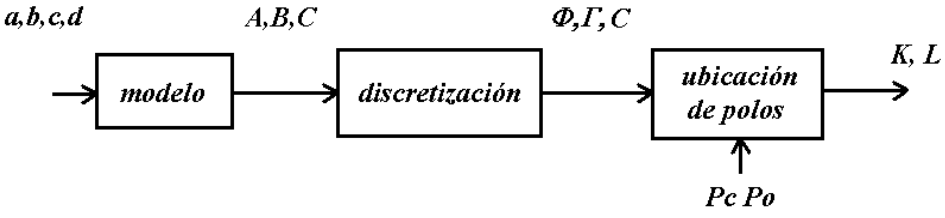


Figura 4.4: Algoritmo de diseño del controlador en Variables de Estado

De esta forma resulta muy fácil recalcular los vectores del controlador en caso de modificarse los parámetros del sistema o los valores deseados para los polos.

4.1.6. Implementación

La implementación resulta muy sencilla utilizando el esquema visto en la sección 2.3.2. Simplemente se agregaron las líneas de código correspondientes al control y a la estimación del vector de estados, dentro del bucle principal.

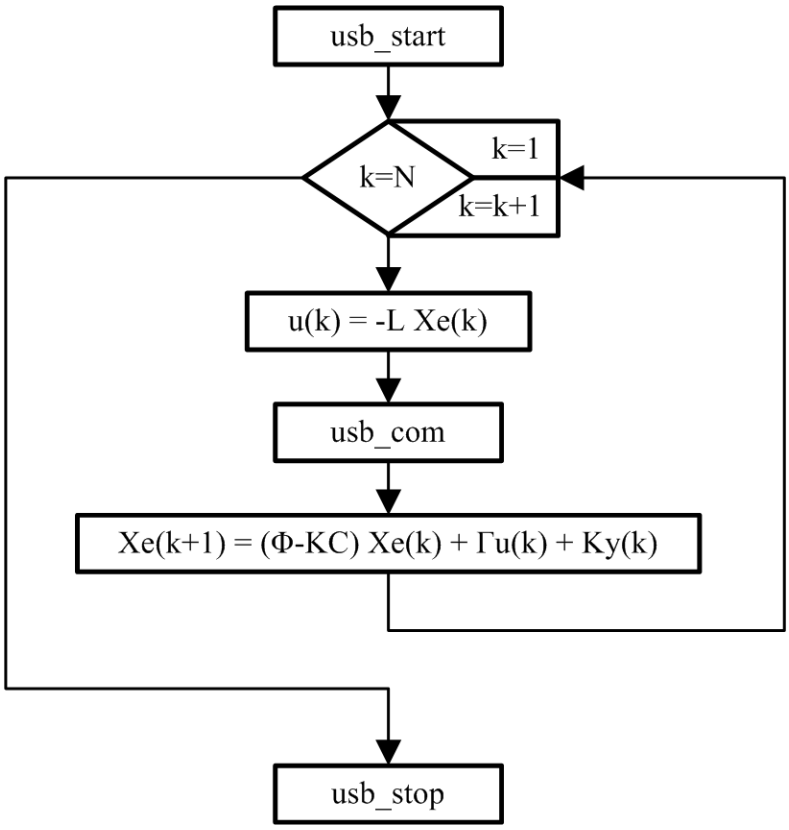


Figura 4.5: Diagrama de Flujo del Algoritmo de Control en Variables de Estado

```
h=0.02;
N=1000;
usb_start;

for (i=1:N)
     u(i)=-L*xe(:,i);
     usb_com;
     xe(:,i+1)=(F-K*C)*xe(:,i)+G*u(i)+K*[x(i);f(i)];
end

usb_stop;
```

Figura 4.6: Código fuente en MATLAB del algoritmo de control por Variables de Estado

El algoritmo se debe iniciar con el péndulo en proximidades de la posición vertical superior, ya que de lo contrario, no son válidas las aproximaciones hechas en la deducción del modelo y por lo tanto el sistema resultaría ser no lineal.

4.1.7. Resultados

A continuación se presentan los resultados de algunos de los ensayos realizados con el equipo y se los compara con simulaciones del modelo teórico del sistema. En la figura 4.7 se muestra la respuesta del sistema ante una condición inicial 0.6 en la posición del carro.

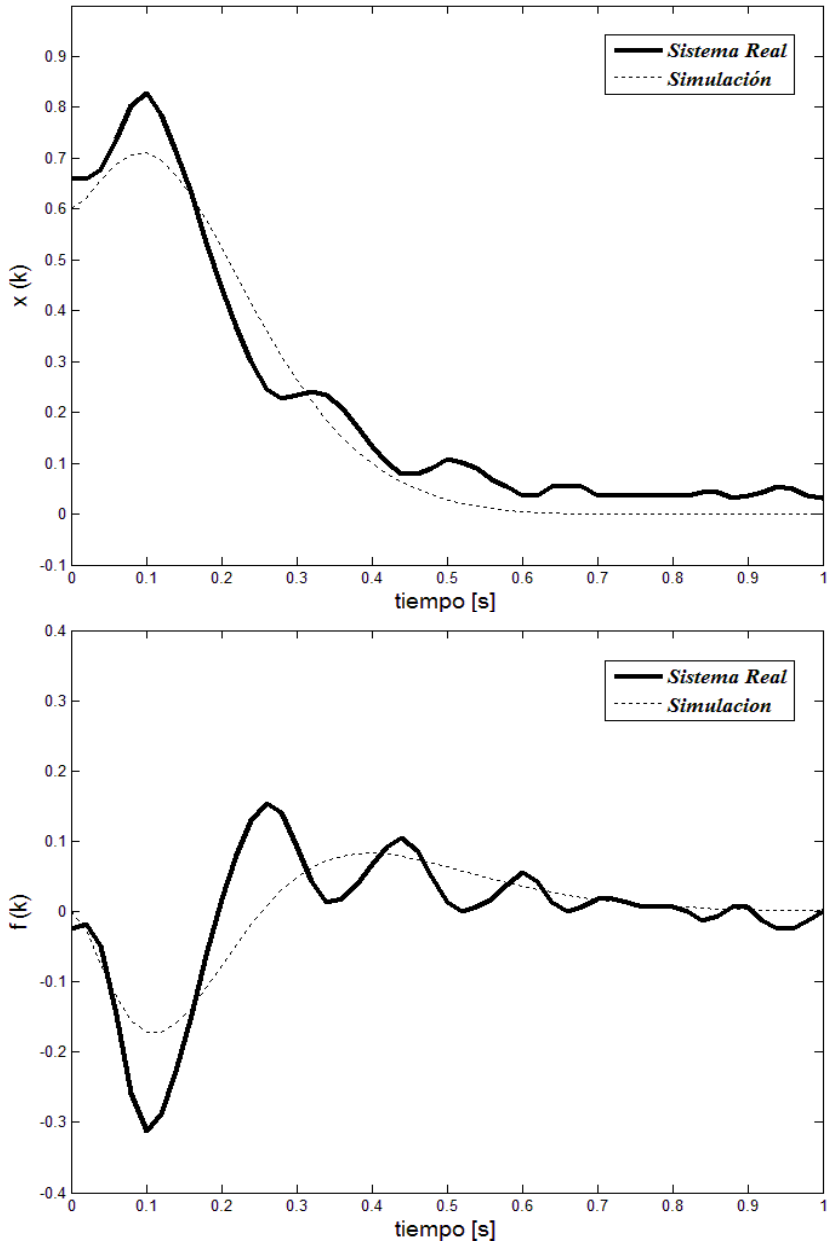


Figura 4.7: Respuesta del sistema con control en Variables de Estado ante una condición inicial en la posición

Se observa la presencia de oscilaciones de alta frecuencia de aproximadamente 10 Hz, producidas por la interacción entre el ruido la dinámica del observador *(ver Sección 4.3.5 y 4.6.)*.

En la Figura 4.8 podemos observar la simulación y la respuesta del sistema real ante una entrada de onda cuadrada en la posición deseada del carro.

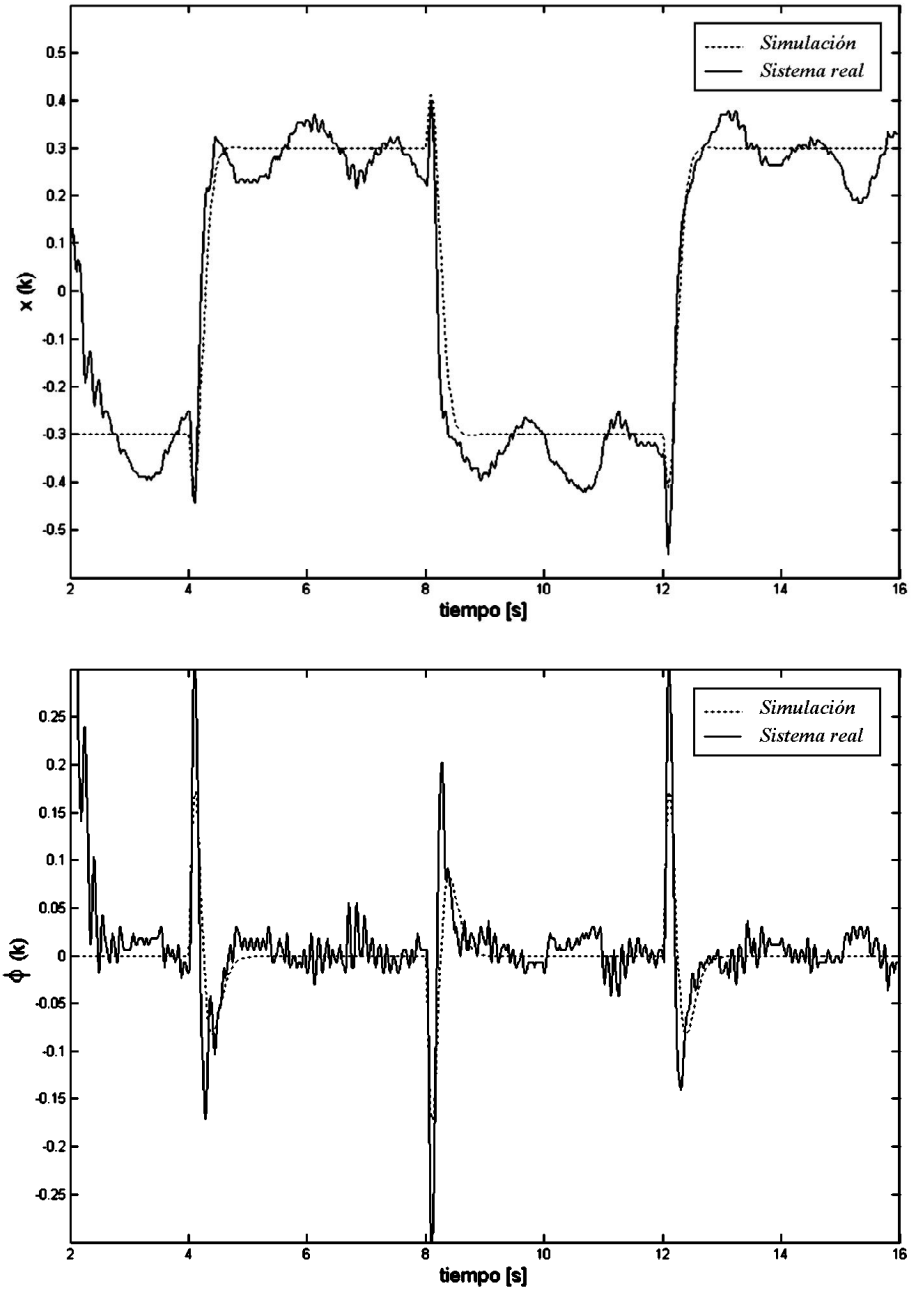


Figura 4.8: Simulación y respuesta del sistema real ante variaciones en la posición deseada

Aquí vemos la presencia de oscilaciones de baja frecuencia de aproximadamente 0.5 Hz, producidas por la dinámica del observador *(ver Sección 4.3.5 y 4.6.).*

4.2. Control en Enfoque Polinomial

Esta sección trata el problema del control del péndulo invertido utilizando funciones de transferencia discreta, en transformada z.

4.2.1. Función de Transferencia de la Realimentación de Estados

A partir del controlador basado en variables de estado desarrollado en la sección anterior, es posible obtener un controlador equivalente en función de transferencia.

Aplicando transformada Z a la ecuación del observador y combinando con la ley de realimentación de estados, obtenemos:

$$\hat{X}(k+1) = \Phi\,\hat{X}(k) + \Gamma\,u(k) + K\left[\, y(k) - C\,\hat{X}(k)\,\right]$$

$$z\hat{X} = (\Phi - KC)\,\hat{X} - \Gamma L\,\hat{X} + K\,y$$

$$(zI - \Phi + KC + \Gamma L)\,\hat{X} = K\,y$$

$$\hat{X} = (zI - \Phi + KC + \Gamma L)^{-1}\,K\,y$$

$$u = -L\hat{X} = -L(zI - \Phi + KC + \Gamma L)^{-1}\,K\,y$$

$$\frac{u}{y} = -L(zI - \Phi + KC + \Gamma L)^{-1}\,K$$

Utilizando esta función de transferencia es posible calcular directamente la señal de control u(k) a partir de las mediciones de la salida y(k). El funcionamiento y desempeño de este controlador es igual al del controlador de variables de estado de la sección anterior ya que es en realidad, una realización alternativa del mismo controlador.

4.2.2. Función de Transferencia de la planta

A partir del modelo de variables de estado es posible obtener un modelo en función de trasferencia aplicando transformada Z.

$$X(k+1) = \Phi X(k) + \Gamma u(k)$$

$$zX = \Phi X + \Gamma u$$

$$(zI - \Phi)X = \Gamma u$$

$$X = (zI - \Phi)^{-1}\Gamma u$$

$$y = CX = C(zI - \Phi)^{-1}\Gamma u$$

$$y = BA^{-1} u$$

Donde

$$C(zI - \Phi)^{-1}\Gamma = BA^{-1}$$

Siendo A y B matrices de polinomios en z de 1x1 y 2x1 respectivamente. La matriz polinomial A puede calcularse como

$$A = \det(zI - \Phi)$$

Mientras que B se calcula mediante

$$B = C(zI - \Phi)^{-1}\Gamma A$$

4.2.3. Asignación de Polos

Si aplicamos un controlador de la forma

$$u = -R^{-1}S\, y + R^{-1}T\, u_c$$

Con R, S y T siendo matrices de polinomios en z de 1x1, 1x2 y 1x1 respectivamente, obtenemos el siguiente sistema de lazo cerrado:

$$u = -R^{-1}S\, BA^{-1} u + R^{-1}T\, u_c$$

$$\left(1 + R^{-1}SBA^{-1}\right) u = R^{-1}T\, u_c$$

$$\left(AR + SB\right) u = AT\, u_c$$

$$u = \left(AR + SB\right)^{-1} AT\, u_c$$

$$y = BA^{-1} u = B\left(AR + SB\right)^{-1} T\, u_c$$

$$y = BT\left(AR+SB\right)^{-1} u_c$$

Las conmutaciones realizadas, son posibles debido a que A, R y SB resultan ser escalares. Podemos ver que los polos del sistema de lazo cerrado son las raíces del polinomio $\left(AR+SB\right)$. Este resulta ser el polinomio característico del sistema. Igualando el mismo, al polinomio característico deseado, podemos despejar los valores de los polinomios R y S necesarios para ubicar arbitrariamente los polos de lazo cerrado.

$$\left(AR+SB\right) = (z-p_1)(z-p_2)...(z-p_k)$$

4.2.4. Algoritmo de Diseño

Al igual que en el caso del diseño en variables de estado, se implementó un programa en MATLAB para facilitar el proceso de cálculo de los coeficientes del controlador. El mismo ensambla el sistema, realiza la discretización, calcula los coeficientes de las matrices A y B, y resuelve la ecuación de diseño de forma automática a partir de los parámetros a, b, c, y d. (ver ***modelado7.m*** en el Anexo IV)

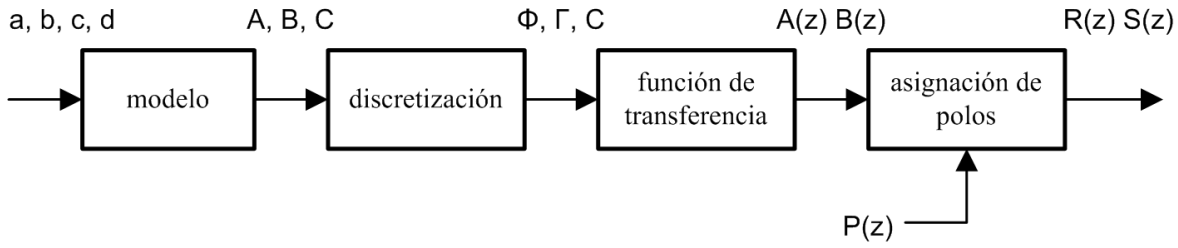


Figura 4.9: Algoritmo de diseño en enfoque polinomial

De esta forma resulta muy fácil recalcular los coeficientes del controlador en caso de modificarse los parámetros del sistema o los valores deseados para los polos.

Debido a que MATLAB no presenta funciones específicas para resolver ecuaciones diofánticas polinomiales, como lo es la ecuación de diseño, se hizo necesario elaborar un algoritmo propio a tal fin.

Para esto se ensambla una ecuación matricial, que representa el sistema de ecuaciones lineales, que resulta de igualar los coeficientes de mismo grado a ambos lados de la igualdad. El primer paso es determinar el grado de los polinomios contenidos en R y S, para esto se iguala el número de ecuaciones del sistema con la cantidad de incógnitas.

$$Ecuaciones \quad = \quad Incógnitas$$

$$grA + grR + 1 = (grR + 1) + 2(grS + 1)$$

El grado mínimo de los polinomios que satisfacen esta condición resulta

$$grR = grS = 1$$

Entonces

$$AR \quad = \quad (a_4 z^4 + a_3 z^3 + a_2 z^2 + a_1 z + a_0)(r_1 z + r_0)$$

$$SB \quad = \quad (b_{13} z^3 + b_{12} z^2 + b_{11} z + b_{10})(s_{11} z + s_{10}) + (b_{23} z^3 + b_{22} z^2 + b_{21} z + b_{20})(s_{21} z + s_{20})$$

$$P \quad = \quad (z - p_1)(z - p_2)...(z - p_5) \quad = \quad q_5 z^5 + q_4 z^4 + q_3 z^3 + q_2 z^2 + q_1 z + q_0$$

Expresando la ecuación de diseño en forma matricial

$$(AR + SB) \quad = \quad P$$

$$\begin{bmatrix} a_4 & 0 & 0 & 0 & 0 & 0 \\ a_3 & a_4 & b_{13} & 0 & b_{23} & 0 \\ a_2 & a_3 & b_{12} & b_{13} & b_{22} & b_{23} \\ a_1 & a_2 & b_{11} & b_{12} & b_{21} & b_{22} \\ a_0 & a_1 & b_{10} & b_{11} & b_{20} & b_{21} \\ 0 & a_0 & 0 & b_{10} & 0 & b_{20} \end{bmatrix} \begin{bmatrix} r_1 \\ r_0 \\ s_{11} \\ s_{10} \\ s_{21} \\ s_{20} \end{bmatrix} = \begin{bmatrix} q_5 \\ q_4 \\ q_3 \\ q_2 \\ q_1 \\ q_0 \end{bmatrix}$$

La solución del sistema de ecuaciones se obtiene invirtiendo la matriz

$$\begin{bmatrix} r_1 \\ r_0 \\ s_{11} \\ s_{10} \\ s_{21} \\ s_{20} \end{bmatrix} = \begin{bmatrix} a_4 & 0 & 0 & 0 & 0 & 0 \\ a_3 & a_4 & b_{13} & 0 & b_{23} & 0 \\ a_2 & a_3 & b_{12} & b_{13} & b_{22} & b_{23} \\ a_1 & a_2 & b_{11} & b_{12} & b_{21} & b_{22} \\ a_0 & a_1 & b_{10} & b_{11} & b_{20} & b_{21} \\ 0 & a_0 & 0 & b_{10} & 0 & b_{20} \end{bmatrix}^{-1} \begin{bmatrix} q_5 \\ q_4 \\ q_3 \\ q_2 \\ q_1 \\ q_0 \end{bmatrix}$$

4.2.5. Implementación

La implementación resulta muy sencilla agregando las líneas de código correspondientes al cálculo de la señal de control en el esquema visto en la sección 2.3.2.

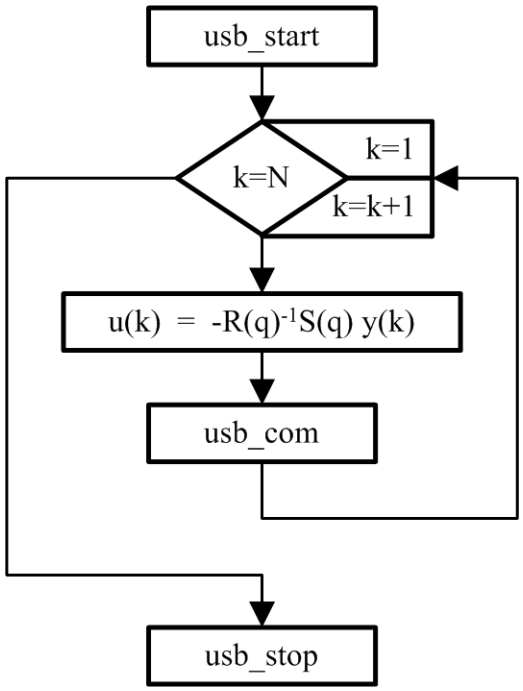


Figura 4.10: Diagrama de Flujo del Algoritmo de Control Polinomial

La implementación de la función de transferencia del controlador se realizó mediante el esquema Directo I, multiplicando los valores presentes y pasados de las señales $x(k)$, $\varphi(k)$ y $u(k)$ por los respectivos coeficientes de la función de transferencia.

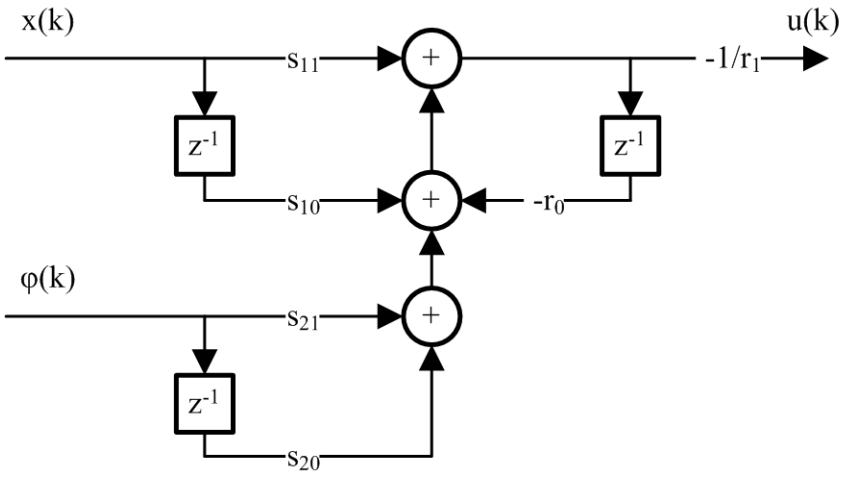


Figura 4.11: Implementación de la Función de Transferencia

```
h=0.02;
N=1000;
usb_start;

for (i=1:N)
     u(i)=-1*(S1(1)*x(i-1)+S1(2)*x(i-2)+S2(1)*f(i-1)+S2(2)*f(i-2))-R(2)*u(i-1);
     usb_com;
end

usb_stop;
```

Figura 4.12: Código fuente en MATLAB del algoritmo de Control Polinomial

Al igual que en el caso del control en variables de estado y por las mismas razones, algoritmo se debe iniciar con el péndulo en proximidades de la posición vertical superior.

4.2.6. Resultados

En la Figura 4.13 se muestra la respuesta del sistema ante una condición inicial de 0,6 en la posición.

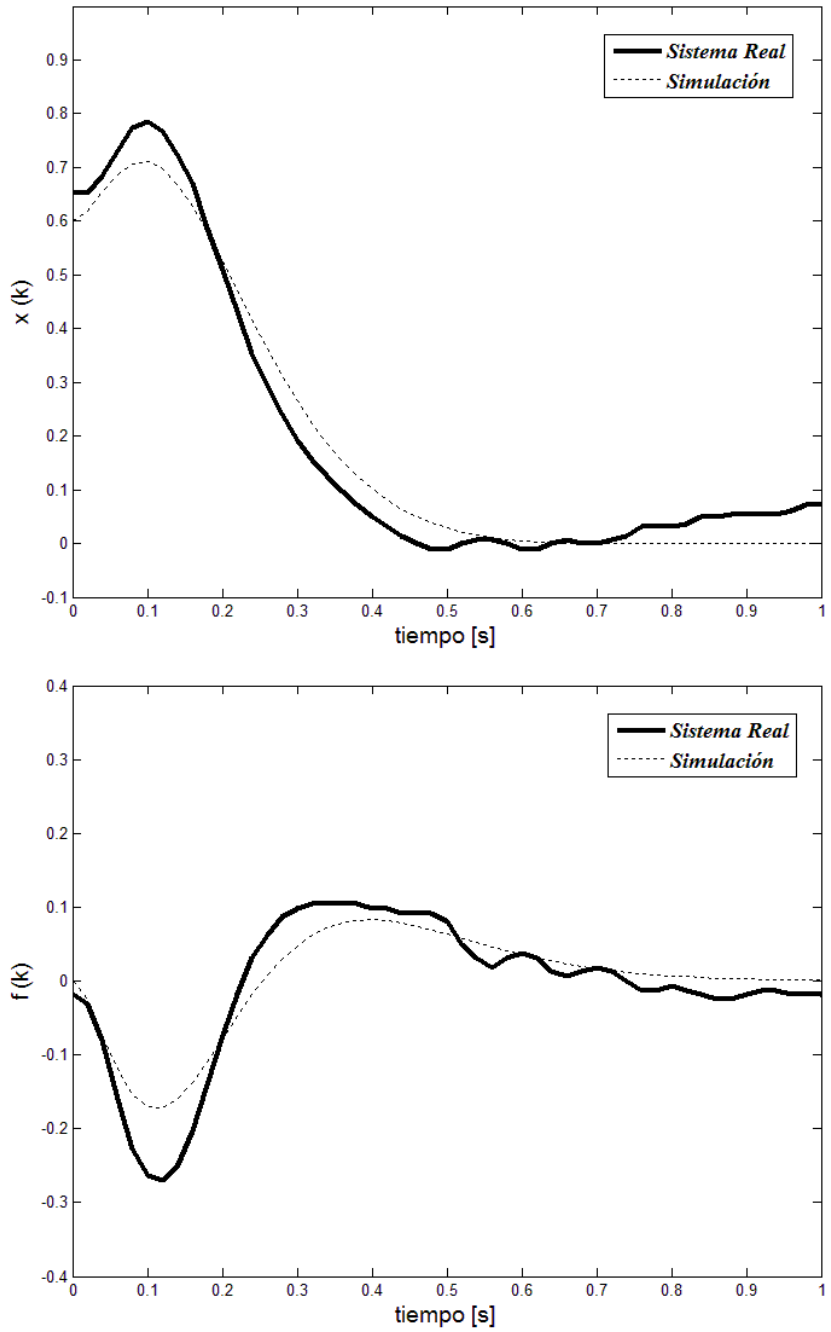


Figura 4.13: Respuesta del sistema con control polinomial ante una condición inicial en la posición

Se observa un mejor desempeño en comparación con el control en Variables de Estado, presentando un comportamiento más amortiguado.

En la Figura 4.14 podemos observar la simulación y la respuesta del sistema real ante una entrada de onda cuadrada en la posición deseada del carro.

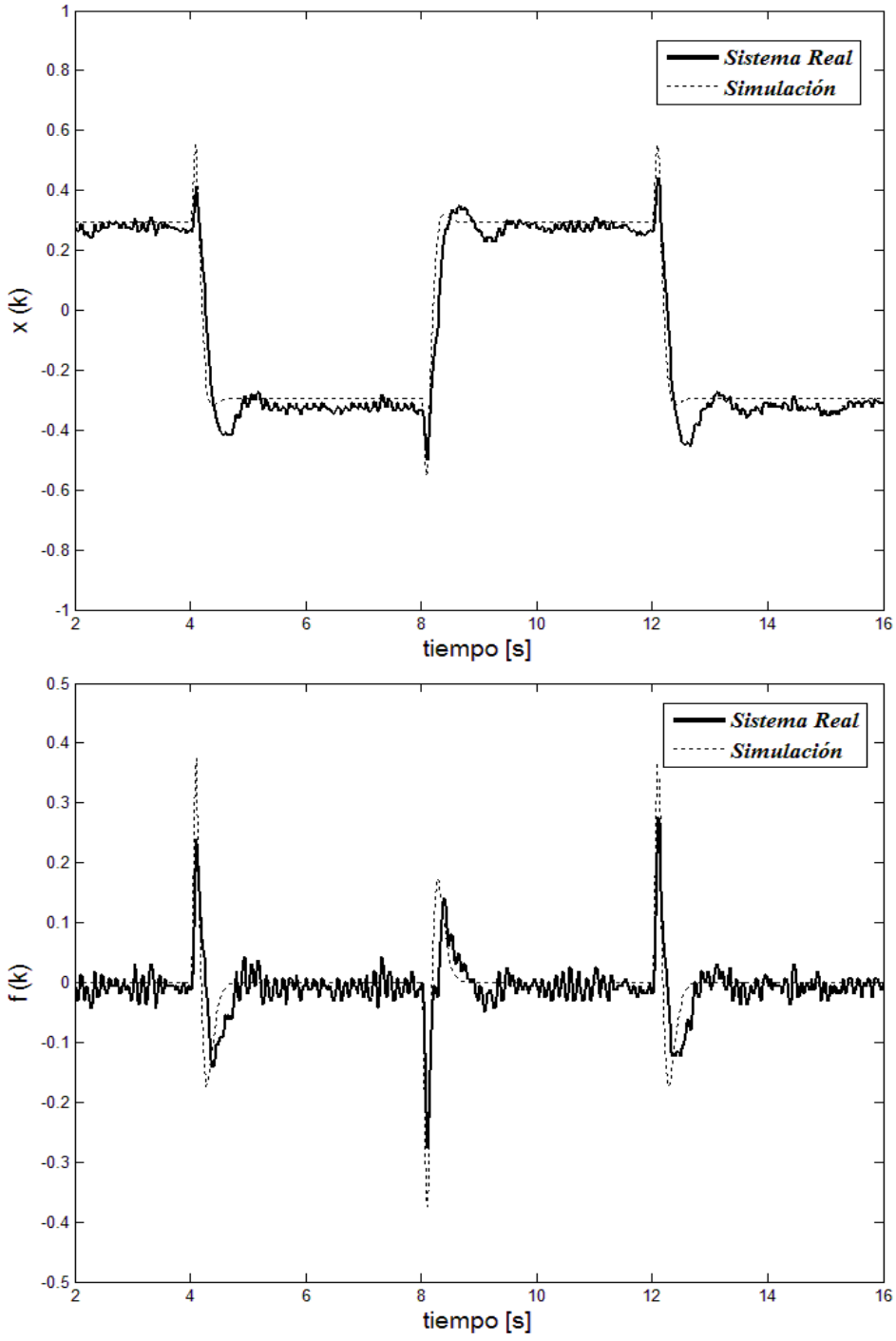


Figura 4.14: Simulación y respuesta del sistema real ante variaciones en la posición deseada

Vemos un mejor desempeño en comparación con el control en Variables de Estado, oscilaciones más amortiguadas y con una menor varianza en estado estacionario *(ver Sección 4.6.)*

4.3. Filtro de Kalman

El filtro de Kalman[18] es un algoritmo desarrollado por Rudolf E. Kalman en 1960 que sirve para poder identificar el vector de estados de un sistema dinámico lineal (de forma similar al observador de estados) cuando el sistema está sometido a ruido blanco aditivo. La ventaja que presenta el filtro de Kalman es que la ganancia de realimentación del error puede ser calculada de forma óptima cuando se conocen las varianzas de los ruidos que afectan al sistema.

4.3.1. Ganancia Óptima

Supongamos que modelo del sistema discreto esta dado por

$$X(k+1) \;=\; \Phi\, X(k) \;+\; \Gamma\, u(k) \;+\; v(k)$$

$$y(k) \;=\; C\, X(k) \;+\; e(k)$$

Donde $v(k)$ y $e(k)$ son procesos de ruido blanco discreto gaussianos independientes, con media cero y matrices de covarianza

$$\operatorname{cov} v(k) \;=\; E\left(v(k)v^T(k)\right) \;=\; R_1$$

$$\operatorname{cov} e(k) \;=\; E\left(e(k)e^T(k)\right) \;=\; R_2$$

Considerando un estimador (predictor de un paso) de la forma

$$\hat{X}(k+1) \;=\; \Phi\, \hat{X}(k) \;+\; \Gamma\, u(k) \;+\; K(k)\left[\; y(k) - C\, \hat{X}(k) \;\right]$$

Donde $\hat{X}(k+1)$ es la predicción de $X(k)$ para el próximo paso, dados los datos hasta el instante k. Su diferencia es el error de estimación:

$$\tilde{X}(k) = X(k) - \hat{X}(k)$$

Restando la ecuación del observador de la ecuación del sistema obtenemos

$$X(k+1) - \hat{X}(k+1) = \Phi\, X(k) + \Gamma\, u(k) + v(k) - \Phi\, \hat{X}(k) - \Gamma\, u(k) - K(k)\left[\, C\, x(k) + e(k) - C\, \hat{X}(k)\right]$$

$$\tilde{X}(k+1) \;=\; \left[\Phi - K(k)C\right]\, \tilde{X}(k) + v(k) - K(k)e(k)$$

El criterio para encontrar el K óptimo es minimizar la varianza del error de estimación

$$P(k+1) \;=\; E\left(\tilde{X}(k+1)\tilde{X}(k+1)^T\right)$$

$$=\; E\left(\left[\Phi - K(k)C\right]\, \tilde{X}(k)\tilde{X}(k)^T\left[\Phi - K(k)C\right]^T + v(k)v(k)^T + K(k)e(k)e(k)^T K(k)^T\right)$$

$$P(k+1) \;=\; \left[\Phi - K(k)C\right]\, P(k)\left[\Phi - K(k)C\right]^T + R_1 + K(k)R_2K(k)^T$$

Ya que $\tilde{X}$, $v(k)$ y $e(k)$ son independientes, tenemos

$$P(k+1) \;=\; \Phi P(k)\Phi^T + R_1 - K(k)CP(k)\Phi^T - \Phi P(k)C^T K^T(k) + K(k)\left(R_2 + CP(k)C^T\right)K^T(k)$$

El mínimo se obtiene derivando [19] esta expresión con respecto a $K(k)$ e igualando a cero

$$\frac{\partial P(k+1)}{\partial K(k)} \;=\; -2\Phi P(k)C^T + 2K(k)\left(R_2 + CP(k)C^T\right) \;=\; 0$$

Obteniéndose

$$K(k) = \Phi P(k)C^T\left(R_2 + CP(k)C^T\right)^{-1}$$

$$P(k+1) \;=\; \Phi P(k)\Phi^T + R_1 - \Phi P(k)C^T\left(R_2 + CP(k)C^T\right)^{-1} CP(k)\Phi^T$$

Estas expresiones permiten estimar recursivamente la varianza del error de estimación $P(k)$, y a partir de ésta, calcular el vector de realimentación $K(k)$ óptimo.

4.3.2. Caracterización del Ruido

Para calcular correctamente la ganancia de realimentación del error, es necesario conocer primero las matrices de covarianza de los ruidos R_1 y R_2.

El cálculo de R_2 se realiza a partir de supuestos sobre la distribución de probabilidad del error de medición. El error de medición $e(k)$ consiste únicamente del error de cuantificación de los encoders. El mismo suele considerarse como ruido blanco aleatorio con distribución uniforme [20] en el intervalo $\pm\frac{\Delta x}{2}$, donde Δx es la distancia minima que puede distinguir el encoder. Esta distribución tiene media cero y varianza $\frac{\Delta x^2}{12}$.

$$e_x(k) \sim U\left(-\frac{\Delta x}{2}, \frac{\Delta x}{2}\right) \qquad E\left(e_x(k)\right) = 0 \qquad E\left(e_x(k)e_x(k)^T\right) = \frac{\Delta x^2}{12}$$

$$e_\varphi(k) \sim U\left(-\frac{\Delta \varphi}{2}, \frac{\Delta \varphi}{2}\right) \qquad E\left(e_\varphi(k)\right) = 0 \qquad E\left(e_\varphi(k)e_\varphi(k)^T\right) = \frac{\Delta \varphi^2}{12}$$

La resolución de los encoders se calcula dividiendo el rango de variación de la variable por la cantidad de niveles distinguibles sobre dicho rango.

$$\Delta x = \frac{2}{334} = 5.98\times10^{-3} \qquad \Delta \varphi = \frac{2\pi}{1024} = 6.13\times10^{-3}$$

El vector $e(k)$ esta compuesto por los ruidos del encoder de posición y el de ángulo

$$e(k) = \begin{bmatrix} e_x(k) \\ e_\varphi(k) \end{bmatrix}$$

La matriz de covarianza de $e(k)$ resulta

$$R_2 \;=\; E(e(k)e(k)^T) \;=\; \begin{bmatrix} 2.98 & 0 \\ 0 & 3.13 \end{bmatrix}\times10^{-6}$$

Para la implementación del filtro de Kalman supondremos que este ruido tiene distribución Gaussiana con la misma media y matriz de covarianza.

La estimación de la matriz de covarianza del ruido de carga R_1, en cambio, se deduce a partir del modelo del sistema y de los valores medidos de $y(k)$. Teniendo en cuenta el modelo del sistema en lazo cerrado

$$X(k+1) \quad = \quad (\Phi - \Gamma L)\, X(k) \quad + \quad v(k)$$

Si calculamos la covarianza a ambos lados de la igualdad, obtenemos

$$E\left(X(k+1)X^T(k+1)\right) \quad = \quad E\left(\left((\Phi - \Gamma L)\, X(k) + v(k)\right)\left((\Phi - \Gamma L)\, X(k) + v(k)\right)^T\right)$$

Ya que X(k) y v(k) son independientes, tenemos

$$R_X(k+1) \quad = \quad \Phi_{LC} R_X(k) {\Phi_{LC}}^T + R_1$$

Donde $R_X(k) = \mathrm{cov}\left(X(k)\right)$ y $\Phi_{LC} = (\Phi - \Gamma L)$.

Luego, en estado estacionario $R_X(k) = R_X(k+1)$, por lo que R_1 puede calcularse a partir de R_X

$$R_1 \quad = \quad R_X \quad - \quad \Phi_{LC} R_X {\Phi_{LC}}^T$$

Si aplicamos el mismo procedimiento a la ecuación de salida del sistema

$$y(k) \quad = \quad C\, X(k) \quad + \quad e(k)$$

Obtenemos

$$R_y \quad = \quad C\, R_X C^T \quad + \quad R_2$$

Debido a que R_y y R_2 son matrices de 2x2, no es posible deducir R_X directamente de esta ecuación, ya que se tiene un sistema con 3 ecuaciones y 10 incógnitas. Para calcular el valor de R_X, es necesario utilizar información adicional, más precisamente, conocer los estados no medibles del sistema.

Para esto se ensambla un vector de estimación de estados $Y(k)$, a partir de los valores medidos de $y(k)$ y la aproximación de sus derivadas con diferencias finitas centradas de primer orden.

$$Y(k) = \begin{bmatrix} \dfrac{y_x(k+1) - y_x(k-1)}{2h} \\ y_x(k) \\ \dfrac{y_\varphi(k+1) - y_\varphi(k-1)}{2h} \\ y_\varphi(k) \end{bmatrix}$$

Luego, como

$$\begin{aligned} y_x(k) &= x(k) + e_x(k) \\ y_\varphi(k) &= \varphi(k) + e_\varphi(k) \end{aligned}$$

Llegamos a

$$Y(k) = \begin{bmatrix} \dfrac{x(k+1) - x(k-1)}{2h} + \dfrac{e_x(k+1) - e_x(k-1)}{2h} \\ x(k) + e_x(k) \\ \dfrac{\varphi(k+1) - \varphi(k-1)}{2h} + \dfrac{e_\varphi(k+1) - e_\varphi(k-1)}{2h} \\ \varphi(k) + e_\varphi(k) \end{bmatrix} \approx \begin{bmatrix} x'(k) \\ x(k) \\ \varphi'(k) \\ \varphi(k) \end{bmatrix} + \begin{bmatrix} \dfrac{e_x(k+1) - e_x(k-1)}{2h} \\ e_x(k) \\ \dfrac{e_\varphi(k+1) - e_\varphi(k-1)}{2h} \\ e_\varphi(k) \end{bmatrix}$$

Aplicando covarianza a ambos lados de la igualdad, y como $X(k)$ $e(k-1)$ $e(k)$ y $e(k+1)$ son independientes, obtenemos

$$R_Y = \operatorname{cov}\big(Y(k)\big) \approx \operatorname{cov}\big(X(k)\big) + \operatorname{cov}\left(\begin{bmatrix} \dfrac{e_x(k+1) - e_x(k-1)}{2h} \\ e_x(k) \\ \dfrac{e_\varphi(k+1) - e_\varphi(k-1)}{2h} \\ e_\varphi(k) \end{bmatrix}\right) = R_X + R_E$$

Donde

$$R_E = \begin{bmatrix} \dfrac{\operatorname{cov}(e_x)}{h} & 0 & 0 & 0 \\ 0 & \operatorname{cov}(e_x) & 0 & 0 \\ 0 & 0 & \dfrac{\operatorname{cov}(e_\varphi)}{h} & 0 \\ 0 & 0 & 0 & \operatorname{cov}(e_\varphi) \end{bmatrix}$$

Luego

$$R_X = R_Y - R_E$$

Y consecuentemente

$$R_1 = \left(R_Y - R_E\right) - \Phi_{LC}\left(R_Y - R_E\right)\Phi_{LC}{}^T$$

4.3.3. Algoritmo de Diseño

Para el cálculo de la matriz de realimentación de error $K(k)$ óptima, se implementó un algoritmo de cálculo en MATLAB. La estructura del mismo sigue los pasos que se explican a continuación.

Primero, se ensayo el equipo, con un controlador de variables de estado para obtener una secuencia de mediciones de $y(k)$ de un tamaño suficiente (por ejemplo 1600 muestras).

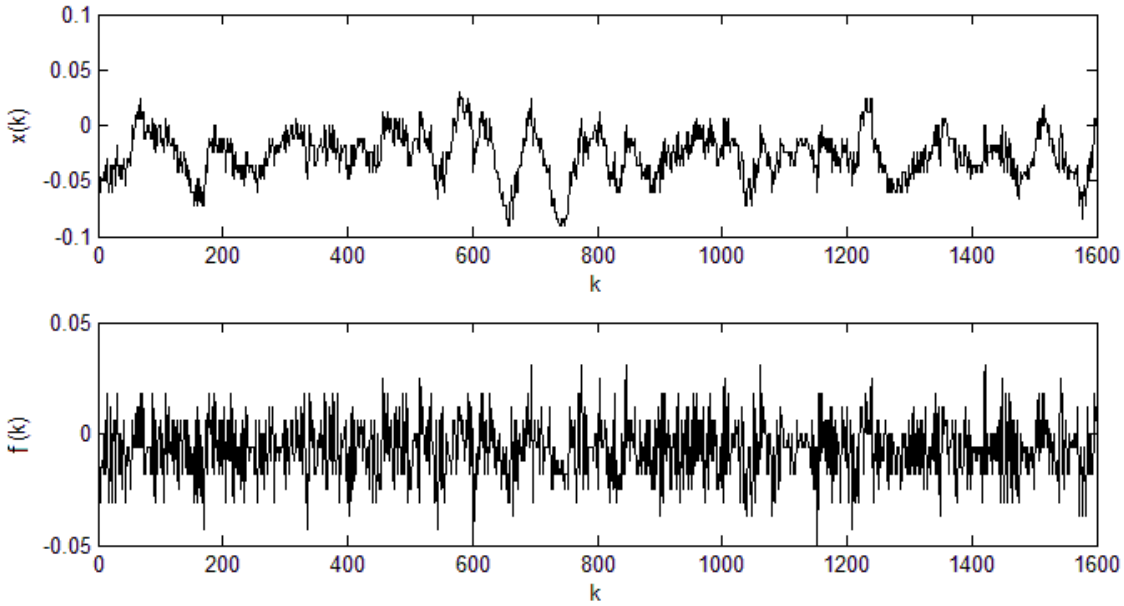


Figura 4.15: Salidas del sistema en estado estacionario con control en Variables de Estado

A partir de esos datos, se construye el vector $Y(k)$ y se calcula su matriz de covarianza R_Y, así como también la matriz de covarianza R_E del ruido presente en $Y(k)$.

$$R_Y = \begin{bmatrix} 0.1043 & 0.0000 & 0.1474 & -0.0001 \\ 0.0000 & 0.0004 & 0.0001 & 0.0002 \\ 0.1474 & 0.0001 & 0.2353 & 0.0000 \\ -0.0001 & 0.0002 & 0.0000 & 0.0002 \end{bmatrix}$$

$$R_E = \begin{bmatrix} 0.1494 & 0 & 0 & 0 \\ 0 & 0.0030 & 0 & 0 \\ 0 & 0 & 0.1569 & 0 \\ 0 & 0 & 0 & 0.0031 \end{bmatrix} \times 10^{-3}$$

Aplicando $R_X = R_Y - R_E$ obtenemos

$$R_X = \begin{bmatrix} 0.1041 & 0.0000 & 0.1474 & -0.0001 \\ 0.0000 & 0.0004 & 0.0001 & 0.0002 \\ 0.1474 & 0.0001 & 0.2351 & 0.0000 \\ -0.0001 & 0.0002 & 0.0000 & 0.0002 \end{bmatrix}$$

Finalmente, aplicando $R_1 = R_X - \Phi_{LC} R_X {\Phi_{LC}}^T$ obtenemos

$$R_1 = \begin{bmatrix} 0.0337 & -0.0003 & 0.0863 & 0.0008 \\ -0.0003 & 0.0000 & -0.0002 & 0.0000 \\ 0.0863 & 0.0001 & 0.1751 & 0.0009 \\ 0.0008 & 0.0000 & 0.0009 & 0.0000 \end{bmatrix}$$

A partir de estos valores de R_1 y R_2 se calculan recursivamente los valores de $P(k)$ y $K(k)$ utilizando, partiendo de un valor inicial arbitrario de $P(0)$

$$P(k+1) = \Phi P(k) \Phi^T + R_1 - \Phi P(k) C^T \left(R_2 + C P(k) C^T \right)^{-1} C P(k) \Phi^T$$

$$K(k) = \Phi P(k) C^T \left(R_2 + C P(k) C^T \right)^{-1}$$

La convergencia se logra prácticamente en 3 pasos, obteniéndose el valor de estado estacionario de $K(\infty)$.

$$K(\infty) = \begin{bmatrix} 5.46 & -0.204 \\ 0.928 & -0.026 \\ -31.2 & 49.1 \\ -0.527 & 1.91 \end{bmatrix}$$

Debido a esta convergencia tan rápida, pierde sentido la implementación de la versión variante en el tiempo del Filtro de Kalman. Es por esto que en este trabajo solamente se implementó la versión de estado estacionario.

4.3.4. Implementación

El algoritmo utilizado para la implementación del Filtro de Kalman es el mismo que se utilizó en la sección 4.1.6 para el control en Variables de Estado. Simplemente se reemplaza el vector de realimentación del error K por su valor óptimo estacionario $K(\infty)$.

4.3.5. Resultados

En la Figura 4.16 se muestra la respuesta del sistema ante una condición inicial de 0,6 en la posición.

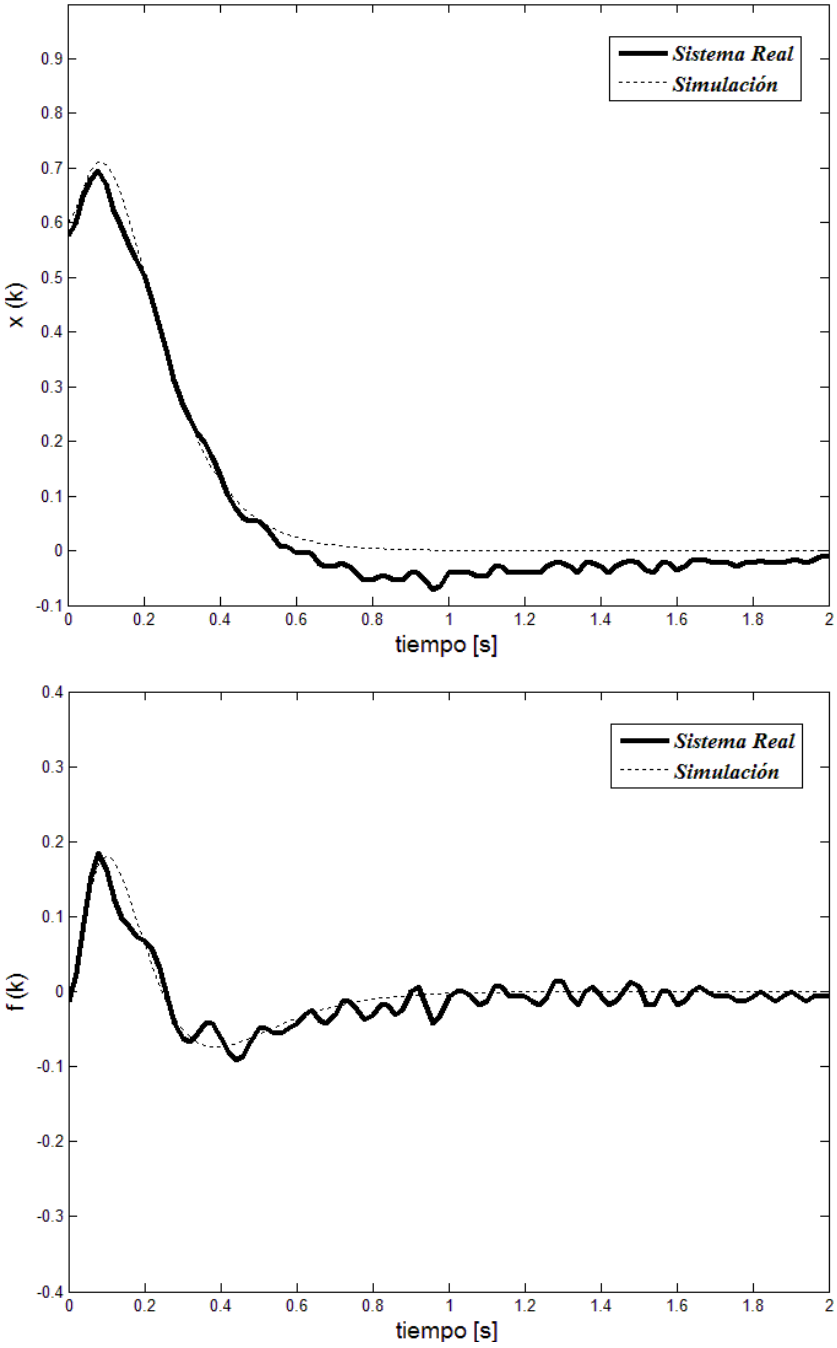


Figura 4.16: Respuesta del sistema con Filtro de Kalman ante una condición inicial en la posición

En la Figura 4.17 podemos observar la simulación y la respuesta del sistema real ante una entrada de onda cuadrada en la posición deseada del carro.

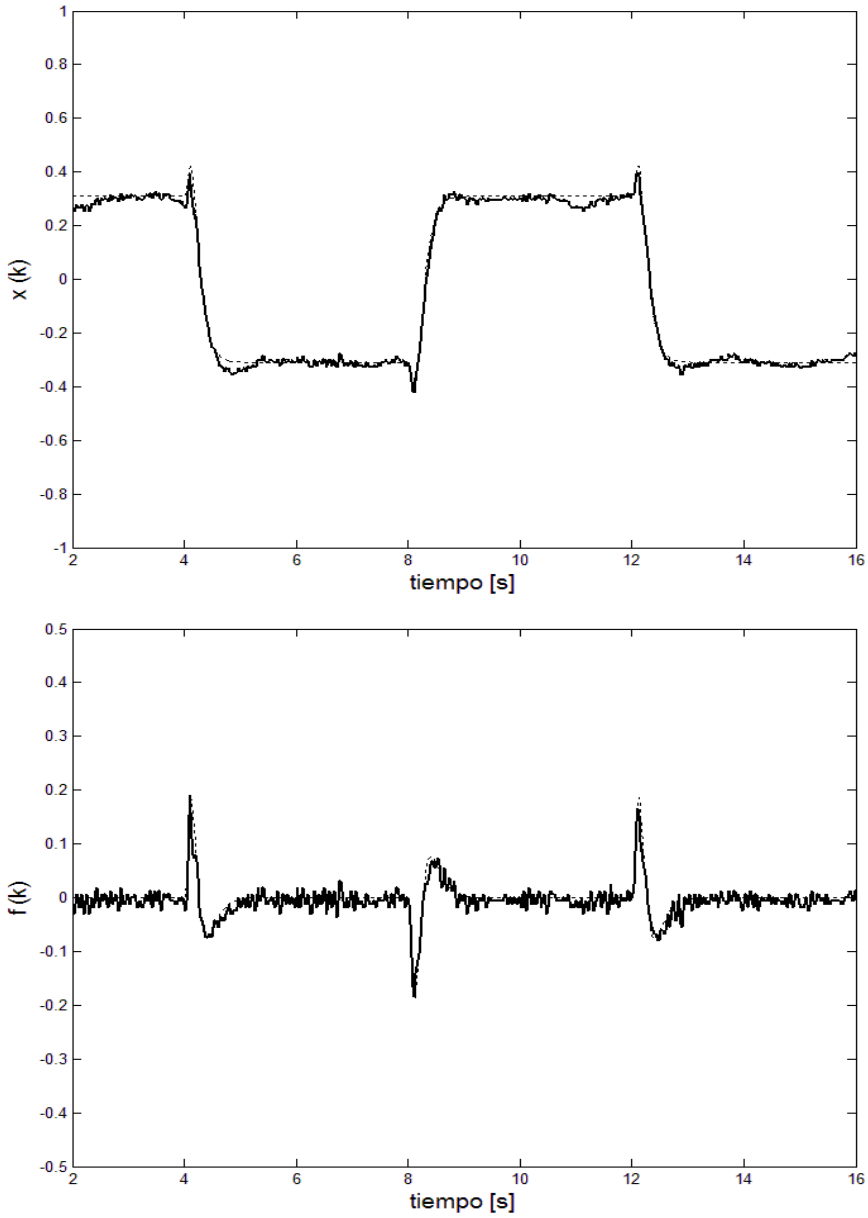


Figura 4.17: Simulación y respuesta del sistema real ante variaciones en la posición deseada

Vemos una notable mejoría en el desempeño en comparación con el control en Variables de Estado, una reducción del sobreimpulso y con una menor varianza en las partes planas. Si se tiene en cuenta que el vector de realimentación de estados L es el mismo, y que la única diferencia entre ambos es la elección de la ganancia K, podemos atribuir los problemas del controlador en Variables de Estado a su Observador *(ver Sección 4.6.)*.

4.4. Control No Lineal

Debido a que el péndulo en si mismo es un sistema no lineal, el control Polinomial y en Variables de Estado solamente funciona dentro de un rango reducido de variación del ángulo $\phi(k)$ (alrededor de su posición vertical). Para poder llevar el péndulo, desde su posición inferior hasta la posición superior, es necesario implementar un algoritmo de control no lineal, como por ejemplo, el control basado en energía [21][22].

4.4.1. Control Basado en Energía

El Control Basado en Energía consiste en aplicar una estrategia de control tal que la energía del péndulo se incremente hasta alcanzar el valor que tendría en la posición superior. Luego es posible capturar y estabilizar el péndulo en esa posición conmutando al control clásico.

Volviendo al modelo del péndulo, esta vez no realizaremos simplificaciones que reduzcan los límites de variación del ángulo

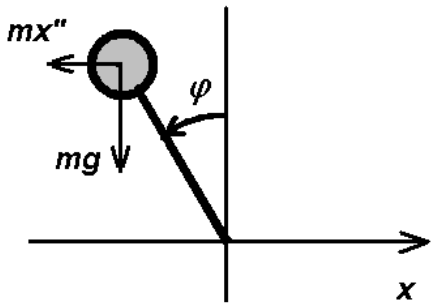


Figura 4.18: Modelo Simplificado del Péndulo Invertido

Visto desde un marco de referencia inercial solidario al eje del péndulo, la energía del péndulo es la suma de su energía potencial gravitatoria y su energía cinética de rotación.

$$E = E_p + E_c$$

$$E = mg\ell\,(\cos\varphi + 1) + \frac{1}{2} J\varphi'^{\,2}$$

Donde la posición de reposo corresponde a la energía cero. La derivada de dicha energía es

$$\frac{dE}{dt} = -mg\ell\,\varphi' sen\varphi + J\varphi'\varphi''$$

Combinando esta ecuación con la ecuación diferencial del péndulo

$$J\varphi'' = \ell mg\ sen\varphi + \ell m x''\cos\varphi$$

Obtenemos:

$$\frac{dE}{dt} = -mg\ell\,\varphi' sen\varphi + \varphi'\left(\ell mg\ sen\varphi + \ell m x''\cos\varphi\right)$$

$$\frac{dE}{dt} = \ell m\ x''\varphi'\cos\varphi$$

Para que la energía del péndulo aumente, su derivada debe ser positiva. Para esto es necesario que:

$$sign(x'') = sign(\varphi'\cos\varphi)$$

Para que la energía del péndulo disminuya el signo de la aceleración del carro debe ser el contrario

$$sign(x'') = -sign(\varphi'\cos\varphi)$$

Entonces, se puede aplicar una ley de control proporcional para llevar la energía del péndulo al valor deseado

$$x'' = -k\left(E - E_0\right) sign\left(\varphi'\cos\varphi\right)$$

Donde E_0 es el nivel de energía deseado y k es la ganancia de la realimentación. El valor de k es un parámetro del diseño que puede determinarse empíricamente.

A fines prácticos, podemos considerar el carro prácticamente detenido durante la fase de levantamiento del péndulo (swing-up), es decir $x' \approx 0$.

$$\frac{x}{u} = \frac{ba}{s\left(s+a\right)} \qquad \Rightarrow \qquad x'' + ax' = bau \qquad \Rightarrow \qquad x'' \approx ba\,u$$

En estas condiciones la aceleración del carro es proporcional a la señal de control, lo que nos lleva a la siguiente ley de control

$$u = -sat\left(\frac{k}{ba}\left(E - E_0\right)\right) sign\left(\varphi'\cos\varphi\right)$$

En la misma se incluye una función de saturación para evitar que el valor de la señal de control alcance valores excesivamente grandes.

4.4.2. Implementación

Al igual que en los casos anteriores, la implementación resulta muy sencilla utilizando el esquema visto en la sección 2.3. Simplemente se agregaron las líneas de código correspondientes al control por energía, al control por variables de estado y a la estimación del vector de estados, dentro del bucle de ejecución. Se incluyó además, una estructura de decisión para elegir cual de los dos controles se utilizará en cada momento.

A diferencia de los casos anteriores, el algoritmo se puede iniciar con el péndulo en cualquier posición, incluso con el péndulo detenido, ya que el control por energía se encarga de realizar los movimientos necesarios para elevarlo a su posición vertical superior.

Otra forma de implementar el algoritmo para el cálculo del signo de la señal de control, es haciendo uso de la derivada del seno del ángulo de desviación, ya que

$$\frac{d}{dt} sen\varphi = \varphi' \cos\varphi$$

El signo de esta derivada puede calcularse comparando el valor del seno de φ en el instante actual con el valor anterior.

$$sign\left(\varphi' \cos\varphi\right) = sign\left(sen\varphi(k) - sen\varphi(k-1)\right)$$

Entonces, es posible obtener otra realización del algoritmo reemplazando esta parte de la ley de control.

Para hacer más fácil la implementación del algoritmo, se normalizó el valor de la energía para que la misma sea cero en el estado de reposo y uno en la posición de equilibrio superior. El valor deseado para la energía es entonces $E_0 = 1$.

En las figuras 4.19 y 4.20 se muestran el diagrama de flujo y el código fuente en MATLAB del algoritmo de control implementado

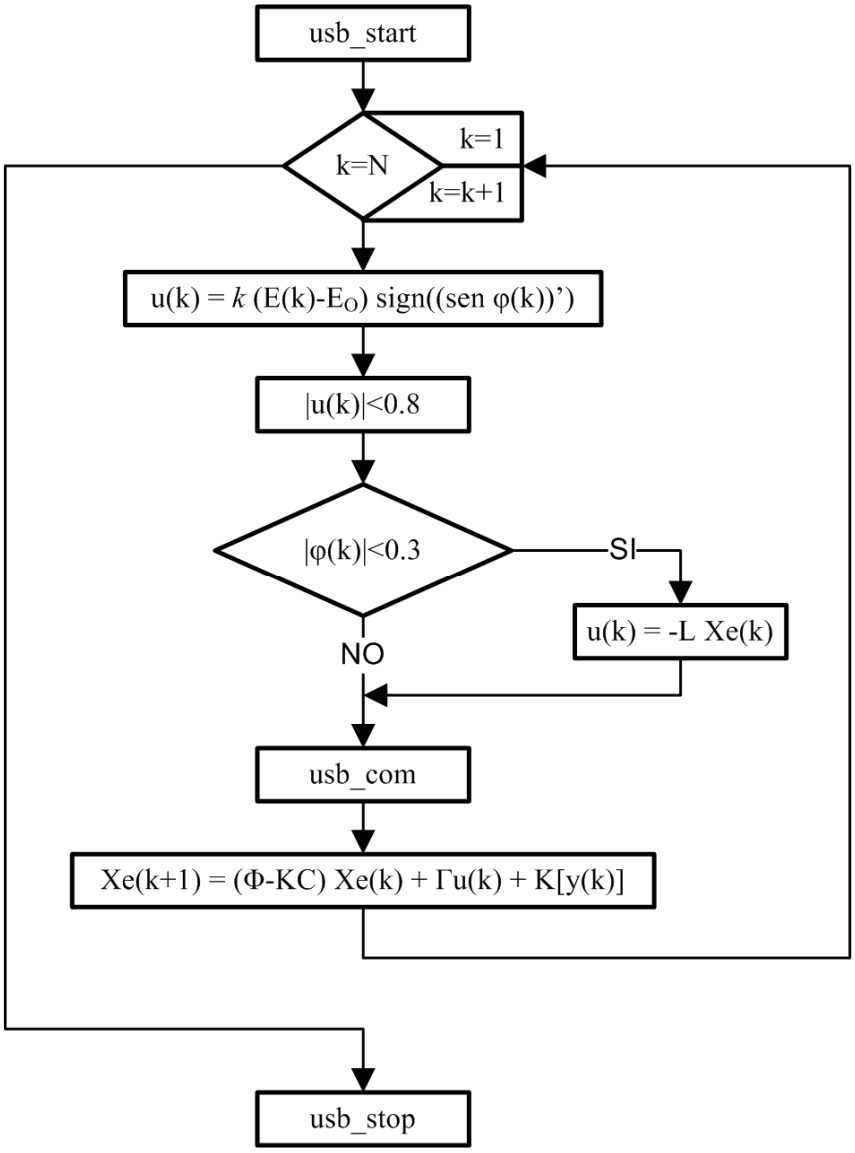


Figura 4.19: Diagrama de Flujo del Algoritmo de Control por Energía

```
h=0.02;
N=1000;

usb_start;

for (i=3:N)

     u(i)= -3*(E(i-1)-1)*sign(sen_fi(i-1)-sen_fi(i-2))-0.1*x(i-1);

     if(u(i)>0.8)
        u(i)=0.8;
     end
     if(u(i)<-0.8)
        u(i)=-0.8;
     end

     if((abs(f(i-1))<0.3)&(E(i-1)<1.3))
        u(i)=-L*xe(:,i);
     end

     usb_com;
     xe(:,i+1)=(F-K*C)*xe(:,i)+G*u(i)+K*[x(i);f(i)];

end

usb_stop;
```

Figura 4.20: Código fuente en MATLAB del algoritmo de Control por Energía

4.4.3. Resultados

En las figura 4.21 se muestran los resultados de los ensayos realizados con el control por energía. En las mismas se observa como el péndulo realiza oscilaciones de amplitud creciente hasta alcanzar el punto de equilibrio superior en aproximadamente 3 segundos. Luego, cuando la desviación angular es menor a 0.3 radianes, se aplica el control por variables de estado para mantener al péndulo estabilizado en dicha posición.

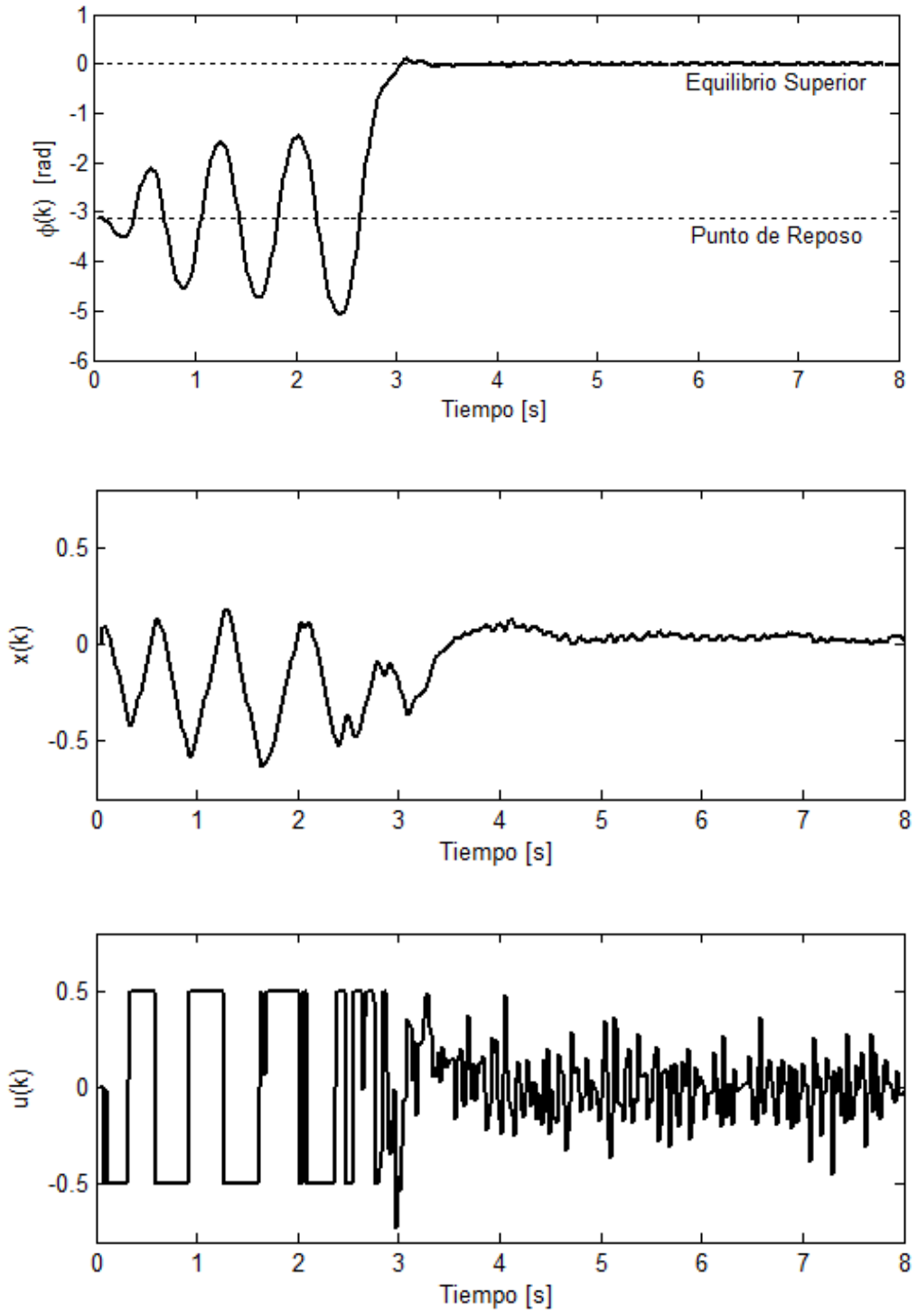


Figura 4.21: Angulo de desviación del péndulo, posición del carro y señal de control en función del tiempo, para el algoritmo de Control por Energía

En la figura 4.22 se muestra el aumento de energía producido por el algoritmo. Puede observarse que su valor es uno a partir del momento en que se alcanza la posición de equilibrio superior.

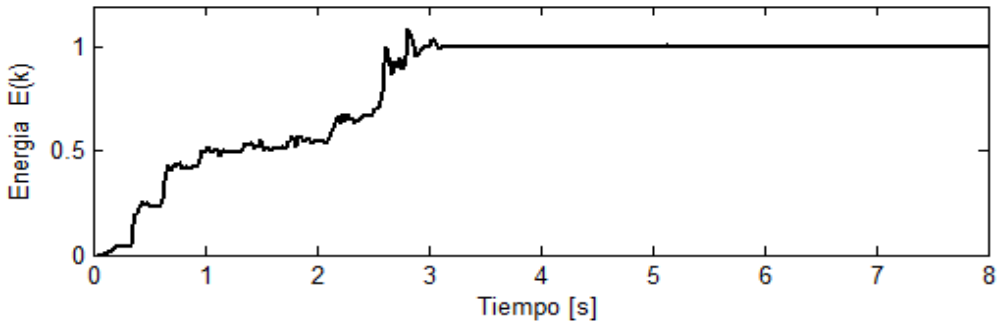


Figura 4.22: Energía del péndulo en función del tiempo

Otra cuestión interesante para analizar es el intercambio que se produce entre la energía potencial y la energía cinética del péndulo a medida que la energía total va aumentando. En la figura 4.23 se grafican los valores que van tomando dichas energías, a medida que el péndulo alcanza su punto de equilibrio superior. Se pueden observar trazos cuya pendiente es predominantemente de -45º. Esto se debe a la facilidad que tiene el péndulo de intercambiar energía potencial y cinética, en contraste a la dificultad de incrementar la energía total.

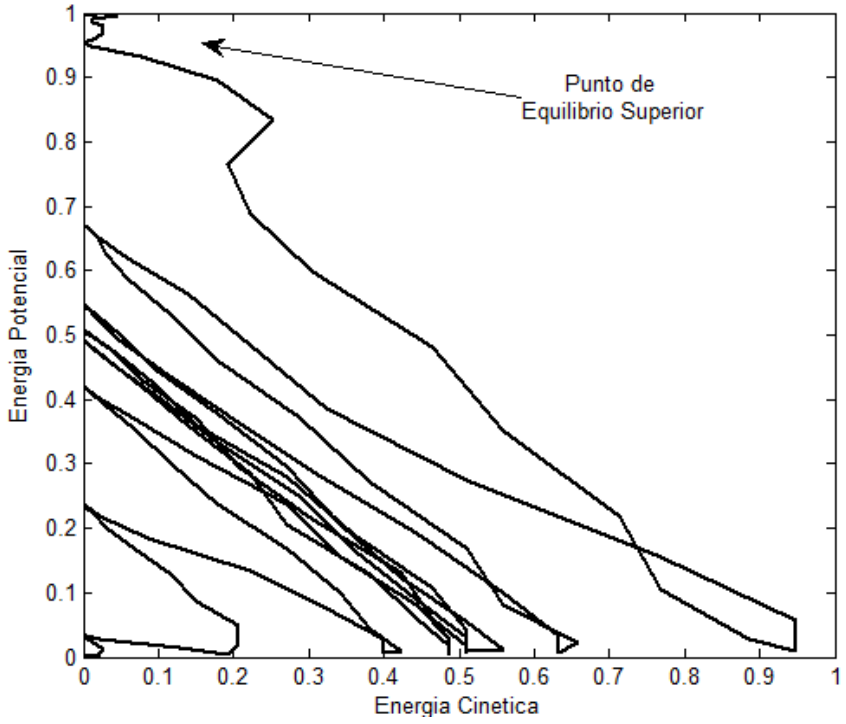


Figura 4.23: Intercambio de energía potencial y cinética a medida que el péndulo alcanza el equilibrio superior

Por último, es posible realizar un análisis del comportamiento del péndulo, en función de sus variables de estado φ y φ'.

La energía total del péndulo, expresada en función de dichas variables de estado, genera una superficie tridimensional. En estado libre, cuando el péndulo no es excitado, y despreciando el rozamiento, podemos decir que el péndulo tenderá a conservar su estado energético. Es decir que solamente podrá transitar, las líneas de nivel, de igual energía, de dicha superficie. En la figura 4.24 se han graficado dichas líneas de nivel, en función de las variables de estado φ y φ'.

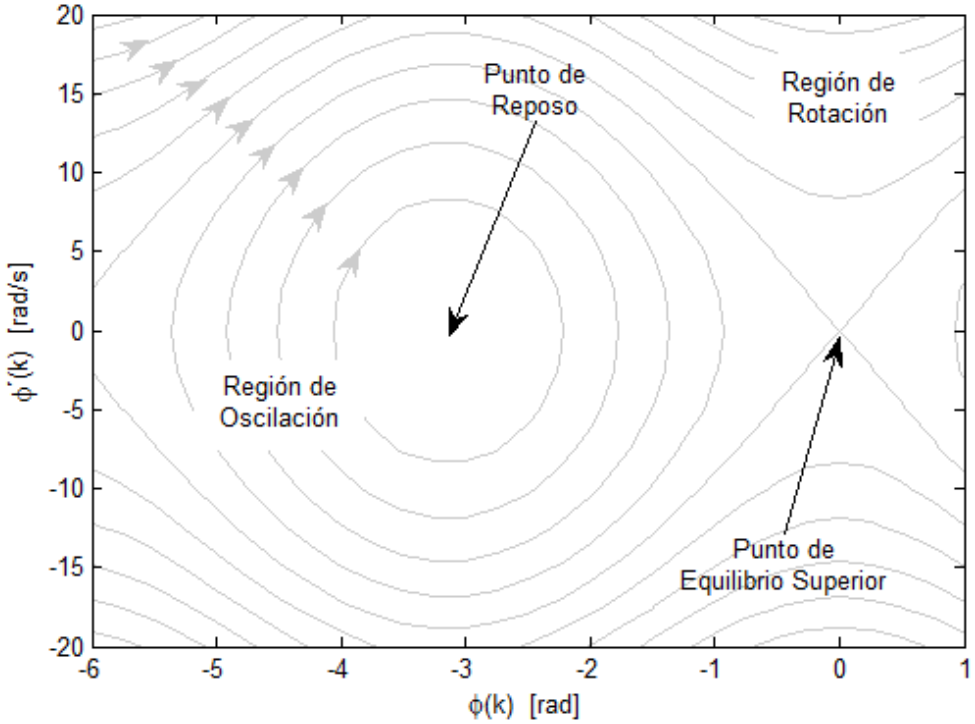


Figura 4.24: Curvas de nivel de igual energía en función de las variables de estado

En esta figura puede observarse que el punto de energía minima corresponde al punto de reposo (posición vertical inferior). Alrededor del mismo se encuentra la Región de Oscilación en la cual el péndulo oscila siguiendo el círculo de la energía correspondiente. Al adquirir la energía E=1, el péndulo puede oscilar o rotar, alcanzando el Punto de Equilibrio Superior. Mas allá de este punto, para energías superiores, se encuentra la Región de Rotación, en la cual el péndulo rota sin detenerse nunca.

Al aplicar el algoritmo de Control Basado en Energía, el péndulo puede ahora pasar de una curva de nivel a la siguiente, aumentando su energía sucesivamente. Este proceso continua hasta alcanzar la curva E=1, la que lo lleva luego hasta el Punto de Equilibrio Superior. En la figura 4.25 puede observarse la trayectoria que describe el péndulo al aplicar el algoritmo de Control Basado en Energía, comenzando desde el Punto de Reposo hasta el Punto de Equilibrio Superior.

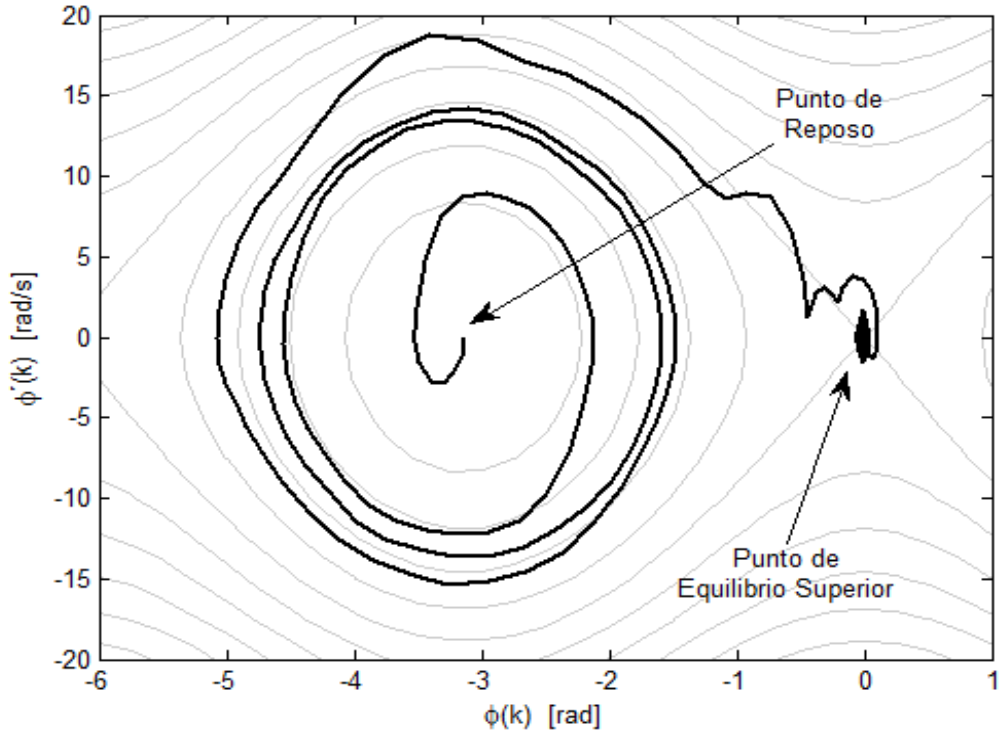


Figura 4.25: Evolución de las variables de estado al aplicar Control por Energía

4.5. Control con Coeficientes Variantes

Otra forma de mejorar el control del péndulo invertido es modelar la no linealidad del sistema utilizando un modelo lineal de coeficientes variantes. Si suponemos que el ángulo φ tiene poca variación durante un periodo de muestreo, podemos modelar el sistema como un sistema lineal variante en el tiempo, cuyos parámetros dependerán del valor de φ.

4.5.1. Modelado

Partiendo de la ecuación del péndulo invertido

$$J\varphi'' = \ell mg\ sen\varphi + \ell mx''\cos\varphi$$

Podemos reescribir la misma como

$$\varphi'' = c_\varphi\varphi + d_\varphi c_\varphi x''$$

donde

$$c_\varphi = \frac{\ell mg\ sen\varphi}{J\varphi} \qquad\qquad d_\varphi c_\varphi = \frac{\ell m}{J}\cos\varphi$$

son coeficientes que dependen de φ, los cuales supondremos como constantes para poder aplicar la transformada de Laplace.

$$s^2\varphi = c_\varphi\varphi + s^2 d_\varphi c_\varphi x$$

Obteniendo

$$\frac{\varphi}{x} = \frac{s^2 d_\varphi c_\varphi}{\left(s^2 - c_\varphi\right)} = \frac{s^2 dc\cos\varphi}{\left(s^2 - c\dfrac{sen\varphi}{\varphi}\right)}$$

Vemos que la función de transferencia del sistema es similar a la del sistema lineal. La misma se obtiene al reemplazar

$$c \quad\Rightarrow\quad c_\varphi = c\frac{sen\varphi}{\varphi} \qquad \text{y} \qquad dc \quad\Rightarrow\quad d_\varphi c_\varphi = dc\cos\varphi$$

De forma análoga, el sistema en variables de estado resulta

$$\begin{bmatrix} x''(t) \\ x'(t) \\ \varphi''(t) \\ \varphi'(t) \end{bmatrix} = \begin{bmatrix} -a & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ -adc\cos\varphi & 0 & 0 & c\dfrac{sen\varphi}{\varphi} \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x'(t) \\ x(t) \\ \varphi'(t) \\ \varphi(t) \end{bmatrix} + \begin{bmatrix} ba \\ 0 \\ badc\cos\varphi \\ 0 \end{bmatrix} u(t)$$

$$y(t) = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x'(t) \\ x(t) \\ \varphi'(t) \\ \varphi(t) \end{bmatrix}$$

O en forma compacta

$$X'(t) = A(\varphi)\, X(t) + B(\varphi)\, u(t)$$

$$y(t) = C\, X(t)$$

4.5.2. Discretización de φ

Si dividimos el intervalo $(-\pi;+\pi)$ en N sub-intervalos iguales, podemos calcular las matrices $A(\varphi)$ y $B(\varphi)$ correspondientes a cada uno de los mismos. Luego, realizando la discretización se obtienen las matrices $\Phi(\varphi)$ y $\Gamma(\varphi)$, por último se calculan los vectores de realimentación $L(\varphi)$ y $K(\varphi)$ correspondientes a la ubicación deseada de los polos del sistema de lazo cerrado para cada sub-intervalo.

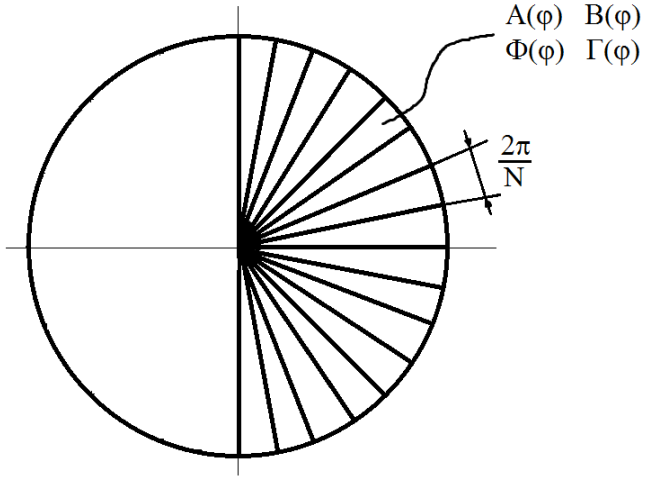


Figura 4.26: División del ángulo φ en sub-intervalos

4.5.3. Implementación

Una vez calculadas las matrices $\Phi(\varphi)$, $\Gamma(\varphi)$, $L(\varphi)$ y $K(\varphi)$ puede establecerse una ley de control de coeficientes variables. El algoritmo de control deberá entonces determinar en cual de los sub-intervalos se encuentra el ángulo φ, y luego aplicar la ley de realimentación de estados correspondiente a dicho sub-intervalo. La desventaja de utilizar este método de control es que no se puede garantizar la estabilidad, ya que se esta conmutando de un controlador a otro constantemente.

El diagrama de flujo es similar al visto en la sección 4.1 control en variables de estado, pero en este caso, tanto el vector de realimentación como las matrices del observador dependen del valor que toma el ángulo φ en el periodo de muestreo anterior. Se agregó una función de saturación para evitar que la señal de control alcance valores excesivos cuando los valores del ángulo φ sean muy elevados.

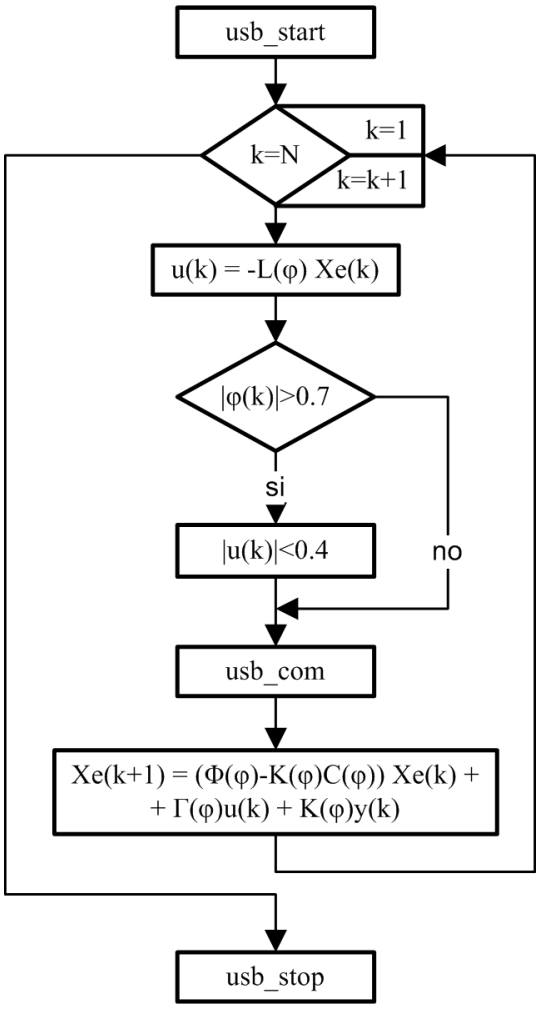


Figura 4.27: Diagrama de Flujo del Algoritmo de Control con Coeficientes Variables

```
%control coeficientes variables
h=0.02;
N=1000;
usb_start;
if (conectado == 1)
    for (i=3:N)
     k=fix(abs(NN*f(i-1)/pi))+1;
     u(i)=-LL(:,:,k)*xe(:,i);
     if(abs(f(i-1))>0.7)
       if(u(i)>0.4)
          u(i)=0.4;
       end
       if(u(i)<-0.4)
          u(i)=-0.4;
       end
     end
     usb_com;
     xe(:,i+1)=FKC(:,:,k)*xe(:,i)+GG(:,:,k)*u(i)+KK(:,:,k)*[x(i);f(i)];
    end
end
usb_stop;
```

Figura 4.28: Código Fuente del Algoritmo de Control con Coeficientes Variables

4.5.4. Resultados

En la figura 4.29 se muestran los resultados de los ensayos realizados con el control con coeficientes variantes. En la misma se observa como el péndulo realiza oscilaciones de amplitud creciente hasta alcanzar el punto de equilibrio superior en aproximadamente 3 segundos.

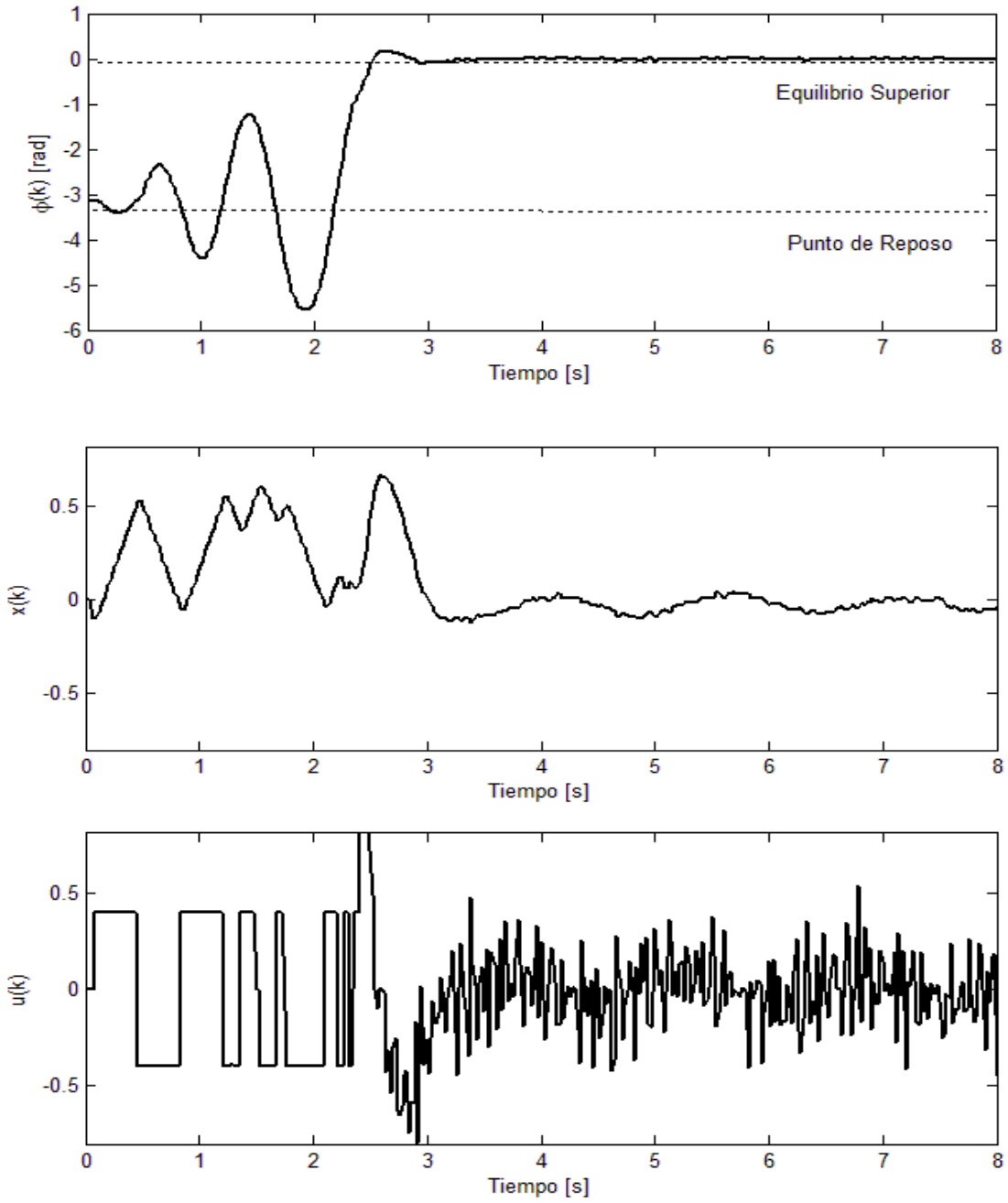


Figura 4.29: Angulo de desviación del péndulo, posición del carro y señal de control en función del tiempo, para el algoritmo de Control con Coeficientes Variantes

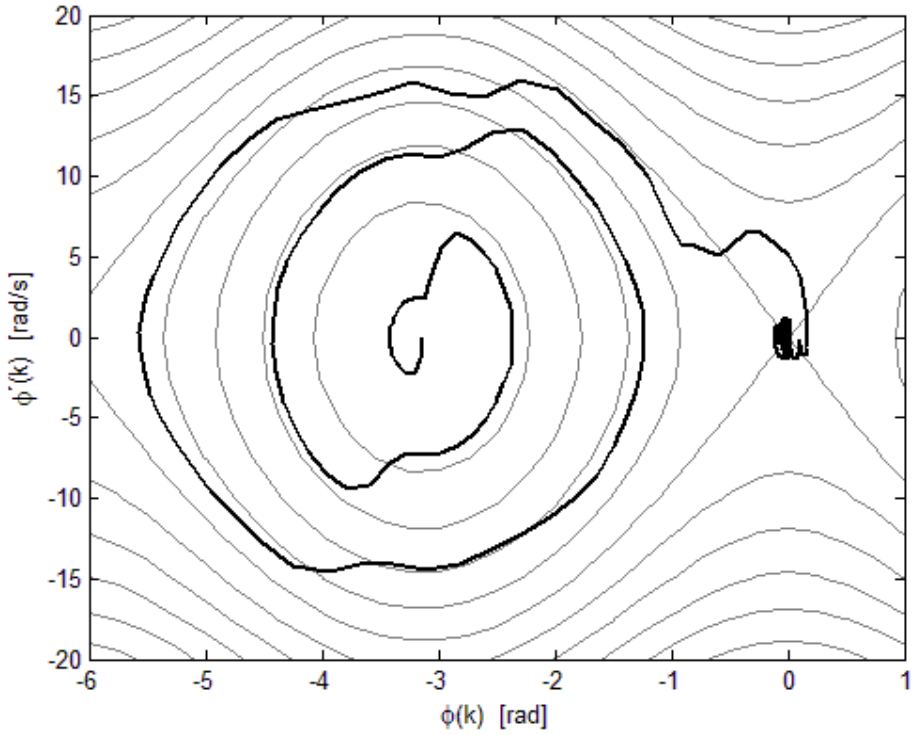


Figura 4.30: Evolución de las variables de estado al aplicar Control con Coeficientes Variantes

En la figura 4.30 se muestra el diagrama de fase del sistema. En la misma podemos observar que el péndulo realiza tres ciclos de oscilación antes de alcanzar la posición de equilibrio.

El desempeño de este controlador es similar al del control por energía. La gran diferencia consiste en que en este caso no existe la necesidad de conmutar entre cos controladores, ya que el control con coeficientes variantes realiza sin dificultad, tanto el levantamiento, como la estabilización del péndulo.

4.6. Comparación de los Métodos de Control

En esta sección se realiza una breve comparación de los algoritmos de control utilizados. Para realizar una comparación equitativa, en todos los algoritmos, se ubicaron los polos de lazo cerrado en el punto z = 0.7.

En la Figura 4.31 pueden observarse gráficamente, los índices de desempeño utilizados en la comparación.

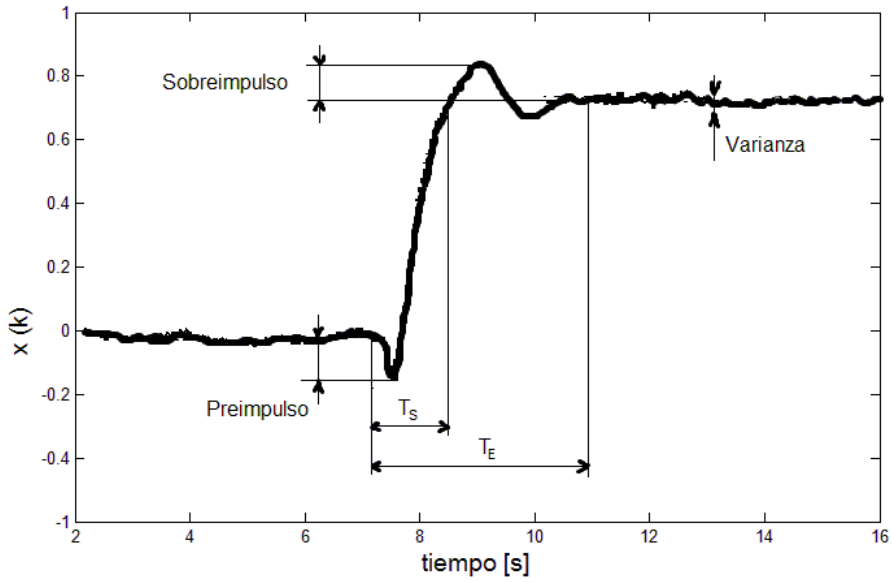


Figura 4.31: Índices de desempeño utilizados

En primer lugar, se calcularon las varianzas de las variables bajo control x(k) y φ(k), para comparar el desempeño de los algoritmos en estado estacionario. Por otro lado, se calculó tanto el sobreimpulso como el pre-impulso que presentaron ante una entrada escalón en la posición deseada, para evaluar las características dinámicas de los controladores. Así mismo, se calculó el tiempo de subida y el tiempo de establecimiento para comparar la velocidad de respuesta de los sistemas.

Los resultados pueden observarse en la Tabla 4.

	Varianza [10^{-4}]		Sobreimpulso [%]		Pre-impulso [%]		Tiempo [s]	
	x(k)	φ(k)	x(k)	φ(k)	x(k)	φ(k)	T_S	T_E
Variables de Estado	4.95	1.92	16	33	33	50	0.6	-
Control Polinomial	2.95	1.50	12	25	16	36	0.4	2
Filtro de Kalman	1.83	1.41	7	14	14	30	0.48	1

Tabla 4: comparación del desempeño de los diferentes Controladores

Podemos observar una notable superioridad del Filtro de Kalman en todos los indicadores, con excepción del tiempo de subida, en el que el Control Polinomial tuvo un tiempo menor.

Como se mencionó anteriormente, debido a que el vector de realimentación de estados es el mismo, tanto para el Filtro de Kalman como en Variables de Estado, podemos atribuir el bajo desempeño del Control en Variables de Estado al bajo rendimiento del Observador de Estados en presencia de ruido.

En cuanto a la comparación de los algoritmos de levantamiento (swing-up), los dos algoritmos utilizados, el Control por Energía y el Control con Coeficientes Variantes, mostraron un comportamiento similar en cuanto a la velocidad de levantamiento.

Cabe destacar que, debido a que se establece una limitación en la señal de control (para evitar daños en el equipo), los tiempos de levantamiento dependen de la amplitud de ese límite. Por lo tanto existe una relación de compromiso en cuanto a la velocidad de levantamiento y la violencia con la cual el motor ejecuta el algoritmo.

Para un mismo valor límite en la señal de control, los tiempos medios de levantamiento fueron de 3 segundos en ambos casos.

En cuanto a una evaluación cualitativa del desempeño general, podemos decir que el algoritmo de Control por Energía mostró un desempeño más estable frente a perturbaciones externas, logrando reducir el exceso de energía aplicada externamente de forma eficaz. El algoritmo de Control con Coeficientes Variantes en cambio, presentó ocasionalmente, comportamiento errático inesperado.

5. Conclusiones

A continuación se destacan los principales resultados alcanzados y las conclusiones extraídas de la realización del trabajo:

Se logró cumplir el objetivo de diseñar y construir un prototipo operativo del péndulo invertido, apto para ser utilizado en las prácticas de laboratorio de la cátedra de Sistemas Controlados por Computadora. El mismo, como fue demostrado a lo largo del trabajo, posee la capacidad de auto-elevarse desde la posición de reposo y mantenerse erguido el tiempo que sea necesario. Se realizaron múltiples experimentos que permitieron aplicar diferentes técnicas de identificación de sistemas, estrategias de control lineal y no lineal, obteniéndose excelentes resultados.

Se han elaborado las guías de laboratorio necesarias para que los alumnos puedan llevar a cabo los experimentos de identificación, control en variables de estado y control polinomial, así como también un breve manual de usuario que explica como conectar e instalar el equipo.

La gran mayoría de los péndulos de laboratorio implementados en el mundo son instalaciones fijas que ocupan grandes volúmenes y requieren un ambiente específico para ellos. Uno de los grandes méritos de este trabajo, fue desarrollar un péndulo portátil, de dimensiones reducidas (67cm), que pudiera ser transportado y utilizado en un aula común. Para lograr esto hubo que reducir la longitud del péndulo, lo que hace que su caída sea muy rápida, y constituye todo un desafío de control, ya que plantea grandes exigencias para los sistemas mecánicos, eléctricos y electrónicos.

Otra gran ventaja del equipo desarrollado es que puede ser operado desde cualquier PC o notebook con MATLAB, sin la necesidad de instalar ningún hardware o software adicional. Esto le brinda una gran versatilidad y compatibilidad.

Resulta importante destacar, la gran cantidad de temáticas abarcadas para la realización del proyecto, como por ejemplo: diseño de circuitos electrónicos analógicos y digitales, diseño con microcontroladores, comunicación USB, programación en diferentes lenguajes (C, Visual Basic, MATLAB), procesamiento digital de señales, sistemas controlados por computadora, procesos estocásticos, máquinas eléctricas, entre otras, lo cual le confiere un alto carácter integrador al trabajo realizado. Además se ha incursionado en algunas temáticas nuevas, no vistas durante la carrera, como por ejemplo: control polinomial multi-variable, identificación de sistemas, control no lineal y diseño mecánico.

Todos los registros generados durante el desarrollo del trabajo se adjuntan en un CD para que el equipo pueda ser replicado o reparado en caso de desperfecto. Estos registros incluyen, planos mecánicos, circuitos esquemáticos, lay-out de circuito impreso, código fuente de MATLAB y microcontroladores, listado de componentes utilizados.

Por último, es importante mencionar que el equipamiento construido queda a disposición del Departamento de Electrotecnia de la Facultad, para ser usado en las prácticas de laboratorio de las materias del Área de Control Automático y Sistemas.

5.1. Propuestas de Mejora y Futuras Líneas de Trabajo

Existen varias propuestas para perfeccionar el equipo desarrollado que podrían llevarse a cabo en futuras líneas de trabajo. A continuación se mencionan algunas de ellas:

- Implementar un lazo de control local en el microcontrolador para reducir la alinealidad del motor. Esto reduciría las incertezas en la identificación del sistema.
- Mejorar la construcción mecánica del equipo, por ejemplo utilizando un carro con menor rozamiento, evitar el uso de poleas (ya que agrega dinámica y alinealidades), utilizar un riel mas largo (que permita mayor amplitud de movimiento), etc.
- Investigar las variaciones en los parámetros del sistema provocados por factores externos como por ejemplo la temperatura.

Bibliografía

Karl J. Astrom. “Sistemas Controlados por Computadora” Paraninfo. Año 1988

Katsuhiko Ogata “Sistemas de Control en Tiempo Discreto” Prentice Hall. Año 1996

Dorf R. C., Bishop R. H. “Sistemas de Control Moderno” Prentice Hall. Año 2005

Proakis, Manolakis. “Tratamiento Digital de Señales”. Prentice Hall. Año 1998

Alan V.Oppenhein, Alan S. Wilson “Signals and Systems” Pearson Education Año 1998

Vladimir Kucera “Analysis and Design of Discrete Linear Control Systems” Prentice Hall Año 1991

Walter E. Pronzato L. “Identification of Parametric Models” Springler Año 1997

Jer-Nan Juang Minh Q. Phan “Identification and Control of Mechanical Systems” Cambridge University Press. Año 2004

Ljung L., “System Identification” Prentice-Hall, 1987.

Robert Resnick, David Halliday, “Fisica” Vol. 1 Cuarta Edición, Compañía Editorial Continental, Año 2001

Boylestad, Nashelsky, “Electrónica: Teoría de Circuitos y Dispositivos Electrónicos”, Prentice Hall, Año 2003

B. Wilamowski J. Irwin “Control and Mechatronics”, CRC Press, Año 2011

Referencias

[1] Segway, http://www.segway.com/

[2] Honda Motor Company. “U3-X Unicycle” http://world.honda.com/U3-X/

[3] Norris, James Caspar (1951), “An analysis of a compound pendulum rocket suspension”. Engineer's thesis, California Institute of Technology

[4] B. Vanderborght, B. Verrelst, R. Van Ham, M. Van Damme, and D Lefeber, (2008), “Objective locomotion parameters based inverted pendulum trajectory generador”, Robotics and Autonomous Systems, 56, 9, , 738-750 .

[5] N. J. Peruzzi and J. M. Balthazar (2007), Nonlinear Dynamics and Control of an Ideal/Nonideal Load Transportation System With Periodic Coefficients, J. Comput. Nonlinear Dynamics, 2, 1, 32-40.

[6] Y. Takahashi, O. Tsubouchi (2005), “Modern Control Approach for Robotic Wheelchair with Inverse Pendulum Control”,. 5th International Conference on Intelligent Systems Design and Applications (ISDA'05), 2005364-369, 364-369.

[7] MATLAB “The language of technical computing” http://www.mathworks.com/products/matlab

[8] USB “Universal Serial Bus” http://www.usb.org/

[9] Microchip Technology Inc. (2009) – PIC18F4550 Datasheet - High-Performance, Enhaced Flash, USB Microcontrollers with nanoWatt Technology

[10] Custom Computer Services Inc. http://www.ccsinfo.com/

[11] Microchip Technology Inc. http://www.microchip.com/

[12] Gabriel J. Pool Balam (2009), “Transferencia y procesamiento de datos a alta velocidad, mediante el uso de MATLAB, el puerto USB 2.0 y PIC18F2455 de Microchip” Universidad Modelo. Escuela de Ingenieria.

[13] International Rectifier – IRF540 N-Chanel HEXFET Power MOSFET Datasheet - PD-9.373H

[14] International Rectifier – IRF9530 P-Chanel HEXFET Power MOSFET Datasheet - PD-9.320G

[15] Robert Resnick, David Halliday (2001), “Fisica” Vol. 1 Cuarta Edición Pag. 362-364

[16] Karl J. Astrom (1988), “Sistemas Controlados por Computadora” Paraninfo. Pag. 53-78, 240-243, 389-391.

[17] Walter E. Pronzato L. (1997) “Identification of Parametric Models” Springler , Pag. 1-6, 83-90, 238

[18] Kalman, R. E. (1960). “A new approach to linear filtering and prediction problems” Transactions of the ASME, Journal of Basic Engineering, 82, 34–45.

[19] Petersen & Pedersen (2008), “The Matrix Cookbook” , Pag. 9-10

[20] R. Kustra, O. Tujsnaider (1984), “Principios de Transmisión de Señales Digitales”, ENTEL, Pag. 115-116

[21] .J. Astrom and K. Furuta (1996), Swinging-up a pendulum by energy control. Proc. IFAC Congress 37-95.

[22] K. Graichen, M. Treuer, and M. Zeitz (2007), “Swing-up of the double pendulum on a cart by feedforward and feedback control with experimental validation” , 43, 1, 63-71.

ANEXOS

ANEXO I – Manual de Usuario

En este anexo se describen los pasos a seguir para la puesta en funcionamiento del equipo experimental de laboratorio (*).

1. Ubique el equipo experimental sobre una base bien firme y nivelada horizontalmente.
2. Ubique la PC o notebook que se utilizará para realizar los experimentos cerca del equipo experimental y enciéndala.
3. Conecte el cable de alimentación del equipo experimental a la red eléctrica y el cable USB a la PC o notebook.
4. Espere a que el péndulo se encuentre detenido, luego encienda el equipo accionando la tecla de alimentación.
5. Si es la primera vez que conecta el equipo experimental a esta PC o notebook, deberá instalar los drivers correspondientes. Los mismos se encuentran en el CD que viene incluido con el equipo.
6. Inicie el entorno MATLAB y cargue los programas que desee ejecutar con el equipo (**).
7. Ejecute los programas y observe el funcionamiento del equipo.
8. Los datos obtenidos del experimento quedan registrados en los vectores **f**, **x** y **u** del workspace de MATLAB, las mismas pueden graficarse o ser almacenadas en un archivo para su posterior uso.

(*) Si tiene dudas sobre la construcción física del equipo, consulte el capítulo 2 del presente trabajo.

(**) Si tiene dudas con respecto a los métodos de identificación, consulte el capítulo 3. Si tiene dudas con respecto a las técnicas de control, consulte el capítulo 4.

ANEXO II – Guías de Laboratorio

Trabajo Práctico de Laboratorio Nº 1 – Control en Variables de Estado

Introducción

Una de las aplicaciones más importantes del control automático es el control de sistemas mecánicos inestables. El presente trabajo práctico consiste en la implementación de un controlador en variables de estado, utilizando el equipo experimental provisto por la cátedra.

El objetivo es que los alumnos apliquen los conocimientos adquiridos en las clases teóricas, adquieran experiencia en el diseño de controladores en variables de estado y tomen contacto con las realizaciones prácticas los mismos.

Descripción

El trabajo se realizará siguiendo las consignas planteadas en esta guía de laboratorio. Se utilizará el entorno MATLAB para realizar el diseño del vector de realimentación y el observador para el sistema planteado. Posteriormente se implementará dicho controlador utilizando el equipo experimental y se evaluará su desempeño.

Se trabajará en forma grupal, en grupos de no más de 4 integrantes. Se deberá presentar un informe escrito describiendo el trabajo realizado, los resultados obtenidos y las conclusiones.

Datos

Para la realización del diseño utilizaremos los siguientes datos

El modelo del sistema es

$$\begin{bmatrix} x''(t) \\ x'(t) \\ \phi''(t) \\ \phi'(t) \end{bmatrix} = \begin{bmatrix} -a & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ -acd & 0 & 0 & c \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x'(t) \\ x(t) \\ \phi'(t) \\ \phi(t) \end{bmatrix} + \begin{bmatrix} ab \\ 0 \\ abcd \\ 0 \end{bmatrix} u(t)$$

$$y(t) = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x'(t) \\ x(t) \\ \varphi'(t) \\ \varphi(t) \end{bmatrix}$$

Con $a = 80$ $b = 5$ $c = 88$ $d = 0.015$

El período de muestreo es h = 0.02

La ubicación deseada para los polos del observador es alrededor del punto z = 0.1

La ubicación deseada para los polos de lazo cerrado es alrededor del punto z = 0.8

Desarrollo

1. Utilizando el entorno MATLAB realice la discretización del sistema planteado.
2. Luego, diseñe un vector de realimentación L utilizando la función ***place.***
3. Del mismo modo, diseñe un vector de realimentación K para el observador del sistema.
4. Elabore un programa en MATLAB que implemente tanto el observador como la realimentación de los estados del sistema.
5. Conecte y encienda el equipo siguiendo el manual de usuario del mismo.
6. Ejecute el programa elaborado en el paso 4, la señal de entrada quedará almacenada en el vector u(k) y las señales de salida en los vectores x(k) y f(k).
7. Verifique el correcto funcionamiento del controlador, de lo contrario busque y corrija errores, luego repita los pasos 1 a 6.
8. Guarde los datos obtenidos en un archivo .mat
9. Evalúe el desempeño del controlador calculando la varianza de x(k) y f(k)
10. Compare con los desempeños obtenidos por otros grupos

Trabajo Práctico de Laboratorio Nº 2 – Control en Enfoque Polinomial

Introducción

Una de las aplicaciones más importantes del control automático es el control de sistemas mecánicos inestables. El presente trabajo práctico consiste en la implementación de un controlador en Enfoque Polinomial para el problema del péndulo invertido, utilizando el equipo experimental provisto por la cátedra.

El objetivo es que los alumnos apliquen los conocimientos adquiridos en las clases teóricas, adquieran experiencia en el diseño de controladores utilizando Enfoque Polinomial y tomen contacto con las realizaciones prácticas los mismos.

Descripción

El trabajo se realizará siguiendo las consignas planteadas en esta guía de laboratorio. Se utilizará el entorno MATLAB para realizar el diseño de la función de transferencia del controlador para el sistema planteado. Posteriormente se implementará dicho controlador utilizando el equipo experimental y se evaluará su desempeño.

Se trabajará en forma grupal, en grupos de no más de 4 integrantes. Se deberá presentar un informe escrito describiendo el trabajo realizado, los resultados obtenidos y las conclusiones.

Datos

Para la realización del diseño utilizaremos los siguientes datos

El modelo del sistema es

$$\begin{bmatrix} x''(t) \\ x'(t) \\ \phi''(t) \\ \phi'(t) \end{bmatrix} = \begin{bmatrix} -a & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ -acd & 0 & 0 & c \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x'(t) \\ x(t) \\ \phi'(t) \\ \phi(t) \end{bmatrix} + \begin{bmatrix} ab \\ 0 \\ abcd \\ 0 \end{bmatrix} u(t)$$

$$y(t) = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x'(t) \\ x(t) \\ \varphi'(t) \\ \varphi(t) \end{bmatrix}$$

Con $a = 80$ $b = 5$ $c = 88$ $d = 0.015$

El período de muestreo es h = 0.02

La ubicación deseada para los polos de lazo cerrado es alrededor del punto z = 0.7

Desarrollo

1. Utilizando el entorno MATLAB realice la discretización del sistema planteado para obtener las matrices Φ y Γ del sistema discreto.
2. Calcule la función de transferencia del sistema mediante

$$A(z) = \det(zI - \Phi) \qquad B(z) = C(zI - \Phi)^{-1}\Gamma A$$

3. Calcule los coeficientes q_i del polinomio característico deseado

$$(z - p_1)(z - p_2)...(z - p_5) = q_5 z^5 + q_4 z^4 + q_3 z^3 + q_2 z^2 + q_1 z + q_0$$

4. Calcule el vector de coeficientes a partir de

$$\begin{bmatrix} r_1 \\ r_0 \\ s_{11} \\ s_{10} \\ s_{21} \\ s_{20} \end{bmatrix} = \begin{bmatrix} a_4 & 0 & 0 & 0 & 0 & 0 \\ a_3 & a_4 & b_{13} & 0 & b_{23} & 0 \\ a_2 & a_3 & b_{12} & b_{13} & b_{22} & b_{23} \\ a_1 & a_2 & b_{11} & b_{12} & b_{21} & b_{22} \\ a_0 & a_1 & b_{10} & b_{11} & b_{20} & b_{21} \\ 0 & a_0 & 0 & b_{10} & 0 & b_{20} \end{bmatrix}^{-1} \begin{bmatrix} q_5 \\ q_4 \\ q_3 \\ q_2 \\ q_1 \\ q_0 \end{bmatrix}$$

Donde a_i y b_{ij} son los coeficientes de A(z) y B(z)

5. La función de transferencia del controlador resulta

$$R^{-1}S = \begin{bmatrix} \dfrac{s_{11}z + s_{10}}{r_1 z + r_0} \\ \dfrac{s_{21}z + s_{20}}{r_1 z + r_0} \end{bmatrix}$$

6. Elabore un programa en MATLAB que implemente dicho controlador
7. Conecte y encienda el equipo siguiendo el manual de usuario del mismo.
8. Ejecute el programa elaborado en el paso 6, la señal de entrada quedará almacenada en el vector u(k) y las señales de salida en los vectores x(k) y f(k).
9. Verifique el correcto funcionamiento del controlador, de lo contrario busque y corrija errores, luego repita los pasos 1 a 8.
10. Guarde los datos obtenidos en un archivo .mat
11. Evalúe el desempeño del controlador calculando la varianza de x(k) y f(k)
12. Compare con los desempeños obtenidos por otros grupos

Trabajo Práctico de Laboratorio Nº 3 – Identificación de Sistemas

Introducción

En cualquier aplicación de los conceptos de control vistos en clase, es muy importante primero, conocer con cierta exactitud las características del sistema a controlar. Para esto existen varias técnicas de identificación de sistemas, el presente trabajo práctico consiste en la implementación de una de ellas, utilizando el equipo experimental provisto por la cátedra.

El objetivo es que los alumnos apliquen los conocimientos adquiridos en las clases teóricas y tomen contacto con las realizaciones prácticas de los sistemas controlados por computadora.

Descripción

El trabajo se realizará siguiendo las consignas planteadas en esta guía de laboratorio. Se utilizará el entorno MATLAB para inyectar diferentes señales de prueba en el sistema y se registraran las señales de entrada y salida. Posteriormente se utilizaran dichas señales para identificar la función de transferencia del sistema utilizando las técnicas de identificación de sistemas.

Se trabajará en forma grupal, en grupos de no más de 4 integrantes. Se deberá presentar un informe escrito describiendo el trabajo realizado, los resultados obtenidos y las conclusiones.

Desarrollo

1. Elabore un programa en MATLAB para realizar la obtención de los datos, el mismo deberá incluir:
   1.1. La especificación del periodo de muestreo y la cantidad de pasos a ejecutar h=0.02 N=500
   1.2. La especificación de la señal de pruebas u(k) para $1 < k < N$
   1.3. La llamada al comando usb_start
   1.4. La llamada al comando usb_com , N veces utilizando un bucle for

1.5. La llamada al comando sub_stop

2. Conecte y encienda el equipo siguiendo el manual de usuario del mismo
3. Ejecute el programa elaborado en el paso 1, la señal de entrada quedará almacenada en el vector u(k) y la señal de salida en el vector x(k)
4. Guarde los datos obtenidos en un archivo .mat
5. A partir de los datos obtenidos, ensamble las matrices Ψ e Y siguiendo la forma

$$Y = \begin{bmatrix} x(3) \\ x(4) \\ x(5) \\ \vdots \end{bmatrix} \quad \Psi = \begin{bmatrix} x(2) & x(1) & u(2) & u(1) \\ x(3) & x(2) & u(3) & u(2) \\ x(4) & x(3) & u(4) & u(3) \\ \vdots & \vdots & \vdots & \vdots \end{bmatrix}$$

6. Calcule el vector de parámetros utilizando

$$\hat{\theta} = \left(\Psi^T \Psi\right)^{-1} \Psi^T Y$$

7. La función de transferencia del sistema resulta

$$x(z) = \frac{b_1 z + b_0}{z^2 + a_1 z + a_0} u(z)$$

Donde los coeficientes b_1 b_0 a_1 y a_0 se extraen del vector de parámetros

$$\hat{\theta} = \begin{bmatrix} -a_1 \\ -a_0 \\ b_1 \\ b_0 \end{bmatrix}$$

8. Realice la simulación de la respuesta de la función de transferencia obtenida a la entrada u(k)
9. Grafique la respuesta del sistema real y la simulación para verificar que los resultados son correctos.

ANEXO III – Código Fuente de los Microcontroladores

master_t.c

```
#include "a18F4550_t.h"

//~~~ 20MHZ OSCILLATOR CONFIGS ~~~//
#fuses HSPLL,NOWDT,NOPROTECT,NOLVP,NODEBUG,USBDIV,PLL1,CPUDIV1,VREGEN,CCP2C1,NOMCLR
#use delay(clock=48M)

#include "axl_usb_t.c"
#include "defines_t.c"
#include "rutinas_t.c"

////////////////////////////////////////////////////////////////////////
//PROGRAMA PRINCIPAL
////////////////////////////////////////////////////////////////////////

void main()
{
   INIT();
   while (TRUE)
   {
      usb_task();
      if(usb_enumerated())
      {

         if (usb_kbhit(1))
         {

            if(usb_get_packet(1, rxdata, 8)==8);
            {
               DATA();
               if ((L0)&&(US))
                  U1=0;
               if ((L1)&&(!US))
                  U1=0;

               usb_put_packet(1, rxdata, 8, USB_DTS_TOGGLE);
               PWM();

            }
         }
      }

      L0=input(LIM0);
      L1=input(LIM1);

      if ((L0)&&(US))
      {
         U1=0;
         PWM();
      }
      if ((L1)&&(!US))
      {
         U1=0;
         PWM();
      }
   }
}
```

defines_t.c

```
////////////////////////////////////////////////////////////////////////////
//PINES
////////////////////////////////////////////////////////////////////////////
#define LED1 PIN_E0
#define LED2 PIN_A4
#define LED3 PIN_A5

#define BUTTON1 PIN_A0
#define BUTTON2 PIN_A1
#define BUTTON3 PIN_A2
#define BUTTON4 PIN_A3

#define BUSCON1 PIN_C6
#define BUSCON2 PIN_C7

#define PWM1ENABLE PIN_E2
#define PWM2ENABLE PIN_E1

#define LIM0 PIN_B6
#define LIM1 PIN_B7

////////////////////////////////////////////////////////////////////////////
//TIMES
////////////////////////////////////////////////////////////////////////////
#DEFINE TBUSCON1   8        //TIEMPO DE DURACION DEL PULSO BUSCON1
#DEFINE TBUSCON2   8        //TIEMPO DE DURACION DEL PULSO BUSCON2
#DEFINE TBUSCON3   1        //TIEMPO DE REPOSO DESPUES DEL PULSO BUSCON
////////////////////////////////////////////////////////////////////////////
//DEFINES
////////////////////////////////////////////////////////////////////////////
#define LED_ON output_high
#define LED_OFF output_low

////////////////////////////////////////////////////////////////////////////
//VARIABLES
////////////////////////////////////////////////////////////////////////////

unsigned int8 T0;
#BIT T00 = 0x25.0
#BIT T01 = 0x25.1
#BIT T02 = 0x25.2
#BIT T03 = 0x25.3
#BIT T04 = 0x25.4
#BIT T05 = 0x25.5
#BIT T06 = 0x25.6
#BIT T07 = 0x25.7

unsigned int8 T1;

#BIT T10 = 0x26.0
#BIT T11 = 0x26.1
#BIT T12 = 0x26.2
#BIT T13 = 0x26.3
#BIT T14 = 0x26.4
#BIT T15 = 0x26.5
#BIT T16 = 0x26.6
#BIT T17 = 0x26.7

unsigned int8 D;
#BIT D0 = 0x27.0
#BIT D1 = 0x27.1
#BIT D2 = 0x27.2
#BIT D3 = 0x27.3
#BIT D4 = 0x27.4
#BIT D5 = 0x27.5
#BIT D6 = 0x27.6
#BIT D7 = 0x27.7

unsigned int8 B;
#BIT B0 = 0x28.0
#BIT B1 = 0x28.1
#BIT B2 = 0x28.2
#BIT B3 = 0x28.3
```

```
#BIT B4 = 0x28.4
#BIT B5 = 0x28.5
#BIT B6 = 0x28.6
#BIT B7 = 0x28.7

unsigned int16 Xref;
unsigned int8 UA,trash, A0,A1,A2;

unsigned int8 rxdata[8];

#BIT US = 0x30.0

//#WORD U = 0x31
//#BYTE U0 = 0x31
#BYTE U1 = 0x32

#WORD F = 0x33
#BYTE F0 = 0x33
#BYTE F1 = 0x34

#WORD X = 0x35
#BYTE X0 = 0x35
#BYTE X1 = 0x36

#BYTE L = 0x37
#BIT L0 = 0x37.0
#BIT L1 = 0x37.1

int1 USA;

unsigned int8 rxdata2[8];
```

rutinas_t.c

```
void DATA()
{

   output_high(BUSCON1);          //REQUEST X
   delay_us(TBUSCON1);
   output_low(BUSCON1);
   delay_us(TBUSCON3);

   D=input_d();                 //DATA IN
   B=input_b();

   T00=B5;
   T01=B4;
   T02=B3;
   T03=B2;
   T04=B1;
   T05=B0;
   T06=D7;
   T07=D6;

   T1=0;
   T10=D5;

   X0=T0;
   X1=T1;

   X=((X-Xref)&(0x01FF));


   output_high(BUSCON2);          //REQUEST F
   delay_us(TBUSCON2);
   output_low(BUSCON2);
   delay_us(TBUSCON3);

   D=input_d();                 //DATA IN
   B=input_b();

   T00=B5;
   T01=B4;
   T02=B3;
   T03=B2;
   T04=B1;
```

```
   T05=B0;
   T06=D7;
   T07=D6;

   T1=0;
   T10=D5;
   T11=D4;

   F0=T0;
   F1=T1;


   L=0;
   L0=input(LIM0);
   L1=input(LIM1);
}


void INIT()
{
   trash=input_b();
   trash=input_d();
   output_low(BUSCON1);
   output_low(BUSCON2);



   setup_ccp1(CCP_PWM);
   setup_ccp2(CCP_PWM);
   setup_timer_2(T2_DIV_BY_16, 255, 1);
   set_pwm1_duty(0);
   set_pwm2_duty(0);

   U1=0;
   UA=0;



   //referencia de X
   U1=64;
   if(input(LIM1))
   {
      output_low(PWM1ENABLE);
      set_pwm1_duty(0);
      output_high(PWM2ENABLE);
      set_pwm2_duty(U1);
      delay_ms(200);
   }
      output_low(PWM2ENABLE);
      set_pwm2_duty(0);
      delay_ms(5);
      output_high(PWM1ENABLE);
      set_pwm1_duty(U1);

   while(!input(LIM1)) {}
   U1=0;

   output_low(PWM2ENABLE);
   set_pwm2_duty(0);
   output_high(PWM1ENABLE);
   set_pwm1_duty(0);
   delay_ms(200);
   Xref=0;
   DATA();
   Xref=(X-100);

   usb_init();
}


void PWM()
{
   if (US!=USA)
   {
      output_low(PWM1ENABLE);
      output_low(PWM2ENABLE);
      set_pwm1_duty(0);
      set_pwm2_duty(0);
```

```
      delay_us(200);
      USA=US;
      UA=0;
   }

   if (U1!=UA)
   {
      if (US)
      {
         output_low(PWM1ENABLE);
         set_pwm1_duty(0);
         output_high(PWM2ENABLE);
         set_pwm2_duty(U1);
      }
      else
      {
         output_low(PWM2ENABLE);
         set_pwm2_duty(0);
         output_high(PWM1ENABLE);
         set_pwm1_duty(U1);
      }
      UA=U1;
   }
}
```

Slave_t.c

```
#include "a18F4550_t.h"

//~~~ 48MHZ OSCILLATOR CONFIGS ~~~// 4Mhz Crystal
#fuses HSPLL,NOWDT,NOPROTECT,NOLVP,NODEBUG,USBDIV,PLL1,CPUDIV1,VREGEN,CCP2C1,NOMCLR
#use delay(clock=48M)

#include "defines_s_t.c"

////////////////////////////////////////////////////////////////////////////
//PROGRAMA PRINCIPAL
////////////////////////////////////////////////////////////////////////////

void main()
{

   X=0x0;
   F=0x0;

   while (TRUE)
   {

//////////////////////////////////////
//DATA
//////////////////////////////////////


      if(input(BUSCON1))
      {
         output_d(X0);
         output_b(X1);
      }

      if(input(BUSCON2))
      {
         output_d(F0);
         output_b(F1);
      }


////////////////////////////////////////////////////////////////////////////
//encoder A
////////////////////////////////////////////////////////////////////////////
   enca1p=input (ENCA1);
   enca2p=input (ENCA2);

   if(enca1p)
   {
      if(enca2p)
```

```
   {
      if(enca1a)
      {
         if(enca2a)
         {
            //1111
            ZZ++; //nada
         }
         else
         {
         delay_cycles(1);
            //1110
            F++;
         }
      }
      else
      {
      delay_cycles(1);
         if(enca2a)
         {
            //1101
            F--;
         }
         else
         {
         delay_cycles(1);
            //1100
            ZZ++; //error
         }
      }
   enca2a=1;
   }
   else     //C7=0
   {
      delay_cycles(1);
      if(enca1a)
      {
         if(enca2a)
         {
            //1011
            F--;
         }
         else
         {
            delay_cycles(1);
            //1010
            ZZ++; //nada
         }
      }
      else
      {
      delay_cycles(1);
         if(enca2a)
         {
            //1001
            ZZ++; //error
         }
         else
         {
         delay_cycles(1);
            //1000
            F++;
         }
      }
   enca2a=0;
   }
enca1a=1;
}
else     //C6=0
{
delay_cycles(1);
   if(enca2p)
   {
      if(enca1a)
      {
         if(enca2a)
         {
            //0111
```

```
               F++;
            }
            else
            {
            delay_cycles(1);
               //0110
               ZZ++; //error
            }
         }
         else
         {
         delay_cycles(1);
            if(enca2a)
            {
               //0101
               ZZ++; //nada
            }
            else
            {
            delay_cycles(1);
               //0100
               F--;
            }
         }
      enca2a=1;
      }
      else     //C7=0
      {
      delay_cycles(1);
         if(enca1a)
         {
            if(enca2a)
            {
               //0011
               ZZ++; //error
            }
            else
            {
            delay_cycles(1);
               //0010
               F--;
            }
         }
         else
         {
         delay_cycles(1);
            if(enca2a)
            {
               //0001
               F++;
            }
            else
            {
            delay_cycles(1);
               //0000
               ZZ++; //nada
            }
         }
      enca2a=0;
      }
   enca1a=0;
   }


////////////////////////////////////////////////////////////////////////////
//encoder B
////////////////////////////////////////////////////////////////////////////
   encb1p = input (ENCB1);
   encb2p = input (ENCB2);

   if(encb1p)
   {
      if(encb2p)
      {
         if(encb1a)
         {
            if(encb2a)
            {
```

```
            //1111
            ZZ++; //nada
         }
         else
         {
         delay_cycles(1);
            //1110
            X++;
         }
      }
      else
      {
      delay_cycles(1);
         if(encb2a)
         {
            //1101
            X--;
         }
         else
         {
         delay_cycles(1);
            //1100
            ZZ++; //error
         }
      }
   encb2a=1;
   }
   else     //C7=0
   {
      delay_cycles(1);
      if(encb1a)
      {
         if(encb2a)
         {
            //1011
            X--;
         }
         else
         {
            delay_cycles(1);
            //1010
            ZZ++; //nada
         }
      }
      else
      {
      delay_cycles(1);
         if(encb2a)
         {
            //1001
            ZZ++; //error
         }
         else
         {
         delay_cycles(1);
            //1000
            X++;
         }
      }
   encb2a=0;
   }
encb1a=1;
}
else     //C6=0
{
delay_cycles(1);
   if(encb2p)
   {
      if(encb1a)
      {
         if(encb2a)
         {
            //0111
            X++;
         }
         else
         {
         delay_cycles(1);
```

```
                //0110
                ZZ++; //error
            }
         }
         else
         {
         delay_cycles(1);
            if(encb2a)
            {
               //0101
               ZZ++; //nada
            }
            else
            {
            delay_cycles(1);
               //0100
               X--;
            }
         }
      encb2a=1;
      }
      else     //C7=0
      {
      delay_cycles(1);
         if(encb1a)
         {
            if(encb2a)
            {
               //0011
               ZZ++; //error
            }
            else
            {
            delay_cycles(1);
               //0010
               X--;
            }
         }
         else
         {
         delay_cycles(1);
            if(encb2a)
            {
               //0001
               X++;
            }
            else
            {
            delay_cycles(1);
               //0000
               ZZ++; //nada
            }
         }
      encb2a=0;
      }
   encb1a=0;
   }


   }
}
```

Defines_s_t.c

```
////////////////////////////////////////////////////////////////////////////
//PINES
////////////////////////////////////////////////////////////////////////////


#define ENCA1 PIN_C0
#define ENCA2 PIN_C1
#define ENCB1 PIN_E0
#define ENCB2 PIN_E1

#define BUSCON1 PIN_C6
#define BUSCON2 PIN_C7


////////////////////////////////////////////////////////////////////////////
//DEFINES
////////////////////////////////////////////////////////////////////////////
#define LED_ON output_high
#define LED_OFF output_low

////////////////////////////////////////////////////////////////////////////
//VAR
////////////////////////////////////////////////////////////////////////////

signed int16 F;
#BYTE F0=5
#BYTE F1=6

signed int16 X;
#BYTE X0=7
#BYTE X1=8

int1 enca1p,enca2p,enca1a,enca2a,encb1p,encb2p,encb1a,encb2a;

signed int16 ZZ;
```

ANEXO IV – Código Fuente en MATLAB

usb_start.m

```
LIMU=1;
LIMX=0.5;
u=zeros(1,N);
f=zeros(1,N);
x=zeros(1,N);

xe=zeros(4,N+1);
sen_fi=zeros(1,N);
E=zeros(1,N);
Ec=zeros(1,N);
Ep=zeros(1,N);

loadlibrary mpusbapi _mpusbapi.h alias libreria

data_in = eye(1,8,'uint8');       % Se declara el vector de datos de entrada
data_out = eye(1,8,'uint8');      % Se declara el vector de datos de salida
len = 8*ones(1,4,'uint8');
waittime=uint8(50);

vid_pid_norm = libpointer('int8Ptr',[uint8('vid_04d8&pid_1011') 0]);
out_pipe = libpointer('int8Ptr',[uint8('\MCHP_EP1') 0]);
in_pipe = libpointer('int8Ptr',[uint8('\MCHP_EP1') 0]);

[conectado] = calllib('libreria','MPUSBGetDeviceCount',vid_pid_norm);
if conectado == 1

    [my_out_pipe] = calllib('libreria', 'MPUSBOpen',uint8 (0), vid_pid_norm, out_pipe, uint8(0),
uint8 (0)); % Se abre el tunel de envio
    [my_in_pipe] = calllib('libreria', 'MPUSBOpen',uint8 (0), vid_pid_norm, in_pipe, uint8 (1),
uint8 (0)); % Se abre el tunel de recepción
end
t1=cputime;
tic
```

usb_stop.m

```
toc;
Elapsed_time=cputime-t1
h_medio=Elapsed_time/N
if (conectado == 1)
    data_out(3)=uint8(0);
    calllib('libreria', 'MPUSBWrite',my_out_pipe, data_out, len(1), len(2), waittime); % Se envia el
dato al PIC
    [aa,bb,data_in,dd] = calllib('libreria', 'MPUSBRead',my_in_pipe, data_in, len(3), len(4),
waittime); % Se recibe el dato que envia el PIC

    calllib('libreria', 'MPUSBClose', my_in_pipe);  % Se cierra el tunel de recepción
    calllib('libreria', 'MPUSBClose', my_out_pipe); % Se cierra el tunel de envio
end
unloadlibrary libreria
%close all

clear t1 ans data_in data_out uc aa bb dd conectado i in_pipe len my_in_pipe my_out_pipe out_pipe
vid_pid_norm waittime
```

usb_com.m

```
     if(u(i)>LIMU)
         u(i)=LIMU;
     end
     if(u(i)<-LIMU)
         u(i)=-LIMU;
     end
     if(u(i)>0)
         if(x(i-1)>LIMX)
             u(i)=0;
             u0=0;
         else
             data_out(1)=0;
             u0=u(i);
         end
     else
         if(x(i-1)<-LIMX)
             u(i)=0;
             u0=0;
         else
             data_out(1)=1;
             u0=-u(i);
         end
     end

     u1=u0*18.3333;
     if(u0>0.012)
         u1=(u0*1.0448)+0.2075;
     end
     if(u0>0.28)
         u1=(u0*0.7813)+0.2813;
     end


     u0=fix(200*u0);
    if(u0>0)
         u0=u0+35;
    else
          u0=0;
    end
    data_out(3)=uint8(u0);


    calllib('libreria', 'MPUSBWrite',my_out_pipe, data_out, len(1), len(2),waittime); % Se envia el
dato al PIC
    [aa,bb,data_in,dd]    =    calllib('libreria',    'MPUSBRead',my_in_pipe,    data_in,    len(3),
len(4),waittime); % Se recibe el dato que envia el PIC

     f(i)=(single(data_in(4))+256*single(data_in(5)));
     f(i)=f(i)-512;
     f(i)=(-pi/512)*f(i);
     x(i)=(-1)*(single(data_in(6))+256*single(data_in(7))-258)/167;


     sen_fi(i)=sin(f(i));
     Ec(i)=(7*(f(i)-f(i-1))^2);                    %aproximacion de primer orden
     if(Ec(i)>2)
         Ec(i)=Ec(i-1);
     end
     Ep(i)=0.25*cos(f(i))+0.25*cos(f(i-1))+0.5;
     E(i)=Ec(i)+Ep(i);


     ttt=toc;
     while (ttt<h)
         ttt=toc;
     end
     tic
```

modelado2.m

```
%modelado del sistema en variables de estado
%péndulo arriba

  h=0.02;
  a=50;
  b=4;
  c=88;
  d=0.016;


A=[-a 0 0 0 ; 1 0 0 0 ; -a*c*d 0 0 c ; 0 0 1 0];
B=[a*b ; 0 ; a*b*c*d ; 0];
C=[0 1 0 0 ; 0 0 0 1];

SS=ss(A,B,C,0);
SSD=c2d(SS,h);

F=SSD.a;
G=SSD.B;

Po=[0.02 0.021 0.022 0.023];         %polos del observador
K=place(F',C',Po).';

Pc=[0.8 0.85 0.75 0.7];          %polos del control
L=place(F,G,Pc);


clear SS SSD A B Po Pc
```

modelado6.m

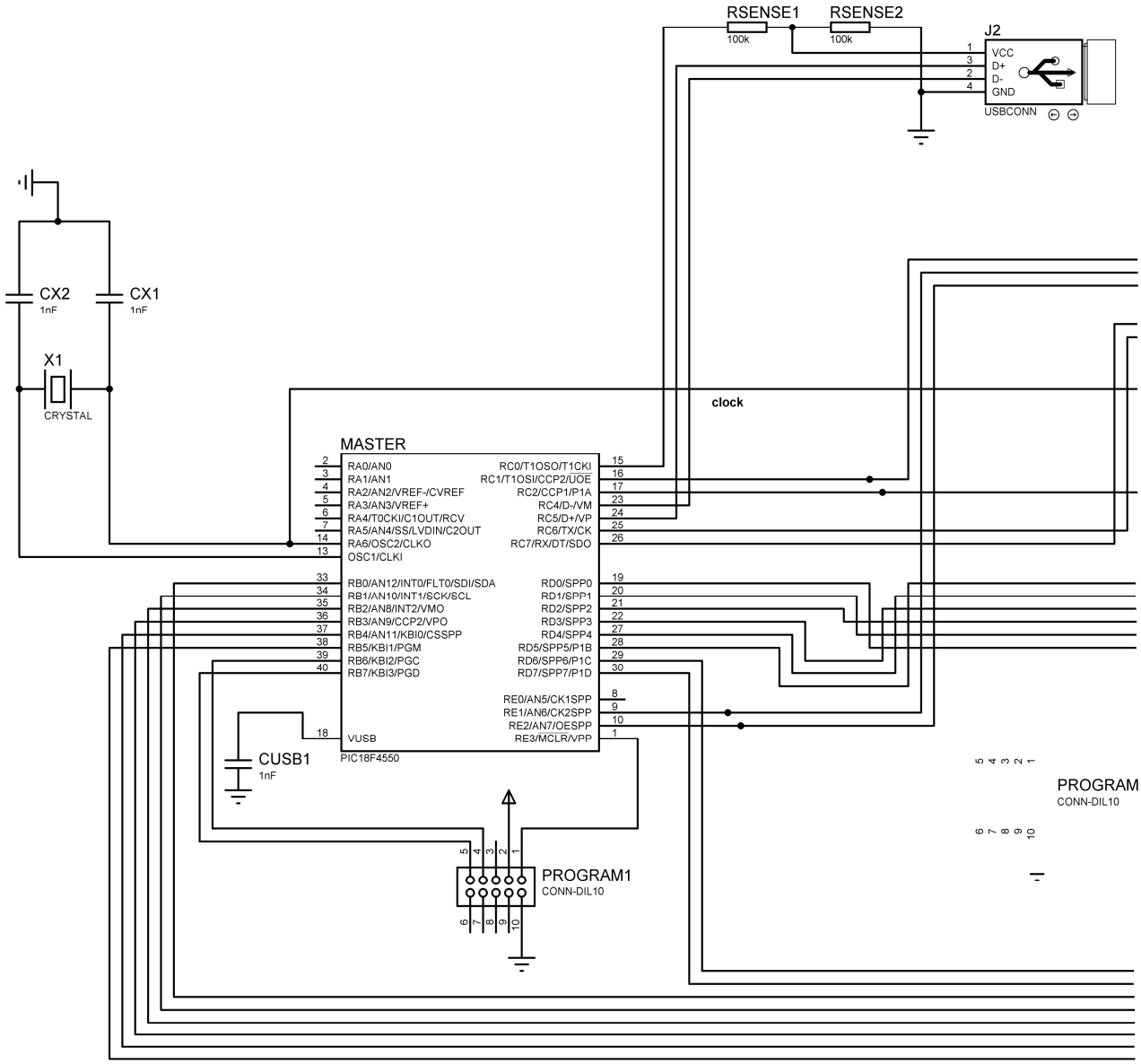
```
%modelado del sistema en enfoque polinomial
%péndulo arriba

  h=0.02;
  a=50;
  b=5;
  c=88;
  d=0.015;


A=[-a 0 0 0 ; 1 0 0 0 ; -a*c*d 0 0 c ; 0 0 1 0];
B=[a*b ; 0 ; a*b*c*d ; 0];
C=[0 1 0 0 ; 0 0 0 1];

SS=ss(A,B,C,0);
SSD=c2d(SS,h);

F=SSD.a;
G=SSD.B;

Po=[0.1 0.11 0.12 0.13];         %polos del observador
K=place(F',C',Po).';

Pc=[0.7+0.1*i 0.7-0.1*i 0.71 0.72];         %polos del control
L=place(F,G,Pc);


clear A B SS SSD
syms z
I=eye(4);

AA=expand(det(z*I-F));
BB=expand(simplify(AA*C*inv(z*I-F)*G));
A=fliplr(eval(coeffs(AA,z)));
B1=fliplr(eval(coeffs(BB(1),z)));
B2=fliplr(eval(coeffs(BB(2),z)));
```

```
AoAm=expand((z-0.7)*(z-0.72)^4);
AoAm=fliplr(eval(coeffs(AoAm)));
M=[[A';0] [0;A'] [0;B1';0] [0;0;B1'] [0;B2';0] [0;0;B2']];
X=M\AoAm';

R=[X(1) X(2)];
S1=[X(3) X(4)];
S2=[X(5) X(6)];

clear AA BB A B1 B2 AoAm X M I z
```

modelado_param_v.m

```
%modelado del sistema en variables de estado
%con parametros variantes
%pendulo arriba

  h=0.02;
  a=50;
  b=3.8;
  c=88;
  d=0.015;

  NN=100;             %numero de divisones en fi

for i=1:NN;
    fi=pi*i/NN
    i
    A=[-a 0 0 0 ; 1 0 0 0 ; -a*c*d*cos(fi) 0 0 c*sinc(fi/pi) ; 0 0 1 0];
    B=[a*b ; 0 ; a*b*c*d*cos(fi) ; 0];
    C=[0 1 0 0 ; 0 0 0 1];

    SS=ss(A,B,C,0);
    SSD=c2d(SS,h);

    F=SSD.a;
    G=SSD.B;

    Po=[0.1 0.11 0.12 0.13];         %polos del observador
    K=place(F',C',Po).';

    if(i==NN/2)
        L=[0 0 0 0];                 %para fi=pi/2 no hay control
    else
        if(i==NN)

        else
            Pc=[0.8 0.85 0.75 0.7];         %polos del control
            L=place(F,G,Pc);
        end
    end

    LL(:,:,i)=L;%/(10*sin(fi)*sin(fi)+1);
    KK(:,:,i)=K;
    GG(:,:,i)=G;
    FF(:,:,i)=F;
    FGL(:,:,i)=(F-G*L);
    FKC(:,:,i)=(F-K*C);

end
    LL(:,:,i+1)=L;
    KK(:,:,i+1)=K;
    GG(:,:,i+1)=G;
    FF(:,:,i+1)=F;
    FGL(:,:,i+1)=(F-G*L);
    FKC(:,:,i+1)=(F-K*C);
clear SS SSD A B Po Pc
```

ANEXO V – Circuitos Esquemáticos



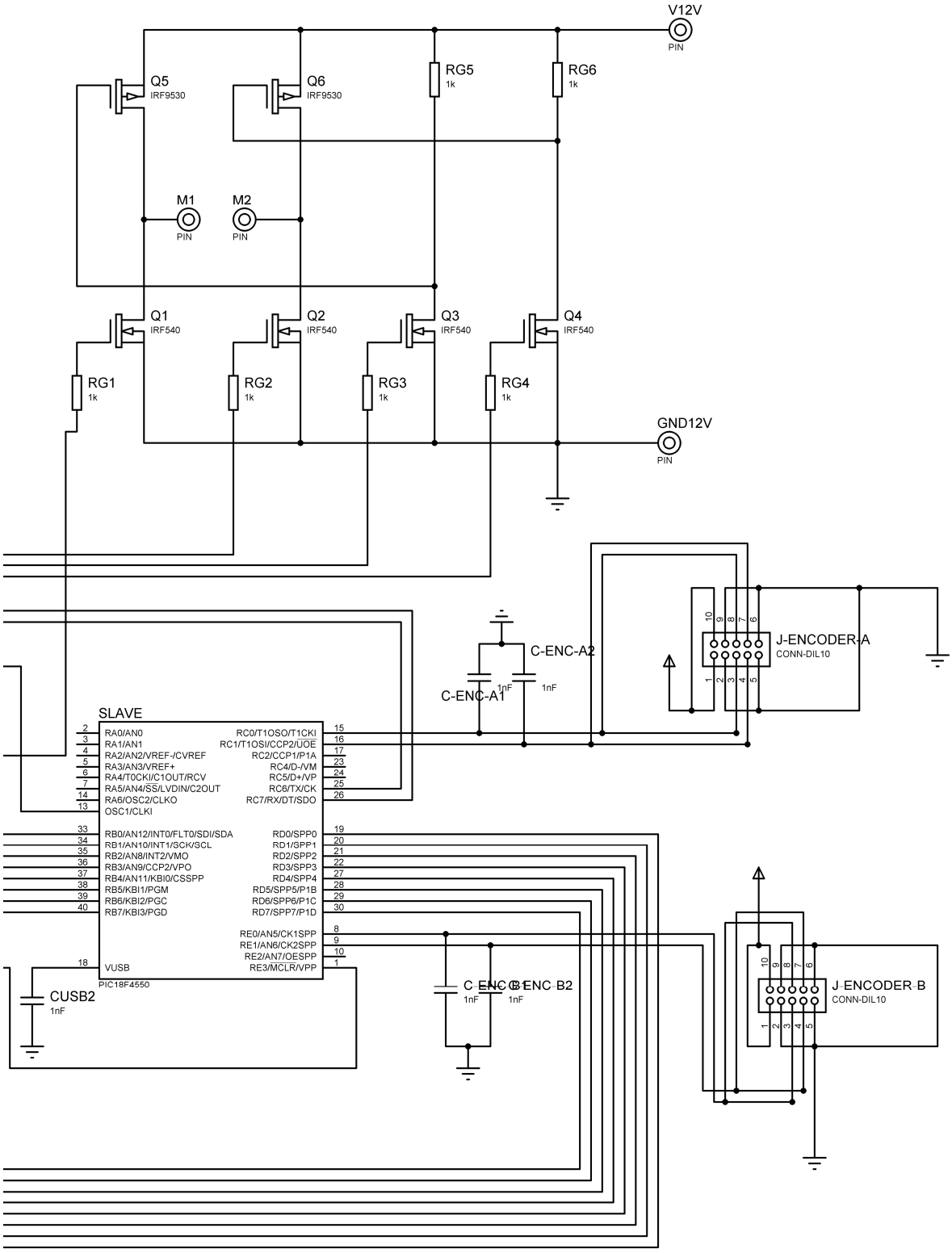


Figura A.1: Circuito esquemático de la placa principal

	Nombre	Nº de PIN	MAESTRO		ESCLAVO	
			TIPO	FUNCION	TIPO	FUNCION
Puerto A	RA0…RA5	2-3-4-5-6-7	-	No utilizado	-	No utilizado
	OSC1–OSC2	13 - 14	O	Cristal 4MHz	O	Clock
Puerto B	RB0…RB5	33-34-35-36-37-38	E	Bus de datos Protocolo Paralelo	S	Bus de datos Protocolo Paralelo
	RB6-RB7	39-40	E/P	ICSP Límites de carrera	-	ICSP
Puerto C	RC0	15	E	Pin de censado de conexión USB	E	Encoder A
	RC1	16	S	PWM	E	Encoder A
	RC2	17	S	PWM	-	No utilizado
	RC3	18	A	Regulador USB 3.3V	-	No utilizado
	RC4-RC5	23-24	E/S	Señal USB	-	No utilizado
	RC6-RC7	25-26	S	Banderas del Protocolo Paralelo	E	Banderas del Protocolo Paralelo
Puerto D	RD0…RD7	19-20-21-22 27-28-29-30	E	Bus de datos Protocolo Paralelo	S	Bus de datos Protocolo Paralelo
Puerto E	RE0	8	-	No utilizado	E	Encoder A
	RE1	9	S	Control del Puente H	E	Encoder A
	RE2	10	S	Control del Puente H	-	No utilizado
	MCLR	1	P	ICSP Reset	P	ICSP Reset
Alimentación	VDD	11-32	A	+5V	A	+5V
	VSS	12-31	A	GND	A	GND
E-Entrada, S-Salida, P-Programación, A-Alimentación, O- Oscilador						

Tabla 5: Descripción de Puertos y Pines, detallando Tipo y Función de cada uno