UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

Levantamiento y estabilización del péndulo invertido

TESIS

Que para obtener el título de

Ingeniero Eléctrico Electrónico

P R E S E N T A

Fernando Castaños Luna

DIRECTOR DE TESIS

M.I. Rolando A. Carrera Méndez

Ciudad Universitaria, Cd. Mx., 2003

Capítulo 1

Introducción

El péndulo invertido es un sistema mecánico clásico para probar nuevas ideas en la disciplina del control. Tiene la ventaja de ser, por un lado, un mecanismo relativamente sencillo, y por el otro, un sistema que contiene puntos inestables. El péndulo invertido se ha usado ampliamente como patrón para comparar tanto algoritmos de control, como el *hardware* para implementarlos. Otra de las cualidades de este dispositivo es que su dinámica es similar a la de un transporte aéreo y a la de un robot bípedo con la capacidad de caminar. Los algoritmos utilizados para controlarlo pueden ser adaptados al control de otros mecanismos más complejos.

1.1. Descripción del péndulo invertido

El péndulo invertido es un servo mecanismo que consta de un riel sobre el cual se puede deslizar un carro, sobre éste está montado un péndulo que puede girar libremente. El sistema está instrumentado, de tal suerte que se puede medir el ángulo del péndulo con respecto a la vertical, así como la posición y la velocidad del carro. A través de un motor y una banda conectada al carro, se puede hacer que éste se deslice sobre el riel, el cual mide aproximadamente 1.2m. (Ver Figura 1.1)

Si se considera al péndulo separado del carro, el péndulo tiene dos puntos de equilibrio: uno estable, abajo; y otro inestable, arriba. El objetivo del control es cambiar la dinámica del sistema para que en la posición vertical, arriba, se tenga un punto de equilibrio estable. En otras palabras, la idea es encontrar la fuerza que ha de aplicarse al carro para que el péndulo no se caiga, incluso si se le perturba con un empujón tipo escalón o impulso.

Figura 1.1: Esquema del péndulo invertido

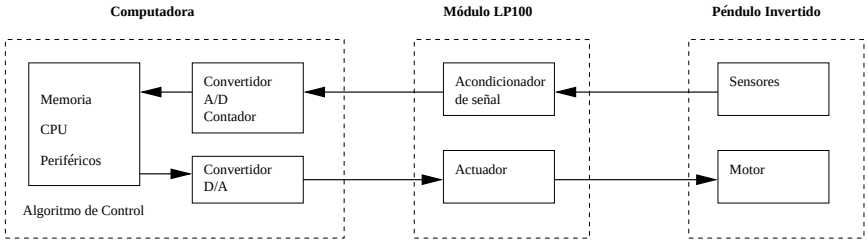

Figura 1.2: Flujo de señales entre la PC y el péndulo

1.2. Control propuesto

El controlador podría ser continuo, usando amplificadores operacionales por mencionar algún ejemplo; o discreto, usando un microcontrolador, un procesador digital de señales (DSP) o una computadora personal (PC). El diseño de un controlador continuo es más sencillo que el de uno discreto, pero un controlador discreto es mucho más flexible. En la Coordinación de Automatización, Instituto de Ingeniería, U.N.A.M. se cuenta actualmente con un péndulo invertido que ya incluye una tarjeta de adquisición de datos diseñada para incorporarse a una PC, un módulo que contiene los acondicionadores de señal para los sensores y el actuador para el motor. El uso de una PC para realizar el controlador es pues, la implementación más rápida. La flexibilidad con que cuentan las PC ofrece también la opción más didáctica. En la Figura 1.2 se muestra un diagrama que muestra la interconexión de los elementos mencionados, y en la Figura 1.3 se muestra una perspectiva (fuera de escala) de los componentes esenciales del sistema.

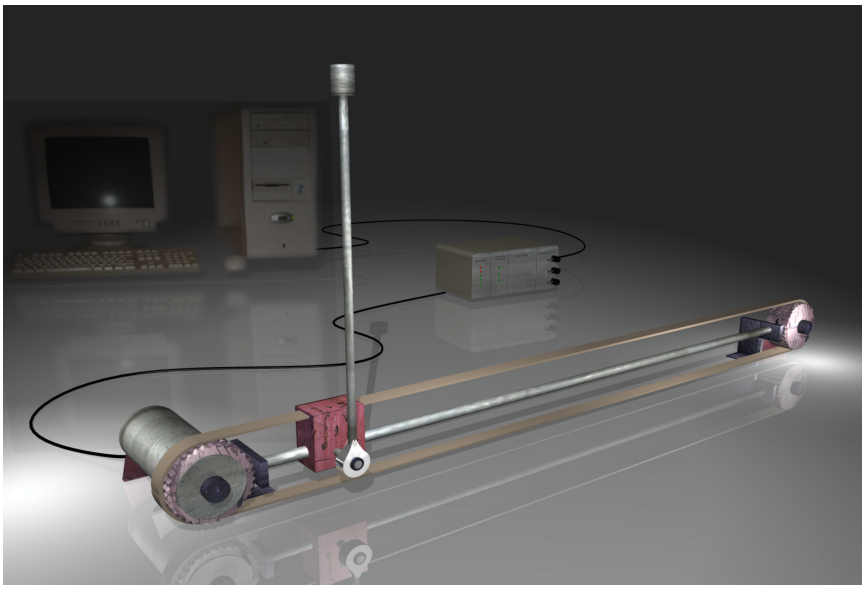

Figura 1.3: Perspectiva del péndulo invertido

1.3. Objetivos

En general, los sistemas de control requieren un tiempo de muestreo preciso. Esta precisión se consigue fácilmente con un microcontrolador o un DSP, pero si ha de usarse una PC, se requiere normalmente de un sistema operativo diseñado específicamente para trabajar en tiempo real. Cualquiera de estas dos opciones implica un esfuerzo para pasar del diseño a la implementación. Uno de los objetivos es proporcionar una plataforma sencilla para que futuros estudiantes de control puedan probar sus diferentes algoritmos. La sencillez de dicha plataforma radicaría principalmente en la separación del proceso de diseño y el proceso de implementación. Para verificar la validez de dicha plataforma, naturalmente será necesario diseñar un control que estabilice al péndulo.

El sistema, además de ser contener puntos inestables, es no lineal y presenta singularidades. Dado que al estabilizar el péndulo se espera que las variaciones en el ángulo sean pequeñas, se puede utilizar un modelo linealizado para simplificar el diseño del control que lo estabiliza. El algoritmo de estabilización requiere que el péndulo se encuentre en la posición vertical superior, por lo que se propone como objetivo adicional, diseñar un algoritmo que levante al péndulo y un conmutador entre el esquema que lo levanta

y el que lo estabiliza.

Capítulo 2

Modelo Matemático del Péndulo

El propósito de este capítulo es obtener un modelo matemático que describa al péndulo invertido. Primero se analiza la dinámica del sistema. Después, por medio de una serie de Taylor se linealiza el modelo obtenido para poder utilizar una ley de control de tipo LQR. Dado que el control propuesto es por medio de una computadora digital, se calcula al final el modelo discreto.

2.1. Ecuaciones diferenciales

Las variables que definen la condición del sistema en todo momento son la posición r del carro y el ángulo Φ del péndulo con respecto a la vertical. El objetivo de esta sección, es encontrar la ecuación o el conjunto de ecuaciones diferenciales que describan el comportamiento de estas variables para una fuerza F, aplicada en el carro a través de la banda.

El péndulo invertido se puede concebir como un cuerpo rígido cuyo movimiento se limita a dos dimensiones. Las ecuaciones fundamentales de movimiento plano de un cuerpo rígido son (ver Beer y Russell [2], capítulo 16)

$$\sum F_i = ma_i \tag{2.1}$$

$$\sum F_j = ma_j \tag{2.2}$$

$$\sum M_G = I\alpha_g \tag{2.3}$$

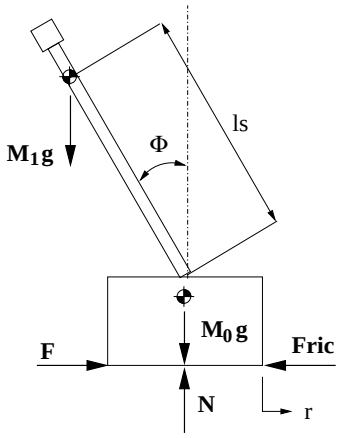

Figura 2.1: Diagrama de cuerpo libre del péndulo invertido

Las ecuaciones 2.1 y 2.2 son la segunda ley de Newton para las componentes horizontal i y vertical j de la fuerza F y la aceleración a experimentada por el cuerpo rígido de masa m. La ecuación 2.3, derivada también de la segunda ley de Newton, establece que la suma de momentos M de las fuerzas que actúan sobre un cuerpo rígido alrededor de un punto G cualquiera, es igual al momento de inercia I por la aceleración angular α (con dirección g) alrededor del cuerpo rígido.

En la Figura 2.1 se encuentra el diagrama de cuerpo libre del péndulo invertido. Sobre el péndulo invertido actúan F, la fuerza de fricción $Fric$, los pesos del péndulo y el carro, y la reacción normal N que ejerce el riel sobre el carro (tercera ley de Newton) M_0 es la masa del carro, M_1 la del péndulo, y g es la aceleración ejercida por la tierra.

Para simplificar el análisis, se puede dividir el péndulo invertido en dos cuerpos: el carro y el péndulo.

En el lado izquierdo de la Figura 2.2 se puede observar, además del peso del péndulo, las fuerzas de reacción H y V que actúan sobre su articulación. Para obtener la aceleración horizontal del péndulo, es necesario tomar en cuenta que la posición horizontal de su centro de gravedad depende de r y de Φ. Si se aplica 2.1 al péndulo, se tiene

$$H = M_1 \frac{d^2}{dt^2}\Big(r - l_s \operatorname{sen} \Phi\Big) \tag{2.4}$$

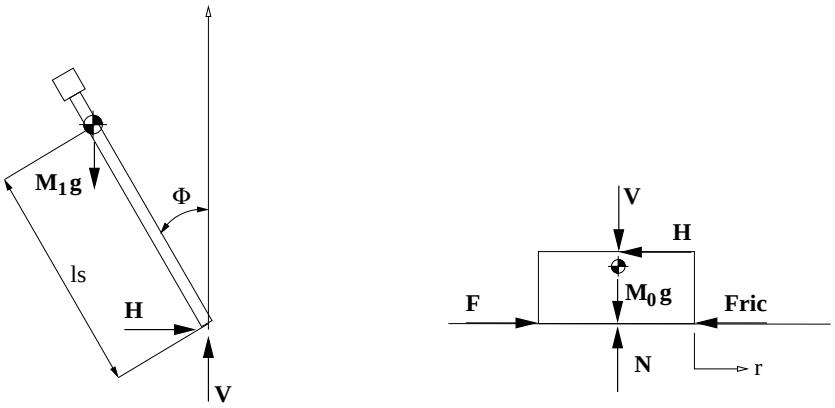

Figura 2.2: Diagramas de cuerpo libre del péndulo (izquierda) y del carro (derecha)

y la ecuación 2.2 da

$$V = M_1\left[\frac{d^2}{dt^2}\Big(l_s\cos\Phi\Big) + g\right] \tag{2.5}$$

Sobre el centro de gravedad del péndulo actúan tres momentos. Uno se debe a la fuerza H, otro a la fuerza V, y el tercero a la fricción que produce la articulación. De acuerdo con 2.3

$$\Theta_s\frac{d^2\Phi}{dt^2} = Vl_s\,\mathrm{sen}\,\Phi + Hl_s\cos\Phi - C\frac{d\Phi}{dt} \tag{2.6}$$

Θ_s es la masa del momento de inercia de la articulación con respecto al centro de gravedad del péndulo y C es la constante de fricción del péndulo.

Desarrollando las ecuaciones 2.4 y 2.5 se tiene

$$H = M_1\Big(\ddot{r} - l_s\ddot{\Phi}\cos\Phi + l_s\dot{\Phi}^2\,\mathrm{sen}\,\Phi\Big) \tag{2.7}$$

$$V = M_1l_s\Big(-\ddot{\Phi}\,\mathrm{sen}\,\Phi - \dot{\Phi}^2\cos\Phi\Big) + M_1g \tag{2.8}$$

Sustituyendo 2.7 y 2.8 en la ecuación 2.6, se obtiene

$$\begin{aligned}\Theta_s\ddot{\Phi} = {} & M_1l_s^2\,\mathrm{sen}\,\Phi\Big(-\ddot{\Phi}\,\mathrm{sen}\,\Phi - \dot{\Phi}^2\cos\Phi\Big) + M_1l_sg\,\mathrm{sen}\,\Phi + \\ & + M_1l_s^2\cos\Phi\Big(-\ddot{\Phi}\cos\Phi + \dot{\Phi}^2\,\mathrm{sen}\,\Phi\Big) + M_1l_s\ddot{r}\cos\Phi - C\dot{\Phi}\end{aligned} \tag{2.9}$$

$$\Theta_s\ddot{\Phi} + C\dot{\Phi} + M_1l_s^2\ddot{\Phi} - M_1l_s\ddot{r}\cos\Phi - M_1l_sg\,\mathrm{sen}\,\Phi = 0 \tag{2.10}$$

Si se hace $\Theta = \Theta_s + M_1 l_s^2$, entonces

$$\Theta\ddot{\Phi} + C\dot{\Phi} - M_1 l_s\Big(\ddot{r}\cos\Phi + g\,\mathrm{sen}\,\Phi\Big) = 0 \tag{2.11}$$

La expresión 2.11 es una ecuación diferencial con dos variables. Para resolverla es necesaria otra ecuación que relacione las mismas variables. Esta segunda ecuación se puede obtener analizando el diagrama de cuerpo libre del carro (Figura 2.2). Aplicando 2.1 al carro

$$F - Fric - H = M_0\frac{d^2r}{dt} \tag{2.12}$$

La fuerza de fricción $Fric$ es proporcional a la velocidad del carro, y F_r es la constante de proporcionalidad. Desarrollando $Fric$ y sustituyendo H

$$M_0\ddot{r} + F_r\dot{r} + M_1\ddot{r} + M_1 l_s\Big(\dot{\Phi}^2\,\mathrm{sen}\,\Phi - \ddot{\Phi}\cos\Phi\Big) = F \tag{2.13}$$

Si hacemos $M = M_0 + M_1$, entonces

$$M\ddot{r} + F_r\dot{r} + M_1 l_s\Big(\dot{\Phi}^2\,\mathrm{sen}\,\Phi - \ddot{\Phi}\cos\Phi\Big) = F \tag{2.14}$$

Las ecuaciones 2.11 y 2.14 representan un sistema de dos ecuaciones diferenciales simultáneas con dos incógnitas que describen al péndulo invertido.

Hasta ahora se han considerado las fricciones viscosas tanto del carro como del péndulo, pero no se consideró la fricción seca o de Coulomb. La fricción seca se comporta de acuerdo a la curva mostrada en la Figura 2.3. Cuando se le aplica una fuerza F a un cuerpo colocado en una superficie rugosa, se produce una fuerza de fricción seca F_s que se opone al movimiento. Al principio, cuando F es pequeña, el cuerpo no se mueve, lo que significa que F_s es de igual magnitud: conforme F aumenta, F_s aumenta. Pero F_s tiene un límite (F_m), y cuando F lo sobrepasa el cuerpo se comienza a mover, F_s disminuye un poco y luego se mantiene constante de ahí en adelante. A la fuerza que mantiene al cuerpo en equilibrio se le llama fricción estática, y a la que se tiene cuando el cuerpo se mueve se le denomina fricción cinética (F_k). Ver [2], capítulo 8.

La fuerza de fricción seca es relativamente difícil de modelar, por lo que de momento se dejará de lado. Puede considerarse después como una perturbación que el controlador debe compensar o puede estimarse y agregarse a la señal de control $u(k)$. En el capítulo 3 se volverá a tratar el tema de la fricción seca.

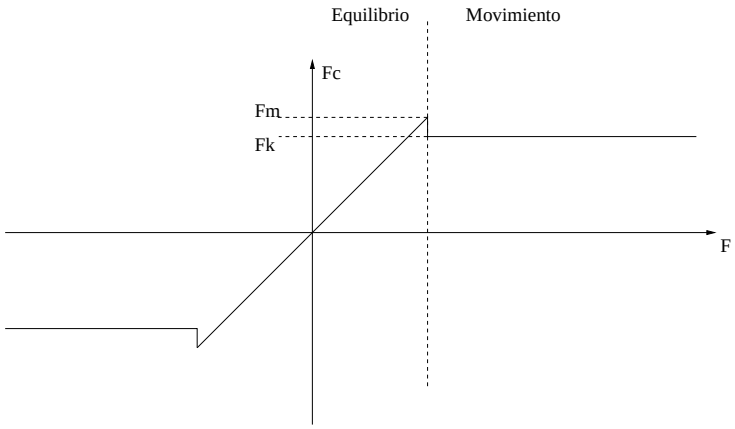

Figura 2.3: Fricción seca o de Coulomb

2.2. Ecuación de estado

Cualquier conjunto de ecuaciones diferenciales lineales ordinarias puede representarse por otro conjunto de ecuaciones de primer orden. A esta representación se le llama *ecuación de estado*. Una forma general de expresar la dinámica de un sistema lineal es (Ver Franklin y Powell [3])

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u} \tag{2.15}$$

$$\mathbf{y} = \mathbf{C}\mathbf{x} + \mathbf{D}\mathbf{u} \tag{2.16}$$

donde el vector $\mathbf{x}$ es el *estado* del sistema y contiene n elementos para un sistema de orden n. $\mathbf{u}$ es el vector de entrada y contiene m elementos. $\mathbf{y}$ contiene p elementos y es el vector de salida. $\mathbf{A}$, de dimensión $n \times n$ es la matriz del sistema. $\mathbf{B}$, de dimensión $n \times m$ es la matriz de entrada. $\mathbf{C}$, de dimensión $p \times n$ es la matriz de salida y $\mathbf{D}$ es una matriz de dimensión $p \times m$.

Este tipo de representación tiene la ventaja de que permite conocer el comportamiento interno del sistema, además de que se puede trabajar con sistemas cuyas condiciones iniciales sean diferentes de cero. Otra ventaja es que se facilita el diseño asistido por computadora, ya que los paquetes de *software* normalmente dependen de esta representación[1].

En la sección 2.1 se obtuvieron dos ecuaciones de segundo orden, por lo

[1] Además de que la representación como ecuación de estado es más compacta, es más sencillo programar operaciones entre matrices que operaciones con funciones de transferencia o ecuaciones diferenciales.

que el vector $\mathbf{x}$ tiene cuatro elementos: r, Φ, $\dot{r}$ y $\dot{\Phi}$

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} r \\ \Phi \\ \dot{r} \\ \dot{\Phi} \end{bmatrix},$$

el vector $\mathbf{u}$ tiene un elemento: la fuerza aplicada al carro.

$$u = F$$

El péndulo invertido consta de tres sensores, uno para r, otro para Φ y otro para $\dot{r}$. Por lo que se define a la salida del sistema como

$$\mathbf{y} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix},$$

con lo que se pueden determinar las matrices $\mathbf{C}$ y $\mathbf{D}$ fácilmente

$$\mathbf{C} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}, \qquad \mathbf{D} = 0.$$

Para encontrar $\mathbf{A}$ y $\mathbf{B}$ es necesario expresar las ecuaciones 2.11 y 2.14 en la forma

$$\dot{\mathbf{x}} = f(\mathbf{x}, u) \tag{2.17}$$

para $\dot{x}_1$ y $\dot{x}_2$ se tiene

$$\dot{x}_1 = f_1(\mathbf{x}, u) = x_3 \tag{2.18}$$

$$\dot{x}_2 = f_2(\mathbf{x}, u) = x_4 \tag{2.19}$$

De la ecuaciones 2.11 y 2.14, haciendo $\alpha = M_1 l_s$

$$\dot{x}_3 = \frac{\alpha \dot{x}_4 \cos x_2 - \alpha x_4^2 \operatorname{sen} x_2 - F_r x_3 + u}{M} \tag{2.20}$$

$$\dot{x}_4 = \frac{\alpha \dot{x}_3 \cos x_2 + \alpha g \operatorname{sen} x_2 - C x_4}{\Theta} \tag{2.21}$$

sustituyendo 2.21 en 2.20 y despejando $\dot{x}_3$

$$\begin{aligned} \dot{x}_3 = f_3(\mathbf{x}, u) = {} & \frac{\alpha^2 g \operatorname{sen} x_2 \cos x_2}{\Theta M - \alpha^2 \cos^2 x_2} - \frac{F_r \Theta}{\Theta M - \alpha^2 \cos^2 x_2} x_3 - \\ & - \frac{\alpha C \cos x_2}{\Theta M - \alpha^2 \cos^2 x_2} x_4 - \frac{\alpha \operatorname{sen} x_2}{\Theta M - \alpha^2 \cos^2 x_2} x_4^2 + \\ & + \frac{\Theta}{\Theta M - \alpha^2 \cos^2 x_2} u \end{aligned} \tag{2.22}$$

La ecuación 2.22 no es lineal. Para para poder representarla como ecuación de estado lineal debe tener la siguiente forma

$$\dot{x}_i = \sum_{j=1}^{n} a_{ij} x_j + b_i u$$

donde a_{ij} y b_i son constantes. Para linealizar la ecuación 2.22 se puede expresar $f_3(\mathbf{x}, u)$ como una serie de Taylor y utilizar únicamente el primer término.

$$\dot{x}_3 \approx \underline{\dot{x}}_3 = \sum_{i=1}^{4} \left[\left. \frac{\partial f_3(\mathbf{x}, u)}{\partial x_i} \right|_{\mathbf{x}=0, u=0} \Delta x_i \right] + \left. \frac{\partial f_3(\mathbf{x}, u)}{\partial u} \right|_{\mathbf{x}=0, u=0} \Delta u \qquad (2.23)$$

Al calcular las derivadas parciales y hacer $\beta = \Theta M - \alpha^2$ se tiene

$$a_{31} = \left. \frac{\partial f_3(\mathbf{x}, u)}{\partial x_1} \right|_{\mathbf{x}=0, u=0} = 0 \qquad (2.24)$$

$$a_{32} = \left. \frac{\partial f_3(\mathbf{x}, u)}{\partial x_2} \right|_{\mathbf{x}=0, u=0} = \frac{\alpha^2 g}{\beta} \qquad (2.25)$$

$$a_{33} = \left. \frac{\partial f_3(\mathbf{x}, u)}{\partial x_3} \right|_{\mathbf{x}=0, u=0} = -\frac{\Theta F_r}{\beta} \qquad (2.26)$$

$$a_{34} = \left. \frac{\partial f_3(\mathbf{x}, u)}{\partial x_1} \right|_{\mathbf{x}=0, u=0} = -\frac{\alpha C}{\beta} \qquad (2.27)$$

$$b_3 = \left. \frac{\partial f_3(\mathbf{x}, u)}{\partial u} \right|_{\mathbf{x}=0, u=0} = \frac{\Theta}{\beta} \qquad (2.28)$$

Para $\dot{x}_4$ se puede seguir un procedimiento similar. Al sustituir 2.20 en 2.21 se observa que

$$\begin{aligned} \dot{x}_4 = f_4(\mathbf{x}, u) = {} & \frac{\alpha M g \operatorname{sen} x_2}{\Theta M - \alpha^2 \cos^2 x_2} - \frac{\alpha F_r \cos x_2}{\Theta M - \alpha^2 \cos^2 x_2} x_3 - \\ & - \frac{MC}{\Theta M - \alpha^2 \cos^2 x_2} x_4 - \frac{\alpha^2 \operatorname{sen} x_2 \cos x_2}{\Theta M - \alpha^2 \cos^2 x_2} x_4^2 + \\ & + \frac{\alpha \cos x_2}{\Theta M - \alpha^2 \cos^2 x_2} u \end{aligned} \qquad (2.29)$$

y las derivadas parciales son

$$a_{41} = \left.\frac{\partial f_4(\mathbf{x}, u)}{\partial x_1}\right|_{\mathbf{x}=0,u=0} = 0 \tag{2.30}$$

$$a_{42} = \left.\frac{\partial f_4(\mathbf{x}, u)}{\partial x_2}\right|_{\mathbf{x}=0,u=0} = \frac{\alpha M g}{\beta} \tag{2.31}$$

$$a_{43} = \left.\frac{\partial f_4(\mathbf{x}, u)}{\partial x_3}\right|_{\mathbf{x}=0,u=0} = -\frac{\alpha F_r}{\beta} \tag{2.32}$$

$$a_{44} = \left.\frac{\partial f_4(\mathbf{x}, u)}{\partial x_1}\right|_{\mathbf{x}=0,u=0} = -\frac{MC}{\beta} \tag{2.33}$$

$$b4 = \left.\frac{\partial f_4(\mathbf{x}, u)}{\partial u}\right|_{\mathbf{x}=0,u=0} = \frac{\alpha}{\beta} \tag{2.34}$$

A partir de este punto, ya se pueden determinar los coeficientes de $\mathbf{A}$ y $\mathbf{B}$

$$\mathbf{A} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & a_{32} & a_{33} & a_{34} \\ 0 & a_{42} & a_{43} & a_{44} \end{bmatrix} \qquad \mathbf{B} = \begin{bmatrix} 0 \\ 0 \\ b_3 \\ b_4 \end{bmatrix}.$$

La ecuación de estado es entonces

$$\dot{\underline{\mathbf{x}}} = \mathbf{A}\underline{\mathbf{x}} + \mathbf{B}\Delta u, \tag{2.35}$$

donde $\underline{\mathbf{x}}$ son las desviaciones del estado alrededor del punto de operación.

Los parámetros del péndulo invertido se pueden encontrar en [1] y se muestran en la tabla 2.1.

Como ya se mencionó, el péndulo invertido contiene tres sensores. Los sensores de posición y velocidad entregan un voltaje proporcional a la variable medida. El sensor de posición angular consta de un *encoder* óptico conectado a un contador. El valor de dicho contador –un entero de 12 bits– es proporcional al ángulo del péndulo. Para simplificar el modelado de los sensores se optó por multiplicar el valor del contador por una constante, de tal suerte que se pudiera conceptualizar este sensor como un transductor más. El sistema tiene, además, un motor y un amplificador que transforman un voltaje en un par mecánico. Se puede entender estos transductores como una transformación lineal al vector de estado y a la entrada

$$\mathbf{x_n} = \mathbf{N}\underline{\mathbf{x}} \tag{2.36}$$

$$u = K_f u_s \tag{2.37}$$

	Parámetro	Valor	Unidad
Parámetros Básicos	M_0	3.2	Kg
	M_1	0.329	Kg
	l_s	0.44	m
	Θ_s	0.008	Kgm^2
	F_r	6.2	Kg/s
	C	0.009	Kgm^2/s
Parámetros Compuestos	M	3.529	Kg
	Θ	0.072	Kgm^2
	α	0.1448	Kgm
	β	0.23313	Kgm

Cuadro 2.1: Parámetros del péndulo invertido

Parámetro	Valor	Unidad
K_f	2.6	N/V
n_{11}	14.9	V/m
n_{22}	52.27	V/rad
n_{33}	-7.64	Vs/m
n_{44}	52.27	Vs/rad

Cuadro 2.2: Parámetros de los transductores

donde $\mathbf{x_n}$ es el vector de estado normalizado, $\mathbf{N}$ es una matriz diagonal cuyos parámetros se encuentran en la tabla 2.2, u_s es la señal de entrada normalizada y K_f es la relación que existe entre el voltaje a la entrada del amplificador y la fuerza aplicada al carro.

$$\mathbf{N} = \begin{bmatrix} n_{11} & 0 & 0 & 0 \\ 0 & n_{22} & 0 & 0 \\ 0 & 0 & n_{33} & 0 \\ 0 & 0 & 0 & n_{44} \end{bmatrix} \tag{2.38}$$

Los elementos n_{11} a n_{33} son los factores que relacionan las cantidades eléctricas con las mecánicas. Como $\dot{\Phi}$ no se puede medir directamente, ha de ser estimado más tarde. El elemento n_{44}, se puede, entonces, elegir arbitrariamente. Por simplicidad se eligió $n_{44} = n_{22}$ s.

Si se sustituyen las ecuaciones 2.36 y 2.37 en 2.35 se puede ver que

$$\mathbf{N^{-1}}\dot{\mathbf{x}}_\mathbf{n} = \mathbf{A}\mathbf{N^{-1}}\mathbf{x_n} + \mathbf{B}K_f u_s \tag{2.39}$$

y al despejar $\dot{\mathbf{x}}_\mathbf{n}$

$$\dot{\mathbf{x}}_\mathbf{n} = \mathbf{NAN^{-1}x_n} + \mathbf{NB}K_f u_s \tag{2.40}$$

de donde se puede obtener la matriz normalizada del sistema

$$\mathbf{A_n} = \mathbf{NAN^{-1}} \tag{2.41}$$

y la matriz normalizada de entrada

$$\mathbf{B_n} = \mathbf{NB}K_f \tag{2.42}$$

Calculando

$$\mathbf{A} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0.883 & -1.916 & -0.0056 \\ 0 & 21.53 & -3.868 & -0.1369 \end{bmatrix} \tag{2.43}$$

$$\mathbf{B} = \begin{bmatrix} 0 \\ 0 \\ 0.30896 \\ 0.62382 \end{bmatrix} \tag{2.44}$$

y

$$\mathbf{A_n} = \begin{bmatrix} 0 & 0 & -1.95 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & -0.129 & -1.916 & 0.00082 \\ 0 & 21.53 & 26.461 & -0.1369 \end{bmatrix} \tag{2.45}$$

$$\mathbf{B_n} = \begin{bmatrix} 0 \\ 0 \\ -6.137 \\ 84.778 \end{bmatrix} \tag{2.46}$$

Finalmente,

$$\begin{aligned} \dot{\mathbf{x}}_\mathbf{n} &= \mathbf{A_n x_n} + \mathbf{B_n} u \\ \mathbf{y_n} &= \mathbf{C x_n} \end{aligned} \tag{2.47}$$

2.3. Modelo discreto

Al implementar controladores por medio de sistemas digitales se puede elegir normalmente entre dos opciones, la primera consiste en utilizar el

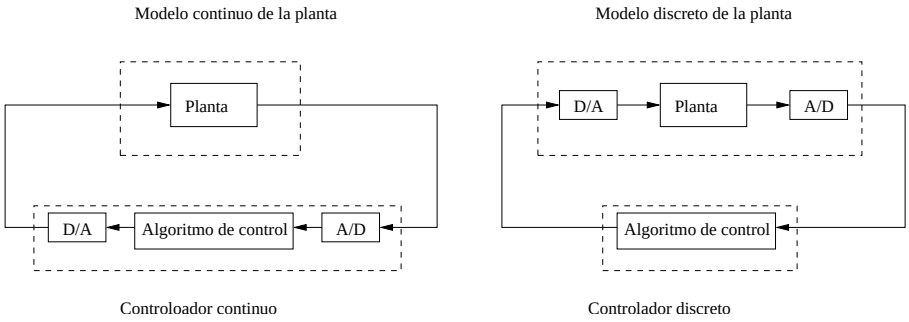

Figura 2.4: Esquemas para diseñar en el tiempo continuo (izquierda) y en el discreto (derecha)

modelo continuo del sistema o planta que se quiere controlar, diseñar el controlador en el tiempo continuo y discretizarlo para construirlo sobre un sistema digital. A este procedimiento se le llama diseño por emulación. La segunda opción es obtener el modelo discreto de la planta y realizar directamente el diseño del controlador en el tiempo discreto.

En la Figura 2.4 se muestran los dos enfoques. Al diseñar por emulación, resulta un controlador que se encuentra formado por los convertidores A/D, D/A y un controlador discreto. Este controlador emula el comportamiento de un control exclusivamente continuo, pero no se comporta exactamente de la misma forma, el grado de exactitud con respecto al controlador exclusivamente continuo depende de la frecuencia de muestreo. En el segundo caso (lado derecho de la figura), la planta se conforma por los convertidores D/A, A/D y la planta continua. En este caso la planta sí exhibe un comportamiento igual al de una planta exclusivamente discreta. En este sentido el segundo enfoque es más exacto.

El siguiente paso es obtener el modelo discreto del sistema. Para llevar esto a cabo se necesita determinar el periodo de muestreo. La frecuencia de muestreo recomendada [3] es

$$20 < \frac{\omega_s}{\omega_b} < 40 \tag{2.48}$$

donde ω_s es la frecuencia de muestreo y ω_b es el ancho de banda del sistema en lazo cerrado. Debido a que el controlador no ha sido diseñado todavía, no se conoce el valor exacto de ω_b. Para fines prácticos ω_b se puede suponer, tentativamente, igual al polo más rápido del sistema. Los polos del sistema

están determinados por los valores característicos de $\mathbf{A_n}$

$$\begin{aligned} \det(s\mathbf{I} - \mathbf{A_n}) &= 0 && (2.49)\\ s_1 &= 0\\ s_2 &= 4.52\\ s_3 &= -4.84\\ s_4 &= -1.73\\ \Rightarrow \omega_b &\approx 4.84 \text{ [rad/s]} && (2.50) \end{aligned}$$

Si se elige una frecuencia de muestreo de 40 veces la del polo más rápido

$$\omega_s = 40 \times 4.84 = 193.48 \text{ [rad/s]} \quad (2.51)$$

$$T_s = \frac{2\pi}{\omega_s} \approx 30 \text{ [ms]} \quad (2.52)$$

Cabe mencionar que la ecuación 2.48 ha de verificarse una vez que se haya diseñado el controlador.

Una vez calculado el periodo de muestreo se pueden utilizar diversas transformaciones para determinar el modelo discreto del sistema. El más directo es el retenedor de orden cero, ya que eso es precisamente lo que hay a la entrada de la planta. Utilizando las rutinas del paquete *Matlab* se calculan las matrices

$$\mathbf{A_d} = \begin{bmatrix} 1 & 1.11 \times 10^{-4} & -5.69 \times 10^{-2} & 4.10 \times 10^{-7} \\ 0 & 1.01 & 1.17 \times 10^{-2} & 3.00 \times 10^{-2} \\ 0 & -3.77 \times 10^{-3} & 9.44 \times 10^{-1} & -3.31 \times 10^{-5} \\ 0 & 6.45 \times 10^{-1} & 7.72 \times 10^{-1} & 1.01 \end{bmatrix} \quad (2.53)$$

$$\mathbf{B_d} = \begin{bmatrix} 5.28 \times 10^{-3} \\ 3.74 \times 10^{-2} \\ -1.79 \times 10^{-1} \\ 2.47 \end{bmatrix} \quad (2.54)$$

$$\mathbf{C_d} = \mathbf{C} \quad (2.55)$$

$$\mathbf{D_d} = \mathbf{D} \quad (2.56)$$

y

$$\mathbf{x_n}(k+1) = \mathbf{A_d}\mathbf{x_n}(k) + \mathbf{B_d}u(k) \quad (2.57)$$

$$\mathbf{y_n}(k) = \mathbf{C_d}\mathbf{x_n}(k) + \mathbf{D_d}u(k) \quad (2.58)$$

Capítulo 3

Estabilización del Péndulo

En este capítulo se describe el diseño del controlador que estabiliza la posición inherentemente inestable del péndulo. Se explica la forma en que se implementó el sistema y al final se presentan los resultados obtenidos.

3.1. Diseño del control

En el diseño propuesto se plantea retroalimentar el estado a través de una ganancia determinada con el método LQR (Linear Quadratic Regulator). El estado que se mide no es completo, por lo que se propone utilizar un observador predictor de orden reducido para estimar parte del estado.

3.1.1. Bases teóricas

La ventaja de tener al sistema representado en el espacio de estados es que el diseño de un controlador LQR resulta más sencillo. La idea es establecer la retroalimentación mediante una combinación lineal de las variables de estado. Otra de las ventajas de este método, es que se puede separar el problema en dos pasos independientes. Se asume que se tienen disponibles todas las variables de estado para ser retroalimentadas; esto, aunque no necesariamente sea cierto[1] nos permite proceder con el primer paso: obtener la ley de control, es decir, la expresión que nos permite calcular la señal de control. El segundo paso consiste en diseñar un observador que estime el estado completo a partir de una porción dada. El algoritmo de control final está formado

[1]De hecho, para un problema de control general, sería poco práctico comprar una gran cantidad de sensores sabiendo que no son necesarios si se utilizan métodos clásicos (respuesta en frecuencia).

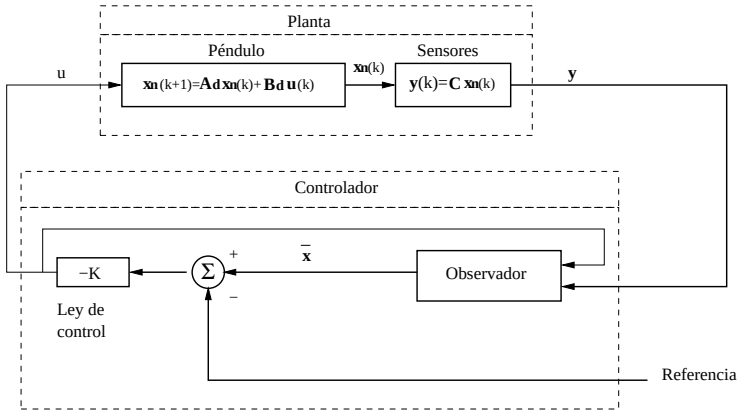

Figura 3.1: Diseño en espacio de estados

entonces, por la combinación del observador y la ley de control.

En la Figura 3.1 se puede ver un esquema del algoritmo de control. $\mathbf{K}$ es la matriz que retroalimenta el estado para obtener la señal de control, es decir: $u(k) = -\mathbf{K}\bar{\mathbf{x}}(k)$.

3.1.2. Retroalimentación del estado

Para obtener la ganancia $\mathbf{K}$ hay varios criterios, uno es asignar arbitrariamente el patrón de polos que se quiere que tenga el sistema en lazo cerrado; otra posibilidad es utilizar un control óptimo variable en el tiempo (*time-varying optimal control*). La opción elegida es un control óptimo LQR (*Linear Quadratic Regulator*), un caso particular del variable en el tiempo.

Se trata de minimizar la función

$$J = \sum_{k=0}^{N} \left[\mathbf{x_n}^T(k)\mathbf{Q}\mathbf{x}_n(k) + \mathbf{u}^T(k)\mathbf{R}\mathbf{u}(k) \right]. \tag{3.1}$$

J es una función de costo que generalmente está asociada con la energía del sistema. Se trata de un control óptimo en el sentido de que se busca minimizar la energía. La expresión $\mathbf{x_n}^T(k)\mathbf{Q}\mathbf{x}_n(k)$ representa la energía que aporta cada estado, $\mathbf{Q}$ es una matriz no negativa definida (una posibilidad es una matriz diagonal, cuyos elementos sean positivos). A través de $\mathbf{Q}$ se puede elegir el *peso* que tiene cada estado en la función J. Se puede conceptualizar $\mathbf{x_n}^T(k)\mathbf{Q}\mathbf{x}_n(k)$, como una x^2 (la energía normalizada) en un sistema con una sola variable. Como en este caso $u(k)$ contiene un sólo elemento, $R \geq 0$

es un elemento que indica el peso que se le quiere dar a la energía asociada con la señal de control.

La ecuación 3.1 está sujeta a la restricción impuesta por el sistema

$$\mathbf{x_n}(k+1) = \mathbf{A_d}\mathbf{x_n}(k) + \mathbf{B_d}u(k) \tag{3.2}$$

Las ecuaciones 3.1 y 3.2 conforman un problema típico de obtención de mínimos con restricciones o condiciones laterales que puede ser resuelto utilizado los multiplicadores de Lagrange (ver Marsden y Tromba [5] capítulo 4); con la gran complicación de que se trata de ecuaciones en diferencias en vez de algebraicas.

La solución no es fácil de obtener. Una forma de resolver el problema es utilizando métodos numéricos. Un método se muestra en [6], capítulo 7 y otro en [3], capítulo 9. La solución en ambos casos resulta en un ley de control variable en el tiempo, es decir

$$u(k) = -\mathbf{K}(k)\mathbf{x_n}(k)$$

Para sistemas con coeficientes constantes, $\mathbf{K}(k)$ permanece fija durante un periodo de tiempo y luego decae a cero. Si se hace $N \to \infty$ en 3.1 entonces la ganancia del controlador permanece fija todo el tiempo

$$\mathbf{K}(k) = \mathbf{K}$$

En este caso, se puede utilizar la ecuación de Hamilton

$$\begin{bmatrix} \mathbf{x_n}(k+1) \\ \lambda(k+1) \end{bmatrix} = \begin{bmatrix} \mathbf{A_d} + \mathbf{B_d}R^{-1}\mathbf{B_d}^{\mathbf{T}}\mathbf{A_d}^{-\mathbf{T}}\mathbf{Q} & -\mathbf{B_d}R^{-1}\mathbf{B_d}^{\mathbf{T}}\mathbf{A_d}^{-\mathbf{T}} \\ -\mathbf{A_d}^{-\mathbf{T}}\mathbf{Q} & \mathbf{A_d}^{-\mathbf{T}} \end{bmatrix} \begin{bmatrix} \mathbf{x_n}(k) \\ \lambda(k) \end{bmatrix} \tag{3.3}$$

donde λ es el multiplicador de Lagrange.

Para un sistema de orden n, la dinámica descrita por la ecuación 3.3 contiene $2n$ polos, de los cuales la mitad son estables y la otra mitad son inestables; de hecho, los polos inestables son los recíprocos de los polos estables. Los polos estables de 3.3 son los polos en lazo cerrado que hacen que se satisfagan 3.1 y 3.2. A partir de estos polos se puede obtener $\mathbf{K}$ utilizando la fórmula de Ackermann. Para un desarrollo más detallado se recomienda consultar [3], capítulo 9.

El valor exacto de J no es relevante, sólo se pretende encontrar la $\mathbf{K}$ que asegure que sea mínimo; lo importante es el valor relativo que tienen los elementos de $\mathbf{Q}$ y R entre sí. Se puede hacer arbitrariamente $R = 1$ y

proponer $\mathbf{Q}$ a partir de ahí. Una posibilidad es

$$\mathbf{Q} = \begin{bmatrix} 650 & 0 & 0 & 0 \\ 0 & 100 & 0 & 0 \\ 0 & 0 & 20 & 0 \\ 0 & 0 & 0 & 10 \end{bmatrix} \tag{3.4}$$

La selección de esta matriz sigue este razonamiento: la sensibilidad del sensor de posición angular es mucho mayor que la del sensor de posición lineal (ver tabla 2.2) Nos importa que tanto la posición angular como lineal tengan pocas variaciones, aunque las velocidades pueden ser grandes. No importa cuanta energía de u se utilice (siempre y cuando no se sature el actuador).

Ahora se puede calcular $\mathbf{K}$,

$$\mathbf{K} = \begin{bmatrix} -2.87469 & 3.46593 & 5.35264 & 0.80618 \end{bmatrix}, \tag{3.5}$$

Una vez que se tiene $\mathbf{K}$, se puede verificar que la frecuencia de muestreo sea correcta, es decir, que cumpla con la desigualdad 2.48. El siguiente paso es calcular el ancho de banda de lazo cerrado ω_b. Existen varias definiciones de ancho de banda. Una de las más comunes es (ver [4], capítulo 2):

El ancho de banda de 3 dB es $f_2 - f_1$, donde a frecuencias dentro de la banda $f_1 < f < f_2$, los espectros de magnitud, es decir $|H(f)|$, se reducen no menos de $1/\sqrt{2}$ veces el valor máximo de $|H(f)|$, y el valor máximo se reduce a una frecuencia dentro de la banda.

En la Figura 3.2 se puede ver la respuesta en frecuencia de las diferentes salidas del sistema en lazo cerrado cuando se considera como entrada la referencia en la posición. En la figura se puede observar que el ancho de banda máximo es aproximadamente igual a 6.5 rad/s. El cociente ω_s/ω_b es de al rededor de 30, lo cual indica que se cumple la desigualdad 2.48.

3.1.3. Observador de Luenberger Reducido

Hasta ahora se ha asumido que se cuenta con el estado completo. En realidad no se cuenta con un sensor para medir $\dot{\Phi}$. Para estimar $\dot{\Phi}$ se puede utilizar un observador de Luenberger. Se puede aprovechar también este observador para estimar la fricción seca F_s de la que se hablaba en el capítulo 2. En la Figura 3.3 se muestra un diagrama de bloques de un observador de orden completo (ver [3] capítulo 8). La ecuación en diferencias que describe al estado estimado $\bar{\mathbf{x}}(k)$ es

$$\bar{\mathbf{x}}(k+1) = \mathbf{A_d}\bar{\mathbf{x}}(k) + \mathbf{B_d}u(k) + \mathbf{L}[\mathbf{y}(k) - \mathbf{C}\bar{\mathbf{x}}(k)] \tag{3.6}$$

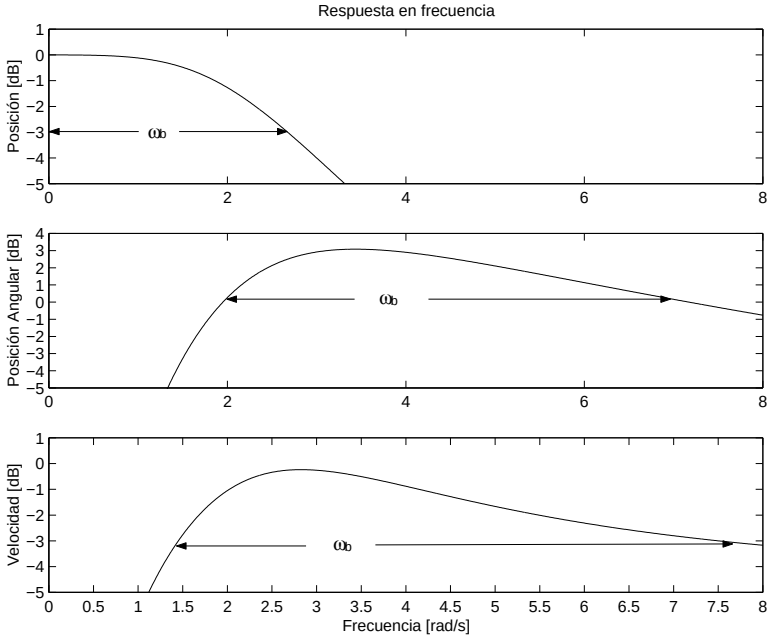

Figura 3.2: Respuesta en frecuencia de las diferentes salidas del sistema en lazo cerrado cuando se considera la referencia en la posición como la entrada del sistema

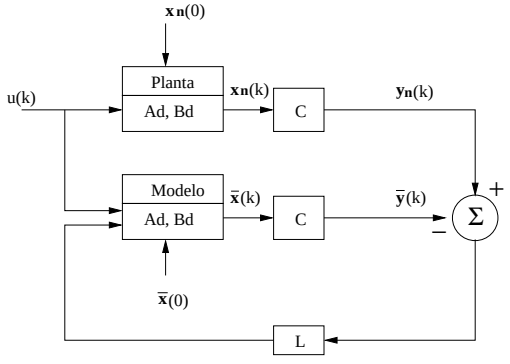

Figura 3.3: Observador de Luenberger

La dinámica del error ($\tilde{\mathbf{x}} = \mathbf{x_n} - \bar{\mathbf{x}}$) depende de $\mathbf{L}$ y está dada por

$$\det(z\mathbf{I} - \mathbf{A_d} + \mathbf{LC}) = 0 \tag{3.7}$$

donde los valores de z corresponden a los polos del estimador.

El observador descrito arriba estima al estado completo. Si se quiere estimar únicamente una parte del estado se puede utilizar un observador de orden reducido. Al diseñar un observador de esta naturaleza, se separa el vector de estados en dos partes, una medida $\mathbf{x_a}$ y otra estimada $\mathbf{x_b}$. La ecuación de estado queda de la siguiente forma:

$$\begin{bmatrix}\mathbf{x_a}(k+1)\\ \mathbf{x_b}(k+1)\end{bmatrix} = \begin{bmatrix}\mathbf{A_{aa}} & \mathbf{A_{ab}}\\ \mathbf{A_{ba}} & \mathbf{A_{bb}}\end{bmatrix}\begin{bmatrix}\mathbf{x_a}(k)\\ \mathbf{x_b}(k)\end{bmatrix} + \begin{bmatrix}\mathbf{B_a}\\ \mathbf{B_b}\end{bmatrix} u(k) \tag{3.8}$$

La parte que describe la dinámica de los elementos no medidos es

$$\mathbf{x_b}(k+1) = \mathbf{A_{bb}}\mathbf{x_b}(k) + \underbrace{\mathbf{A_{ba}}\mathbf{x_a}(k) + \mathbf{B_b}u(k)}_{\text{“entrada” conocida}} \tag{3.9}$$

Los dos últimos elementos son conocidos y pueden considerarse como entradas a la dinámica de $\mathbf{x_b}$. Si se reordena la parte que describe la dinámica de $\mathbf{x_a}$ se tiene que

$$\underbrace{\mathbf{x_a}(k+1) - \mathbf{A_{aa}}\mathbf{x_a}(k) - \mathbf{B_a}u(k)}_{\text{“medición” conocida}} = \mathbf{A_{ab}}\mathbf{x_b}(k) \tag{3.10}$$

Las ecuaciones 3.9 y 3.10 tienen la misma forma que las ecuaciones 2.57 y 2.58 respectivamente, por que lo se puede utilizar la ecuación 3.6 si se hacen las siguientes sustituciones:

$$\begin{aligned}
\mathbf{x_n} &\leftarrow \mathbf{x_b}\\
\mathbf{A_d} &\leftarrow \mathbf{A_{bb}}\\
\mathbf{B_d}u(k) &\leftarrow \mathbf{A_{ba}}\mathbf{x_a}(k) + \mathbf{B_b}u(k)\\
\mathbf{y_n}(k) &\leftarrow \mathbf{x_a}(k+1) - \mathbf{A_{aa}}\mathbf{x_a}(k) - \mathbf{B_a}u(k)\\
\mathbf{C_d} &\leftarrow \mathbf{A_{ab}}
\end{aligned}$$

Al sustituir, se puede ver que la ecuación que describe la porción faltante del estado esta dada por

$$\begin{aligned}
\bar{\mathbf{x}}_\mathbf{b}(k+1) = {} & \mathbf{A_{bb}}\bar{\mathbf{x}}_\mathbf{b}(k) + \mathbf{A_{ba}}\mathbf{x_a}(k) + \mathbf{B_b}u(k) + \\
& + \mathbf{L_r}[\mathbf{x_a}(k+1) - \mathbf{A_{aa}}\mathbf{x_a}(k) - \mathbf{B_a}u(k) - \mathbf{A_{ab}}\bar{\mathbf{x}}_\mathbf{b}(k)]
\end{aligned} \tag{3.11}$$

$\mathbf{L_r}$ es la matriz que se debe encontrar para hacer que el error se comporte de manera estable.

Si se resta la ecuación 3.11 de la 3.9 se obtiene la dinámica del error de estimación

$$\tilde{\mathbf{x}}_\mathbf{b}(k+1) = [\mathbf{A_{bb}} - \mathbf{L_r A_{ab}}]\tilde{\mathbf{x}}_\mathbf{b}(k) \tag{3.12}$$

y la forma de obtener $\mathbf{L_r}$ sería la misma que antes

$$\det(z\mathbf{I} - \mathbf{A_{bb}} + \mathbf{L_r A_{ab}}) = 0 \tag{3.13}$$

La velocidad angular ya se encuentra incluida en el modelo del sistema, pero la fricción seca no. Para incluirla en el modelo, se puede considerar como un quinto estado x_5. Por simplicidad se puede asumir que es constante (tomar únicamente la fricción cinética). La ecuación de estado para esta variable es entonces

$$\dot{x}_5 = 0 \tag{3.14}$$

Como se trata de una fuerza que tiene la misma línea de acción que u, se relaciona con las demás variables de estado a través de $\mathbf{B_d}$. La ecuación, con el nuevo estado es

$$\begin{bmatrix} \mathbf{x_n}(k+1) \\ x_{n5}(k+1) \end{bmatrix} = \begin{bmatrix} \mathbf{A_d} & \mathbf{B_d} \\ \mathbf{0} & 0 \end{bmatrix} \begin{bmatrix} \mathbf{x_n}(k) \\ x_{n5}(k) \end{bmatrix} + \begin{bmatrix} \mathbf{B_d} \\ 0 \end{bmatrix} u(k) \tag{3.15}$$

Si se compara la ecuación 3.8 con la 3.15 se puede ver que

$$\mathbf{A_{aa}} = \begin{bmatrix} A_{d11} & A_{d12} & A_{d13} \\ A_{d21} & A_{d22} & A_{d13} \\ A_{d31} & A_{d32} & A_{d33} \end{bmatrix} \quad \mathbf{A_{ab}} = \begin{bmatrix} A_{d14} & B_{d1} \\ A_{d24} & B_{d2} \\ A_{d34} & B_{d3} \end{bmatrix}$$

$$\mathbf{A_{ba}} = \begin{bmatrix} A_{d41} & A_{d42} & A_{d43} \\ 0 & 0 & 0 \end{bmatrix} \quad \mathbf{A_{bb}} = \begin{bmatrix} A_{d44} & B_{d4} \\ 0 & 0 \end{bmatrix}$$

y

$$\mathbf{B_a} = \begin{bmatrix} B_{d1} \\ B_{d2} \\ B_{d3} \end{bmatrix} \quad \mathbf{B_b} = \begin{bmatrix} B_{d4} \\ 0 \end{bmatrix}$$

En el capítulo anterior se vio que el polo más rápido en lazo abierto es igual -4.84 [rad/s]. La idea es que los polos del observador sean considerablemente más rápidos, por ejemplo

$$s_1 = s_2 = -100 \text{ [rad/s]} \tag{3.16}$$

y pasándolos al plano discreto con $T_s = 0.03$ s

$$k = e^{sT_s} \tag{3.17}$$

$$k_1 = k_2 = 0.04979 \tag{3.18}$$

Para resolver la ecuación 3.13 y encontrar $\mathbf{L_r}$, se puede adaptar un poco el problema y usar la fórmula de Ackermann. La fórmula de Ackermann encuentra la $\mathbf{K}$ que satisface la condición (ver [3], capítulo 8)

$$\det(s\mathbf{I} - \mathbf{A} + \mathbf{BK}) = 0 \tag{3.19}$$

Al trasponer la matriz del sistema en la ecuación que describe la dinámica del error (3.12)

$$[\mathbf{A_{bb}} - \mathbf{L_r A_{ab}}]^T = \mathbf{A_{bb}}^T - \mathbf{A_{ab}}^T \mathbf{L_r}^T \tag{3.20}$$

Se puede usar la fórmula de Ackermann si se sustituye k por s, $\mathbf{A_{bb}}^T$ por $\mathbf{A}$, $\mathbf{A_{ab}}^T$ por $\mathbf{B}$ y $\mathbf{L_r}^T$ por $\mathbf{K}$. Esto da

$$\mathbf{L_r} = \begin{bmatrix} 0.21107 & 0.31814 & -7.1667 \\ -0.00821 & 0.00031 & 0.2781 \end{bmatrix} \tag{3.21}$$

Si se reordena la ecuación 3.11 se tiene

$$\begin{aligned} \bar{\mathbf{x}}_\mathbf{b}(k+1) - \mathbf{L_r x_a}(k+1) = {} & [\mathbf{A_{bb}} - \mathbf{L_r A_{ab}}]\bar{\mathbf{x}}_\mathbf{b}(k) - \\ & - [\mathbf{A_{bb}} - \mathbf{L_r A_{ab}}]\mathbf{L_r x_a}(k) + \\ & + \Big[[\mathbf{A_{bb}} - \mathbf{L_r A_{ab}}]\mathbf{L_r} + \mathbf{A_{ba}} - \mathbf{L_r A_{aa}}\Big]\mathbf{x_a}(k) + \\ & + [\mathbf{B_b} - \mathbf{L_r B_a}]u(k) \end{aligned} \tag{3.22}$$

y si se hace

$$\mathbf{z}(k) = \bar{\mathbf{x}}_\mathbf{b}(k) - \mathbf{L_r x_a}(k) \tag{3.23}$$

$$\begin{aligned} \mathbf{A_L} = \mathbf{A_{bb}} - \mathbf{L_r A_{ab}} &= \begin{bmatrix} 0.04979 & 0 \\ 0 & 0.04979 \end{bmatrix} \\ \mathbf{F_L} = \mathbf{A_L L_r} + \mathbf{A_{ba}} - \mathbf{L_r A_{aa}} &= \begin{bmatrix} -0.20056 & -29.92 & 6.8222 \\ 7.8037 & 0.00075 & -0.2492 \end{bmatrix} \\ \mathbf{B_L} = \mathbf{B_b} - \mathbf{L_r B_a} &= \begin{bmatrix} 0 \\ 0.04979 \end{bmatrix} \end{aligned}$$

la dinámica de $\mathbf{z}$ está dada por

$$\mathbf{z}(k+1) = \mathbf{A_L z}(k) + \mathbf{F_L x_a}(k) + \mathbf{B_L}u(k) \tag{3.24}$$

En la Figura 3.4 se muestra un esquema completo del controlador. Las matrices $\mathbf{C_L}$ y $\mathbf{V_L}$ permiten recuperar $\bar{\mathbf{x}}_\mathbf{b}$ (a partir de la ecuación 3.23), y ponerlo en un vector junto con $\mathbf{x_a}$.

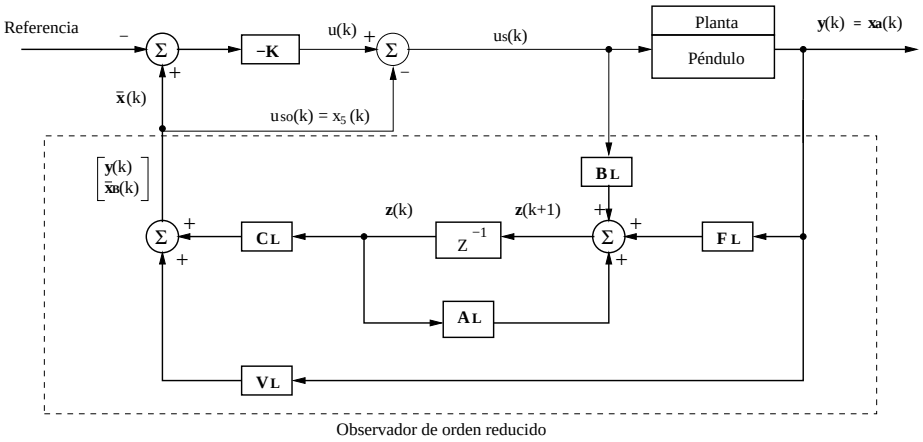

Figura 3.4: Esquema del controlador para estabilizar el péndulo

$$\mathbf{V_L} = \begin{bmatrix} \mathbf{I} \\ \mathbf{L_r} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0.21107 & 0.31814 & -7.1667 \\ -0.00821 & 0.00031 & 0.2781 \end{bmatrix}$$

$$\mathbf{C_L} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix}$$

3.2. Implementación del control

Antes de implementar el esquema mostrado en la Figura 3.4 hay que simularlo para detectar posibles errores. En el capítulo 2 se obtuvo un par de ecuaciones simultáneas diferenciales (2.11 y 2.14) que pueden usarse directamente para modelar el péndulo invertido. Simulink, un módulo de Matlab, permite realizar simulaciones de modelos a través de una interfaz gráfica. Además de las ecuaciones diferenciales, podemos utilizar los parámetros que describen al actuador y a los sensores y agregarlos al modelo. Finalmente, se puede incluir la fricción seca utilizando una curva como la mostrada en la Figura 2.3.

En las Figuras 3.5 y 3.6 se pueden ver los resultados de la simulación del modelo descrito por 2.11, 2.14 y una fricción seca máxima de 1.3 N cuando

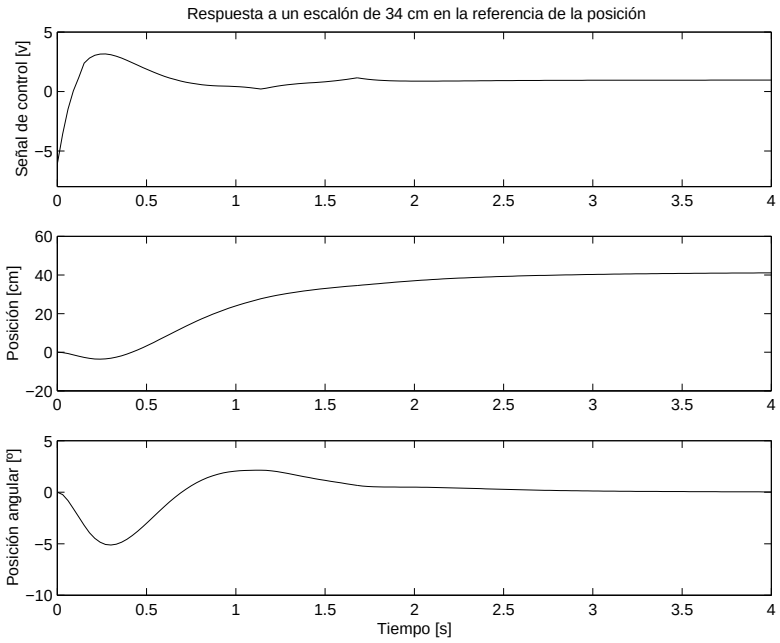

Figura 3.5: Simulación de la señal de control, la posición y la posición angular al aplicar un escalón en la referencia de la posición

se aplica un escalón en la referencia de la posición. Se puede observar que el sistema es estable, que las variaciones en el ángulo no sobrepasan los 7° y que la estimación de la velocidad angular no contiene errores de más del 6 %. La estimación de la fricción contiene errores relativamente grandes. Este error no es muy grave ya que no sobrepasa F_m.

El péndulo trae incluida una tarjeta de adquisición de datos (DAC 6214). Matlab cuenta con un módulo (*Real Time Workshop*) que permite utilizar directamente Simulink para construir un controlador en tiempo real con capacidad para manejar interrupciones[2]. Una opción práctica sería desarrollar una interfaz para manejar la tarjeta desde Matlab y utilizar el modelo del controlador que ya se tiene construido, sustituyendo el modelo de la planta por las interfaces con la tarjeta (ver Figura 3.7)

Una de las ventajas que Matlab tiene, es la opción de que el usuario escriba rutinas en el lenguaje C, las compile, y después las use como si

[2]Si bien el control óptimo es bastante noble en el sentido de que es insensible a las variaciones de los parámetros del sistema, requiere un tiempo de muestreo preciso, determinado por *hardware*, no por *software*.

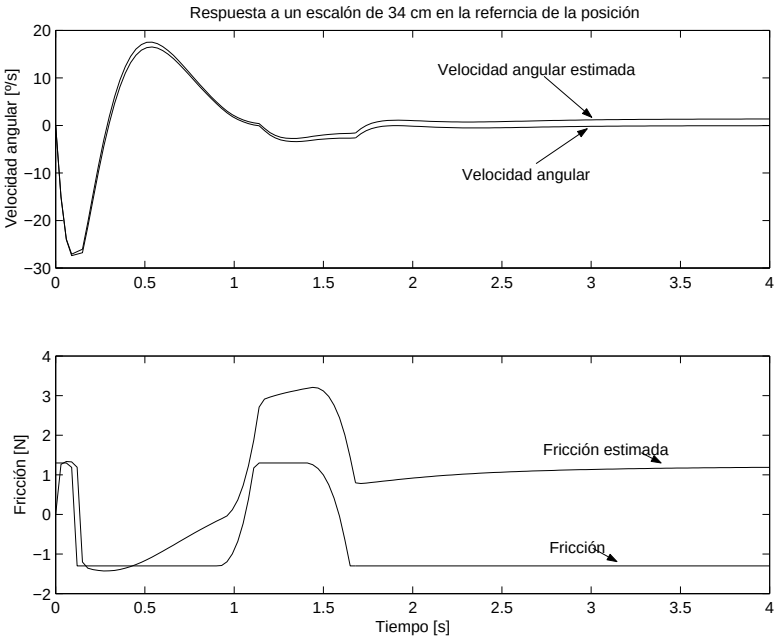

Figura 3.6: Simulación de la señales de velocidad angular, fricción y sus respectivos estimados al aplicar un escalón en la referencia de la posición

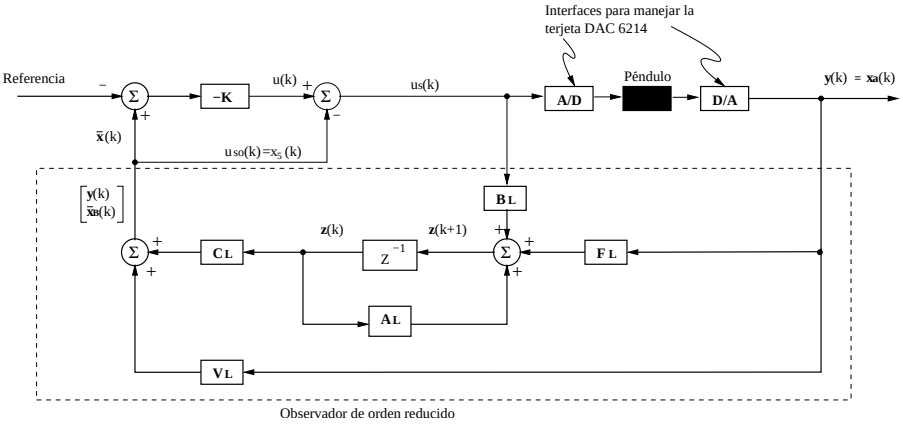

Figura 3.7: Implementación del controlador para estabilizar el péndulo

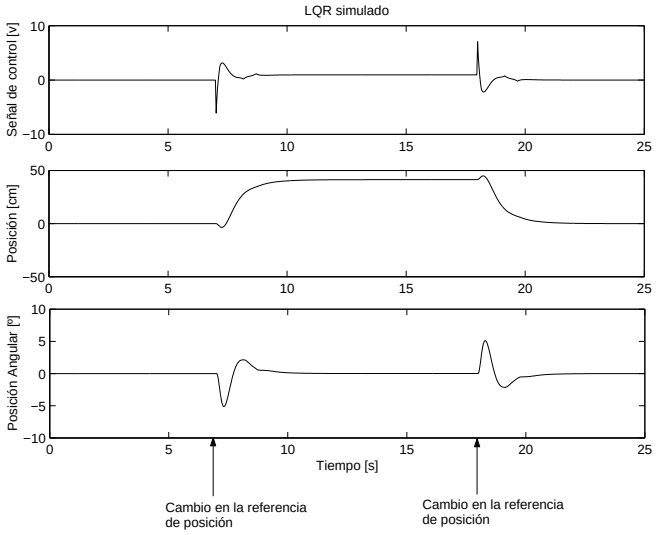

Figura 3.8: Simulación del algoritmo de control usando LQR estimando la fricción

fueran parte de Matlab. Se desarrollaron las interfaces para manejar las tarjetas utilizando esta característica. En el apéndice A se encuentran los códigos fuente.

3.3. Análisis de datos y resultados

Se consideraron tres elementos importantes para probar el control: su insensibilidad a perturbaciones externas (un empujón), su capacidad para seguir a un escalón en la entrada y su habilidad para reaccionar ante una variación en los parámetros (se colocó sobre el péndulo un peso de aproximadamente el 30 % de su valor original, con esto se consiguió cambiar tanto su masa, como su centro de gravedad).

En las Figuras 3.8 y 3.9 se muestran la simulación y los resultados experimentales. Se puede observar que la respuesta de la posición y el ángulo son muy similares a las simuladas. La señal de control es diferente porque compensa el ruido externo. Se puede ver también que responde adecuadamente a perturbaciones externas y que la dinámica del sistema cambia poco cuando se cambian los parámetros del péndulo (la masa).

En las Figuras 3.10 y 3.11 se muestra el mismo experimento, sólo que no se incluyó la estimación de la fricción. El sistema se comporta de manera

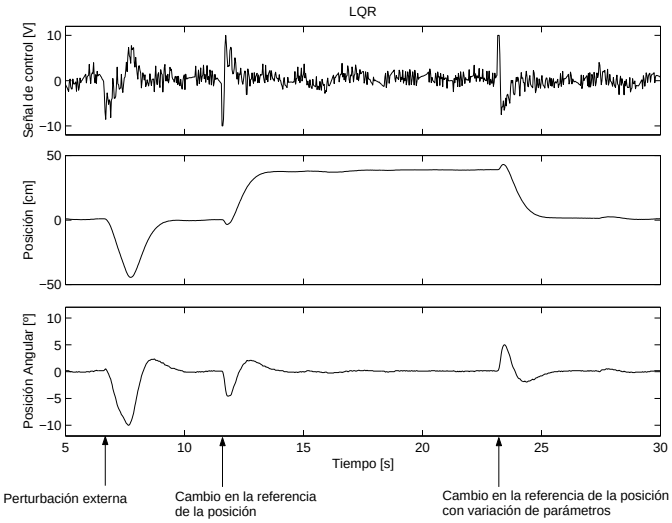

Figura 3.9: Resultados del algoritmo de control usando LQR estimando la fricción

muy similar al mostrado en la Figura 3.9, sólo que presenta oscilaciones tanto en el ángulo como en la posición. De aquí se observa la ventaja de compensar la fricción seca, ya que se eliminan las oscilaciones de manera considerable.

Antes de proseguir con el siguiente capítulo, vale la pena analizar los efectos que tiene la elección de una $\mathbf{Q}$ distinta a la usada en la ecuación 3.4. En la parte superior de la Figura 3.12 se observa la respuesta del sistema cuando

$$\mathbf{Q_1} = \begin{bmatrix} 10 & 0 & 0 & 0 \\ 0 & 10 & 0 & 0 \\ 0 & 0 & 10 & 0 \\ 0 & 0 & 0 & 10 \end{bmatrix} \tag{3.25}$$

Se puede apreciar que la posición r alcanza el estado estacionario en un periodo de tiempo relativamente grande, cerca de 5.5 s. Esto se debe a que los sensores de posición y velocidad son menos sensibles que los de posición angular y velocidad angular. Si se compensa esta diferencia en sensibilidades con una

$$\mathbf{Q_2} = \begin{bmatrix} 60 & 0 & 0 & 0 \\ 0 & 10 & 0 & 0 \\ 0 & 0 & 20 & 0 \\ 0 & 0 & 0 & 10 \end{bmatrix} \tag{3.26}$$

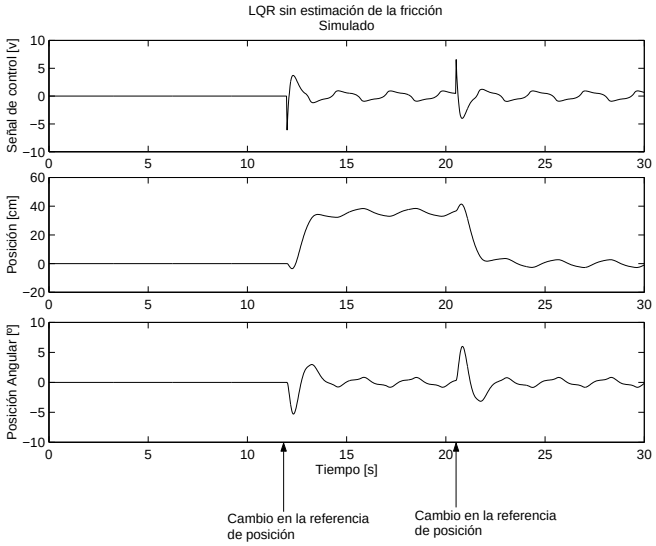

Figura 3.10: Simulación del algoritmo de control usando LQR sin estimar la fricción

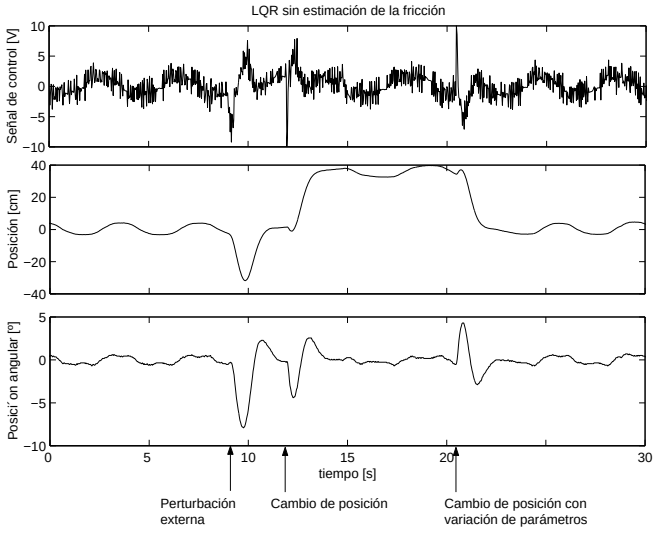

Figura 3.11: Resultados del algoritmo de control usando LQR sin estimar la fricción

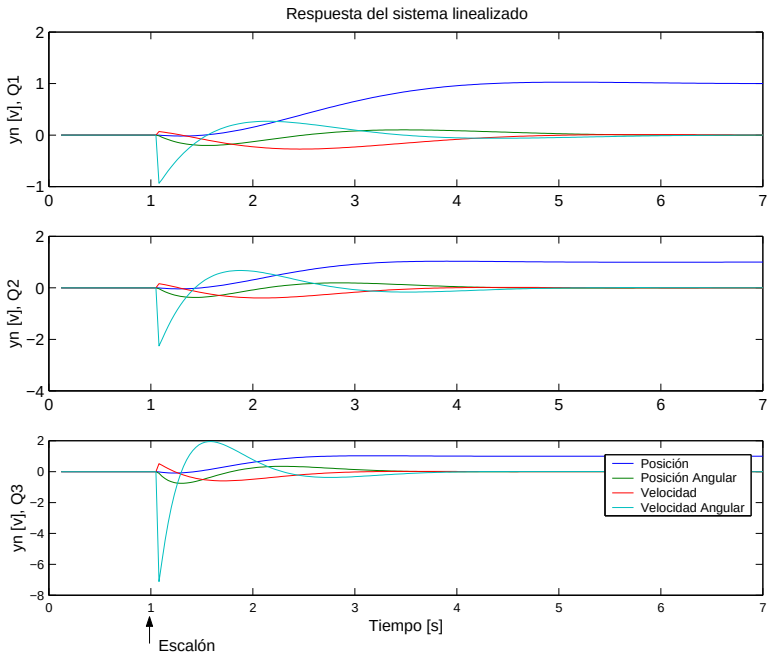

Figura 3.12: Respuestas del sistema para diferentes $\mathbf{Q}$

se obtiene una respuesta como la mostrada en el centro de la Figura 3.12. La respuesta del sistema es más rápida, pero aún puede mejorarse si se le asigna un *peso* relativamente pequeño a la velocidad y a la velocidad angular. Esto se logra, por ejemplo, con una

$$\mathbf{Q_3} = \begin{bmatrix} 650 & 0 & 0 & 0 \\ 0 & 100 & 0 & 0 \\ 0 & 0 & 20 & 0 \\ 0 & 0 & 0 & 10 \end{bmatrix} \tag{3.27}$$

la cual es, precisamente la usada en la ecuación 3.4. La respuesta del sistema es todavía más rápida, como se puede ver en la parte inferior de la Figura 3.12.

Capítulo 4

Levantamiento del Péndulo

En este capítulo se describe el diseño de un algoritmo para levantar el péndulo invertido. Se muestra el levantamiento en simulación, así como los resultados experimentales del proceso de levantamiento.

4.1. Diseño del control

Para levantar el péndulo se propone aplicar una excitación a la frecuencia natural para hacerlo entrar en resonancia. Se propone también utilizar un algoritmo para conmutar entre el controlador que levanta el péndulo y el que lo estabiliza.

4.1.1. Bases teóricas

Cuando la ecuación diferencial que describe la dinámica de un sistema tiene la forma (ver Feynman [7], capítulos 21 y 23)

$$a_1 \frac{d^2x(t)}{dt^2} = -a_0 x(t) \tag{4.1}$$

donde a_0 y a_1 son constantes, se dice que el sistema es un *oscilador armónico*. La solución a la ecuación diferencial es una función cosenoidal

$$x(t) = A\cos(\omega_0 t + \Delta) \tag{4.2}$$

donde A (la amplitud) y Δ (la fase) dependen de las condiciones iniciales. ω_0 es la frecuencia natural y está dada por

$$\omega_0 = \sqrt{\frac{a_0}{a_1}} \tag{4.3}$$

Cuando una excitación externa actúa sobre el sistema se tiene un *oscilador harmónico forzado*. La ecuación diferencial es

$$a_1 \frac{d^2x(t)}{dt^2} + a_0 x(t) = F(t) \tag{4.4}$$

Este tipo de sistema ha sido estudiado con mucho detalle debido a que se presenta en diversos campos de la física: mecánica, acústica, óptica, etc. Cuando la excitación externa es una función cosenoidal de tipo

$$F(t) = F_0 \cos \omega t, \tag{4.5}$$

la solución a la ecuación diferencial tiene la forma

$$x(t) = \frac{F_0}{a_1} \left[\frac{\cos \omega_0 t - \cos \omega t}{(\omega^2 - \omega_0^2)} \right]. \tag{4.6}$$

Si la frecuencia de excitación es igual a la frecuencia natural, la ecuación 4.6 queda indefinida. Es necesario calcular el límite cuando ω tiende a ω_0, es decir

$$\lim_{\omega \to \omega_0} x(t) = \frac{F_0}{a_1} \left[\frac{t \operatorname{sen} \omega_0 t}{2\omega_0} \right]. \tag{4.7}$$

De la ecuación 4.7 se observa que $x(t)$ tiende a infinito conforme pasa el tiempo. En este caso, se dice que el sistema está en resonancia.

En la Figura 4.1 se muestra el esquema de un péndulo fijo si fricción. En ausencia de excitaciones externas las únicas fuerzas son la de gravedad y la reacción en el pivote. La fuerza de gravedad puede separarse en dos componentes: una en la dirección de la barra y otra perpendicular a ésta. La que va en dirección de la barra está compensada por la fuerza de reacción R en el pivote. La fuerza perpendicular es entonces, la resultante. Aplicando la segunda ley de Newton

$$M_1 l_s \frac{d^2\theta(t)}{dt^2} = -M_1 g \operatorname{sen} \theta(t) \tag{4.8}$$

$$l_s \frac{d^2\theta(t)}{dt^2} = -g \operatorname{sen} \theta(t) \tag{4.9}$$

donde M_1 es la masa del péndulo, l_s es la longitud de la barra, g es la aceleración debida a la gravedad y θ es el ángulo con respecto a la vertical.

Si θ es lo suficientemente pequeño, se puede sustituir $\operatorname{sen} \theta$ por θ y obtener un oscilador armónico cuya frecuencia natural es de

$$\omega_0 = \sqrt{\frac{g}{l_s}} \quad \left[\frac{\text{rad}}{\text{s}} \right] \tag{4.10}$$

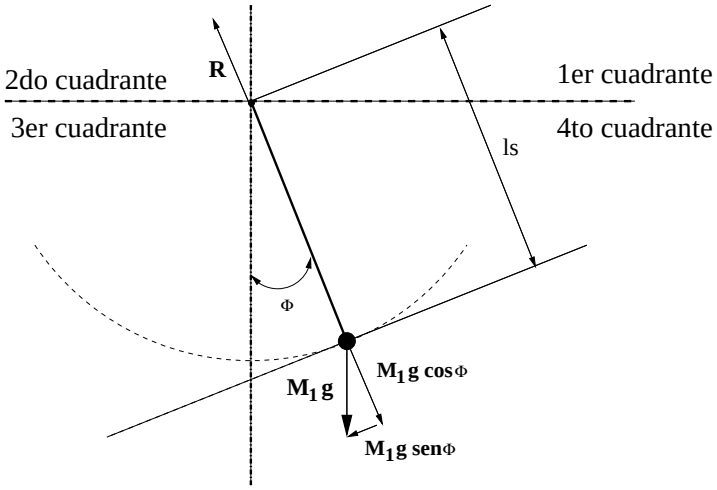

Figura 4.1: Esquema de un péndulo fijo sin fricción

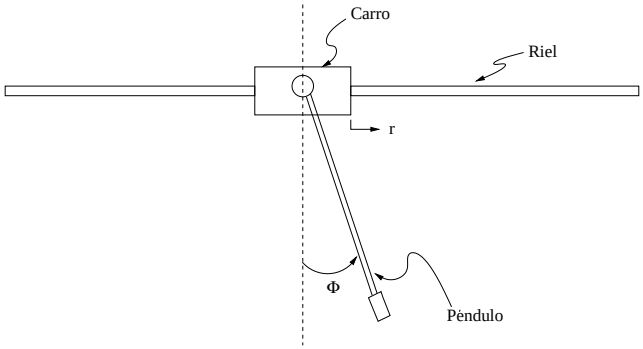

Figura 4.2: Péndulo en la posición inferior

El sistema descrito por 4.9 se comportaría como un oscilador armónico si la excitación se aplicara en el eje. En realidad, la excitación se aplica al carro y en consecuencia se mueve el péndulo. Para analizar este efecto, se puede obtener un modelo descrito en el espacio de estados, linealizado en la posición inferior del péndulo (Figura 4.2). Siguiendo un procedimiento similar al del capítulo 2 pero despreciando la fricción, se llega a

$$\begin{bmatrix} \dot{r} \\ \dot{\Phi} \\ \ddot{r} \\ \ddot{\Phi} \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0.883 & 0 & 0 \\ 0 & -21.53 & 0 & 0 \end{bmatrix} \begin{bmatrix} r \\ \Phi \\ \dot{r} \\ \dot{\Phi} \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0.309 \\ 0.624 \end{bmatrix} F$$

$$\begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} r \\ \Phi \\ \dot{r} \\ \dot{\Phi} \end{bmatrix} \tag{4.11}$$

Los polos del sistema representado por 4.11 son

$$\begin{aligned} s_{1,2} &= 0 \\ s_{3,4} &= 0 \pm 4.6401j. \end{aligned} \tag{4.12}$$

De la ecuación 4.12 se observa que la frecuencia natural del sistema ω_0 es de 4.6401 rad/s. Al aplicar una entrada cosenoidal de esta frecuencia y de amplitud unitaria se obtiene un juego de curvas como el mostrado en la Figura 4.3, de donde se puede apreciar que el principio de resonancia se mantiene.

4.1.2. Excitación

En la sección anterior se calculó la frecuencia de resonancia para el sistema linealizado. La frecuencia de resonancia real seguramente es distinta. Una posibilidad es medir experimentalmente la frecuencia natural del péndulo y aplicar una fuerza que estuviera de acuerdo con la ecuación 4.5. Este enfoque plantea dos problemas principales: uno, que la frecuencia natural del péndulo seguramente varía debido a la fricción seca, dos, aplicar un voltaje al motor de DC en lazo abierto no permite controlar la posición del carro y éste podría rebasar la longitud del riel, chocando contra alguno de los extremos.

Es necesario encontrar la manera de medir dinámicamente la frecuencia natural y mantener la posición del carro dentro de los límites del riel. Una

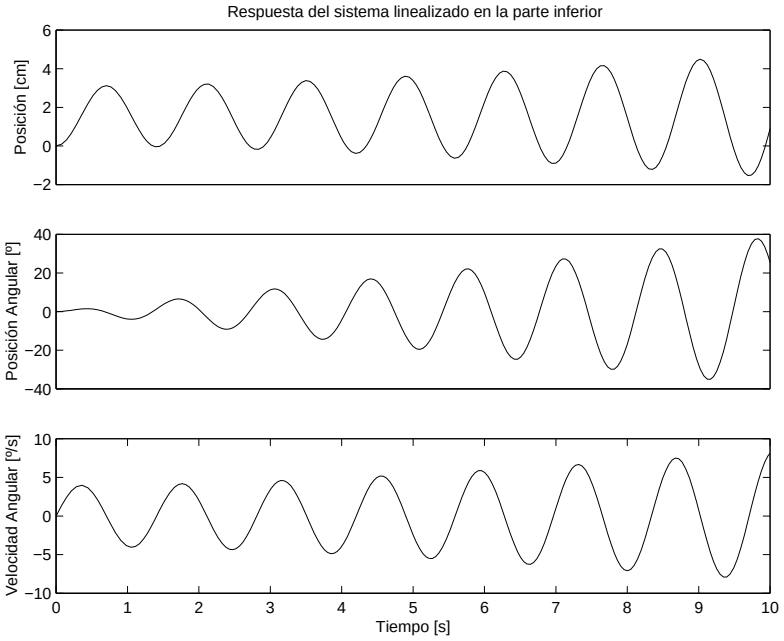

Figura 4.3: Péndulo invertido en resonancia

propuesta es separar el problema en dos partes. La primera consiste en diseñar un algoritmo que controle la posición del carro sin tomar en cuenta el movimiento del péndulo. La segunda parte sería un algoritmo que obtenga la frecuencia del péndulo y a partir de ésta, determine la referencia en la posición que deberá tener el controlador de lazo cerrado.

El controlador de lazo cerrado puede diseñarse utilizando los métodos descritos en los capítulos 2 y 3, pero sin la necesidad de utilizar un observador. Respecto al segundo algoritmo, se puede estimar la velocidad angular del péndulo, y cuando ésta se aproxime a cero, mover el carro de un extremo del riel al otro. Esto es lo que se hace intuitivamente al empujar un columpio, se empuja cuando la velocidad es cero y se ha alcanzado la amplitud máxima para ese ciclo.

Como sólo interesa el instante en el que la velocidad angular es nula, es suficiente si como estimación se realiza la resta del valor actual del ángulo y el valor anterior. Dado que el sistema es discreto, no es posible encontrar el instante preciso en el que esto sucede, es necesario definir un umbral para el cual se pueda considerar que la velocidad es cero. En la Figura 4.4 se muestra el diagrama de flujo.

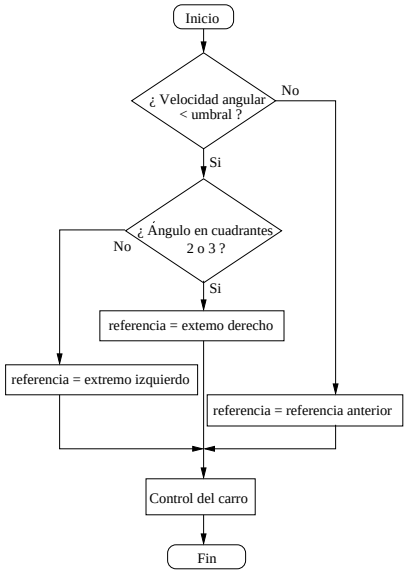

Figura 4.4: Diagrama de flujo para determinar la referencia del controlador del carro

Si se logra que los polos del controlador del carro sean lo suficientemente rápidos, se puede esperar que la excitación se aproxime a una señal cuadrada. Dicha señal cuadrada tendría como frecuencia fundamental, la frecuencia natural del péndulo.

Lo único que falta es un conmutador entre el sistema que levanta el péndulo y el que lo estabiliza. Como para ángulos menores o iguales a $\pm 10^\circ$ se cumple con buena aproximación que $\operatorname{sen}\theta \approx \theta$, se puede utilizar el algoritmo para levantar el péndulo cuando el ángulo se encuentre fuera de ese rango y conmutar al control de estabilización cuando el péndulo entre a ese rango (ver Figura 4.5).

4.1.3. Control del carro

Para controlar el carro se puede utilizar el diagrama de cuerpo libre de la Figura 4.6 y aplicar la segunda ley de Newton

$$F = Fric + M\ddot{r} \tag{4.13}$$

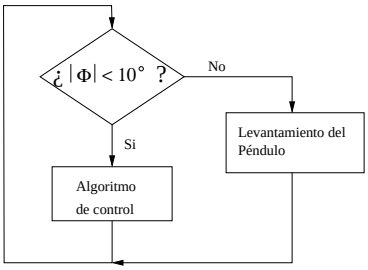

Figura 4.5: Diagrama de flujo para conmutar entre el levantamiento del péndulo y su estabilización

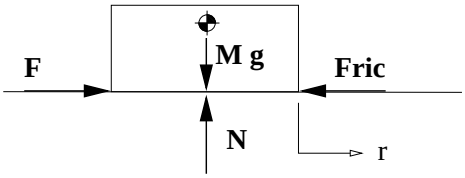

Figura 4.6: Diagrama de cuerpo libre del carro al despreciar el efecto del movimiento del péndulo

donde M es la masa total del carro y el péndulo, r la dirección del movimiento y $Fric$ es la fuerza de fricción dada por

$$Fric = F_r \dot{r} \tag{4.14}$$

por lo que

$$F = F_r \dot{r} + M\ddot{r} \tag{4.15}$$

Siguiendo un procedimiento similar al del capítulo 2, se obtiene la siguiente ecuación de estado

$$\mathbf{x_{nc}}(k+1) = \mathbf{A_{dc}}\mathbf{x_{nc}}(k) + \mathbf{B_{dc}}u(k) \tag{4.16}$$

$$\mathbf{y_{nc}}(k) = \mathbf{C_{dc}}\mathbf{x_{nc}}(k) + \mathbf{D_{dc}}u(k) \tag{4.17}$$

donde x_{nc1} es la posición del carro en volts, x_{nc2} es la velocidad del carro en volts y

$$\mathbf{A_{dc}} = \begin{bmatrix} 1 & -0.05699 \\ 0 & 0.94866 \end{bmatrix} \quad \mathbf{B_{dc}} = \begin{bmatrix} 0.0048543 \\ -0.1644908 \end{bmatrix}$$

$$\mathbf{C_{dc}} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \quad \mathbf{D_{dc}} = \mathbf{0}$$

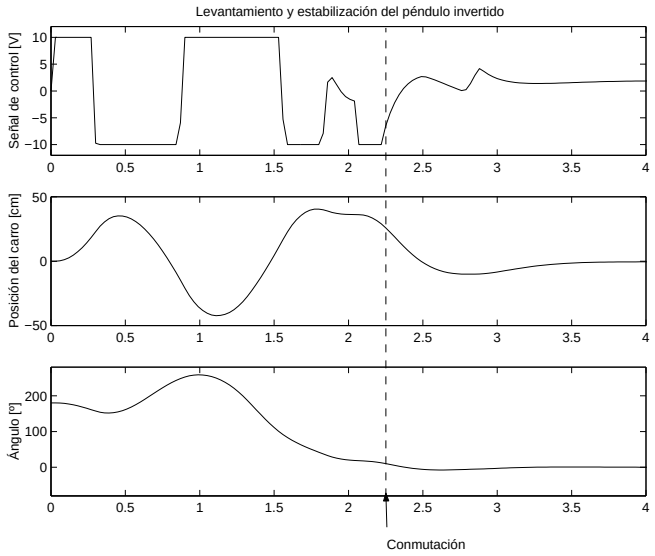

Figura 4.7: Simulación del conmutador y los controles para levantar y estabilizar el péndulo invertido

Para obtener la matriz $\mathbf{K_c}$ que retroalimenta el estado se puede usar la formula de Ackermann. Para un par de polos $z_1 = z_2 = 0.5$, escogidos arbitrariamente, se tiene

$$\mathbf{K_c} = \begin{bmatrix} 25.9767 \\ -5.0006 \end{bmatrix} \tag{4.18}$$

4.2. Implementación del control

Al igual que con el control para estabilizar el péndulo, se llevó a cabo una simulación del sistema. En la simulación se incluyeron: el mismo modelo del péndulo invertido que en el capítulo 3, el algoritmo de control para estabilizar el péndulo, el algoritmo para levantarlo y el conmutador entre los dos algoritmos. La respuesta se muestra en la Figura 4.7

Se llevaron a cabo varias simulaciones para diferentes pares de polos en el controlador del carro. Con la mayoría se obtuvo un sistema capaz de levantar y luego estabilizar al péndulo. La diferencia que se tuvo fue la amplitud del sobretiro en la posición del carro al momento de conmutar los controladores, así que se buscó un par que no excediera los límites de la barra.

Para implementar los algoritmos se utilizó Simulink.

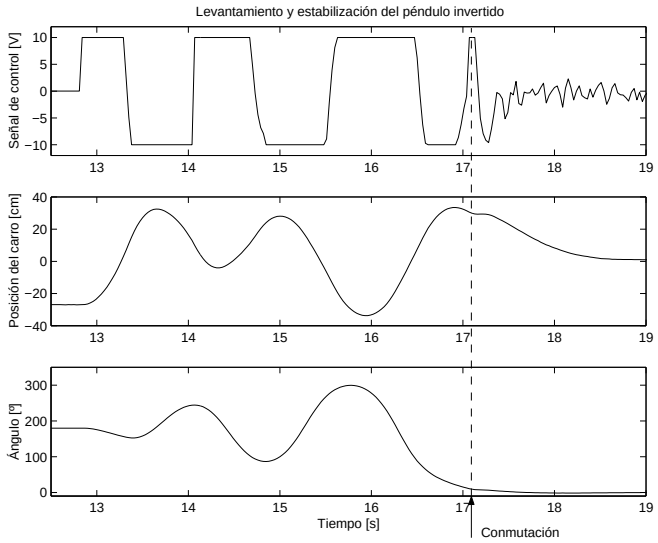

Figura 4.8: Levantamiento y estabilización del péndulo invertido usando LQR para el control del carro

4.3. Análisis de datos y resultados

Es importante mencionar que con el par de polos elegidos en la simulación no fue posible levantar el péndulo en la práctica, y sí fue posible hacerlo con otros pares de polos determinados experimentalmente. Se probó también un control del carro con LQR usando

$$\mathbf{Q_c} = \begin{bmatrix} 300 & 0 \\ 0 & 10 \end{bmatrix} \quad R_c = 1 \tag{4.19}$$

para obtener una

$$\mathbf{K_c} = \begin{bmatrix} 12.0060 \\ -3.3444 \end{bmatrix} \tag{4.20}$$

El resultado fue satisfactorio en la mayoría de pruebas. En la Figura 4.8 se muestra una de ellas.

Åström y Furuta [8] proponen levantar el péndulo retroalimentando la energía del péndulo. La idea, a grandes rasgos, es darle al péndulo una energía igual a la energía potencial que tendría en la posición vertical inestable. A pesar de que ellos usaron un péndulo invertido distinto (uno rotacional), es interesante ver que las curvas que obtuvieron son muy similares a las de la Figura 4.8.

Capítulo 5

Conclusiones

5.1. Sobre la implementación

Como se mencionó en la introducción, uno de los objetivos era desarrollar un plataforma donde fuera posible probar algoritmos de control de manera sencilla. Este objetivo se llevó a cabo satisfactoriamente, ya que la plataforma ha sido utilizada con éxito en la realización de prácticas de control por parte de alumnos de posgrado.

Usando Simulink fue posible realizar un sistema con varias cualidades: tiene una interfaz gráfica, funciona en tiempo real y el esfuerzo para pasar del diseño a la implementación es mínimo.

5.2. Sobre los algoritmos de control

Para diseñar el control que estabiliza al péndulo se usaron métodos relativamente tradicionales (LQR y observador de Luenberger). Como se puede ver en los resultados del capítulo 3, el control diseñado es bueno en el sentido de que es capaz de compensar adecuadamente tanto perturbaciones como variaciones en los parámetros del sistema. La realización de este control cumple un par de propósitos; el primero, aplicar la teoría expuesta tanto en las materias de control como en sus antecedentes, y el segundo, motivar a futuros estudiantes de dichas materias mostrando aplicaciones prácticas.

El levantamiento del péndulo, a diferencia de la estabilización, no tiene muchos antecedentes. Comparado con el de Åström y Furuta tiene la ventaja de que podemos garantizar que el péndulo no se saldrá de los extremos (ellos usaron un péndulo rotacional). La desventaja es que con el algoritmo desarrollado en este trabajo no se puede garantizar analíticamente que el péndulo

llegará a la posición elevada con velocidad angular nula. Este probablemente sea uno de los motivos por los cuales el péndulo no logra levantarse en la totalidad de los intentos.

5.3. Consideraciones finales

El desarrollo de esta tesis se puede separar en dos partes: una en la que se comprueba la teoría expuesta en clases de control y otra innovadora, como es la propuesta para levantar el péndulo.

La plataforma que queda puede usarse para probar algoritmos muy variados, como lógica difusa, redes neuronales, etc. Actualmente las interfaces entre Simulink y las tarjetas de adquisición se encuentran documentadas y disponibles para futuros desarrollos.

Bibliografía

[1] Amira GmbH. *Laboratory Setup Inverted Pendulum.* Alemania 1992

[2] Ferdinand P. Beer y E. Russell Johnson, Jr. *Mecánica vectorial para ingenieros.* Mc. Graw Hill. México 1989

[3] Gene F. Franklin, J. David Powell y Michael Workman. *Digital control of dynamic systems.* Addison-Wesley. E.U.A. 1997

[4] Jerrold Marsden y Anthony Tromba. *Cálculo vectorial.* Addison-Wesley Iberoamericana. Delaware, E.U.A. 1991

[5] Ramos Cruz Julián. *Práctica de control del péndulo invertido. Tesis para obtener el título de Ingeniero Mecánico Electricista.* México, D.F. Marzo de 1995.

[6] Richard P. Feynman, Robert Leighton y Matthew Sands. *The Feynman lectures on physics.* Addison-Wesley. E.U.A. 1969

[7] K. J. Åström y K. Furuta. *Swing up a pendulum by energy control.* IFAC 13th World Congress, San Francisco, California, 1996. www.control.lth.se/~kja/furutapaper.pdf

Apéndice A

Interfaces Matlab - DAC 6214

Para controlar el péndulo desde Simulink fue necesario desarrollar interfaces para poder leer los sensores y mandar la señal de control. Matlab permite escribir rutinas en C, compilarlas y ejecutarlas de forma transparente desde Matlab o Simulink, como si fueran funciones originales de Matlab. A este tipo de funciones se les denomina funciones-S (*S-functions*). Para más detalles se recomienda ver [9]. En este apéndice se muestran los códigos fuente usados en esas funciones.

A.1. Interfaz para el convertidor A/D

Esta es la función-S usada para leer los voltajes y los datos provenientes de los sensores. Matlab incluye una plantilla para escribir este tipo de funciones, sólo es necesario establecer el número de entradas, el número de salidas y asignar valores a dichas salidas. Para obtener los valores de la salidas de esta interfaz, se hacen llamadas a funciones definidas en la sección A.3.

```
/*
 * SamiraADC.c driver para leer datos desde la tarjeta amira utilzando una funcin S
 */

#define S_FUNCTION_NAME  SamiraADC
#define S_FUNCTION_LEVEL 2

#include "simstruc.h"
#include "amiraADC.h"



/*====================================*
 * Configuration and execution methods *
 *====================================*/
```

```
static void mdlInitializeSizes(SimStruct *S)
{
    ssSetNumSFcnParams(S, 0);  // Number of expected parameters
    if (ssGetNumSFcnParams(S) != ssGetSFcnParamsCount(S)) {
        return;
    }

    ssSetNumContStates(S, 0);   // number of continuous states
    ssSetNumDiscStates(S, 0);   // number of discrete states

    if (!ssSetNumInputPorts(S, 1)) return;
    ssSetInputPortWidth(S, 0, 1);
    ssSetInputPortDirectFeedThrough(S, 0, 0);

    if (!ssSetNumOutputPorts(S, 1)) return;
    ssSetOutputPortWidth(S, 0, 3);

    ssSetNumSampleTimes(S, 1);   // number of sample times

    ssSetNumRWork(         S, 0);   // number of real work vector elements
    ssSetNumIWork(         S, 0);   // number of integer work vector elements
    ssSetNumPWork(         S, 0);   // number of pointer work vector elements
    ssSetNumModes(         S, 0);   // number of mode work vector elements
    ssSetNumNonsampledZCs( S, 0);   // number of nonsampled zero crossings


    ssSetOptions(S, SS_OPTION_EXCEPTION_FREE_CODE);
    // general options (SS_OPTION_xx)

} //* end mdlInitializeSizes



static void mdlInitializeSampleTimes(SimStruct *S)
{
    // Register one pair for each sample time
    ssSetSampleTime(S, 0, INHERITED_SAMPLE_TIME);
    ssSetOffsetTime(S, 0, 0.0);

} // end mdlInitializeSampleTimes



/* Function: mdlOutputs =======================================================
 * Abstract:
 *    In this function, you compute the outputs of your S-function
 *    block. Generally outputs are placed in the output vector(s),
 *    ssGetOutputPortSignal.
 */
static void mdlOutputs(SimStruct *S, int_T tid)
{
    InputRealPtrsType uPtrs = ssGetInputPortRealSignalPtrs(S,0);
    real_T         *posicion    = ssGetOutputPortRealSignal(S,0);
    real_T         *angulo;
    real_T         *velocidad;
    int_T          error = 0;

 float Fposicion = 0;
 float Fangulo = 0;
 float Fvelocidad = 0;

 angulo = posicion + 1;
    velocidad = angulo + 1;

    amiraADC(0,&error,&Fposicion);
    amiraADC(0,&error,&Fposicion);
 amiraENC(&Fangulo);
    amiraADC(2,&error,&Fvelocidad);
    amiraADC(2,&error,&Fvelocidad);

 *posicion = Fposicion;
 *angulo = Fangulo;
 *velocidad = Fvelocidad;

    if (error == 1)
    ssSetErrorStatus(S,"Tiempo de conversin agotado. No se encuentra
```

```
    tarjeta o probablemente se encuentra en otra direccin de memoria.");
    return;

} // end mdlOutputs


static void mdlTerminate(SimStruct *S)
{
}


/*=============================*
 * Required S-function trailer *
 *=============================*/

#ifdef  MATLAB_MEX_FILE    /* Is this file being compiled as a MEX-file? */
#include "simulink.c"      /* MEX-file interface mechanism */
#else
#include "cg_sfun.h"       /* Code generation registration function */
#endif
```

A.2. Interfaz para el convertidor D/A

Esta interfaz se usa para enviar la señal de control al motor del carro. La entrada de esta función se manda al convertidor D/A por medio de otras funciones definidas en la la sección A.3.

```
/*
 * SamiraDAC.c driver para escribir datos y generar seal de watchdog
 * a la tarjeta amira utilzando una funcin S
 */

#define S_FUNCTION_NAME  SamiraDAC
#define S_FUNCTION_LEVEL 2

#include "simstruc.h"
#include "amiraADC.h"



/*====================================*
 * Configuration and execution methods *
 *====================================*/

static void mdlInitializeSizes(SimStruct *S)
{
    ssSetNumSFcnParams(S, 0);  // Number of expected parameters
    if (ssGetNumSFcnParams(S) != ssGetSFcnParamsCount(S)) {
        return;
    }

    ssSetNumContStates(S, 0);   // number of continuous states
    ssSetNumDiscStates(S, 0);   // number of discrete states

    if (!ssSetNumInputPorts(S, 1)) return;
    ssSetInputPortWidth(S, 0, 2);
    ssSetInputPortDirectFeedThrough(S, 0, 0);

    if (!ssSetNumOutputPorts(S, 1)) return;
    ssSetOutputPortWidth(S, 0, 1);

    ssSetNumSampleTimes(S, 1);   // number of sample times

    ssSetNumRWork(         S, 0);   // number of real work vector elements
    ssSetNumIWork(         S, 0);   // number of integer work vector elements
    ssSetNumPWork(         S, 0);   // number of pointer work vector elements
    ssSetNumModes(         S, 0);   // number of mode work vector elements
    ssSetNumNonsampledZCs( S, 0);   // number of nonsampled zero crossings
```

```
    ssSetOptions(S, SS_OPTION_EXCEPTION_FREE_CODE);
    // general options (SS_OPTION_xx)

} //* end mdlInitializeSizes



static void mdlInitializeSampleTimes(SimStruct *S)
{
    // Register one pair for each sample time
    ssSetSampleTime(S, 0, INHERITED_SAMPLE_TIME);
    ssSetOffsetTime(S, 0, 0.0);

} // end mdlInitializeSampleTimes



/* Function: mdlOutputs =======================================================
 * Abstract:
 *    In this function, you compute the outputs of your S-function
 *    block. Generally outputs are placed in the output vector(s),
 *    ssGetOutputPortSignal.
 */
static void mdlOutputs(SimStruct *S, int_T tid)
{
    InputRealPtrsType uPtrs = ssGetInputPortRealSignalPtrs(S,0);  //Obtenemos entradas
    real_T         *control    = ssGetOutputPortRealSignal(S,0); //Definimos salidas
    int_T          error = 0;

 float Fcontrol;
 unsigned char Udigital;

    Fcontrol = *uPtrs[0];
    amiraDAC(&Fcontrol);  //Escribimos salida analgica
 *control = Fcontrol;

 Udigital = *uPtrs[1]; //Escribimos salida digital
 amiraDIG(&Udigital);

if (Udigital == 1)
    amiraRST();
    if (error == 1)
    ssSetErrorStatus(S,"Tiempo de conversin agotado. No se encuentra tarjeta o probablemente
    se encuentra en otra direccin de memoria.");
    return;


} // end mdlOutputs


static void mdlTerminate(SimStruct *S)
{
}


/*============================*
 * Required S-function trailer *
 *============================*/

#ifdef  MATLAB_MEX_FILE    /* Is this file being compiled as a MEX-file? */
#include "simulink.c"      /* MEX-file interface mechanism */
#else
#include "cg_sfun.h"       /* Code generation registration function */
#endif
```

A.3. Rutinas utilizadas por las interfaces

Este es el encabezado que se incluye en las interfaces descritas en las secciones anteriores. Contiene las funciones usadas por las dos rutinas. Estas

funciones son las que se comunican directamente con la tarjeta DAC 6214. Para más información se recomienda ver la documentación incluida con el péndulo [1]

```
/*********************************************************************
*     amiraADC.h   *
*                                                                    *
*     Rutinas de lectura y escritura de las tarjetas amira
*                    para el pndulo invertido           *
*                                                                     *
*                                                                     *
**********************************************************************/

#include <conio.h>
#define TIEMPO_LIMITE 100000 //Tiempo lmite para la conversin
#define HWADR 0x210
#define DATR 0x213
#define PI 3.141593


void amiraADC(unsigned char canal, int *error, float *volts);
void amiraENC(float *volts);
void amiraRST();
void amiraDAC(float *volts);
void amiraDIG(unsigned char *digital);

//Implementacin de la conversin A/D
void amiraADC(unsigned char canal, int *error, float *volts)
{
int DIR1 = HWADR; //Direccin de la tarjeta
int DIR2 = DATR; //Direccin del registro de datos
int LB_DATO=0; //Byte bajo del dato a leer D0->D7
   int HB_DATO=0; //Byte bajo del dato a leer D8->D12
   unsigned char LB_DIR = 0x00; //Direccin donde se almacena el byte bajo
   unsigned char HB_DIR = 0x01; //Direccin donde se almacena el byte alto
unsigned char SEL_CANL = 0xA0; //Comando para seleccionar canal
unsigned char ESTADO = 0x60; //Comando para leer el estado de la tarjeta
   int mask = 0x000F;
int mask2 = 0x00FF;

   long double i=0;
int busy;

int DATO_BIN; //Dato ledo en forma binaria

canal <<= 4;   //Desplazamos el nmero de canal
    //   4 bits a la izquierda

outp(DIR1,SEL_CANL);   //Seleccionamos
   outp(DIR2,canal); //   canal
   outp(DIR1,0);

LB_DATO = inp(DIR1); //Comienza conversin

do {
i++;
if(i>TIEMPO_LIMITE) {
*error = 1;
return;
}
outp(DIR1,ESTADO);
busy = inp(DIR2) & 0x0080; //Preguntamos si ha terminado la conversin
} while (busy == 0); //Termina la conversin

outp(DIR1,HB_DIR); //Pedimos leer el byte alto (D8 - D11)
HB_DATO = inp(DIR2);          //Leemos byte alto
outp(DIR1,LB_DIR); //Pedimos leer el byte bajo (D0 - D7)
LB_DATO = inp(DIR2); //Leemos byte bajo
LB_DATO &= mask2; //Eliminamos ruido

   HB_DATO &= mask; //Tomamos los primeros cuatro bits
HB_DATO <<= 8; //Recorremos 8 bits
//(para que sea D8 - D11 en vez de D0 - D3)
```

```
DATO_BIN = LB_DATO + HB_DATO;  //Formamos la palabra completa de 12 bits

*volts = 10 - (DATO_BIN * 20.0)/4096.0; //Convertimos a volts
*error = 0;
}

//Implementacin la lectura del encoder en forma binaria
void amiraENC(float *volts)
{
int DIR1 = HWADR; //Direccin de la tarjeta
int DIR2 = DATR; //Direccin del registro de datos
int LB_DATO=0; //Byte bajo del dato a leer D0->D7
   int HB_DATO=0; //Byte bajo del dato a leer D8->D12
   unsigned char LB_DIR = 0x89; //Direccin donde se almacena el byte bajo
   unsigned char HB_DIR = 0x88; //Direccin donde se almacena el byte alto
   int mask = 0x000F;
int mask2 = 0x00FF;

int DATO_BIN = 0; //Dato binario completo

outp(DIR1,HB_DIR); //Pedimos leer el byte alto (D8 - D11)
HB_DATO = inp(DIR2);                //Leemos byte alto
    HB_DATO &= mask; //Tomamos los primeros cuatro bits
HB_DATO <<= 8; //Recorremos 8 bits
//(para que sea D8 - D11 en vez de D0 - D3)

outp(DIR1,LB_DIR); //Pedimos leer el byte bajo (D0 - D7)
LB_DATO = inp(DIR2); //Leemos byte bajo
LB_DATO &= mask2; //Eliminamos ruido

DATO_BIN = LB_DATO + HB_DATO;  //Formamos la palabra completa de 12 bits
    //*volts = LB_DATO;
*volts =  ( (2047.0 - DATO_BIN)*PI*52.27 ) / 2048.0  ; //Convertimos a volts



}

//Implementacin del reset del encoder
void amiraRST()
{
int DIR1 = HWADR; //Direccin de la tarjeta
int DIR2 = DATR; //Direccin del registro de datos
int pato = 0;

   unsigned char RESET_CONT = 0x80; //Comando de reset del contador

outp(DIR1,RESET_CONT); //Mandamos seal de reset.
pato = inp(DIR2);


}

//Implementacin de la conversin D/A
void amiraDAC(float *volts)
{
int DIR1 = HWADR; //Direccin de la tarjeta
int DIR2 = DATR; //Direccin del registro de datos
int LBLN_DATO = 0; //Nibble bajo del byte bajo a escribir
   int LBHN_DATO = 0; //Nibble alto del byte bajo a escribir
   int HBLN_DATO = 0; //Nibble bajo del byte alto a escribir

int DATO_BIN = 0; //Dato binario completo;
int escribe = 0;

   unsigned char LBLN_DIR = 0x40; //Direccin del LBLN
   unsigned char LBHN_DIR = 0x41; //Direccin del LBHN
   unsigned char HBLN_DIR = 0x42; //Direccin del HBLN
   unsigned char ESCR_DIR = 0x43; //Comando para iniciar conversin

if (*volts > 10)
*volts = 10;
if (*volts < -10)
*volts = -10;
DATO_BIN = ((*volts + 10)*4095)/20; //Convertimos de volts a binario
```

```
LBLN_DATO = DATO_BIN & 0x000F;
LBHN_DATO = (DATO_BIN & 0x00F0) >> 4;
HBLN_DATO = (DATO_BIN & 0x0F00) >> 8;

outp(DIR1,LBLN_DIR); //Mandamos primer nibble
outp(DIR2,LBLN_DATO);
outp(DIR1,LBHN_DIR); //Mandamos segundo nibble
outp(DIR2,LBHN_DATO);
outp(DIR1,HBLN_DIR); //Mandamos tercer nibble
outp(DIR2,HBLN_DATO);

outp(DIR1,ESCR_DIR); //Iniciamos conversin y
outp(DIR2,escribe); //escribimos.

return;
}

void amiraDIG(unsigned char *digital)
{
int DIR1 = HWADR; //Direccin de la tarjeta
int DIR2 = DATR; //Direccin del registro de datos

unsigned char SAL_DIG = 0xA0; //Comando para salida digital

*digital &= 0x0F;

outp(DIR1,SAL_DIG);   //Seleccionamos salida digital
outp(DIR2,*digital);   //Mandamos dato

return;
}
```