Trabajo Fin de Grado
Grado en Ingeniería de las Tecnologías Industriales

Control y Simulación del Péndulo de Furuta

Autor: Carlos Regalo Núñez
Tutor: David Muñoz De La Peña Sequedo

Dep. Ingeniería de Sistemas y Automática
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla

Sevilla, 2016

Trabajo Fin de Grado
Grado en Ingeniería de las Tecnologías Industriales

Control y Simulación del Péndulo de Furuta

Autor:
Carlos Regalo Núñez

Tutor:
David Muñoz De La Peña Sequedo
Profesor titular

Dep. de Ingeniería de Sistemas y Automática
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla
Sevilla, 2016

Trabajo Fin de Grado: Control y Simulación del Péndulo de Furuta

Autor: Carlos Regalo Núñez

Tutor: David Muñoz De La Peña Sequedo

El tribunal nombrado para juzgar el Trabajo arriba indicado, compuesto por los siguientes miembros:

Presidente:

Vocales:

Secretario:

Acuerdan otorgarle la calificación de:

Sevilla, 2016

El Secretario del Tribunal

A mi familia

A mis maestros

A mis amigos

Agradecimientos

La realización del trabajo final de carrera es algo muy importante para todos los alumnos, ya que es la primera vez en la que nos encontramos ante un trabajo completamente individual y al que tenemos que enfrentarnos prácticamente solos, usando los conocimientos que hemos ido adquiriendo a lo largo de tantos años en nuestra escuela de ingenieros.

Aunque, tal y como he dicho, tenemos que enfrentarnos solos, sin la ayuda, las indicaciones y las ideas de David, tutor de este trabajo, nunca hubiera sido posible realizar dicha labor. Por otro lado el apoyo de colegas y familiares también se hace imprescindible para continuar con la ardua tarea en los momentos de dificultad.

Por todo ello escribo estas líneas para mostrarles toda mi gratitud.

Carlos Regalo Núñez

Sevilla, 2016

Resumen

El presente trabajo consiste en la simulación y estudio del control de un péndulo invertido rotatorio, conocido como péndulo de Furuta, un sistema que contiene una dinámica muy rápida cuyo control es muy interesante dentro del campo de la automática.

En la primera parte del proyecto, se realiza un modelo del péndulo en *Simulink* previa obtención de las ecuaciones de movimiento, se explicará cómo hacer un modelo en 3D gracias a la herramienta *V Realm* de *Matlab* para visualizar mejor el péndulo en las simulaciones y se estudian diversas formas de controlarlo ya estemos cerca o lejos del punto de operación.

En la segunda parte trabajamos sobre un péndulo real, *Qube-Servo*, cuyos parámetros identificamos y posteriormente controlamos.

El objetivo del documento es realizar un estudio amplio de las técnicas de control que se pueden aplicar a este sistema altamente inestable.

Abstract

This work involves the study of simulation and control of a rotary inverted pendulum known as Furuta pendulum, a system containing a very fast dynamic whose control is very interesting in the field of automatic control.

In the first part of the project, a model of the pendulum in *Simulink* is made after obtaining the equations of motion, it will be explained how to make a 3D model thanks to the *V Realm* tool *Matlab* in order to display better the pendulum during simulations and it will be studied various ways of control either we are close or far from the point of operation.

In the second part we work on a real pendulum, *Qube-Servo*, whose parameters are identified and subsequently controlled.

The aim of the paper is a comprehensive study of control techniques that can be applied to this highly unstable system.

Índice

Índice de Tablas

Índice de Figuras

Notación

A^*	Conjugado
c.t.p.	En casi todos los puntos
c.q.d.	Como queríamos demostrar
■	Como queríamos demostrar
e.o.c.	En cualquier otro caso
e	número e
IRe	Parte real
IIm	Parte imaginaria
sen	Función seno
tg	Función tangente
arctg	Función arco tangente
sen	Función seno
$\sin^x y$	Función seno de x elevado a y
$\cos^x y$	Función coseno de x elevado a y
Sa	Función sampling
sgn	Función signo
rect	Función rectángulo
Sinc	Función sinc
∂y ∂x	Derivada parcial de y respecto
$x_\circ$	Notación de grado, x grados.
Pr(A)	Probabilidad del suceso A
SNR	Signal-to-noise ratio
MSE	Minimum square error
:	Tal que
<	Menor o igual
>	Mayor o igual
\	Backslash
⇔	Si y sólo si

1 INTRODUCCIÓN

La mente que se abre a una nueva idea nunca vuelve a su tamaño original.

- Albert Einstein -

El avance de la tecnología se muestra imparable ante nuestro ojos en los últimos tiempos, podemos ver impresoras que crean objetos tridimensionales a la vez que podemos ver robots humanoides cada vez más perfeccionados o los famosos drones constantemente en los telediarios, entre otras muchas creaciones fascinantes que eran impensables años atrás. Dentro de toda esta tecnología, se encuentra una disciplina de la ingeniería que ha tenido mucho que ver en la realización de estas técnicas, se trata de la ingeniería de control.

Es a través de la ingeniería de control que somos capaces de dar el par adecuado en las articulaciones de los robots para que estos realicen los movimientos que esperamos, es a través de esta disciplina relativamente joven que controlamos el empuje de las aspas de los drones para obtener la posición adecuada o a gracias a ella somos capaces de controlar la temperatura del material en una impresora 3D.

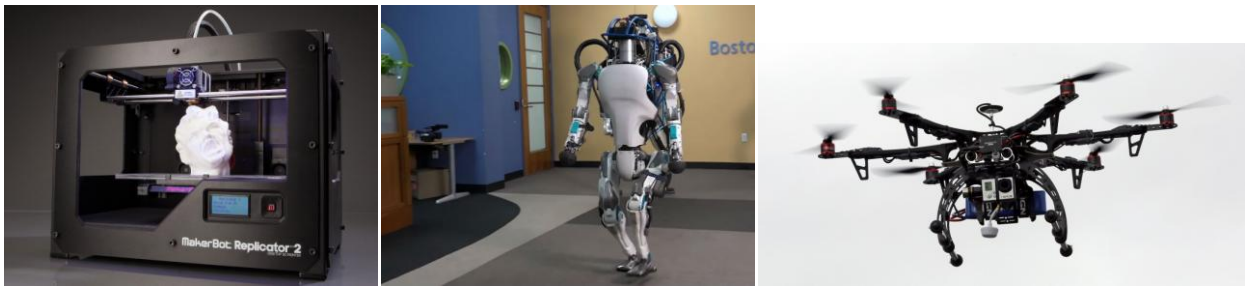

Figura 1-1. Aplicaciones de la ingeniería de control.

Además de las aplicaciones anteriormente mencionadas, el control automático se encuentra presente en elementos aparentemente menos sofisticados, quizás porque estamos más acostumbrados a ellos, como los climatizadores de los vehículos o industrias como la química o petroquímica.

Figura 1-2. La ingeniería de control en industrias tradicionales.

En este trabajo se va a controlar un sistema complejo, pues es altamente inestable y además presenta una dinámica muy rápida, se trata del péndulo de Furuta, un péndulo invertido rotatorio. Dicho péndulo consiste en un péndulo conectado a un brazo que gira sobre una base.

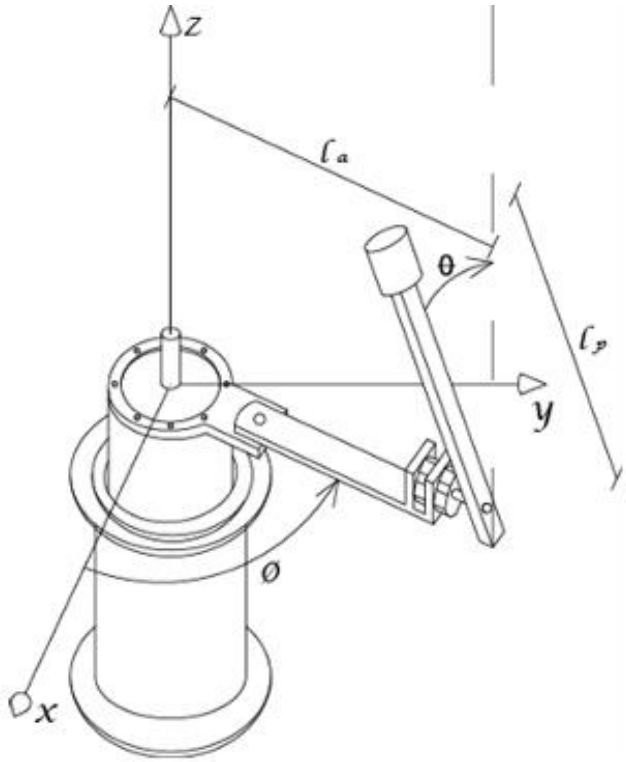

Figura 1-3. Péndulo de Furuta.

En el sistema, previa obtención de las ecuaciones de movimiento, se intentará controlar la posición del péndulo para que este se mantenga vertical hacia arriba, también se controlará la posición del brazo para que esté en un ángulo concreto, como podemos ver, controlaremos dos variables, los ángulos, a partir de una sola, la tensión que aplicamos al motor, esto requiere técnicas de control algo más avanzadas que se aplicarán en este trabajo. Por otro lado, se puede intuir que llevar el péndulo desde la posición inferior a la superior no se puede hacer con un simple controlador PID que tenga como referencia la posición superior del péndulo, ya que si este empieza a girar en un sentido determinado, podríamos llegar como mucho a que el péndulo se mantuviera en posición horizontal en régimen permanente, ya que no se cambiaría el sentido de giro del brazo por que esto probablemente aumentaría el error de posición. Como conseguir llevar el péndulo arriba será otro de los temas que se tratarán en este texto.

Además de la obtención de las ecuaciones de movimiento del péndulo y el posterior control y simulación del modelo a través de la herramienta *Matlab*, también se identifica y controla un péndulo de Furuta real con el que contamos en la escuela de ingenieros, se trata de un péndulo educacional de la compañía *QUANSER* para el aprendizaje del control en las aulas de ingeniería. Se comparan ambos péndulos y se realizan mediciones. La identificación de los parámetros del modelo es clave para el posterior control del mismo, cómo hacerlo es algo que se explicará detalladamente en este documento.

Figura 1-4. Péndulo de *QUBE-Servo*.

El modelado, control y simulación del péndulo de Furuta permite conocer otras técnicas de control y avanzar en el control automático al ser diferente de los típicos sistemas a controlar convencionales.

El control aplicado a sistemas de péndulos invertidos tiene como aplicación directa más notoria los famosos vehículos *segway* que se pueden ver circulando por nuestras ciudades. Aún así, este trabajo no busca una aplicación directa, sino ver y estudiar otras formas de control que permitan aumentar el conocimiento en la disciplina y quizás servir de orientación para otros sistemas de control similares.

Figura 1-5. Paseo en *segway*.

2 DISEÑO, SIMULACIÓN Y CONTROL DEL PÉNDULO DE FURUTA

Si no puedes volar entonces corre, si no puedes correr entonces camina, si no puedes caminar entonces arrástrate, pero sea lo que hagas, sigue moviéndote hacia delante.

- Martin Luther King Jr -

En este apartado del documento se obtienen las ecuaciones de movimiento del péndulo de Furuta, se explica cómo realizar el modelo en *Simulink* a patir de las dichas ecuaciones y también se comentará como usar la herramienta *V Realm* de *Matlab* para crear un modelo en 3D del péndulo para que se pueda visualizar mejor al mismo durante las simulaciones.

Se explica cómo aplicar varias técnicas de control, como el balance de control PD o el control LQR, cuando estamos cerca del punto de funcionamiento, también veremos como linealizar las ecuaciones del péndulo para poder usar el control LQR. Finalmente se verá en qué consiste la técnica *swing-up* que permite llevar el péndulo a una posición cercana al punto de funcionamiento.

En este apartado se abarca completamente la primera parte del trabajo, es decir, la simulación y control del péndulo.

2.1. Modelado no lineal del péndulo de Furuta

En este subapartado veremos cómo obtener las ecuaciones de movimiento, como realizar el modelo en *Simulink* y como crear un modelo 3D con *V Realm*.

2.1.1 Obtención de las ecuaciones de movimiento del péndulo de Furuta

Considerando el péndulo de Furuta, y llamando φ al ángulo que existe entre el brazo del motor, de longitud r y el eje x y llamando θ al ángulo que forma la barra del péndulo, de longitud $2l$ y masa m, con dirección paralela al eje z, ambos son definidos en sentido horario.

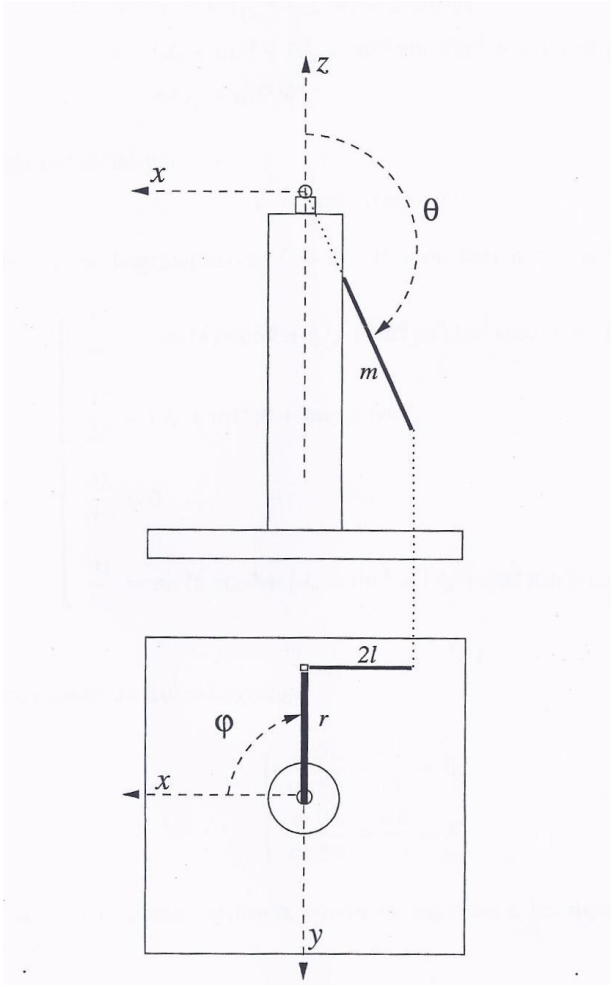

Figura 2-1. Sistemas de coordenadas definido.

Las coordenadas del centro de masa son:

$$x_G = r\cos\varphi - l\sin\theta\, sin\varphi\,,$$
$$y_G = r\sin\varphi + l\sin\theta\cos\varphi\,,$$
$$z_G = l\cos\theta$$

Calculando las derivadas obtenemos que

$$\dot{x}_G = -r\,\dot{\varphi}\sin\varphi - l\,\dot{\varphi}\cos\varphi\sin\theta - l\,\dot{\theta}\sin\varphi\cos\theta\,,$$
$$\dot{y}_G = r\,\dot{\varphi}\cos\varphi - l\,\dot{\varphi}\sin\varphi\sin\theta + l\,\dot{\theta}\cos\varphi\cos\theta\,,$$
$$\dot{z}_G = -l\,\dot{\theta}\sin\theta$$

Por lo tanto, la velocidad del centro de masa es,

$$v^2 = r^2\dot{\varphi}^2 + l^2\dot{\varphi}^2(\sin\theta)^2 + 2rl\dot{\varphi}\dot{\theta}\cos\theta + l^2\dot{\theta}^2$$

Considerando que la suma de los momentos de inercia del brazo y del motor es $J_a = J_{motor} + J_{brazo}$ y el momento de inercia del péndulo es $J_p = \frac{4}{3}ml^2$, la energía cinética del péndulo será

$$2T = mv^2 + J_a\dot{\varphi}^2 + J_p\left(\dot{\theta}^2 + \dot{\varphi}^2(\sin\theta)^2\right)$$
$$= \left(J_a + mr^2 + \left(J_p + ml^2\right)(\sin\theta)^2\right)\dot{\varphi}^2 + 2mrl\dot{\varphi}\dot{\theta}\cos\theta + (J_p + ml^2)\dot{\theta}^2$$

Y la energía potencial es

$$V = mgl(cos\,\theta - 1)$$

El Lagrangiano es $L = T - V$, realizando las ecuaciones de Euler-Lagrange

$$\begin{cases} \dfrac{d}{dt}\dfrac{\partial L}{\partial \dot{\theta}} - \dfrac{\partial L}{\partial \theta} = 0 \\ \dfrac{d}{dt}\dfrac{\partial L}{\partial \dot{\varphi}} - \dfrac{\partial L}{\partial \varphi} = F \end{cases}$$

Y considerando que F es el par externo aplicado, llegamos a las siguientes ecuaciones del movimiento

$$\begin{cases} (J_p + ml^2)(\ddot{\theta} - \dot{\varphi}^2 \sin\theta\, cos\,\theta) + mrl\ddot{\varphi}\cos\theta - mgl\sin\theta = 0 \\ mrl\ddot{\theta}\cos\theta - mrl\dot{\theta}^2\sin\theta + 2(J_p + ml^2)\dot{\theta}\dot{\varphi}\sin\theta\cos\theta + (J_a + mr^2 + (J_p + ml^2)sin^2\theta)\ddot{\varphi} = F \end{cases}$$

Normalizamos estas ecuaciones, es decir, hacemos que el coeficiente de la derivada mayor sea uno definiendo

$$\omega_0 = \sqrt{\frac{mgl}{J_p + ml^2}} = \sqrt{\frac{3g}{7l}}$$
$$\alpha = \frac{mrl}{J_p + ml^2} = \frac{3r}{7l}$$
$$\beta = \frac{J_a + mr^2}{J_p + ml^2}$$
$$\tau = \omega_0 t$$

Las ecuaciones de movimiento pueden entonces ser escritas como

$$\begin{cases} \ddot{\theta} - \dot{\varphi}^2 \sin\theta\cos\theta + \alpha\ddot{\varphi}\cos\theta - \sin\theta = 0 \\ \alpha\ddot{\theta}\cos\theta - \alpha\dot{\theta}^2\sin\theta + 2\dot{\theta}\dot{\varphi}\sin\theta\cos\theta + (\beta + sin^2\theta)\ddot{\varphi} = \gamma u \end{cases}$$

Donde $\gamma = \frac{k_m}{mgl}$, donde k_m es la ganancia equivalente de los subsistemas motor y servo amplificador de potencia y u la acción de control. Estas ecuaciones pueden ser reescritas en forma matricial como

$$\begin{bmatrix} 1 & \alpha\cos\theta \\ \alpha\cos\theta & \beta + sin^2\theta \end{bmatrix}\begin{bmatrix} \ddot{\theta} \\ \ddot{\varphi} \end{bmatrix} + \begin{bmatrix} 0 & -\dot{\varphi}\sin\theta\cos\theta \\ -\alpha\dot{\theta}\sin\theta + \dot{\varphi}\sin\theta\cos\theta & \dot{\theta}\sin\theta\cos\theta \end{bmatrix}\begin{bmatrix} \dot{\theta} \\ \dot{\varphi} \end{bmatrix} + \begin{bmatrix} -\sin\theta \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ \gamma u \end{bmatrix}$$

Premultiplicando la ecuación anterior por

$$\begin{bmatrix} 1 & \alpha\cos\theta \\ \alpha\cos\theta & \beta + sin^2\theta \end{bmatrix}^{-1} = \frac{1}{\Delta}\begin{bmatrix} \beta + sin^2\theta & -\alpha\cos\theta \\ -\alpha\cos\theta & 1 \end{bmatrix}$$

Donde $\Delta = \beta + sin^2\theta - \alpha^2 cos^2\theta$, se obtiene

$$\begin{cases} \Delta\ddot{\theta} = -\alpha^2\dot{\theta}^2\sin\theta\cos\theta + (\beta + sin^2\theta)\dot{\varphi}^2\sin\theta\cos\theta + 2\alpha\dot{\theta}\dot{\varphi}\sin\theta\, cos^2\theta + \beta\sin\theta + sin^3\theta - \gamma u\alpha\cos\theta \\ \Delta\ddot{\varphi} = \alpha\dot{\theta}^2 - \alpha\dot{\varphi}^2\sin\theta\, cos^2\theta - 2\dot{\theta}\dot{\varphi}\sin\theta\cos\theta - \alpha\sin\theta\cos\theta + \gamma u \end{cases}$$

Este sistema de cuarto orden es el modelo del péndulo sin considerar los términos disipativos, se trata de un sistema de cuarto orden $x_1 = \theta, x_2 = \dot{\theta}, x_3 = \varphi\ y\ x_4 = \dot{\varphi}$. Si se consideran los efectos disipativos obtenemos lo siguiente

$$\begin{cases} \Delta\ddot{\theta} = -\alpha^2\dot{\theta}^2 \sin\theta\cos\theta + (\beta + sin^2\theta)\dot{\varphi}^2 \sin\theta\cos\theta + 2\alpha\dot{\theta}\dot{\varphi}\sin\theta\, cos^2\theta + \beta\sin\theta + sin^3\theta - \gamma u\alpha\cos\theta - c_p\dot{\theta} \\ \Delta\ddot{\varphi} = \alpha\dot{\theta}^2\sin\theta - \alpha\dot{\varphi}^2\sin\theta\, cos^2\theta - 2\dot{\theta}\dot{\varphi}\sin\theta\cos\theta - \alpha\sin\theta\cos\theta + \gamma u - c_a\dot{\varphi} \end{cases}$$

Donde c_p y c_a son los parámetros correspondientes a los términos de fricción viscosa del péndulo del brazo y del motor respectivamente.

Estos dos últimos sistemas de ecuaciones diferenciales son las ecuaciones de movimientos del péndulo sin y considerando la fricción, y son los modelos que se implementarán en *Simulink* en el próximo punto.

2.1.2 Creación del modelo en *Simulink*

Simulink es una herramienta muy potente y sencilla para resolver ecuaciones diferenciales, a continuación se va a explicar brevemente como introducir una ecuación diferencial en ella.

Imaginemos que tenemos la siguiente ecuación diferencial

$$m\ddot{x} + c\dot{x} + kx = f(t)$$

Que corresponde a un modelo de masa-muelle-amortiguador donde se aplica una fuerza externa.

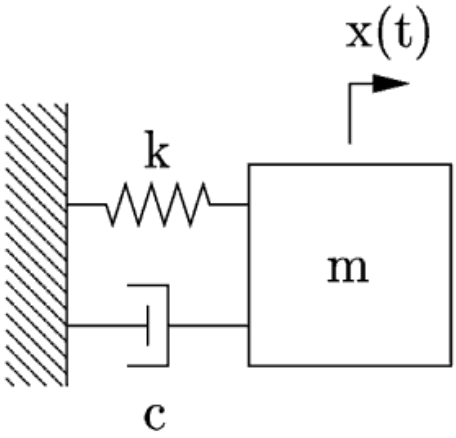

Figura 2-2. Modelo de masa-muelle-amortiguador.

El primer paso es normalizar la ecuación, es decir, hacer que el coeficiente de la derivada de mayor orden sea uno.

$$\ddot{x} + \frac{c}{m}\dot{x} + \frac{k}{m}x = \frac{1}{m}f(t)$$

Luego despejamos la derivada de mayor orden

$$\ddot{x} = \frac{1}{m}f(t) - \frac{c}{m}\dot{x} - \frac{k}{m}x$$

Como la ecuación diferencial es de orden dos, necesitamos dos bloques integradores en los cuales se introducen las condiciones iniciales $x_0 = x(0)$ y $\dot{x}_0 = \dot{x}(0)$. En la figura 2-3 se puede ver el modelo en *Simulink*, en la figura 2-4 se puede ver el contenido que hay dentro del subsistema creado, crear un subsistema es útil principalmente para ahorrar espacio y que se pueda ver más claramente el modelo que realizamos, para ello solo tenemos que seleccionar la parte del modelo que queremos meter dentro del subsistema y (en el caso de *MatlabR2015a*) seleccionar *Diagram > Subsystem & Model Reference > Create Subsystem from Selection.*

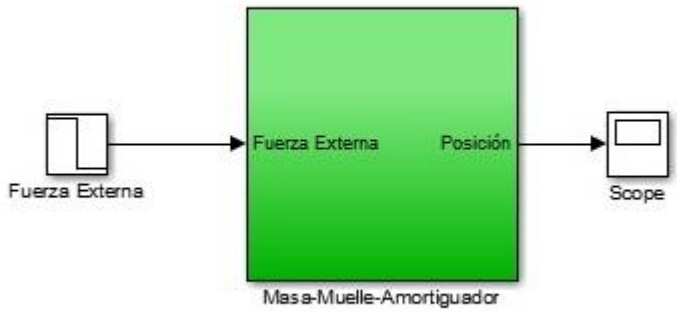

Figura 2-3. Modelo en Simulink®.

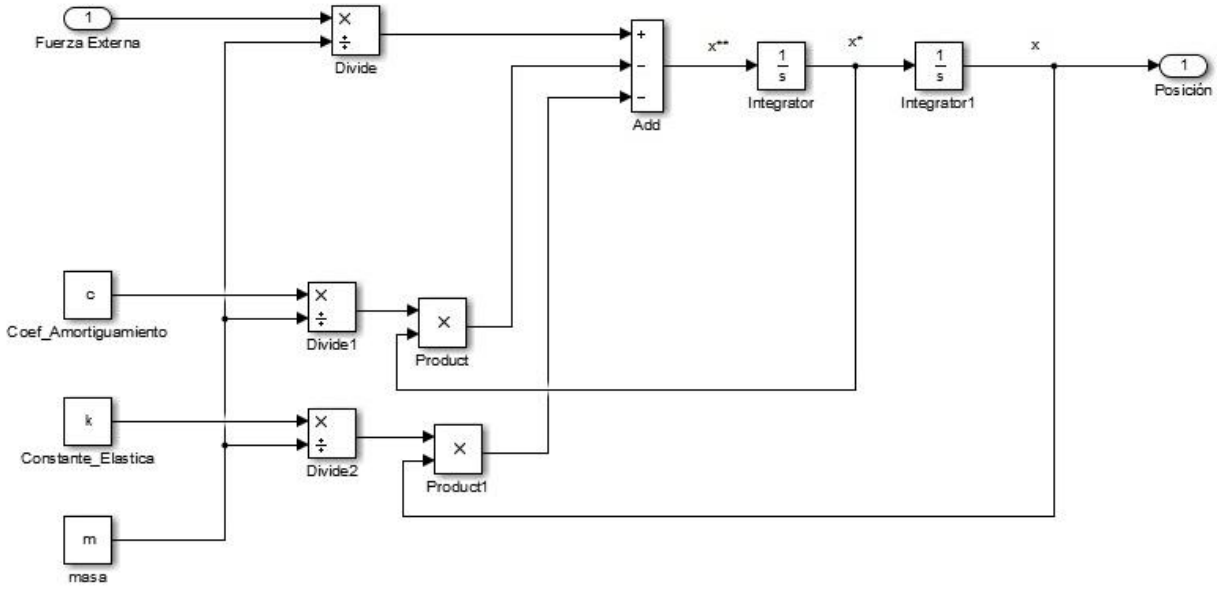

Figura 2-4. Contenido del subsistema.

Si damos valores para los parámetros m, c y k desde el *Command Window*, por ejemplo,

>> m=2; k=1; c=1;

Y con $x_0 = x(0) = 0$ y $\dot{x}_0 = \dot{x}(0) = 0$, simulando obtenemos la respuesta de la figura 2-5.

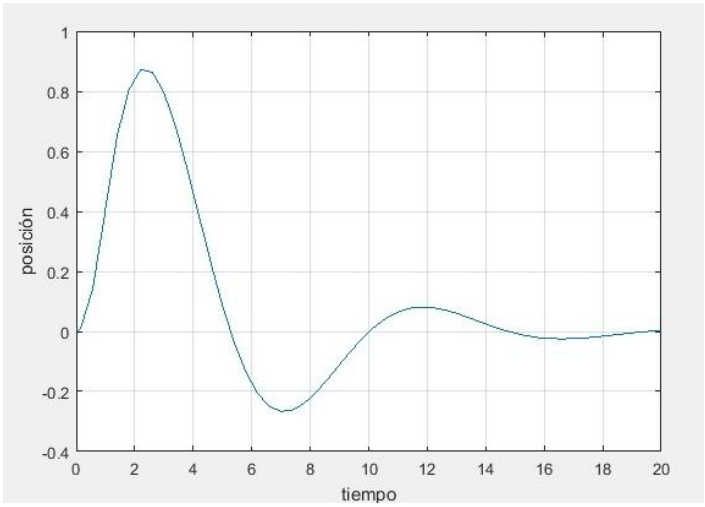

Figura 2-5. Simulación del modelo masa-muelle-amortiguador.

Procediendo de la misma forma, a continuación se va a crear en *Simulink* el modelo del péndulo de Furuta que se obtuvo en el apartado 2.1.1., considerando los parámetros de fricción. Este modelo es,

$$\begin{cases} \Delta\ddot{\theta} = -\alpha^2\dot{\theta}^2 \sin\theta\cos\theta + (\beta + sin^2\theta)\dot{\varphi}^2 \sin\theta\cos\theta + 2\alpha\dot{\theta}\dot{\varphi}\sin\theta\, cos^2\theta + \beta\sin\theta + sin^3\theta - \gamma u\alpha\cos\theta - c_p\dot{\theta} \\ \Delta\ddot{\varphi} = \alpha\dot{\theta}^2\sin\theta - \alpha\dot{\varphi}^2\sin\theta\, cos^2\theta - 2\dot{\theta}\dot{\varphi}\sin\theta\cos\theta - \alpha\sin\theta\cos\theta + \gamma u - c_a\dot{\varphi} \end{cases}$$

Con,

$$\Delta = \beta + sin^2\theta - \alpha^2 cos^2\theta$$

Como se explicó anteriormente, el primer paso para introducir las ecuaciones en *Simulink* es normalizarlas, es decir, dividir lo que tenemos a la derecha de ambas expresiones entre Δ.

El modelo, tras crear un subsistema que además enmascaramos, queda como se puede ver en la figura 2-6.

Figura 2-6. Modelo de Furuta en Simulink.

Lo que encontramos dentro del subsistema, para acceder a ello tenemos que pulsar botón derecho sobre el modelo > *Mask* > *Look Under Mask* ya que el subsistema está enmascarado, es lo que se puede apreciar desde la figura 2-7 hasta la figura 2-12. Aunque pueda parecer complejo, lo que se ha hecho ha sido exactamente lo mismo que para el caso de la masa-muelle-amortiguador pero aplicándolo a dos ecuaciones diferenciales, que no son lineales como se puede ver y además dependen la una de la otra.

Es interesante mencionar lo siguiente del subsistema:

- En la entrada del modelo se ha colocado un bloque de saturación, ya que se tiene en cuenta que la tensión que podemos introducir al QUBE-Servo es de entre -5 y 5 voltios.
- Las condiciones iniciales, en velocidad y posición, tanto de φ como de θ se introducen en los integradores, las condiciones iniciales que se han puesto son $\theta_0 = 0, \varphi_0 = 0\,, \dot{\theta}_0 = 0\ y\ \dot{\varphi}_0 = 0$.
- En el modelo en *Simulink* se usan etiquetas *Goto* y *From* para una mayor claridad en el modelo.
- Por otro lado, los parámetros del sistema, r, l, m, Km, Cp, Ca, g y Ja, se introducen desde el exterior, para tener una mayor facilidad a la hora de cambiar sus valores, algo que nos será muy útil posteriormente para realizar la identificación de los parámetros del sistema.

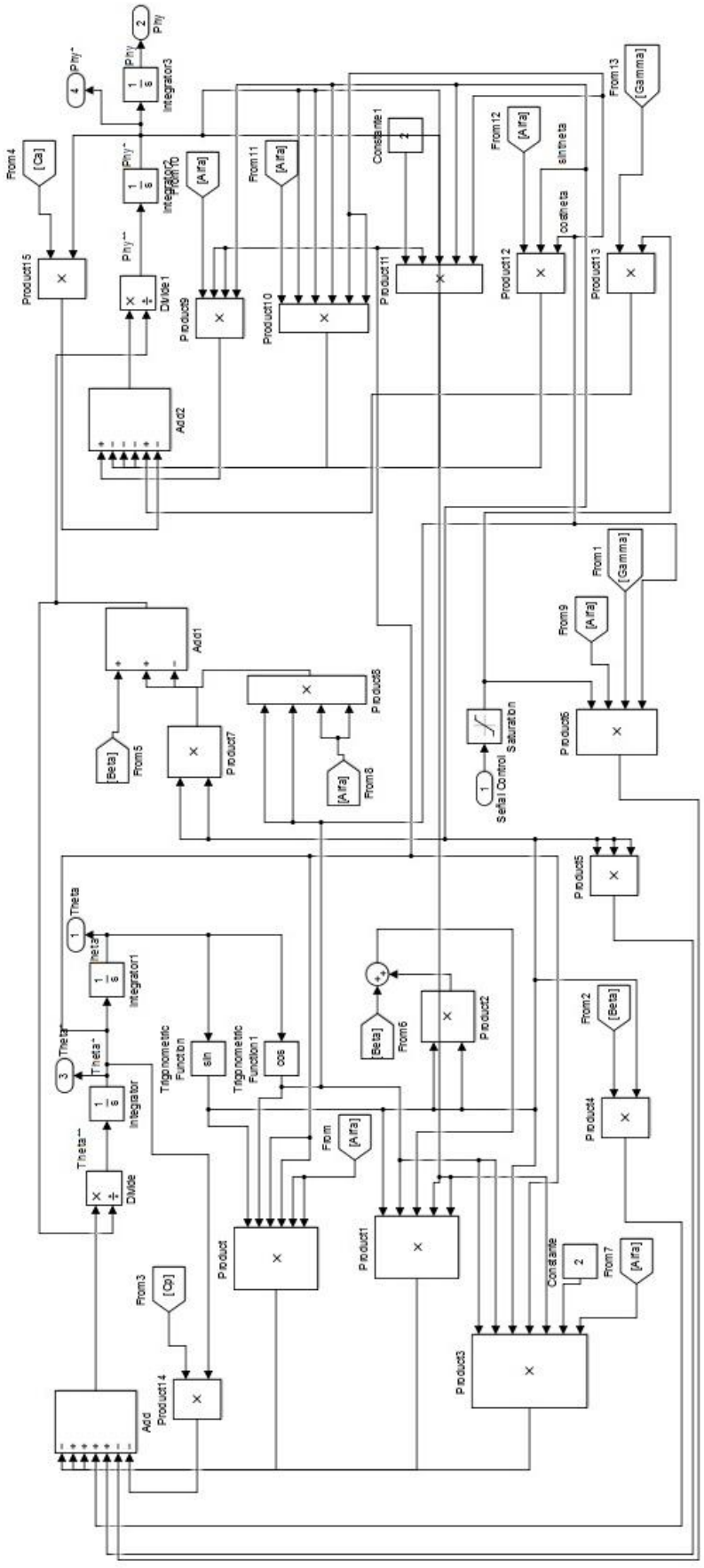

Figura 2-7. Subsistema modelo de Furuta en Simulink (I).

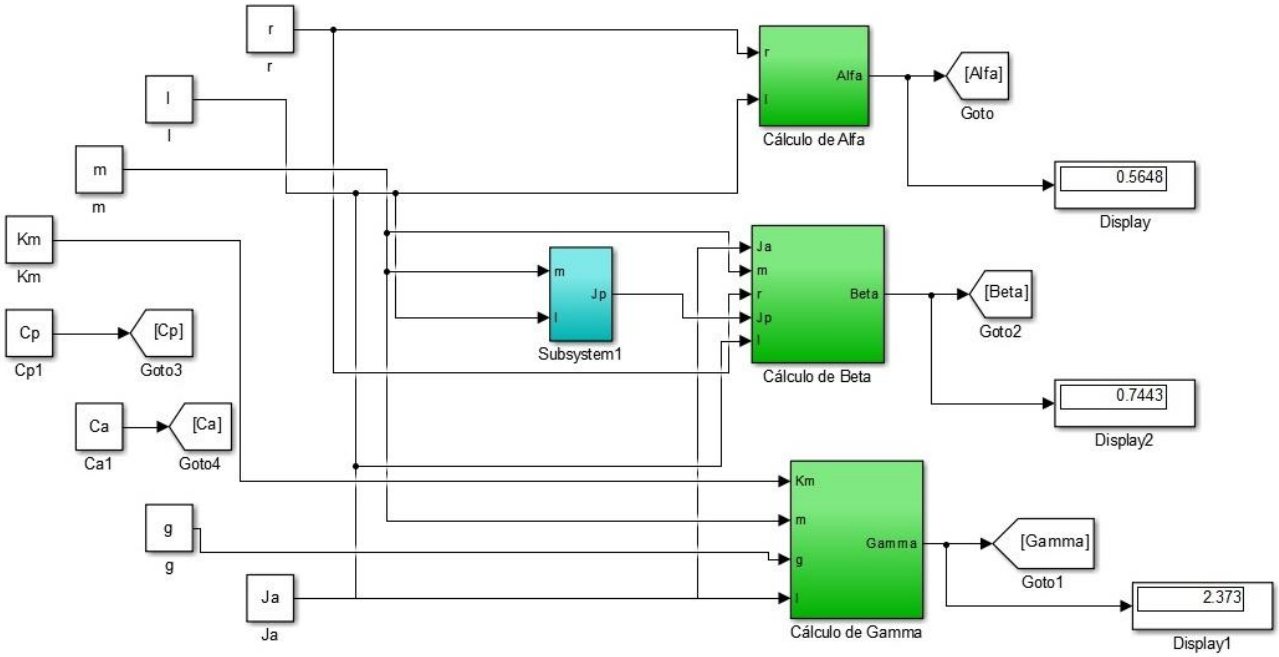

Figura 2-8. Subsistema modelo de Furuta en Simulink (II).

Donde los subsistemas que calculan Alfa, Beta, Gamma y Jp son los que se pueden ver en las figuras 2-9, 2-10, 2-11 y 2-12.

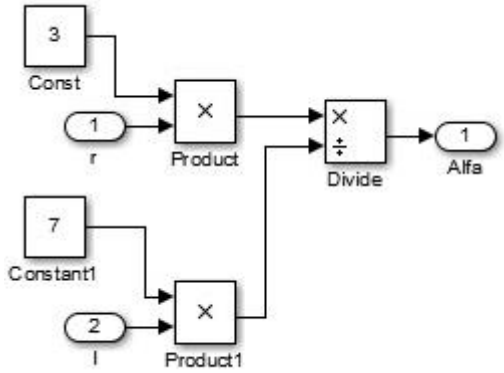

Figura 2-9. Cálculo del parámetro Alfa.

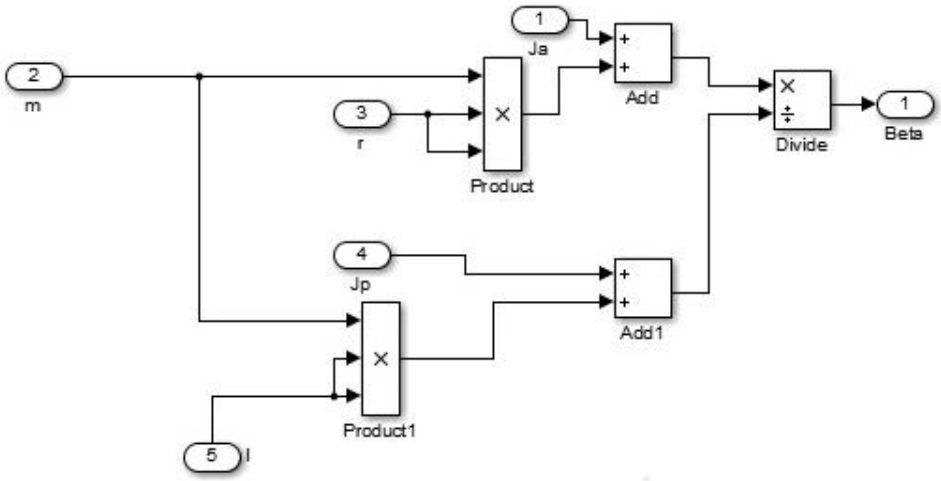

Figura 2-10. Cálculo del parámetro Beta.

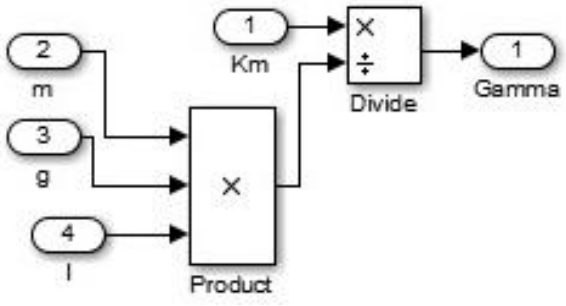

Figura 2-11. Cálculo del parámetro Gamma.

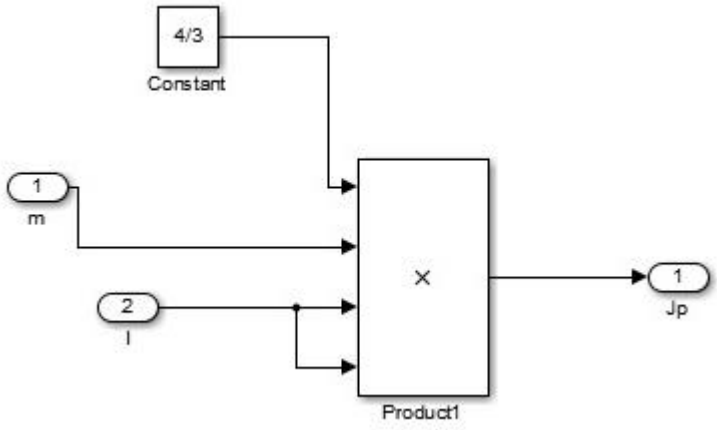

Figura 2-12. Cálculo del parámetro Jp.

Antes se ha comentado que el subsistema está enmascarado, ¿en qué consiste dicho enmascaramiento?

Crear una máscara permite que el subsistema se convierta en un bloque como cualquier otro de *Simulink* permitiéndonos introducir los valores como si fuera un bloque cualquiera, también permite que se haga una descripción del bloque e incluso que en el bloque pueda aparecer un dibujo o esquema. En la Figura 2-13 podemos ver la ventana que se obtiene si se pulsa con el botón izquierdo sobre el subsistema enmascarado.

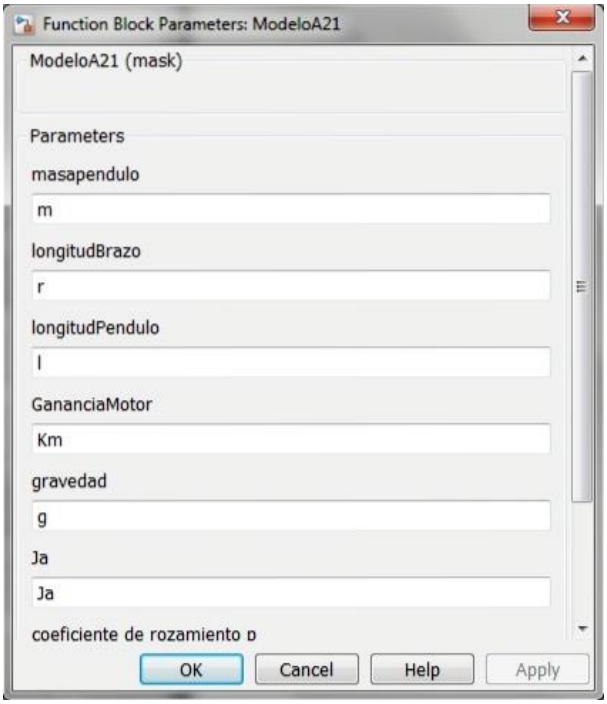

Figura 2-13. Parámetros del ModeloA21 enmascarado.

Para crear el enmascaramiento del subsistema hay que hacer lo siguiente, una vez que tenemos el modelo y creamos el subsistema, pulsamos con el botón derecho sobre el modelo y seleccionamos *Mask > Create Mask*, entonces aparece una ventana con distintas pestañas.

Las pestañas más importantes son la pestaña *Documentation* donde podemos dar una descripción del bloque y la pestaña *Parameters & Dialog* donde se pueden introducir los parámetros que después nos pide el subsistema al clicar dos veces sobre él con el botón izquierdo.

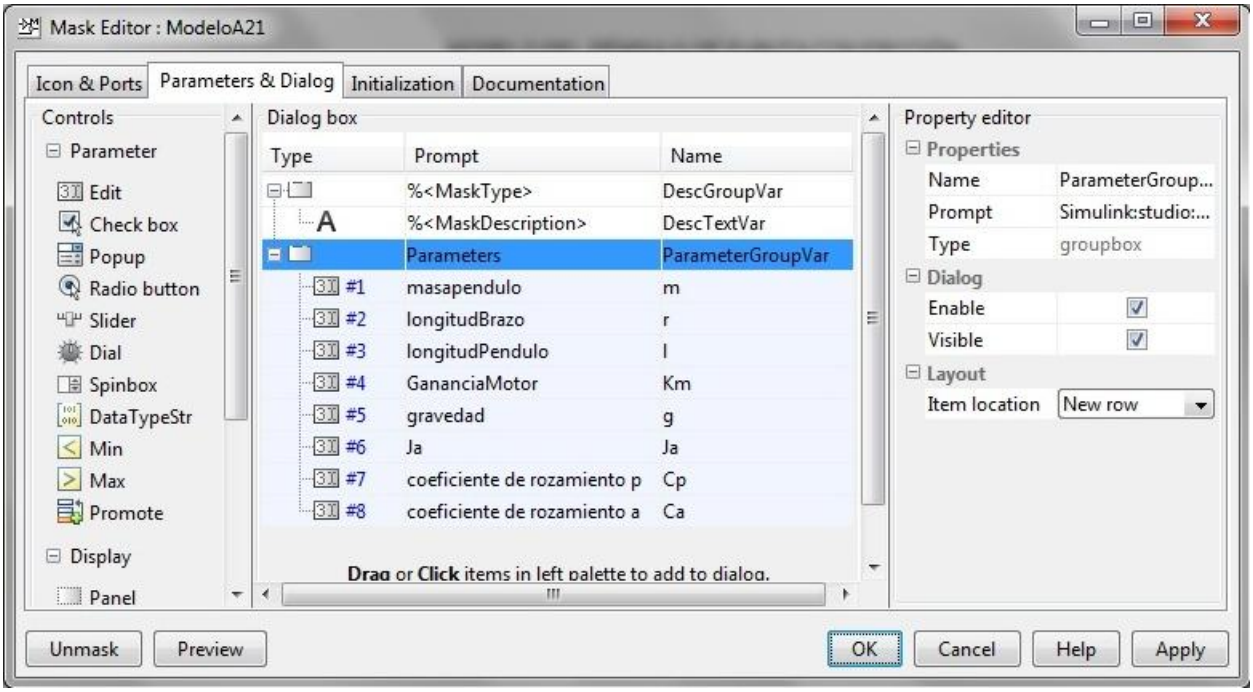

Figura 2-14. Pestaña *Parmeters & Dialog.*

Si queremos introducir algún valor nuevo solo tenemos que pulsar *edit*.

Los parámetros que se aprecian en la tabla 2-1 serán los valores que usaremos para todas las simulaciones que se realizarán hasta el apartado 3 del proyecto.

Tabla 2-1. Parámetros de simulación.

PARÁMETRO	**VALOR**
Masa, m	0.024 kg
Longitud del brazo, r	0.085 m
Mitad de la longitud del péndulo, l	0.0645m
Ganancia del motor, Km	0.036 V/(rad/s)
Gravedad, g	9.8 m/s^2
Momento de inercia brazo y motor, Ja	0 Kg m^2
Fricción viscosa del péndulo, cp	0.5
Fricción viscosa del brazo del motor, ca	0.5

Si además de los parámetros anteriores, aplicamos una tensión de 0 voltios al motor y cambiamos la condición inicial del ángulo del péndulo para que este no parta desde una posición completamente vertical, sino $\theta_0 = 0.1$, obtenemos una respuesta del péndulo como se puede ver en las figuras 2-15 y 2-16.

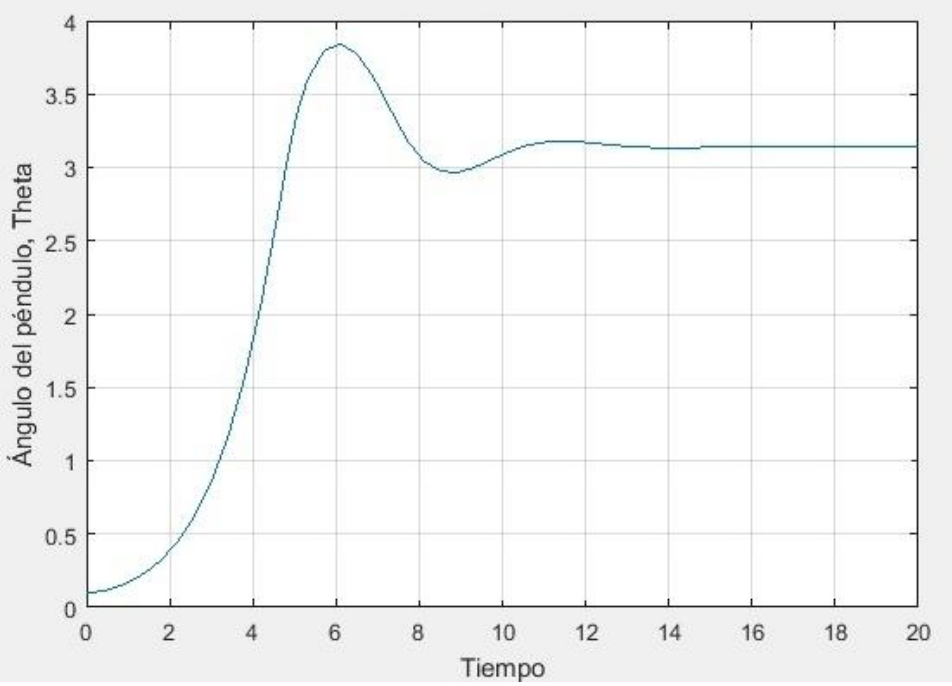

Figura 2-15. Ángulo Theta.

Como se puede apreciar el ángulo del péndulo va desde 0,1 radianes, que es la condición inicial hasta aproximadamente π radianes, que sería el péndulo colgando hacia abajo.

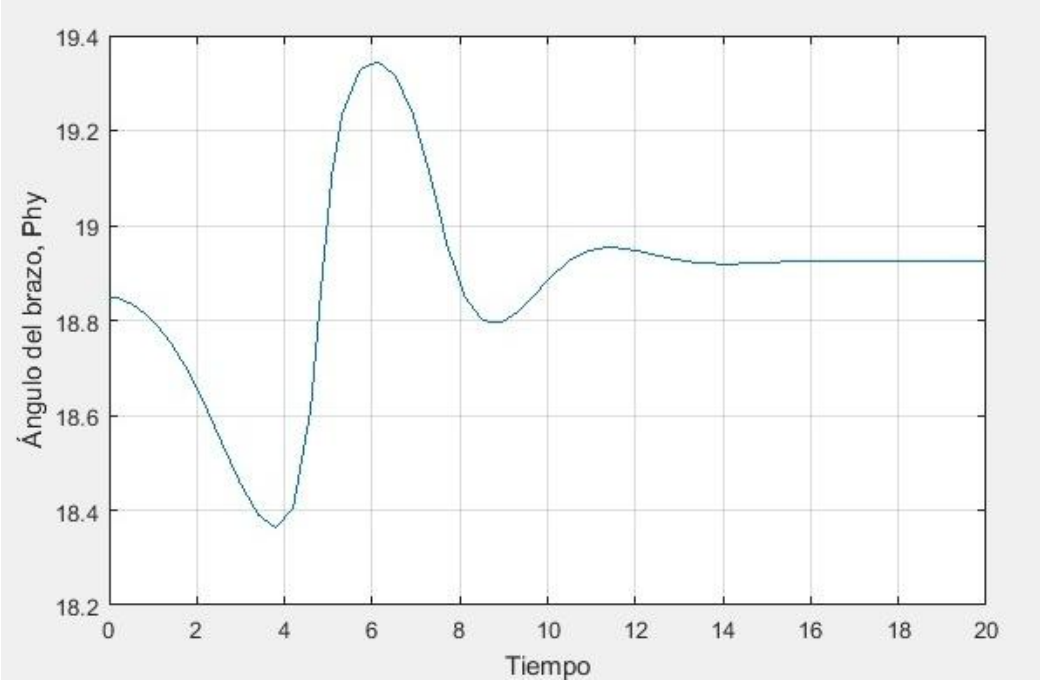

Figura 2-16. Ángulo Phy.

En este caso he puesto que $\varphi_0 = 6\pi$, aunque podría haber sido cualquiera.

Las gráficas anteriores, como se puede apreciar, se han representado en *Matlab* y no en *Simulink*, para ello tenemos que conectar el bloque *simout* en la señal que queremos representar, con *Save format: Array*, y luego en el *Workspace* escribir lo siguiente,

```
>> plot(tout,simout);grid;
>> xlabel('Tiempo');ylabel('Ángulo del brazo, Phy');
```

Como se puede apreciar, visualizar el movimiento del péndulo a partir de las gráficas anteriores puede ser complicado, por ello en el siguiente apartado se va a explicar cómo construir un modelo en 3D del péndulo de Furuta que nos permita observar con gran detalle los movimientos del péndulo.

2.1.3 Creación del péndulo de Furuta en 3D con *V Realm*

Matlab cuenta con un *toolbox* que permite diseñar modelos en 3D que se pueden colocar en *Simulink* y moverse en función de las señales que le llegan, esta herramienta se llama *VRealm*.

Para acceder a *VRealm* hemos de irnos donde tenemos instalado nuestro *Matlab*, meternos en la carpeta de los *toolbox* y buscar *sl3d* a continuación nos metemos en la carpeta *vrealm* y dentro de esta hay que abrir la carpeta *program*, y una vez dentro abrimos el programa *vrbuild2*.

Una vez abierto el programa debemos ver una ventana como la que se muestra en la figura 2-17.

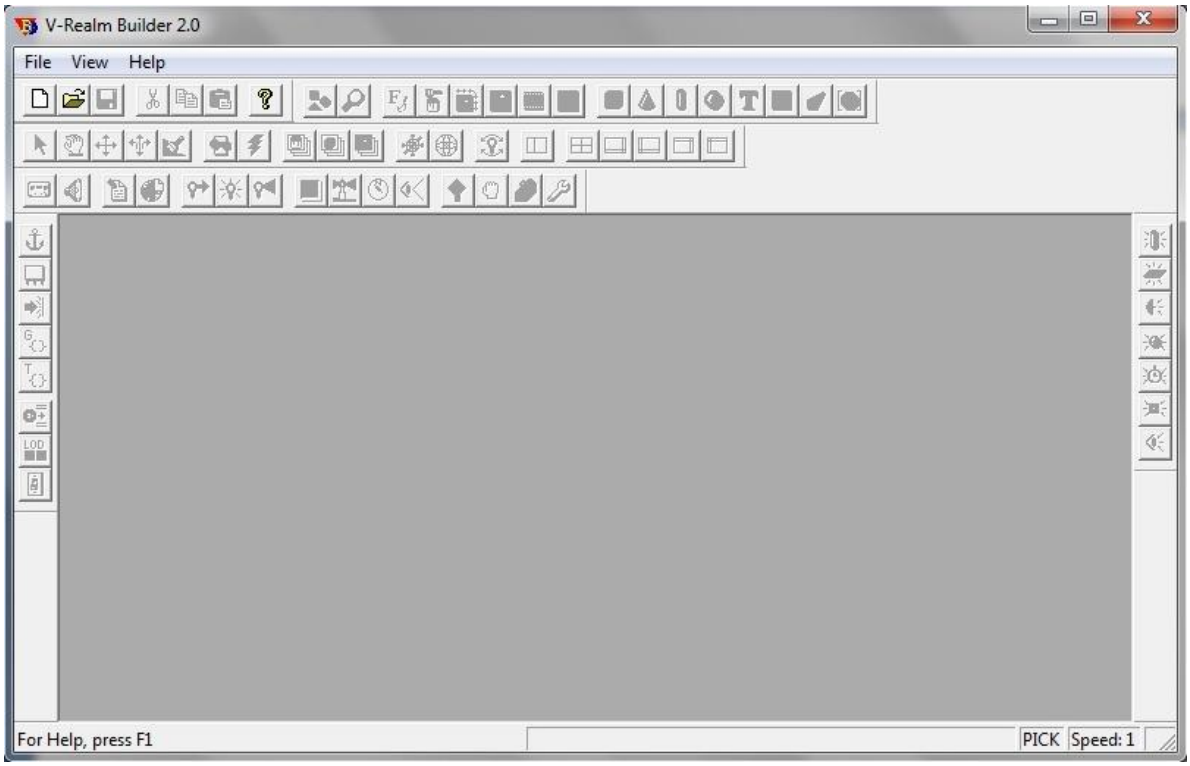

Figura 2-17. V-Realm Builder 2.0

Podemos ir realizando nuestro modelo añadiendo bloques en forma de cubos, cilindros o esferas entre otros. Cuando un nuevo bloque va solidario o pertenece a un bloque anterior este se introduce en el *children* del anterior en un diagrama de árbol que se va creando.

El péndulo que se ha diseñado es el que se presenta en la figura 2-18, está constituido por una base en forma de cubo de color rojo, del que sale un enganche negro que conecta la base al brazo, al final del brazo podemos encontrar otro enganche, negro también, que conecta al brazo con el péndulo.

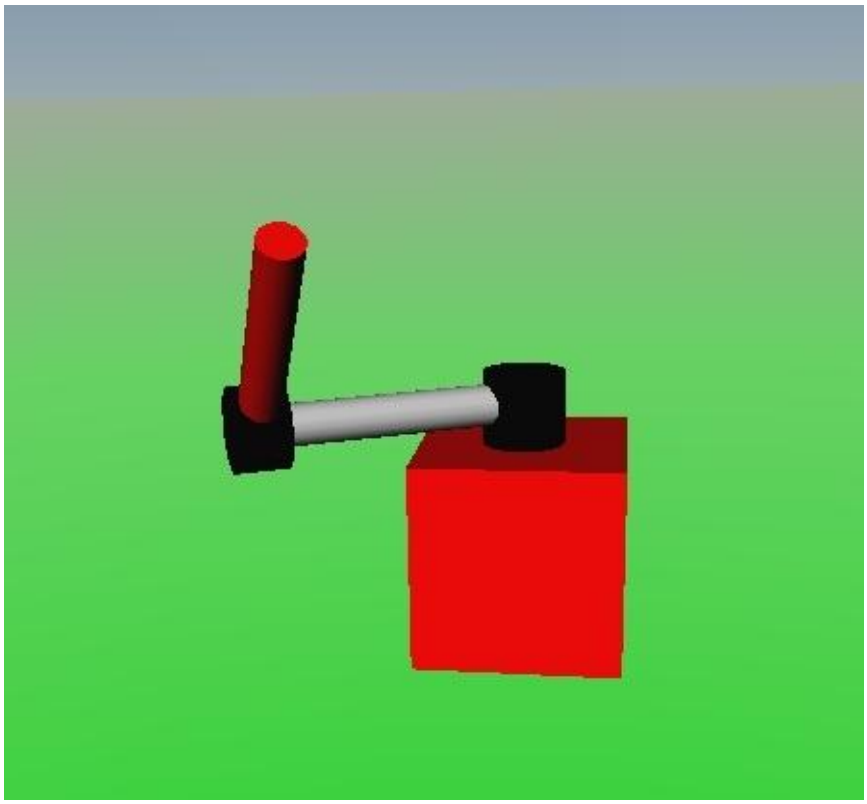

Figura 2-18. Péndulo de Furuta en *VRealm*.

El diagrama de árbol que se ha ido creando para realizar el péndulo es el que se puede apreciar en la figura 2-19, de cada elemento o bloque hay dos partes diferentes que se pueden observar en el árbol.

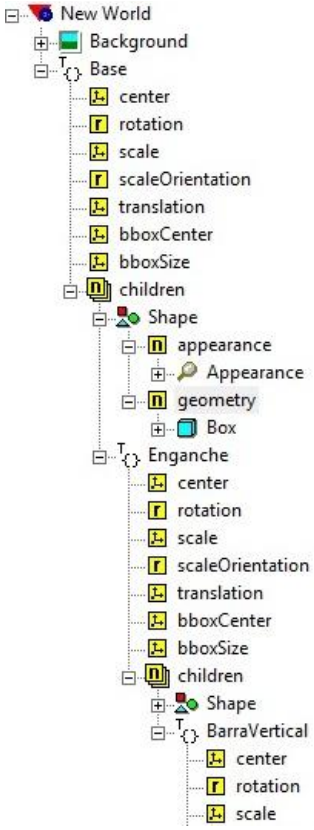

Figura 2-19. Diagrama de árbol de un modelo en *VRealm*.

Lo primero que hay que hacer para comenzar a crear un modelo es pulsar *Insert Background*, dentro de este ya podemos colocar nuestro primer bloque, que en el caso del péndulo de Furuta es un cubo que se ha llamado 'Base', dentro de este bloque se pueden ver dos partes diferenciadas, la parte de *children* que contiene los campos *appearance* y *geometry*, y los campos anteriores. En estos últimos se introducen los parámetros relativos a la posición de los bloques así como los ejes entorno a los que giran o se mueven los bloques, en *appearance* podemos introducir el color de los bloques y en *geometry* sus dimensiones.

Como se ha comentado anteriormente, para que el enganche que conecta la base con el brazo este físicamente conectado a la base, este se debe de crear dentro del *children* del objeto anterior, para ello picamos con el botón izquierdo sobre *children*, y una vez que este está de color azul seleccionamos el nuevo bloque de los distintos que se encuentran en la parte superior del programa.

Con respecto a los movimientos de los distintos elementos es muy importante realizar una configuración correcta. En el caso del modelo del péndulo de Furuta los movimientos, de rotación, lo van a realizar los dos enganches de color negro que se aprecian en el mismo, ya que si uno rota también rotará el resto de los elementos que están conectados al enganche en su *children*. En ambos enganches hay que configurar el campo rotación de la forma que se puede apreciar en la figura 2-20.

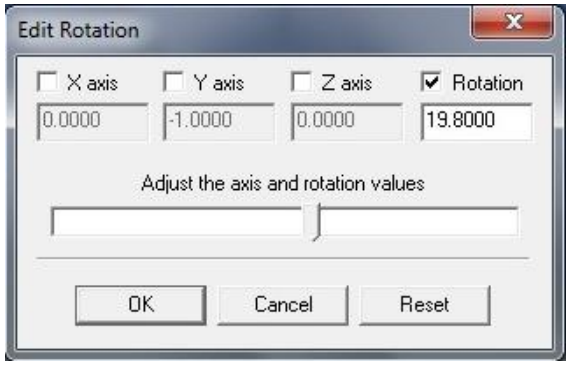

Figura 2-20. Campo rotación en *VRealm*.

Como se puede ver, hay que establecer en primer lugar el eje de rotación, que en ambos casos es el [0 -1 0] en el sistema de referencia que se usa en *VRealm* y si luego marcamos la casilla de rotación y le damos a rotar podemos comprobar que se mueve según lo previsto.

Para poder visualizar el modelo en Simulink hay que introducir una serie de bloques que se encuentran en el campo *Simulink 3D animation* de la librería de bloques de *Simulink*, el bloque *VR Sink* y el *VR Signal Expander*.

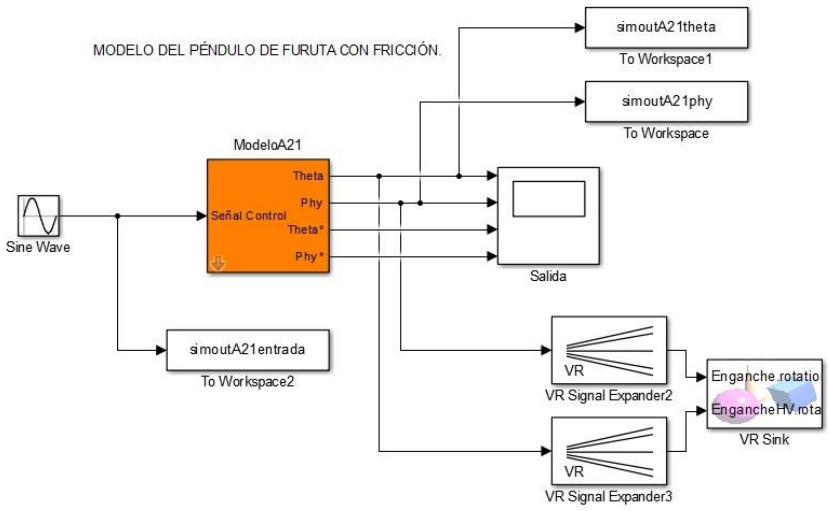

Figura 2-21. Bloques *VR Sink* y *VR Signal* Expander conectados.

En primer lugar se va a comentar como se configura el bloque *VR Sink*, cuando colocamos dicho bloque y picamos sobre el por primera vez, aparece una ventana como la que se muestra en la figura 2-22 donde hay que completar la información de las tres zonas que se muestran en dicha imagen.

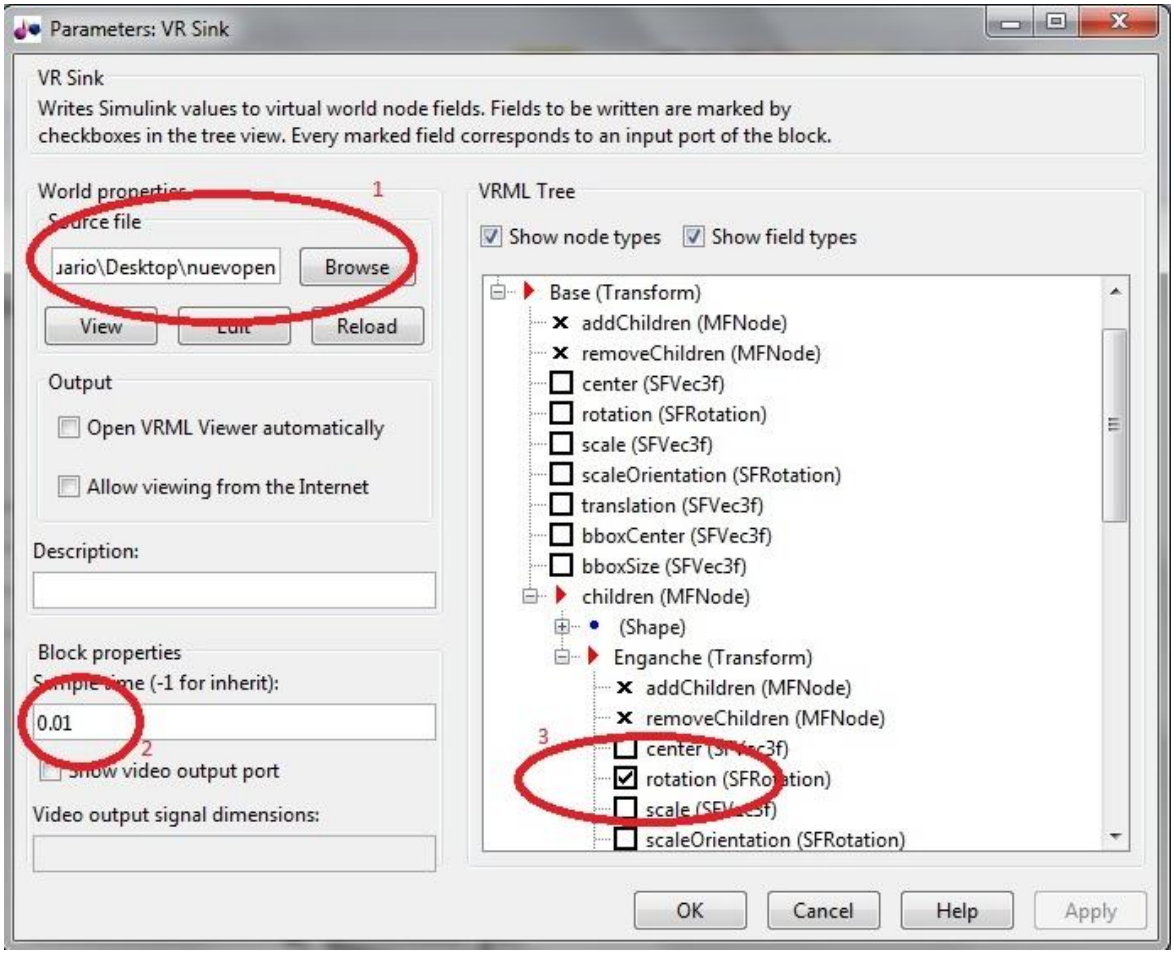

Figura 2-22. Ventana *VR Sink*

En la zona 1 hay que colocar la dirección donde se encuentra el modelo del péndulo que creamos anteriormente en *VRealm*.

En la zona 2 se pone la velocidad a la que se va a ir realizando la simulación y que será interesante cambiar más adelante para ver el movimiento del péndulo con más o menos detalle según interese.

En la zona 3 hay que seleccionar los movimientos que va a tener el péndulo, para ello se marca la casilla 'rotación' en 'Enganche' y en 'EngancheHV', este último no se puede apreciar en la figura 2-22.

Una vez que hemos hecho la configuración por primera vez, si volvemos a clicar sobre el bloque *VR Sink* ya no aparece la ventana de la figura 2-22 sino el entorno virtual donde se encuentra el péndulo. Para navegar por dicho entorno podemos usar el panel de navegación que se encuentra en la parte inferior.

Para simular nuestro modelo podemos hacerlo desde *Simulink* o desde el propio entorno virtual del péndulo pulsando el botón de *play* (ver figura 2-23).

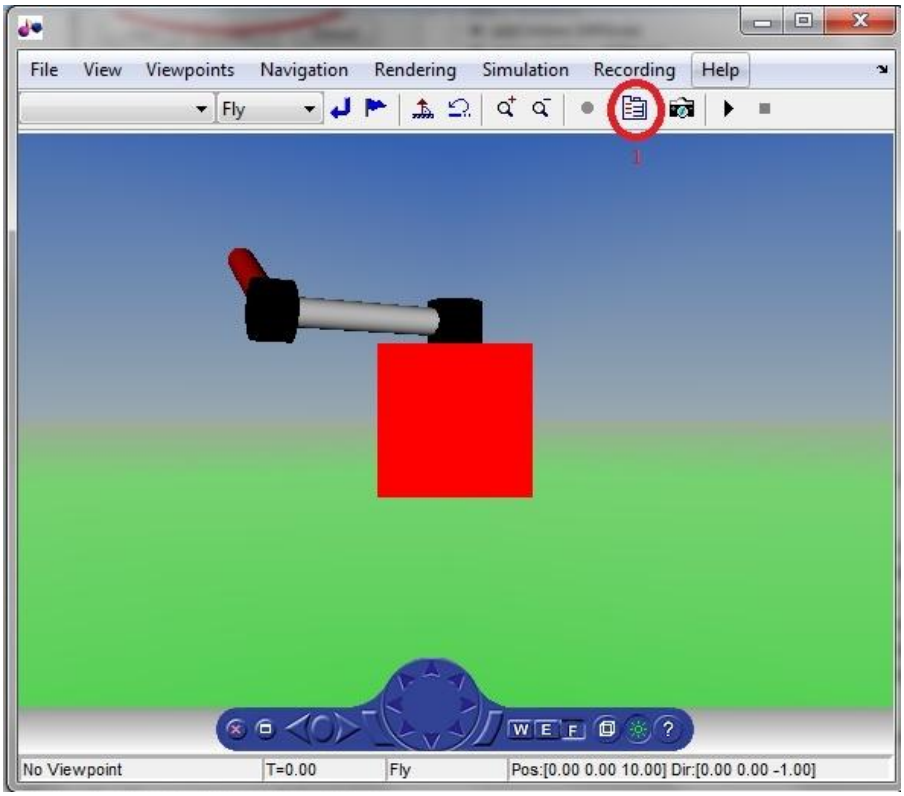

Figura 2-23. Entorno virtual del péndulo en 3D.

Si deseamos cambiar algún parámetro, como por ejemplo la velocidad de simulación de la animación, tenemos que volver a la ventana de la figura 2-22, para ello solo tenemos que picar sobre *Block Parameters*, el icono que se puede ver rodeado de rojo en la figura 2-23.

El otro bloque necesario para poder realizar la simulación de la animación, como se comentó anteriormente, es el *VR Signal Expander*. La función de este bloque es indicar a que parámetro corresponde la entrada que le llega al modelo, es decir, como se pudo ver en la figura 2-20 la rotación tiene cuatro parámetros, *X axis*, *Y axis*, *Z axis* y *Rotation* y de todos estos parámetros solo queremos variar el último, la rotación, ya que los tres primeros son los ejes de rotación que se definieron y que no deben modificar.

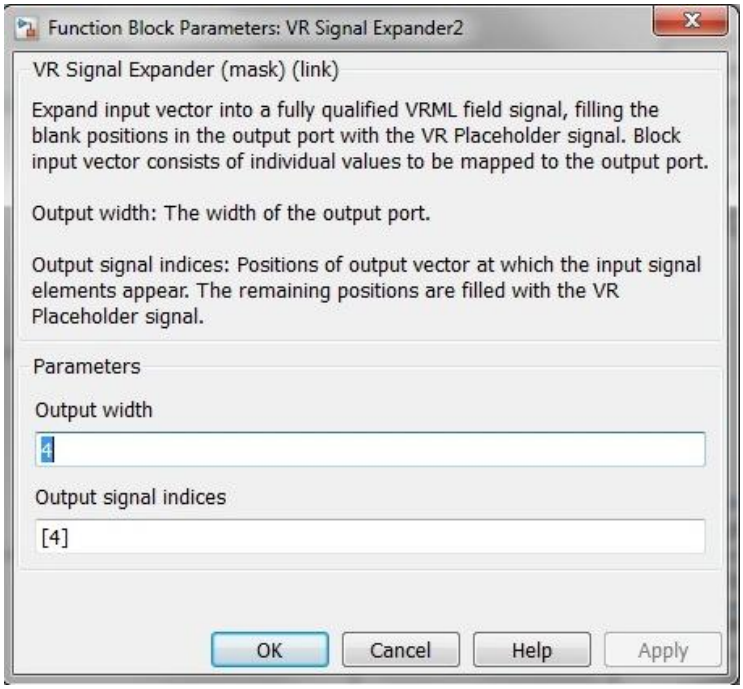

Figura 2-24. Bloque *VR Signal Expander*.

En la figura 2-24 se puede ver que lo que hay que hacer en el bloque *VR Signal* Expander es rellenar *Output width*, donde ponemos cuatro ya que en la rotación hay cuatro parámetros (los que se comentaron anteriormente) y en *Output signal índices* ponemos cuales de los cuatro parámetros que hay va a ser el que se va a modificar en función de la señal de entrada, que es el cuarto, la rotación.

Con esto ya tenemos todo lo necesario para empezar a controlar el péndulo de Furuta, ya que tenemos el modelo del sistema en *Simulink* y también podremos visualizar mucho mejor los movimientos del mismo gracias a la animación 3D que hemos realizado. En el siguiente apartado comenzaremos a explicar cómo controlar el sistema.

2.2. Balance de control PD

En este apartado vamos a controlar el péndulo de Furuta entorno al punto de funcionamiento, es decir, entorno a la posición vertical del mismo a la vez que vamos a intentar controlar también el ángulo del brazo, *phy*. Por tanto se van a controlar dos variables, el ángulo del péndulo y del brazo, con una sola variable de actuación, la tensión de entrada al motor.

Se va a intentar establecer en que rango entorno al punto de funcionamiento el balance de control PD es capaz de controlar adecuadamente y se van a discutir los resultados.

2.2.1. Suma de la actuación de dos controles PD

Como es sabido, un control PD proporciona una entrada al sistema que es la suma de un término proporcional y otro derivativo del error, actuando este último fundamentalmente en el régimen transitorio del sistema que es cuando mayormente varía el error y por tanto la variación del error respecto del tiempo es mayor, como en régimen permanente apenas varía el error la variación del mismo es prácticamente cero provocando una actuación insignificante por parte del término derivativo.

Con un solo control PD es imposible controlar dos salidas, por ello la entrada al sistema va a ser la suma de dos controladores PD, uno aplicado a una de las salidas y el otro PD a la otra salida, de este modo si alcanzamos la referencia para una de las salidas pero no para la otra seguiremos teniendo una señal de entrada al sistema hasta que ambas referencias sean alcanzadas.

Como se puede apreciar en la figura 2-25 la tensión de entrada a nuestro modelo es la suma de dos controladores PD, tal como se comentó anteriormente. El control que se aplica a la posición del ángulo del brazo del péndulo, *phy*, contiene un término proporciona Kp con valor -2 y un término derivativo Kd con valor 0.8, previa a la ganancia derivativa podemos ver un filtro HP con la función de transferencia,

$$Tf(s) = \frac{150s}{s + 150}$$

En las figuras 2-26 y 2-27 podemos ver el diagrama de bode de dicho filtro y que sucede en el error al ser filtrado. Para ello, tras poner un bloque *To Workspace* antes y despúes del filtro de nombres ‘antesfiltrado’ y ‘despuesfiltrar’ que guarda los datos en forma de estructura con tiempo, hemos ejecutado el código que se muestra a continuación.

```
>> TF=tf([150 0],[1 150]);
>> bode(TF);
>> grid;
```

```
>> plot(antesfiltrado.time,antesfiltrado.signals.values,'r');
```

```
>>hold on;
>>grid
>>plot(despuesfiltrar.time,despuesfiltrar.signals.values,'b');
>>legend('Antes de filtrar','Después de filtrar');
>>xlabel('tiempo');ylabel('error filtrado y sin filtrar');
```

Podemos apreciar en la figura 2-27 como al principio, en el régimen transitorio, tras filtrar obtenemos una amplificación del error, lo que provoca que al comienzo tengamos una gran influencia del término derivativo en la respuesta que da el controlador PD y que conforme vamos alcanzando el régimen permanente y las variaciones del error respecto al tiempo son menores la influencia del término derivativo va menguando.

Por otro lado, en el control PD que se realiza sobre el ángulo del péndulo, theta, no hemos puesto referencia ya que esta siempre es cero, que es el punto de funcionamiento entorno al cual el balance de control es válido, el rango de validez del control PD se tratará en el apartado siguiente de este documento. La ganancia proporcional del control PD del ángulo del péndulo es Kp con valor 20 y la ganancia del término derivativo es Kd con valor 1.6.

Los valores Kp y Kd de ambos PD se han obtenido experimentalmente hasta alcanzar unos valores aceptables.

En el apartado siguiente se ven los resultados que proporciona este balance de control PD.

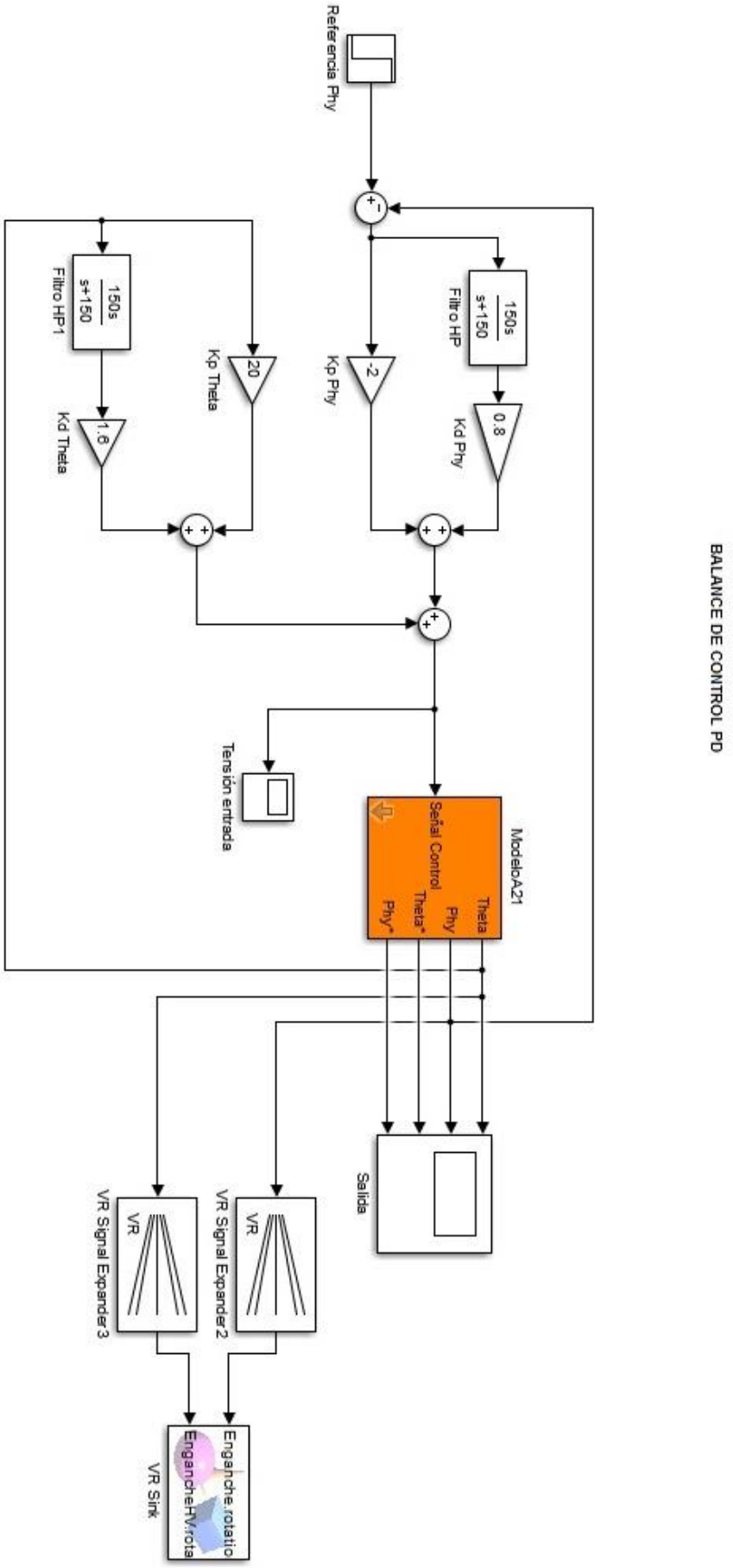

Figura 2-25. Balance de Control PD.

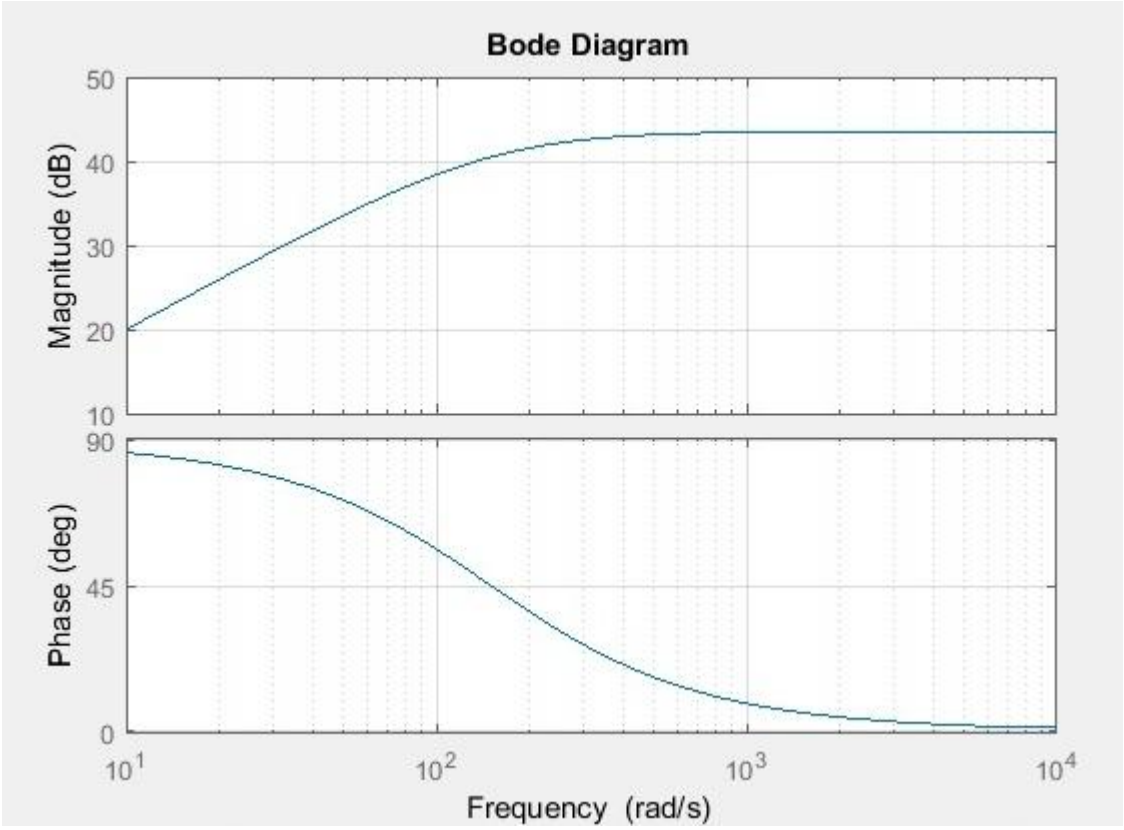

Figura 2-26. Diagrama de bode del filtro HP del balance de control.

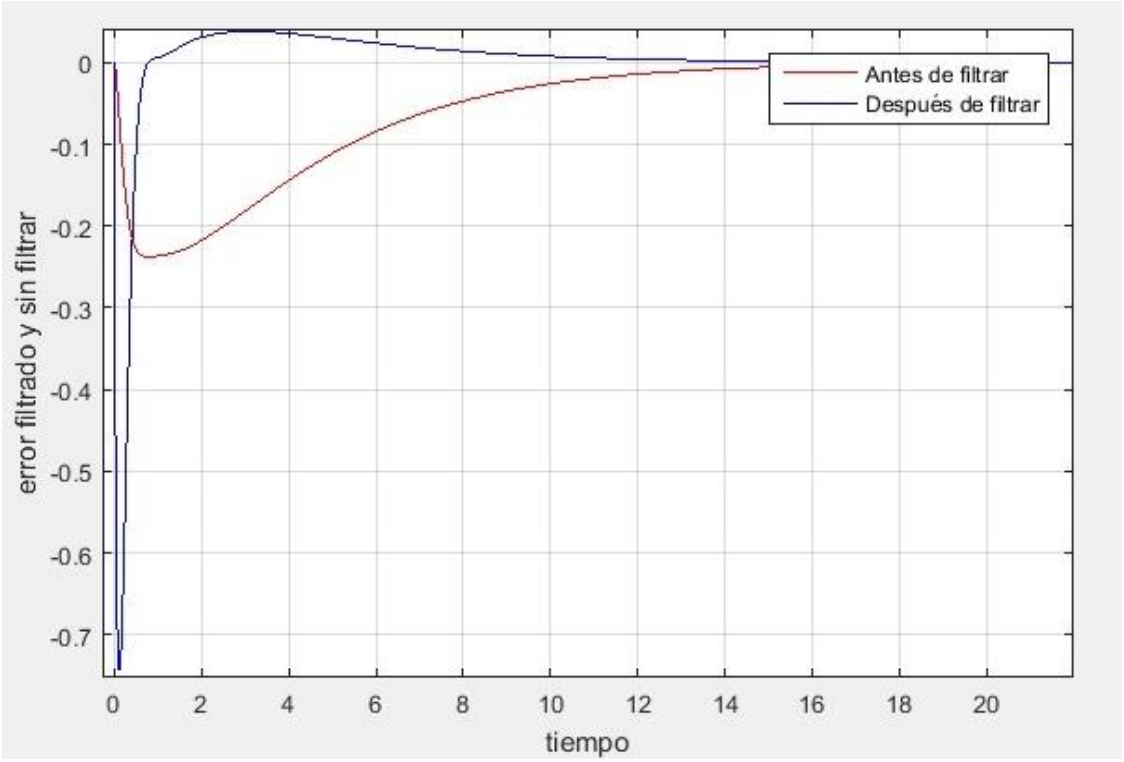

Figura 2-27. Error filtrado y sin filtrar.

2.2.2. Resultados del balance de control y rango de validez

En este subapartado vamos a ver los resultados que proporciona el balance de control PD ante diversas

situaciones.

Simulación del balance de control con condiciones iniciales $\boldsymbol{\theta_0 = 0.1}$, $\boldsymbol{\dot{\theta}_0 = 0}$, $\boldsymbol{\varphi_0 = 0}$ $\boldsymbol{y}$ $\boldsymbol{\dot{\varphi}_0 = 0}$ con referencia de φ=0

Tras poner tres *To Workspace*, uno a la entrada llamado 'entrada'y los otros dos en phy y theta llamados 'physalida' y 'thetasalida' respectivamente y ejecutando el código siguiente, obtenemos el resultado que se puede apreciar en la figura 2-28.

```
>>subplot(3,1,1);
>>plot(entrada.time,entrada.signals.values,'b');
>>grid
>>xlabel('tiempo');ylabel('tension');
>>subplot(3,1,2);
>>plot(thetasalida.time,thetasalida.signals.values,'b');
>>grid
>>xlabel('tiempo');ylabel('theta');
>>subplot(3,1,3);
>>plot(physalida.time,physalida.signals.values,'b');
>>grid
>>xlabel('tiempo');ylabel('phy');
```

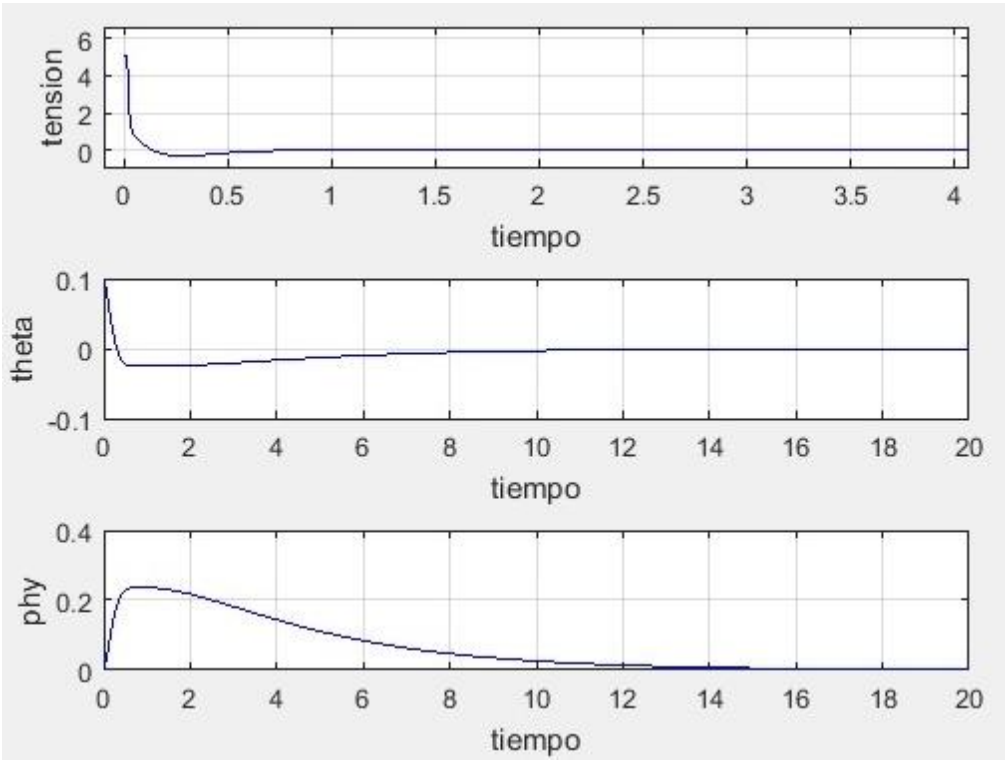

Figura 2-28. Tensión y ángulos de salida en balance de control PD (I).

Simulación del balanace de control con condiciones iniciales $\theta_0 = 0$, $\dot{\theta}_0 = 0$, $\varphi_0 = 0$ y $\dot{\varphi}_0 = 0$ con referencia de φ=pi/2

En este caso, ejecutando el mismo código que en caso anterior tras simular, obtenemos el resultado que se puede apreciar en la figura 2-29.

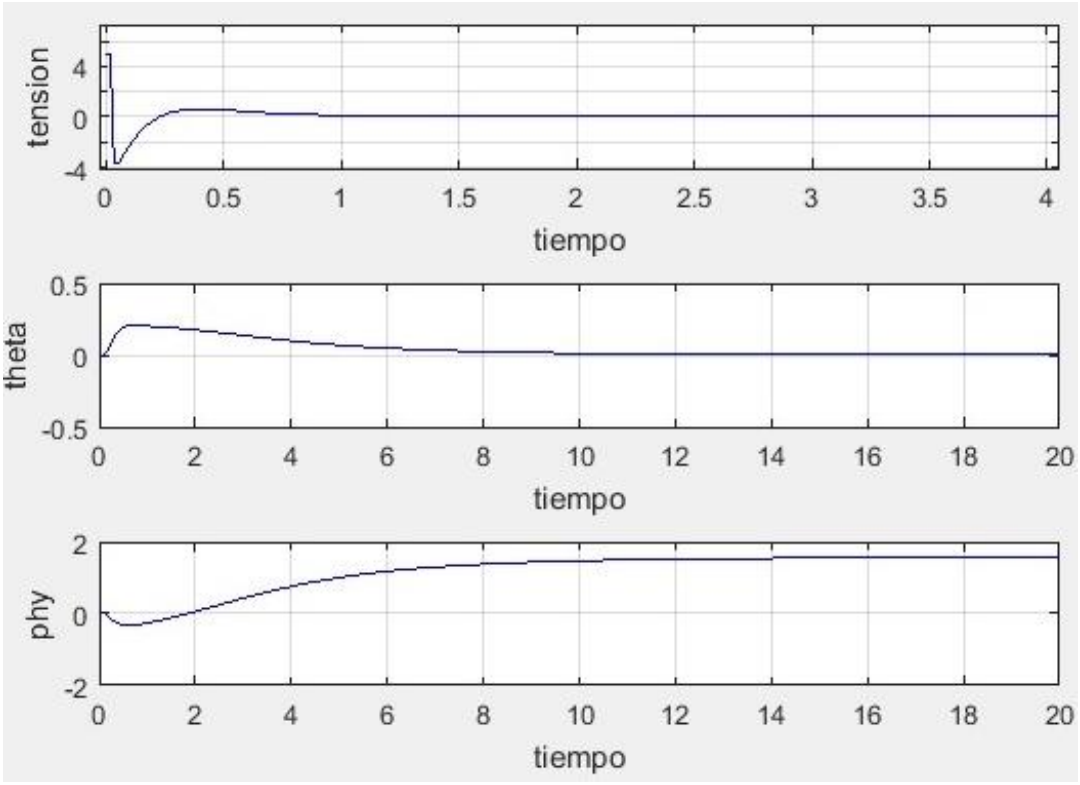

Figura 2-29. Tensión y ángulos de salida en balance de control PD (II).

Rango de validez entorno al punto de funcionamiento

Como hemos podido ver en los apartados anteriores, el balance de control parece responder bien cuando estamos próximos a θ=0, ahora vamos a ver a partir de que valores deja de funcionar el balance de control.

Experimentalmente se puede comprobar cómo es capaz de controlar el péndulo cuando θ pertenece a [-0.57, 0.57] radianes, es decir, θ pertenece a [-32.6, 32,6] grados.

En este caso límite obtenemos el resultado que se aprecia en la figura 2-30 y 2-31.

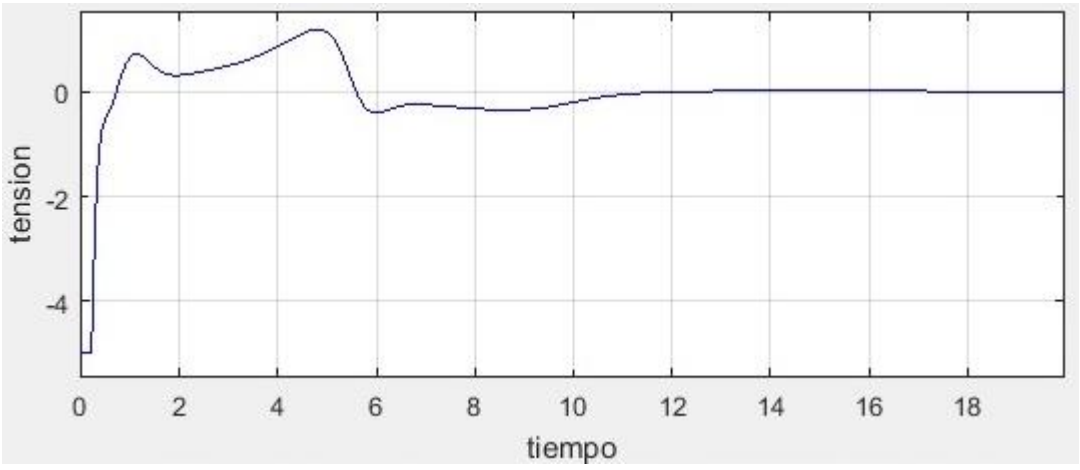

Figura 2-30. Tensión de entrada θ límite.

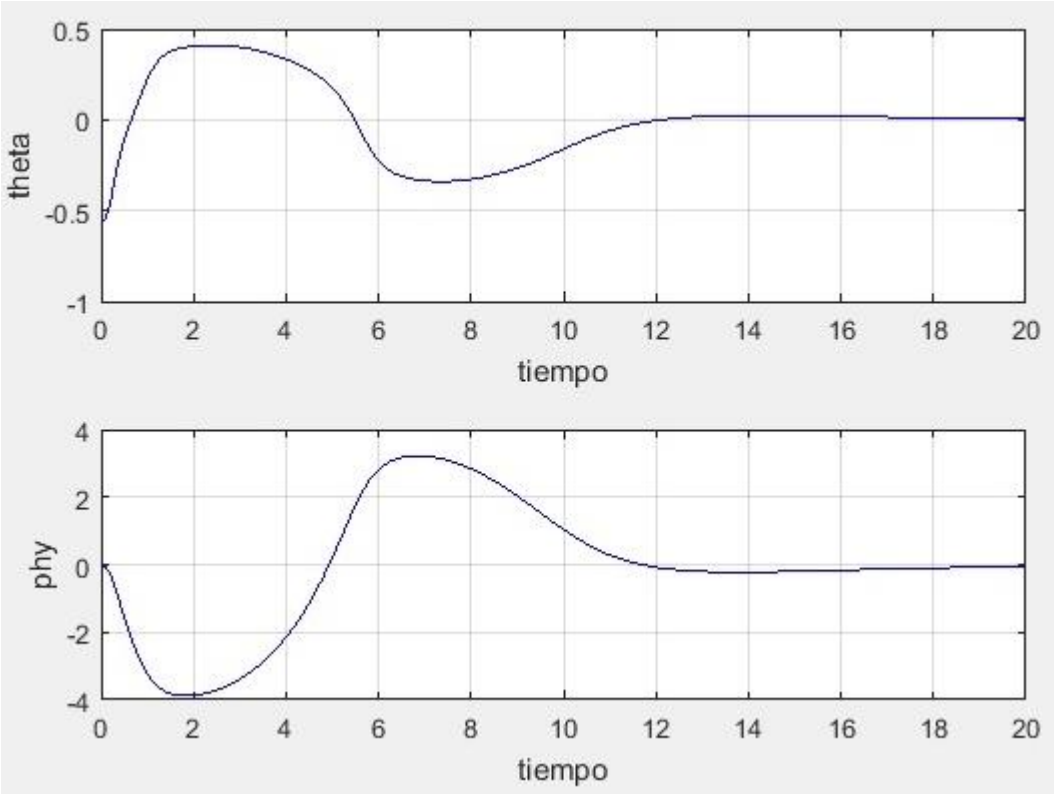

Figura 2-31. Ángulos *theta* y *phy* cuando θ límite.

Podemos concluir que los resultados son aceptables para un amplio rango de validez entorno al punto de funcionamiento.

2.3. Control LQR

En el apartado anterior se controló el sistema entorno al punto de funcionamiento a través de un balance de control PD, en este apartado se va a emplear un tipo de control óptimo, el control LQR, que nos proporcionará un término proporcional que nos permitirá controlar el péndulo. Para usar esta estrategia de control previamente tendremos que linealizar y rescribir en espacio de estados las ecuaciones de movimiento del sistema.

2.3.1. Linealización de ecuaciones diferenciales

Para poder realizar el control LQR previamente tenemos que linealizar nuestro sistema para pasarlo al espacio de estados, en este apartado se intentará explicar detalladamente la linealización de ecuaciones diferenciales.

Antes de linealizar ecuaciones diferenciales veremos como linealizar una ecuación algebraica y se comentará su significado.

Sea la función no lineal $y(t) = t^2$, suponiendo que queremos tener una aproximación lineal a dicha función en $t_o = 2$, la ecuación que rige dicha función lineal es la siguiente,

$$y(t)_{LIN} = y(t_o) + \frac{\partial}{\partial t} y(t)\Big]_{t=t_o} (t - t_o)$$

$$y(t)_{LIN} = 4 + 4(t - 2)$$

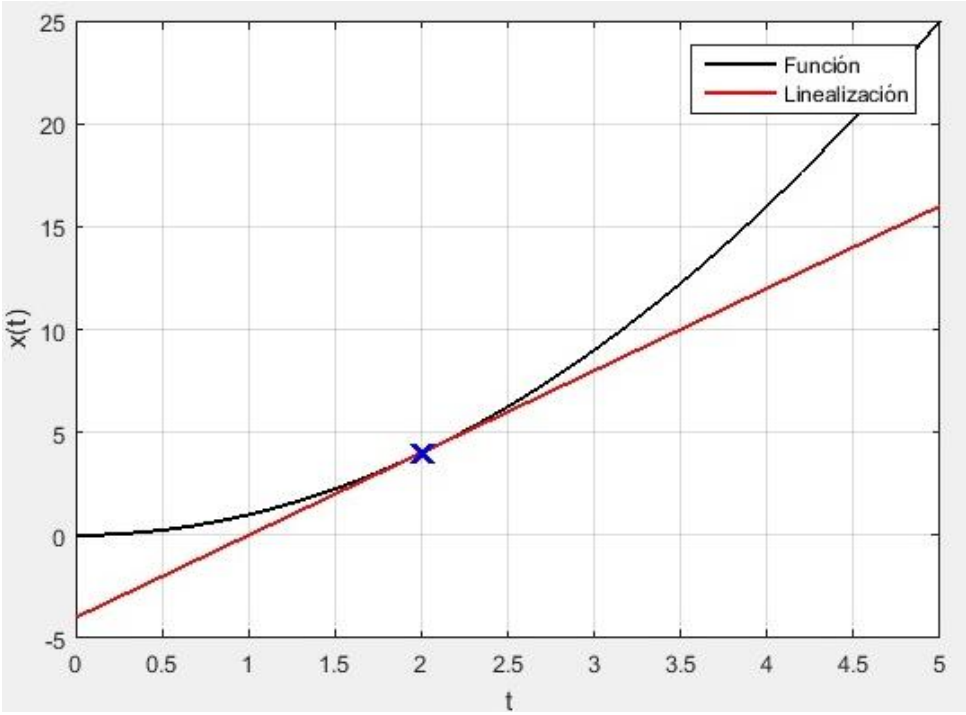

Figura 2-32. Linealización de función algebraica (I).

Se introduce el término de variable incremental, $\alpha(t) = t - t_o$. Como se puede ver en la figura 2-32 para que la aproximación lineal sea válida hemos de estar entorno a t_o, es decir, la varible incremental debe estar próxima a cero. Además de modelo de variables incrementales a veces también se le puede llamar modelo de pequeña señal.

Si tenemos más variables entorno a las que linealizar el procedimiento es idéntico, sea la función $f(x, y) = xy$ se quiere linealizar entorno a $x_o = 0$e $y_o = 0$. Se definen las variables incrementales

$$\alpha(t) = x(t) - x_o$$

$$\beta(t) = y(t) - y_o$$

La linealización responde a la expresión de Taylor,

$$f(x,y)_{LIN} = f(x_o, y_o) + \frac{\partial}{\partial x} f(x,y)\Big]_{x_o,y_o} \alpha(t) + \frac{\partial}{\partial y} f(x,y)\Big]_{x_o,y_o} \beta(t)$$

Que en nuestro caso daría lugar a,

$$f(x,y)_{LIN} = 0$$

En la figura 2-33 se puede apreciar gráficamente dicha linealización, que no es más que un plano tangente en x=0 e y=0 a la función $f(x, y) = xy$.

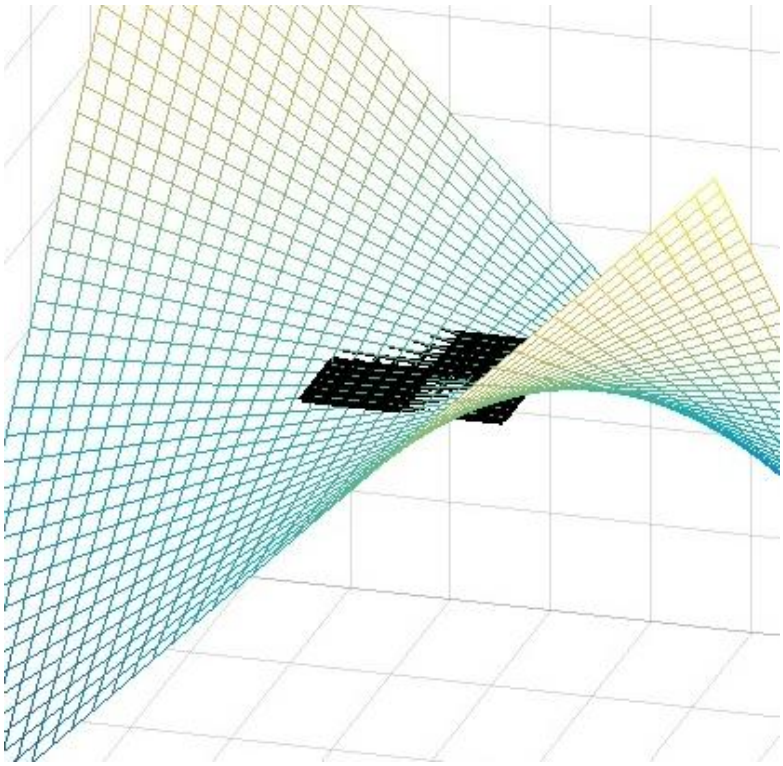

Figura 2-33. Linealización de función algebraica (II).

Para linealizar ecuaciones diferenciales básicamente hacemos lo mismo que para ecuaciones algebraicas pero teniendo en cuenta que:

1) Si hay una parte de la ecuación diferencial que se deriva respecto al tiempo, el primer paso es realizar dicha derivación total, véase el siguiente ejemplo.

$$F(t) = \frac{d}{dt}(a \cdot H^2(t)) \rightarrow F(t) = 2 \cdot a \cdot H(t) \cdot \dot{H}(t)$$

2) Hay que tomar la expresión de Taylor, derivar parcialmente, respecto de todas las variables y sus derivadas, ya que todas tendrán un valor concreto en el punto de funcionamiento.

A continuación se van a linealizar las ecuaciones de movimiento de péndulo de Furuta entorno al punto de funcionamiento que no es otro que $\theta_o = 0, \dot{\theta}_o = 0, \ddot{\theta}_0 = 0, \varphi_o = 0, \dot{\varphi}_o = 0, \ddot{\varphi}_o = 0$ y $u_o = 0$, que sería con el péndulo en posición vertical hacia arriba con el brazo en un ángulo de cero grados, donde no debe de haber movimiento, de ahí que las derivadas temporales sean cero, y con tensión aplicada igual a cero voltios.

Las ecuaciones de movimiento, obtenidas en el apartado 2.1.1., son,

$$\begin{cases} \Delta\ddot{\theta} = -\alpha^2\dot{\theta}^2 \sin\theta\cos\theta + (\beta + sin^2\theta)\dot{\varphi}^2 \sin\theta\cos\theta + 2\alpha\dot{\theta}\dot{\varphi}\sin\theta\, cos^2\theta + \beta\sin\theta + sin^3\theta - \gamma u\alpha\cos\theta - c_p\dot{\theta} \\ \Delta\ddot{\varphi} = \alpha\dot{\theta}^2\sin\theta - \alpha\dot{\varphi}^2\sin\theta\, cos^2\theta - 2\dot{\theta}\dot{\varphi}\sin\theta\cos\theta - \alpha\sin\theta\cos\theta + \gamma u - c_a\dot{\varphi} \end{cases}$$

Con,

$$\Delta = \beta + sin^2\theta - \alpha^2 cos^2\theta$$

Linealizando la primera ecuación término a término queda que el término de la izquierda de la igualdad se reduce a $(\beta - \alpha^2)\ddot{\theta}$, el primer, segundo y tercer sumando de la derecha se reducen a cero, al igual que el quinto, mientras que el cuarto queda como $\beta\theta$, el sexto queda $-\gamma\alpha u$ y el último queda igual al ser ya lineal.

Por otro lado, linealizando la segunda ecuación diferencial queda que $\Delta\ddot{\varphi} \rightarrow (\beta - \alpha^2)\ddot{\varphi}$, $\alpha\dot{\theta}^2\sin\theta \rightarrow 0$, $-\alpha\dot{\varphi}^2\sin\theta\, cos^2\theta \rightarrow 0$, $-2\dot{\theta}\dot{\varphi}\sin\theta\cos\theta \rightarrow 0$, $-\alpha\sin\theta\cos\theta \rightarrow -\alpha\theta$ y los dos términos restantes ya son lineales.

Por tanto, obtenemos la siguiente ecuación de movimiento linealizada entorno al punto de funcionamiento:

$$\begin{cases}(\beta-\alpha^2)\ddot{\theta}=\beta\,\theta-\gamma\,\alpha\,u-c_p\,\dot{\theta}\\(\beta-\alpha^2)\ddot{\varphi}=-\alpha\,\theta+\gamma\,u-c_a\,\dot{\varphi}\end{cases}$$

2.3.2. Obtención del modelo en espacio de estados

Una vez que tenemos las ecuaciones linealizadas de movimiento vamos a tratar de describirlas en espacio de estados, esto es, describirlas de la forma siguiente,

$$\dot{x}=A\,x+B\,u$$

El parámetro *x* es el espacio de estados, que en nuestro caso es,

$$x=[\varphi\quad\theta\quad\dot{\varphi}\quad\dot{\theta}]$$

La ecuación de movimiento, considerando que $\alpha=0.56$, $\beta=0.74$, $\gamma=2.37$, $c_p=0.5$ y $c_a=0.5$, queda como,

$$\begin{cases}\ddot{\theta}=1.737\,\theta-3.115\,u-1.174\,\dot{\theta}\\\ddot{\varphi}=-1.315\,\theta+5.563\,u-1.174\,\dot{\varphi}\end{cases}$$

Que escrito en espacio de estados es,

$$\begin{bmatrix}\dot{\varphi}\\\dot{\theta}\\\ddot{\varphi}\\\ddot{\theta}\end{bmatrix}=\begin{bmatrix}0&0&1&0\\0&0&0&1\\0&-1.315&-1.174&0\\0&1.737&0&-1.174\end{bmatrix}\begin{bmatrix}\varphi\\\theta\\\dot{\varphi}\\\dot{\theta}\end{bmatrix}+\begin{bmatrix}0\\0\\5.563\\-3.115\end{bmatrix}u$$

2.3.3. Uso de *Matlab* para la obtención del parámetro K en control LQR

El control LQR proporciona una ganancia óptima k, de forma que se minimiza un cierto criterio. El control es de la forma que se puede apreciar en la siguiente figura.

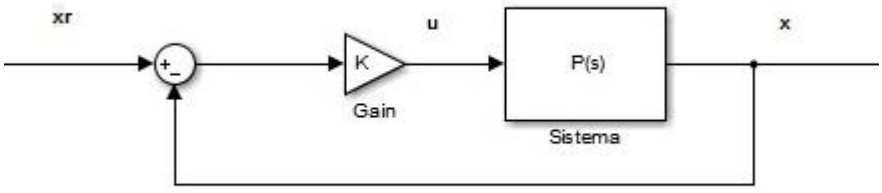

Figura 2-34. Control LQR.

La instrucción *lqr* de *Matlab* tiene la siguiente forma,

[K, S, E]=lqr(A, B, Q, R, N);

Y resuelve el control minimizando el siguiente criterio,

$$J=\int x'\,Q\,x+u'\,R\,u+2\,x'\,N\,u$$

Para nuestro caso,

$$A=\begin{bmatrix}0&0&1&0\\0&0&0&1\\0&-1.315&-1.174&0\\0&1.737&0&-1.174\end{bmatrix}$$

$$B=\begin{bmatrix}0\\0\\5.563\\-3.115\end{bmatrix}$$

Las matrices Q y R son de ponderación, en nuestro caso le asignaremos los siguientes valores, aunque se les podrían haber dado otros,

$$Q = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$R = 1$$

Por otro lado, N es un término que omitiremos dándolo el valor cero. Las salidas que nos genera la función son K (la gancia óptima), S (la solución de la ecuación asociada de Riccati) y E (los polos en bucle cerrado). Introduciendo pues estos valores, obtenemos la siguiente solución.

```
>> [K,S,E]=lqr(A,B,Q,R,N)

K =

  -1.0000   -6.6752   -0.8122   -3.8068



S =

   4.8311   11.3609    3.4233    6.4346
  11.3609   36.7543   10.1043   20.1879
   3.4233   10.1043    3.0609    5.7271
   6.4346   20.1879    5.7271   11.4499



E =

  -6.5615
  -1.5968
  -0.5331
  -0.9966
```

Por tanto la ganancia óptima es,

$$K = [-1 \quad -6.6752 \quad -0.8122 \quad -3.8068]$$

En el siguiente apartado probaremos los resultados de este controlador en nuestro modelo en *Simulink.*

2.3.4. Implementación del control LQR en el modelo en *Simulink* del péndulo de Furuta

En este apartado se verá como implementar el control LQR en nuestro modelo de *Simulink*® sobre el cual hemos de realizar algunas modificaciones al trabajar ahora en espacio de estados.

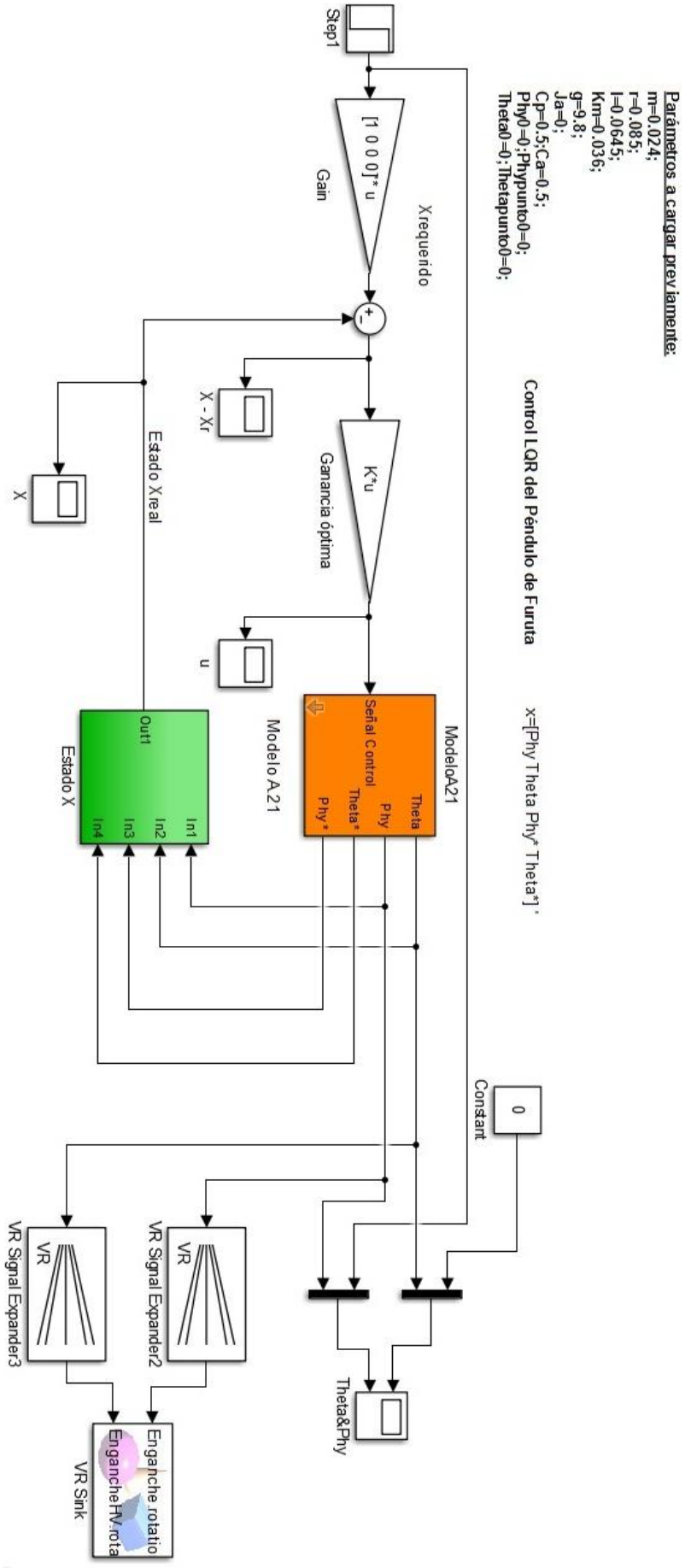

Figura 2-35. Control LQR modelo en Simulink®

En la figura 2-35 se puede ver como la referencia es el ángulo φ, un escalón. También se probará con la referencia del ángulo φ constante igual a cero.

Las ganancias que se han introducido no son como las habituales, son de tipo *Matrix (k*u)*. La primera ganancia contiene lo que se puede apreciar en la figura 2-36, en ella se puede ver como la entrada *u* se multiplica por un vector con un uno en la primera componente y cero en las demás de forma que tras multiplicarse por dicha ganancia la entrada, el resultado es un vector de cuatro filas y una columna donde el primer componente es el ángulo φ de referencia y los demás cero.

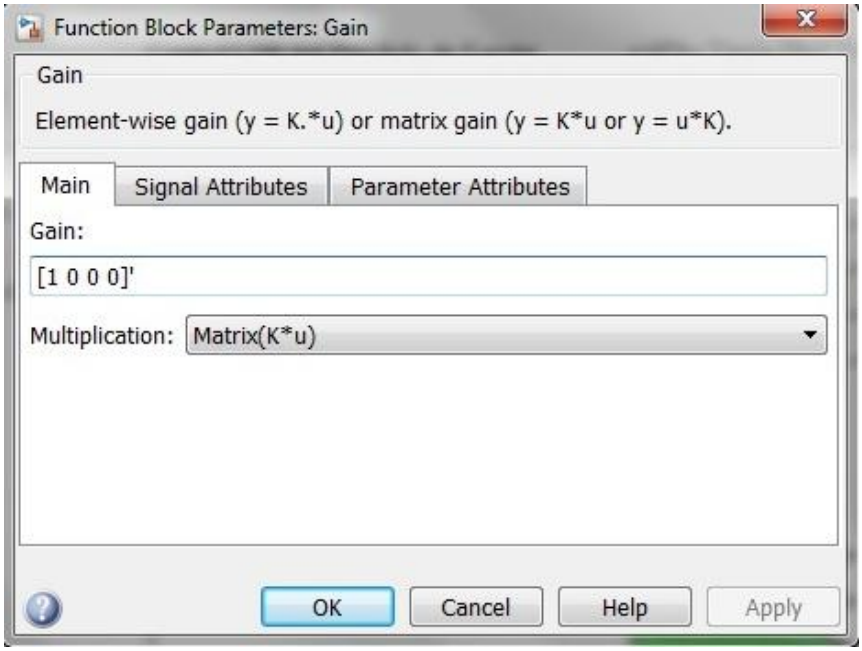

Figura 2-36. Parámetros de la ganacia primera del control LQR.

La segunda ganancia contiene lo que se puede apreciar en la figura 2-37, esto es, la *K* óptima obtenida previamente. Dicha ganancia óptima de dimensiones 1x4 se multiplica por un vector de entrada 4x1, el error de los estados, dando como resultado la tensión de entrada.

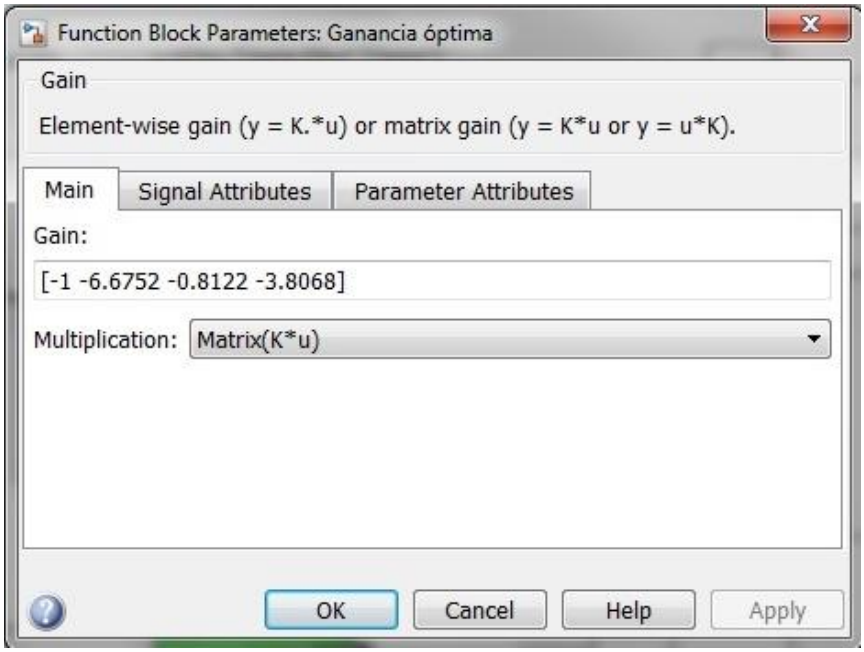

Figura 2-37. Parámetros de la ganancia segunda del control LQR.

En la figura 2-35 también se puede apreciar un bloque verde que genera el estado real a partir de la salida del sistema, este bloque simplemente construye el vector de estados usando ganancias de la misma forma que la explicada anteriormente en la figura 2-26. El contenido de dicho bloque se puede apreciar en la figura 2-38.

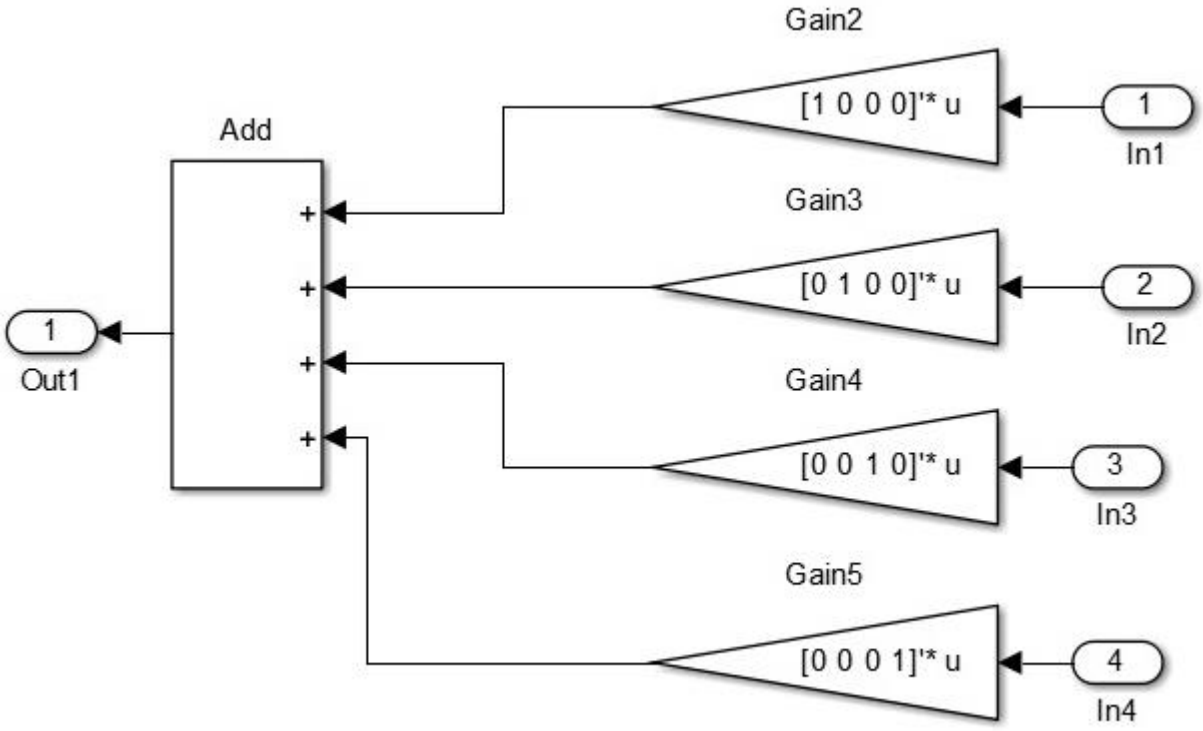

Figura 2-38. Contenido del bloque *Estado X*.

Ya solo nos queda probar los resultados del controlador ante diversas situaciones, esto es lo que se hará en el siguiente apartado.

2.3.5. Resultados del control LQR sobre el modelo del péndulo de Furuta

En primer lugar, vamos a comprobar el comportamiento de nuestro controlador si introducimos como referencia un escalón del ángulo φ del brazo del péndulo hasta un punto distinto del de funcionamiento, que es para donde debería controlar nuestro sistema ya que se linealizó entorno a ese punto y se obtuvo el *K* óptimo considerando ese punto.

Usando como referencia, del ángulo del brazo del péndulo, un escalón que va de cero a pi/2 radianes produciéndose el salto entre ambos valores a partir de cinco, y con la K óptima obtenida previamente obtenemos la respuesta que se puede ver a continuación.

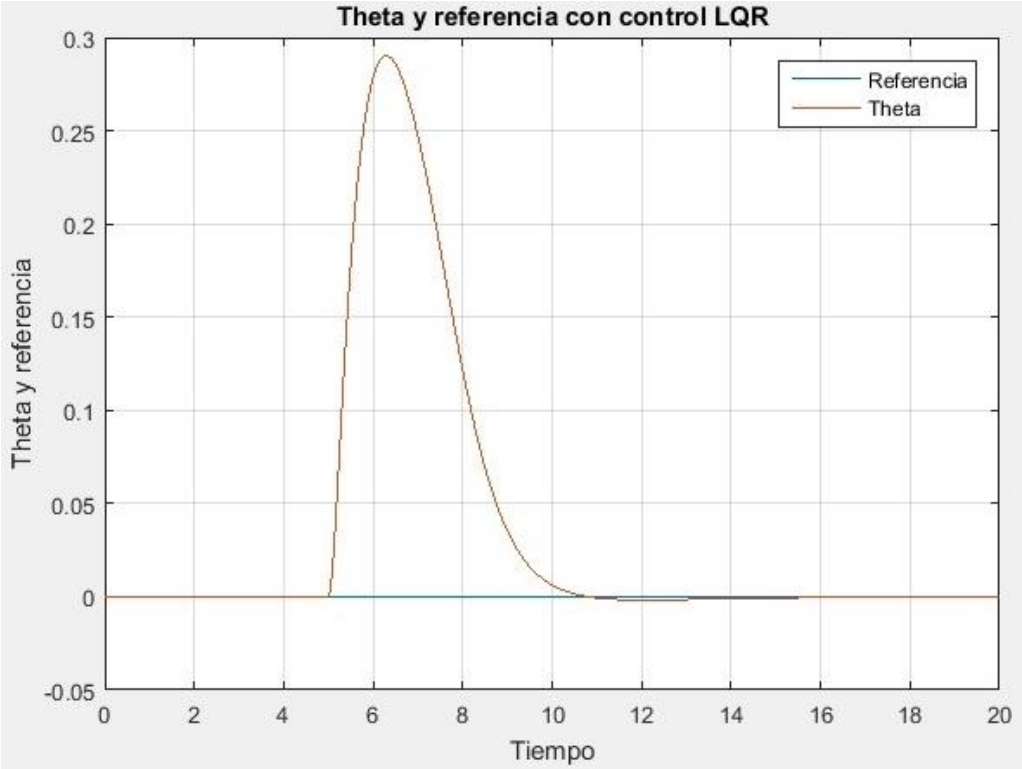

Figura 2-39. Ángulo theta en control LQR.

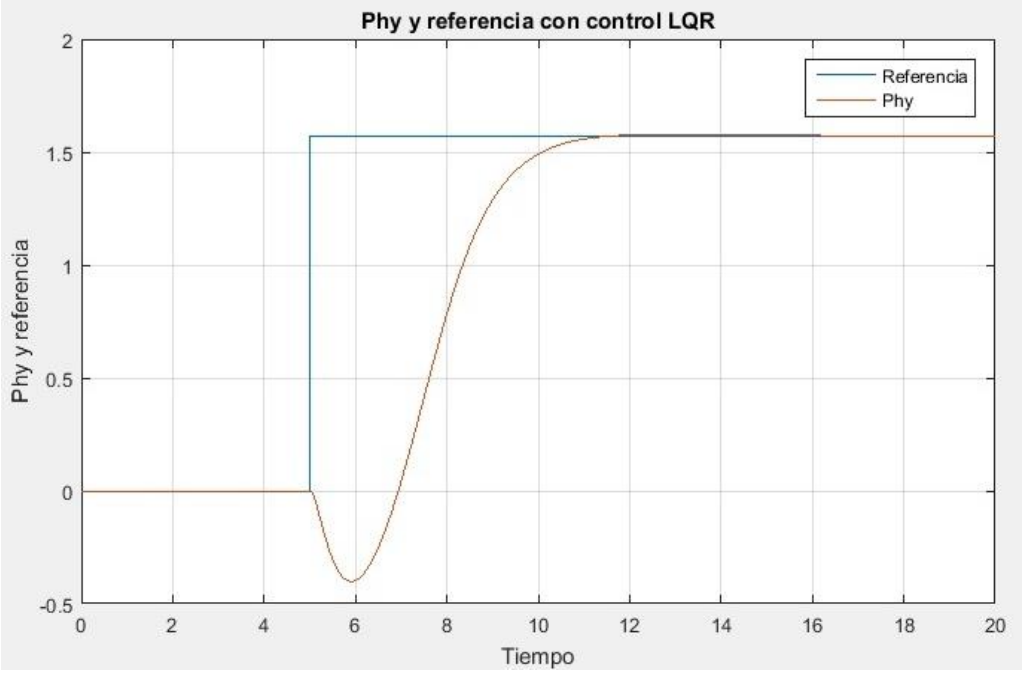

Figura 2-40. Ángulo phy en el control LQR.

Como se puede apreciar, el control es capaz de controlar el péndulo incluso en posiciones alejadas del punto de funcionamiento, también es interesante apreciar la entrada que aplica nuestro sistema con la técnica de control LQR.

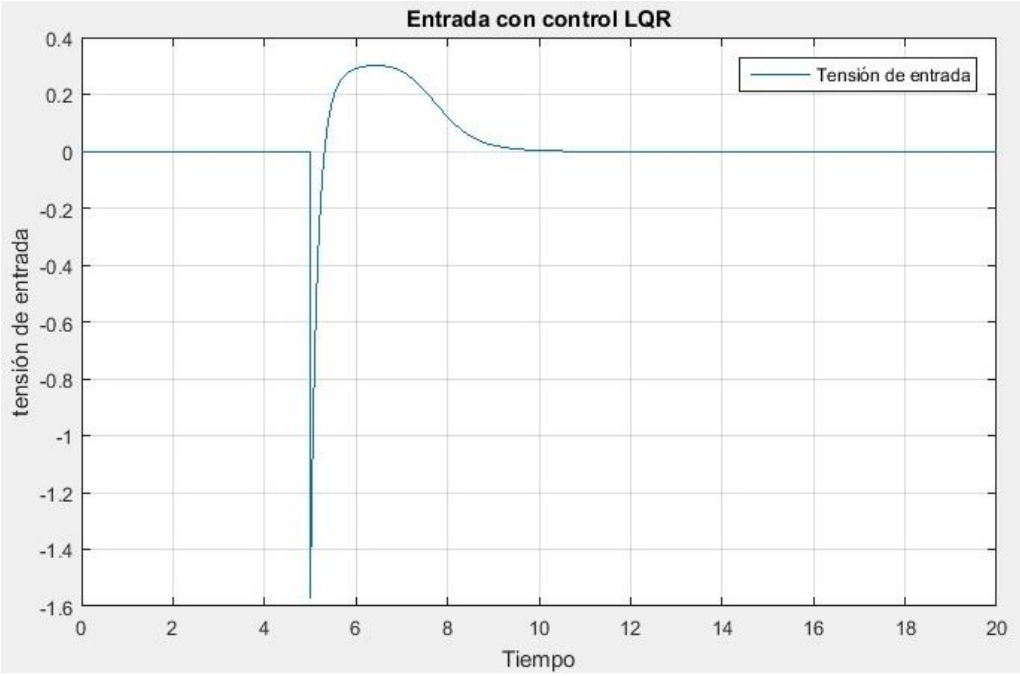

Figura 2-41. Entrada al sistema proporcionada por el control LQR.

Si en lugar de controlar alejados del punto de funcionamiento intentamos controlar entorno a él pero partiendo con que la condición inicial del ángulo del péndulo es $\theta_0 = 0.2$ obtenemos el siguiente resultado.

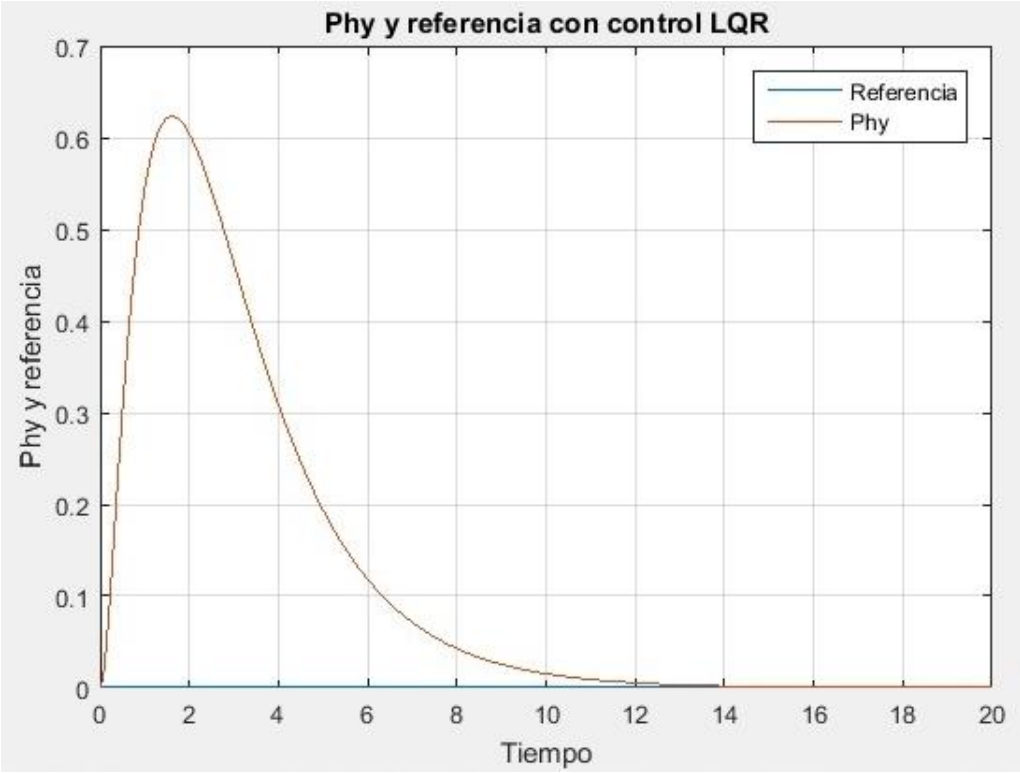

Figura 2-42. Ángulo *phy* en control LQR para $\theta_0 = 0.2$.

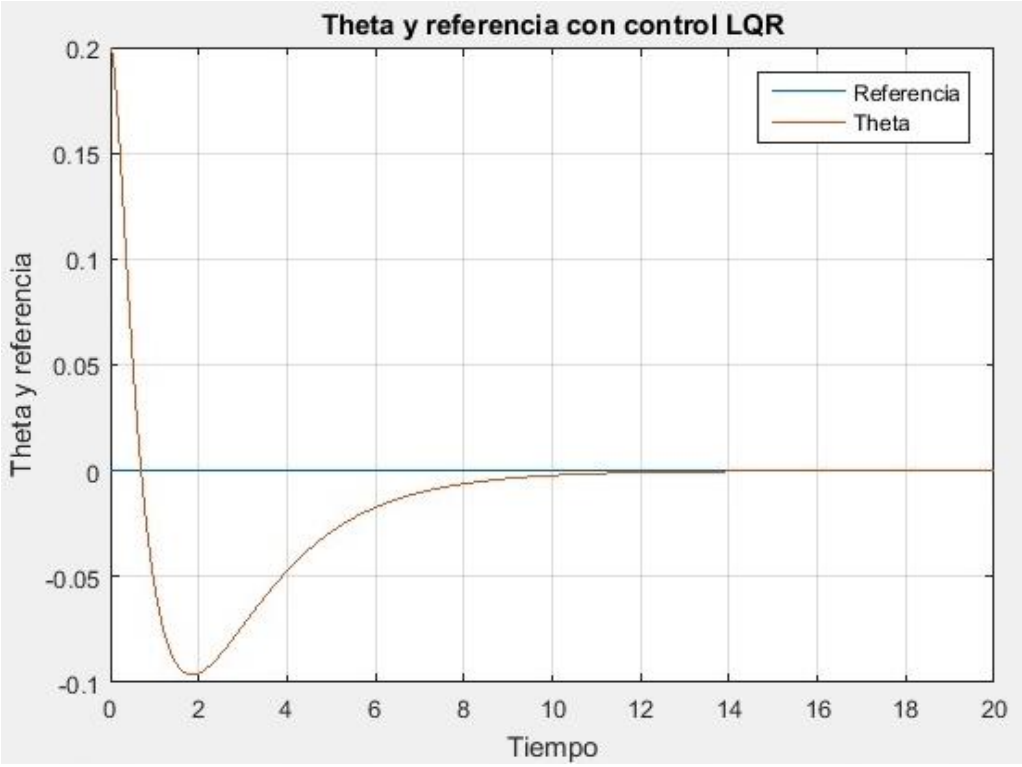

Figura 2-43. Ángulo Theta en control LQR para $\theta_0 = 0.2$.

Las respuestas que hemos estado viendo podría decirse que son buenas ya que son capaces de controlar bien nuestro péndulo incluso en zonas alejadas del punto de funcionamiento para el que se obtuvo la ganancia óptima.

Ahora vamos a intentar comprender cualitativamente como influyen los parámetros de ponderación, para ello vamos a tomar distintos valores de la matriz Q y del parámetro R y vamos a obtener la K óptima correspondiente y vamos a ver los resultados.

1) Si $Q = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 5 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$ y R=1 obtenemos usando *Matlab* que $K = [-1 \quad -7.1333 \quad -0.8143 \quad -3.8638]$.

2) Si $Q = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$ y R=5 obtenemos usando *Matlab* que $K = [-0.4472 \quad -3.4922 \quad -0.3665 \quad -1.9236]$.

A continuación se muestran como varían las respuestas de los ángulos *theta* y *phy* para las tres ganancias óptimas que se han obtenido en el caso de querer controlar el péndulo cuando $\theta_0 = 0.2$.

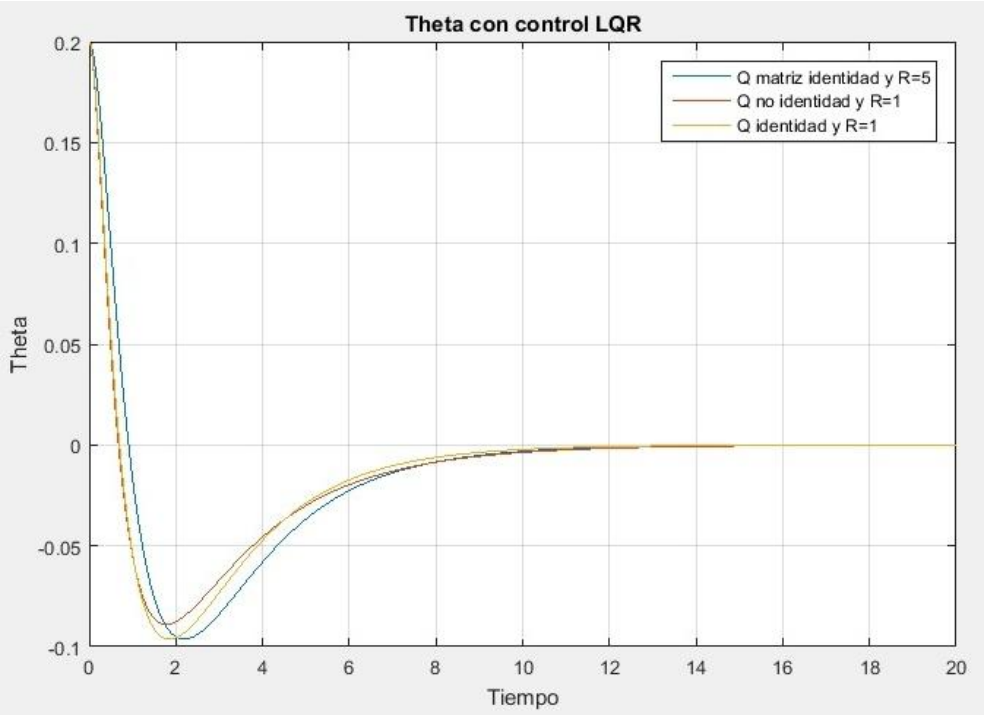

Figura 2-44. Ángulo theta para distintos valores de los parámetros de ponderación.

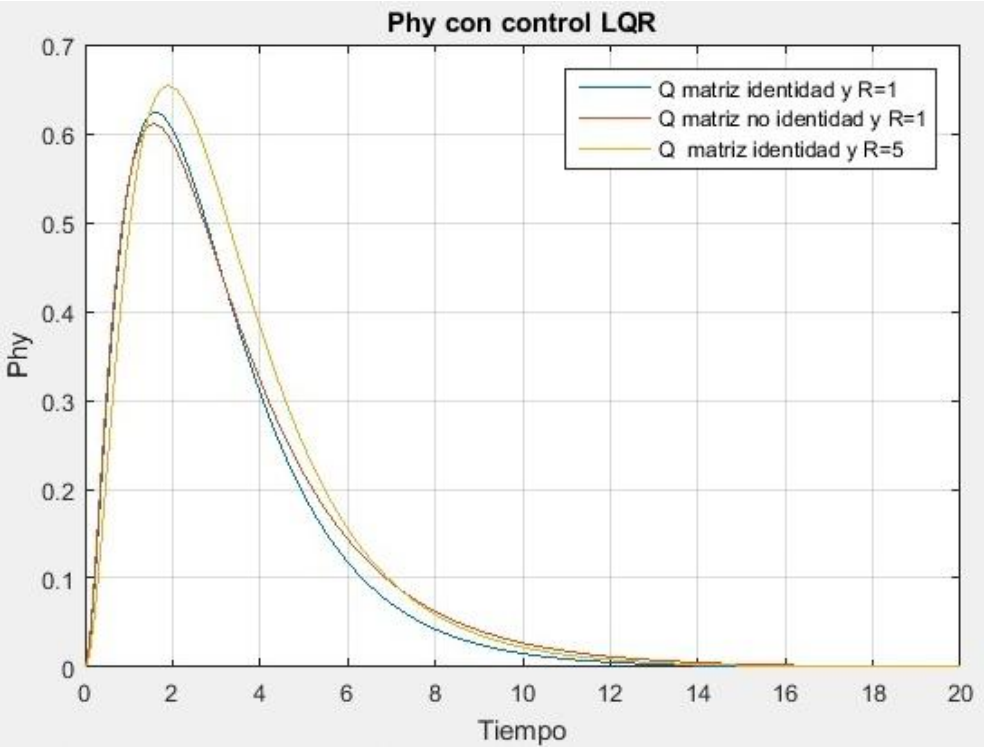

Figura 2-45. Ángulo phy para distintos valores de los parámetros de ponderación.

En función de los resultados obtenidos, se puede decir que comparando las respuestas entre Q matriz identidad con R=1 y Q matriz no identidad, estas son similares en la respuesta que dan en *phy* y en *theta*, teniendo un pico menor la última, lo cual es lógico ya que la respuesta de theta tiene que ser más próxima al punto de funcionamiento ya que la matriz Q tiene un mayor valor de ponderación en la segunda coordenada del espacio de estados, es decir, theta. La respuesta dada cuando cambiamos a R=5 muestra un mayor pico.

2.4. Swing-Up

Hasta este momento hemos estado controlando el péndulo de Furuta cerca del punto de funcionamiento y partiendo de que estamos en una posición cercana al punto de funcionamiento, es decir, con el péndulo casi en posición vertical hacia arriba. Pero, ¿qué sucede si tenemos el péndulo en posición vertical hacia abajo?, ¿cómo llevar el péndulo a una zona controlable por las técnicas vistas anteriormente, esto es, cerca de la posición de equilibrio?

Está claro que si nos imaginamos el péndulo colgando hacia abajo y tenemos un balance de control PD, como el del apartado 2.2., para que el sistema llegue a la posición deseada, vertical hacia arriba, este controlador lo tendrá francamente difícil por no decir imposible, ya que la acción de control es la suma de un término proporcional al error y otro que depende de la variación del error, en el momento que el péndulo comienza a girar el error va disminuyendo (gracias a una actuación que es suma de un término proporcional al error, que va menguando al tener menos error, y un término que va aumentando al haber una variación del error ya que se mueve el péndulo), pero esto será hasta que el péndulo se coloque en posición horizontal, a partir de ese momento se puede intuir que por mucho que aumentara la acción de control el péndulo seguiría en posición horizontal, y cambiar el sentido de giro no es algo que el controlador vaya a hacer porque para ello tendría que cambiar el signo del error o la variación del mismo, que será cero si se mantiene en la posición horizontal, algo que obviamente no sucede. Realmente se podría decir que estamos ante un problema bastante complicado.

El objetivo va a ser intentar llevar el péndulo desde la posición vertical hacia abajo, o posiciones próximas a estas, a una zona donde un control LQR o un balance de control PD ya puedan controlar el péndulo, ¿pero cómo conseguir que el péndulo llegue a zonas controlables? La respuesta es un control que no se base en la posición del péndulo sino en su energía mecánica, esto es, la suma de la energía cinética y potencial del mismo. Definiremos que para $\theta=\pi$, posición vertical hacia abajo, la energía potencial es cero y cuando esté en $\theta=0$ la energía potencial será máxima. Esto es lo que se conoce como *swing-up*.

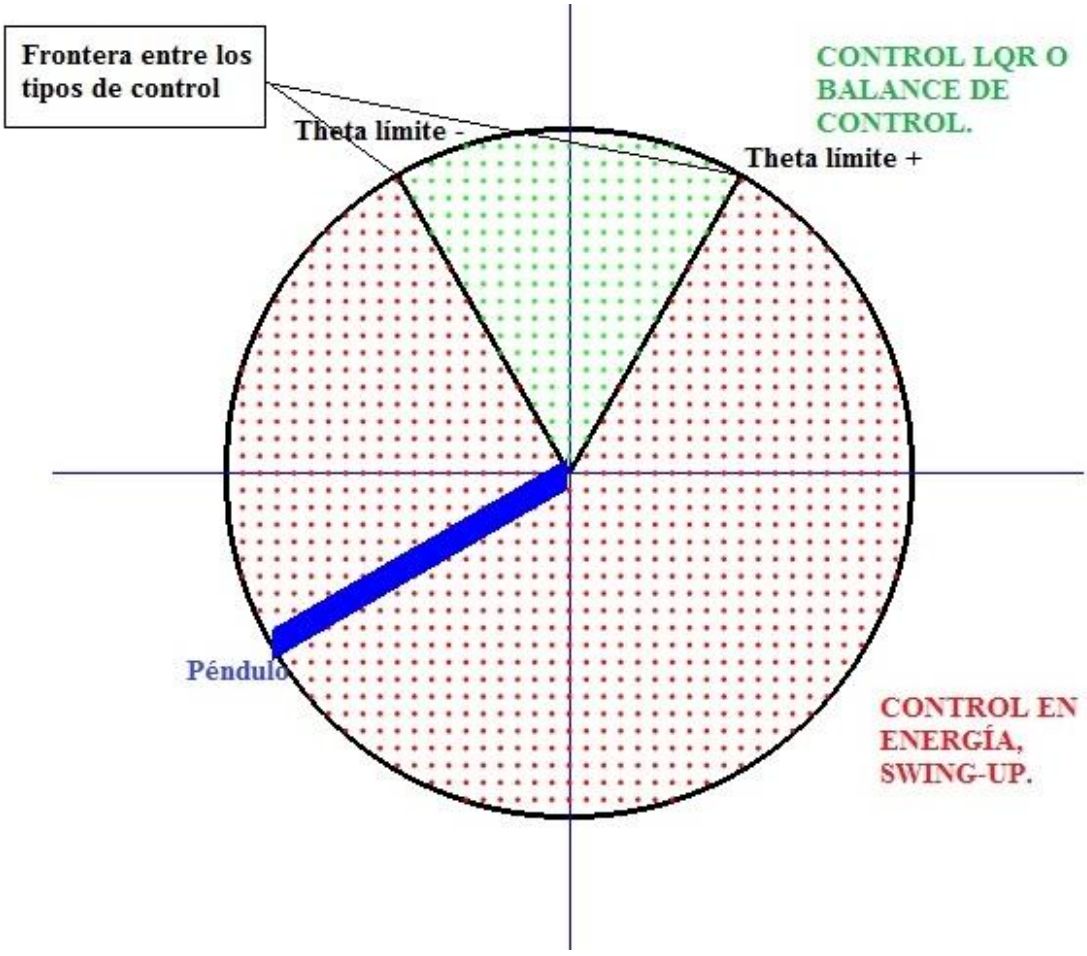

Figura 2-46. Control en energía y control en posición.

En la figura 2-46 se representa, en rojo, la zona donde se realiza un control en energía y la zona, en verde, donde se realiza un control en posición LQR o balance de control PD. Es necesario un tipo de interruptor que cambie de un tipo de control a otro en función de la posición del péndulo.

2.4.1. Control en energía en posiciones alejadas del punto de funcionamiento

Para realizar un control en energía debemos calcular en todo momento la energía del péndulo, cómo hacerlo junto a qué ley de control implementar son los temas que se tratan en este apartado.

La energía del péndulo es,

$$V = m_p\ g\ \frac{l_p}{2}\ (1 + \cos(\theta))$$

$$T = \frac{1}{2}\ J_p \dot{\theta}^2$$

$$E = T + V$$

Donde m_p es la masa, l_p es la longitud y J_p es el momento de inercia del péndulo supuesto con densidad uniforme. Luego ya podemos hacer un modelo en *Simulink* que nos proporcione la energía total a partir de la posición del péndulo y de su velocidad.

Considerando que la masa del péndulo es 0.024 kg y su longitud es 0.129m y considerando que la gravedad es 9,8 m/s^2 podemos calcular que,

$$m_p\ g\ \frac{l_p}{2} = 0.024 \cdot 9.8 \cdot \frac{0.129}{2} = 0.0151704\ \frac{kg\ m^2}{s^2}$$

$$J_p = \frac{4}{3}\ m_p\ {l_p}^2 = 0.0005325\ kg\ m^2$$

El modelo creado para obtener la energía mecánica total es el siguiente,

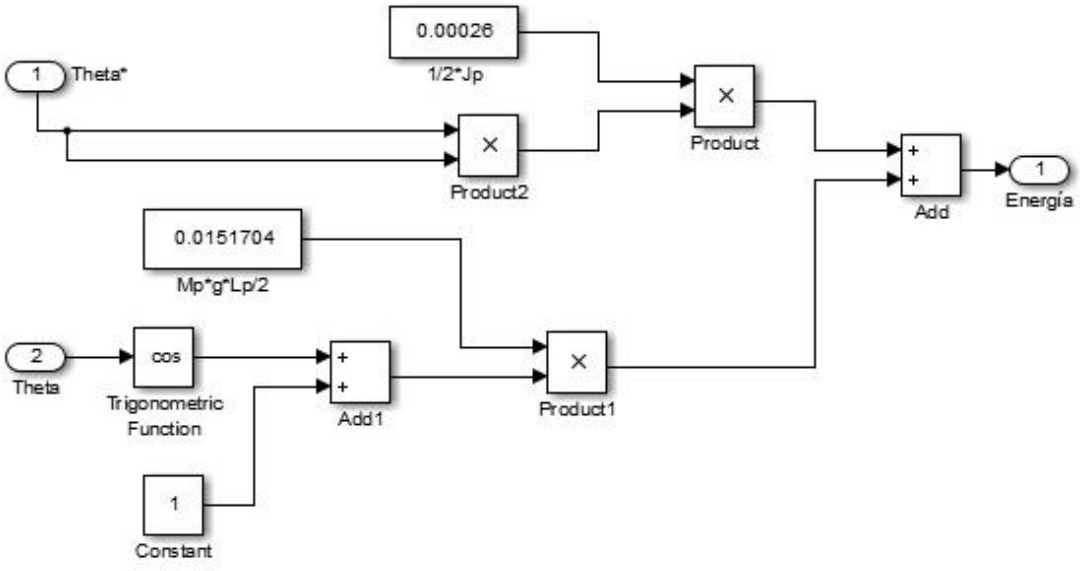

Figura 2-47. Cálculo de la energía mecánica del péndulo.

Este modelo se encuentra dentro de un subsistema,

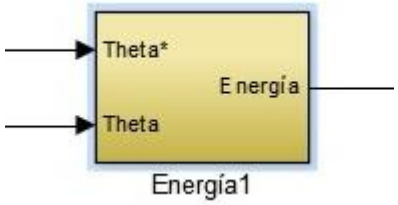

Figura 2-48. Subsistema de cálculo de la energía.

La energía de referencia para el control en energía será aquella que tiene el péndulo cuando este se encuentra en la posición deseada, esto es, en posición vertical hacia arriba, $\theta = 0$, y sin velocidad, $\dot{\theta} = 0$.

$$E_{ref} = m_p\ g\ l_p = 0.0303408\ \frac{kg\ m^2}{s^2}$$

La ley de control que se va a usar en el swing-up es la siguiente,

$$u = K\ \left(E_{ref} - E\right) sign(\cos(\theta)\ \dot{\theta})$$

Donde *sign* es la función signo, que vale uno cuando $\cos(\theta)\ \dot{\theta}$ es positivo, menos uno cuando es negativo y cero cuando es cero. Por tanto se podría decir que la ley de control es un controlador proporcional pero con algo más, la función signo. La ganancia K= -50 y se ha obtenido experimentalmente.

Grosso modo, con la función signo lo que conseguimos es que cambie el sentido de la actuación cada vez que el péndulo pasa por la posición horizontal.

2.4.2. Implementación del *swing-up* en *Simulink*

Ahora vamos a implementar el control completo del péndulo, que va a constar de un *swing-up* para que cuando estemos lejos del punto de funcionamiento nos lleve a una zona próxima y de un control LQR para cuando estemos en una zona controlable cerca del punto de funcionamiento.

Para pasar de un control al otro se va a implementar un interruptor que pase de uno a otro tipo de control en función de la energía, el valor de energía a partir del cual vamos de uno a otro tipo de control es 0.029 $\frac{kg\ m^2}{s^2}$ que correspondería a un valor de $\theta = \pm 24.27º$ si el péndulo estuviera quieto, es decir, sin energía cinética. Este valor se ha obtenido experimentalmente ya que sabemos del apartado 2.2.2.3. que para $\theta \in [-30º,\ 30º]$ el control LQR es capaz de controlar correctamente (ya que el balance de control PD es capaz de controlar), pero como la energía del péndulo también incluye energía cinética podría ser que tuviéramos una energía mayor que la que corresponde a la energía potencial del péndulo en theta igual a 30º (ó -30º), pero no porque estuviéramos en ese ángulo ni cerca, sino porque el movimiento del péndulo le añade energía cinética. De este modo con unos ángulos menores que treinta grados, tendremos mayor energía potencial (ya que estamos a mayor altura) colocándonos del lado de la seguridad si tenemos también energía cinética debido al movimiento.

Una vez que entre el control LQR, ya estamos en un ángulo *theta* próximo al valor de funcionamiento pero el ángulo *phy* podría encontrarse en cualquier posición alejada de la del punto de funcionamiento para la cual se obtuvo la ganancia óptima, esto podría provocar que el controlador LQR fuera incapaz de controlar el péndulo al partir de posiciones de todas las variables de estado alejadas del punto de funcionamiento. Para evitar esto hemos colocado un interruptor de forma que una vez que se cambia del *swing-up* al control LQR se guarde el valor del ángulo *phy* en ese momento y ese sea la referencia del ángulo del brazo.

A continuación se muestran los diagramas en *Simulink* del control LQR y *swing-up*.

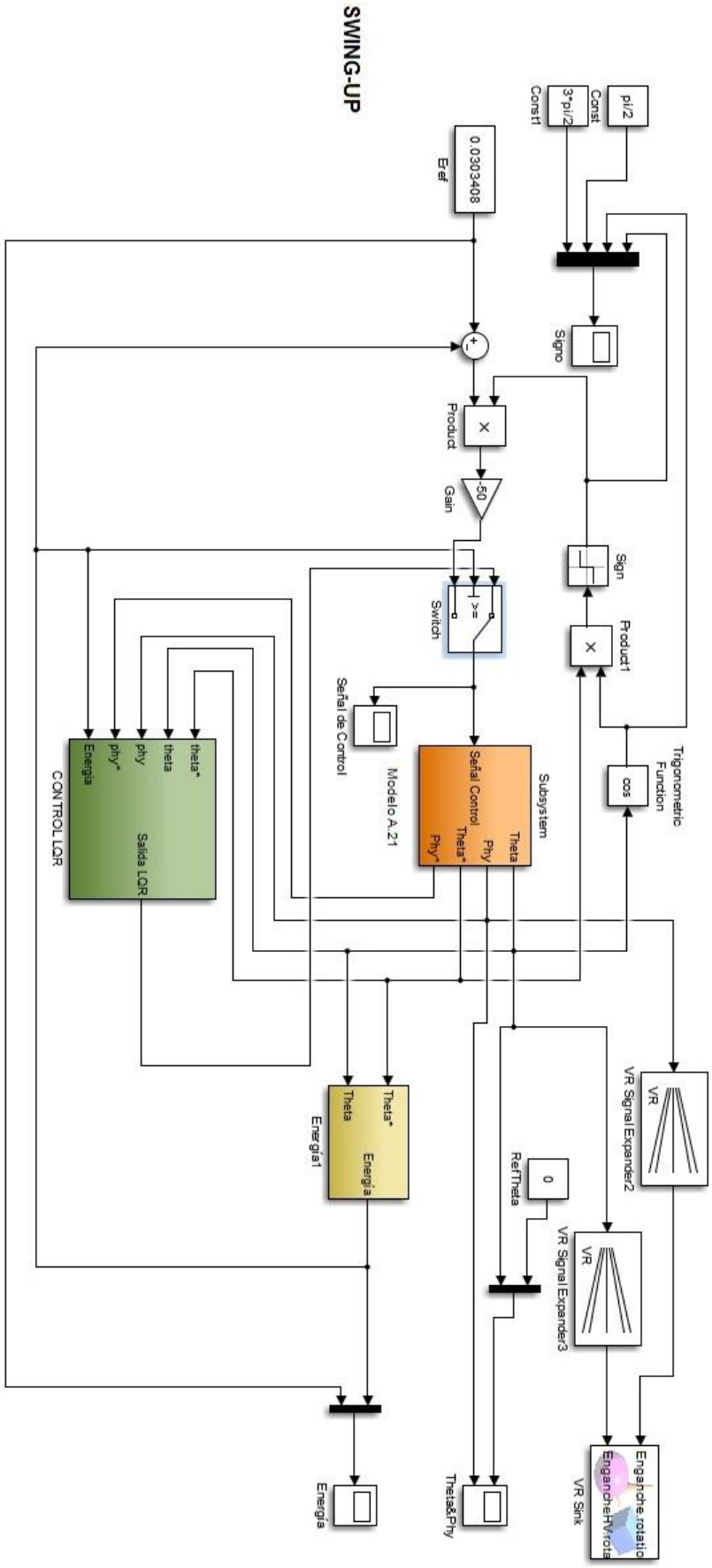

Figura 2-49. Modelo *swing-up* en *Simulink.*

Los subsistemas del modelo A21 y de la energía ya se han presentado previamente y son exactamente iguales, en cambio el subsistema del control LQR se ha modificado al colocarle el interruptor que se comentó anteriormente.

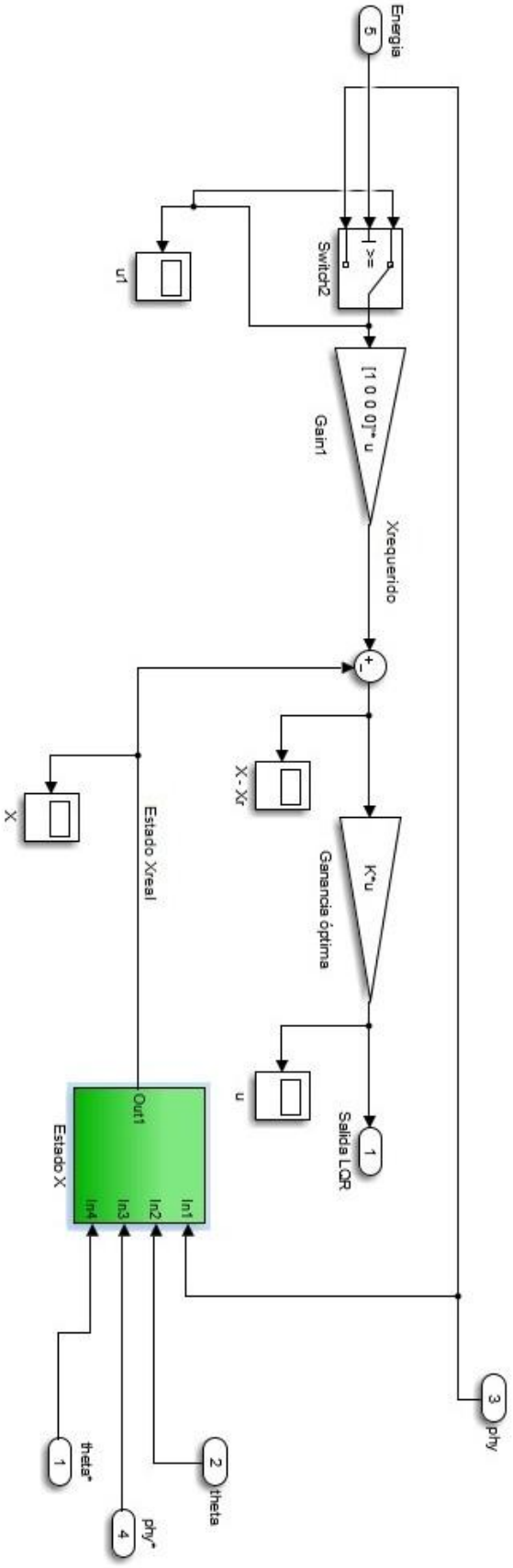

Figura 2-50. Control LQR con interruptor.

Como se puede ver, en el momento que llegamos a la energía a partir de la cual se entra en funcionamiento el LQR se guarda ese ángulo *phy* y es el que se usa como referencia, anteriormente también se está calculando la acción de control del LQR, pero esta no se aplica debido al otro interruptor.

Los contenidos de los bloques interruptores son los siguientes,

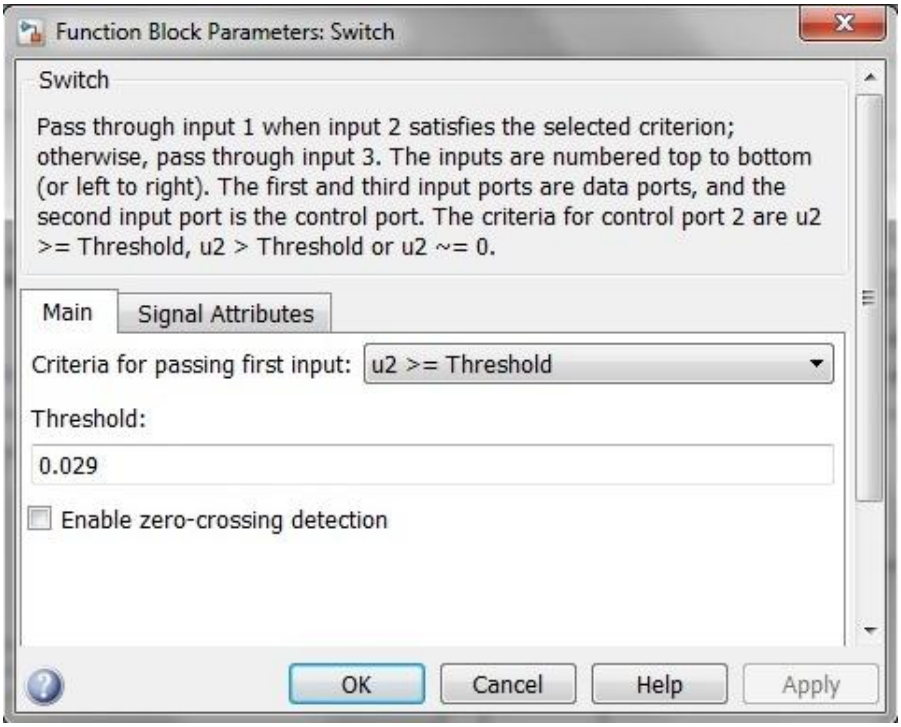

Figura 2-51. Bloque interruptor.

2.4.3. Resultados obtenidos con el *swing-up*

Si partimos con el péndulo parado en posición vertical hacia abajo obtenemos la siguiente respuesta al controlarlo.

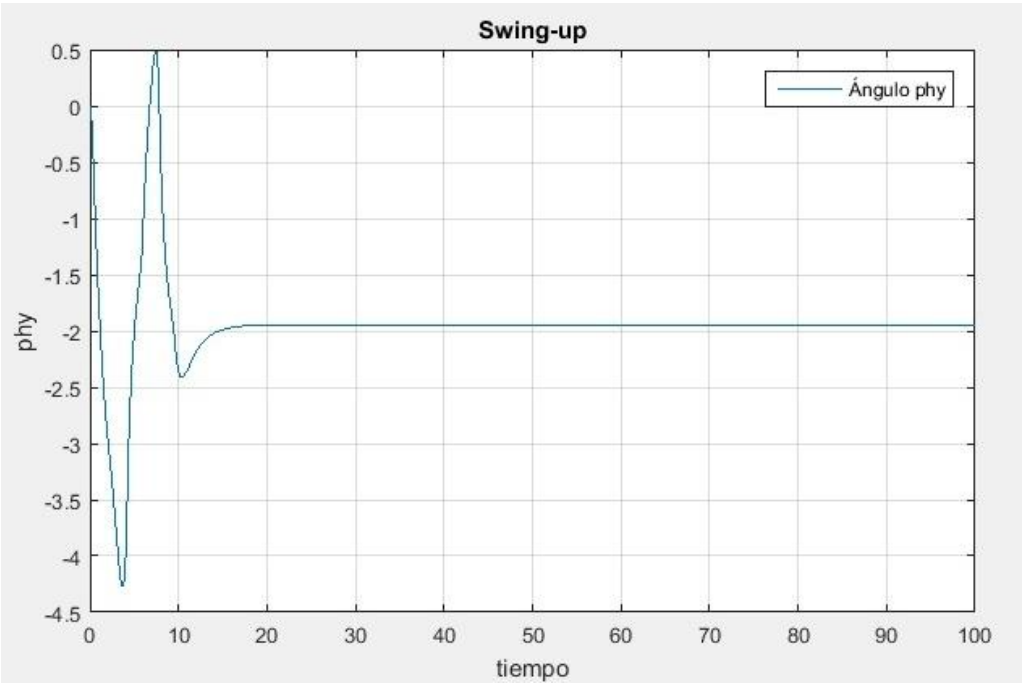

Figura 2-52. Ángulo *phy* control *swing-up*.

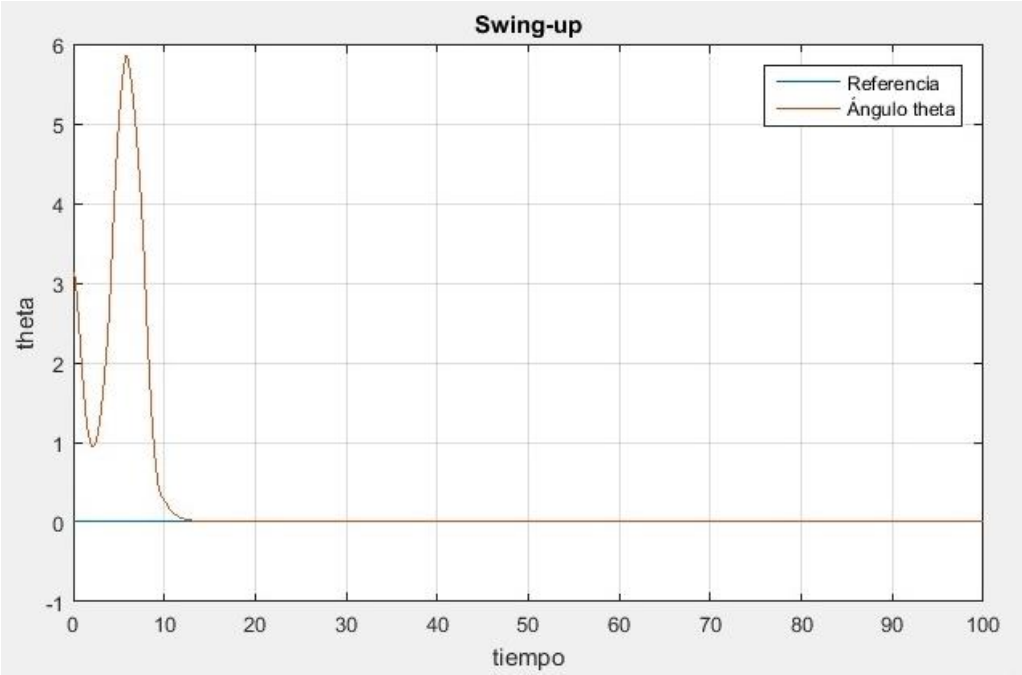

Figura 2-53. Ángulo *theta* control *swing-up.*

La energía del péndulo es la que se muestra a continuación, donde se puede ver como se alcanza la referencia de energía.

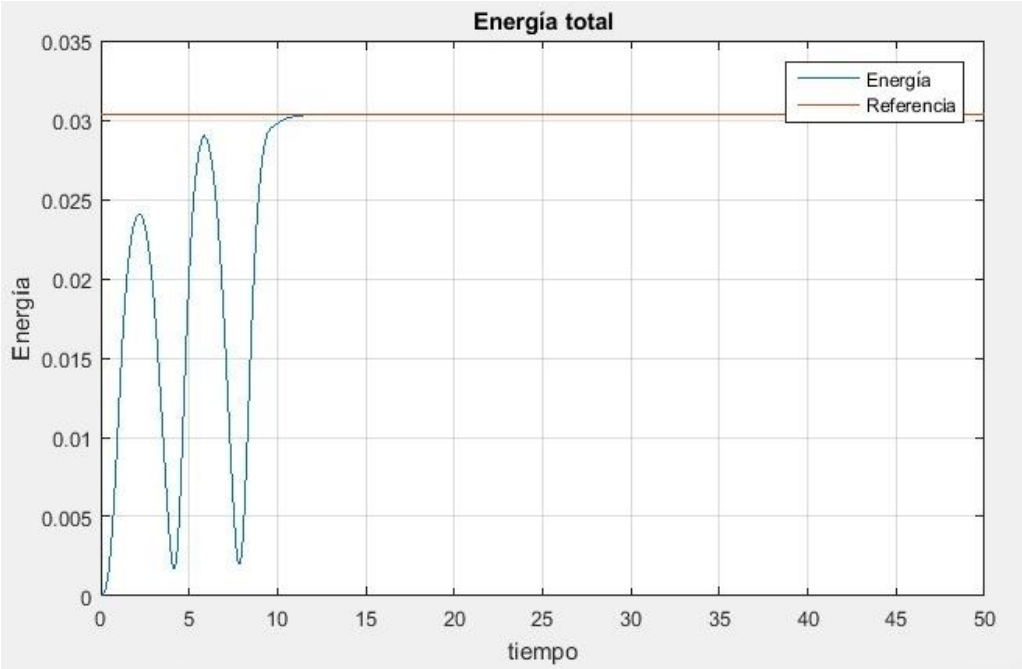

Figura 2-54. Energía del péndulo.

Si partimos de otras condiciones, por ejemplo $\theta_0 = \pi/2$ obtenemos la siguiente respuesta,

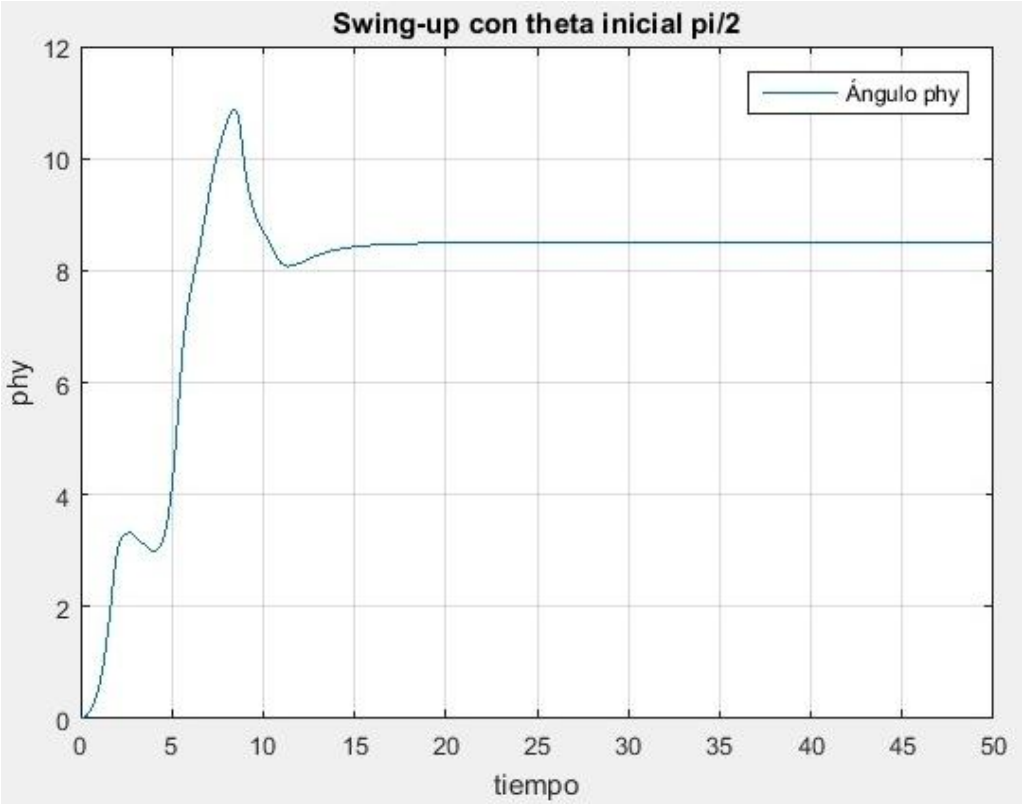

Figura 2-55. Ángulo *phy* en *swing-up* con $\theta_0 = \pi/2$.

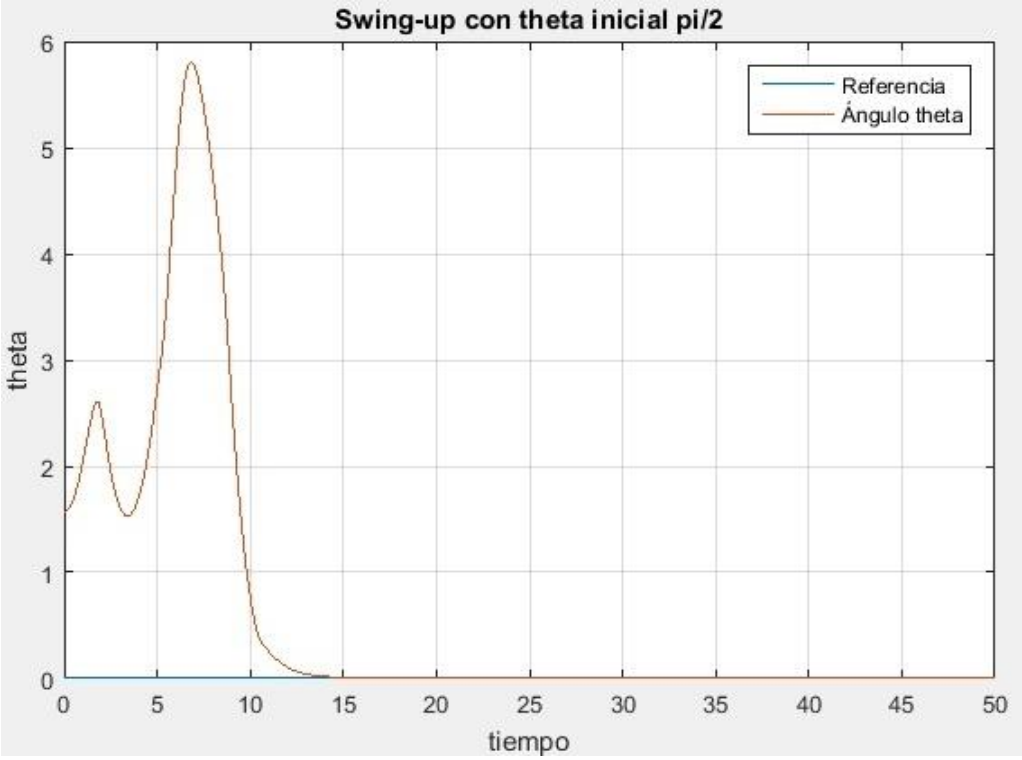

Figura 2-56. Ángulo *theta* en *swing-up* con $\theta_0 = \pi/2$.

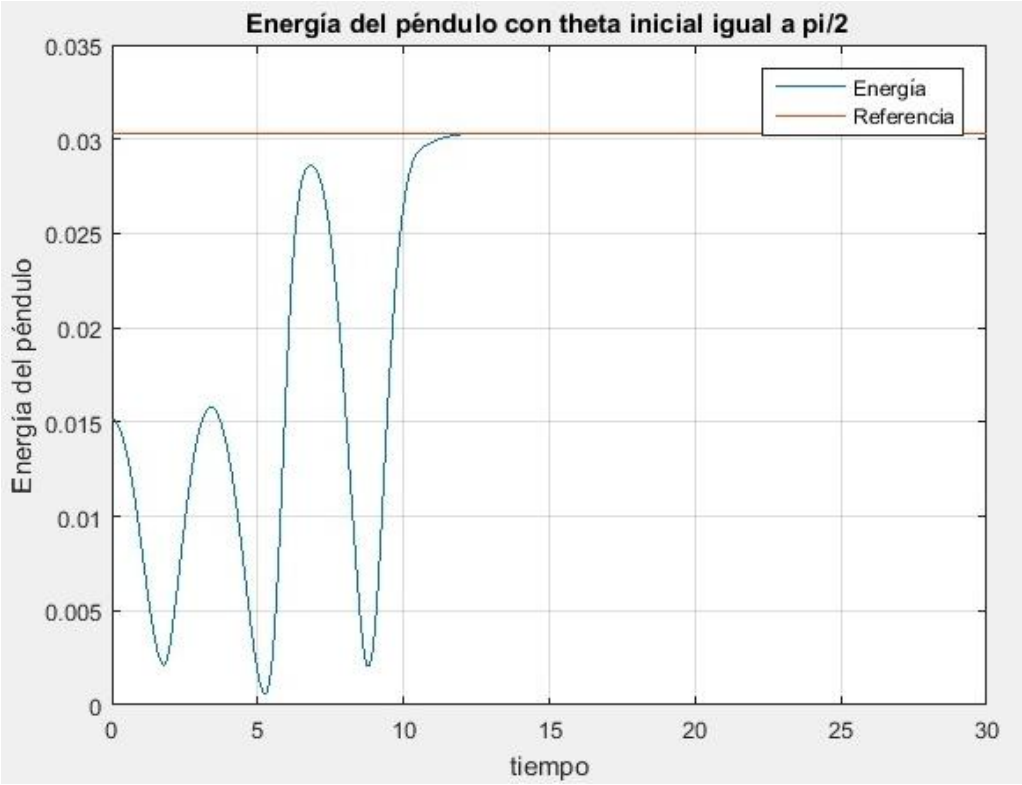

Figura 2-57. Energía en *swing-up* con $\theta_0 = \pi/2$.

En conclusión se puede decir que la respuesta del *swing-up* es ciertamente aceptable siendo capaz de controlar el péndulo partiendo desde cualquier condición inicial con unos buenos resultados.

3 IDENTIFICACIÓN Y CONTROL DEL PÉNDULO REAL DE QUBE-SERVO

En cada rama del conocimiento, el progreso es proporcional a la cantidad de hechos sobre los que se construye, y por tanto a la facilidad de obtención de datos.

- James Clerk Maxwell -

En este apartado se comienza a trabajar con el péndulo *Qube-Servo*, en primer lugar se va a presentar el péndulo con sus componentes, se va a ver el modelo en *Simulink* que usaremos y también como poner en marcha el péndulo, posteriormente se tratará de identificar los parámetros del péndulo a través de distintas técnicas como la identificación a un paso, identificación a N pasos o incluso técnicas de ingeniería inversa a partir de valores proporcionados por el fabricante, finalmente probaremos un controlador obtenido previa identificación del sistema y veremos su resultado.

2.1. El péndulo de Furuta Qube-Servo

El péndulo de *Qube-Servo* que trataremos de controlar es el que se ve en la figura 3-1, como se puede apreciar consta de un péndulo conectado a un *encoder* a través de un brazo, a diferencia del modelo con el que se ha venido trabajando, este péndulo tiene la posición $\theta = 0$ radianes cuando se encuentra vertical hacia abajo y $\theta = \pi$ radianes cuando está hacia arriba, esto deberá tenerse en cuenta a la hora de identificar y controlar el sistema.

Otro detalle importante de este péndulo es que debido a que el *encoder* está conectado a la base a través de un cable, el brazo del péndulo no podrá realizar giros completos sobre sí mismo, algo que el modelo con el que hemos venido trabajando puede hacer sin problemas ya que no se le ha impuesto esa limitación.

Los datos del estado del péndulo así como la actuación pasan a través de una conexión *USB* desde nuestro sistema al ordenador y viceversa, estos datos son tratados directamente en *Simulink* a través de unos bloques proporcionados por el fabricante del péndulo.

Figura 3-1. Péndulo *Qube-Servo*.

En el modelo en *Simulink* que será necesario usar encontramos una serie de bloques, en la figura 3-2 podemos ver un bloque, el *encoder*, que cuenta las pulsaciones y nos indica cual es el ángulo *theta* y *phy* del péndulo y el brazo respectivamente. En el modelo proporcionado por el fabricante llaman *alpha* al ángulo que se ha venido llamando *theta* y *theta* al ángulo que se ha venido llamando *phy*.

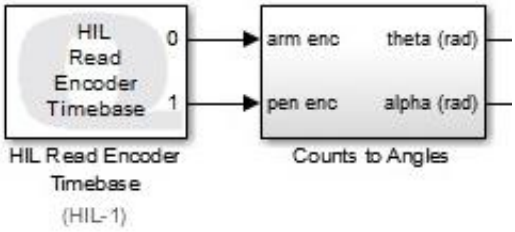

Figura 3-2. Bloque del encoder del péndulo *QUBE-Servo.*

Otro bloque importante es el bloque de actuación, es decir, el bloque al que le mandamos la señal de tensión necesaria para controlar el péndulo. Este bloque se muestra en la figura 3-3.

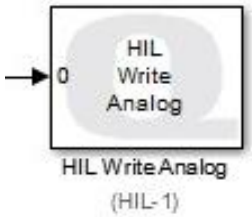

Figura 3-3. Bloque de actuación del péndulo *QUBE-Servo*.

Por tanto, los bloques anteriores son la salida del sistema real, es decir los ángulos, y la entrada al sistema real, la tensión que aplicamos al motor.

Es imprescindible poner en el modelo el bloque de la figura 3-4, puesto que es en el donde seleccionamos las características del aparato que estamos usando.

Figura 3-4. Bloque imprescindible para usar dispositivos *Quarc*.

Si clicamos sobre este último bloque podemos seleccionar las características del dispositivo usado, véase la figura 3-5, aunque normalmente ya está bien por defecto al introducir el péndulo.

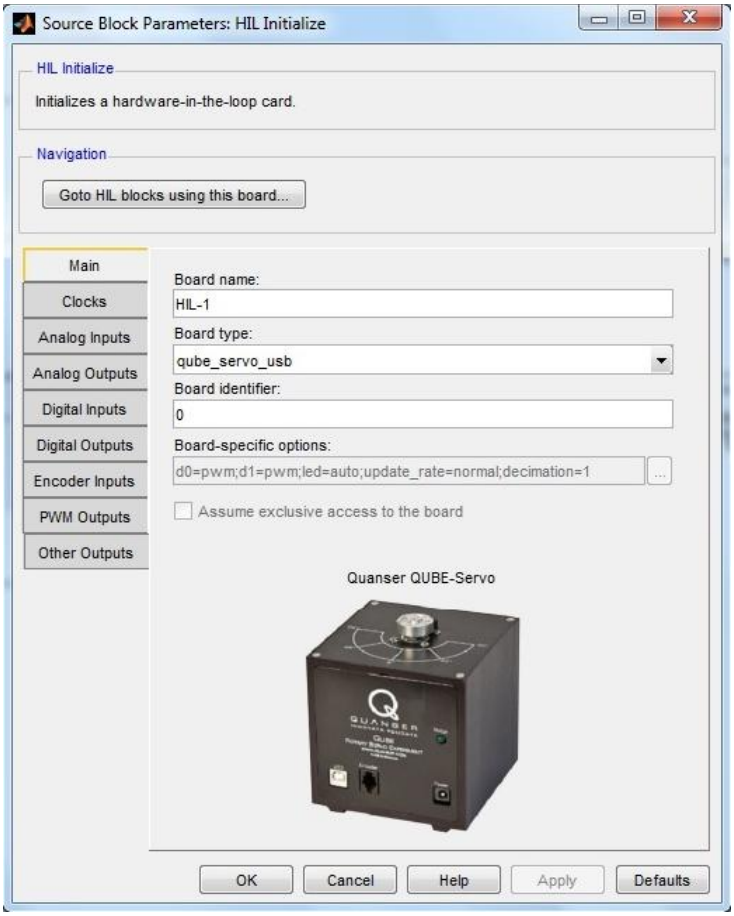

Figura 3-5. Parámetros del aparato de *Quarc*.

Una vez que ya tenemos un modelo montado, tenemos que compilarlo, pero la compilación no es como se hace normalmente en un modelo de *Simulink* que no está conectado con un sistema real, a continuación se indican los tres pasos necesarios para realizar la compilación.

1) En las pestañas superiores de la ventana donde está el modelo de *Simulink*, nos vamos a *QUARC* y seleccionamos *Set default options.*

2) Tras el paso primero, en la misma pestaña seleccionamos *Build*, esto puede tardar unos segundos.

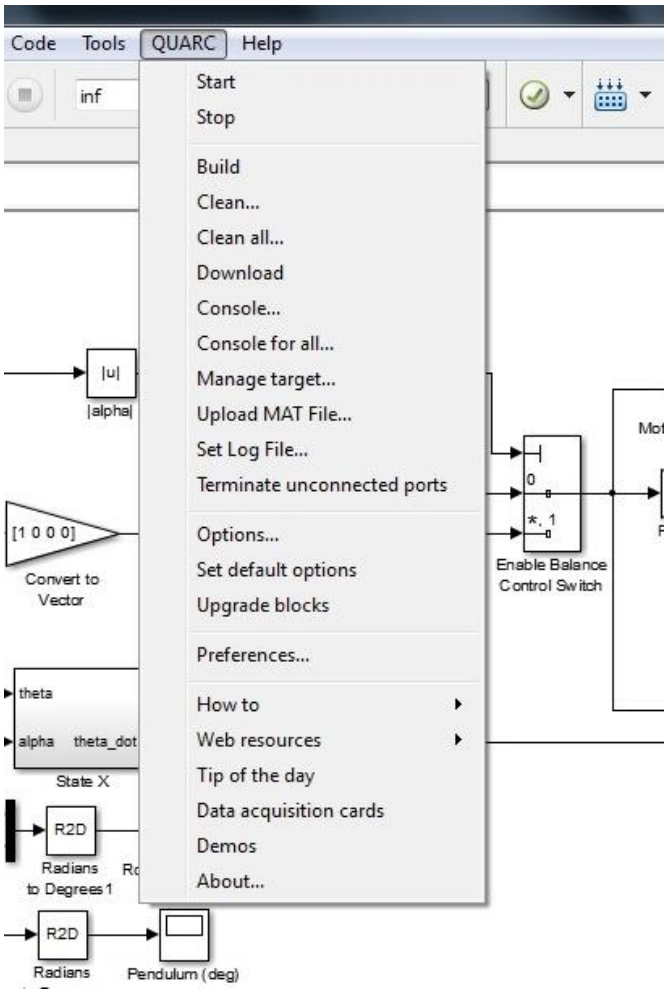

Figura 3-6. Compilación del péndulo de *QUBE-Servo* (I).

3) Tras los pasos anteriores nos vamos a la pestaña *Simulation* y picamos sobre *Connect to target*.

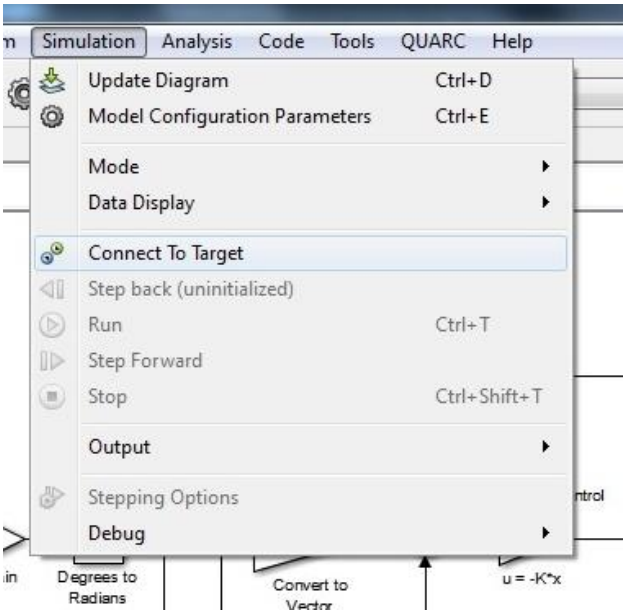

Figura 3-7. Compilación del péndulo de *QUBE-Servo* (II).

Una vez que se realizan los tres pasos comentados, ya podemos darle a la simulación  como hacemos siempre.

3.1 Identificación del péndulo de *QUBE-Servo*

En este apartado vamos a obtener la respuesta del péndulo ante diversas entradas, posteriormente vamos a intentar de identificar los parámetros del péndulo a través de la comparación entre la respuesta obtenida en el modelo real y la obtenida en el modelo matemático ante dichas entradas.

3.1.1 Pruebas de laboratorio y obtención de resultados ante diversas entradas

La primera entrada probada en el sistema, es una entrada aleatoria de tensiones entre -2 y 2 voltios que cambian cada medio segundo, véase la figura 3-8.

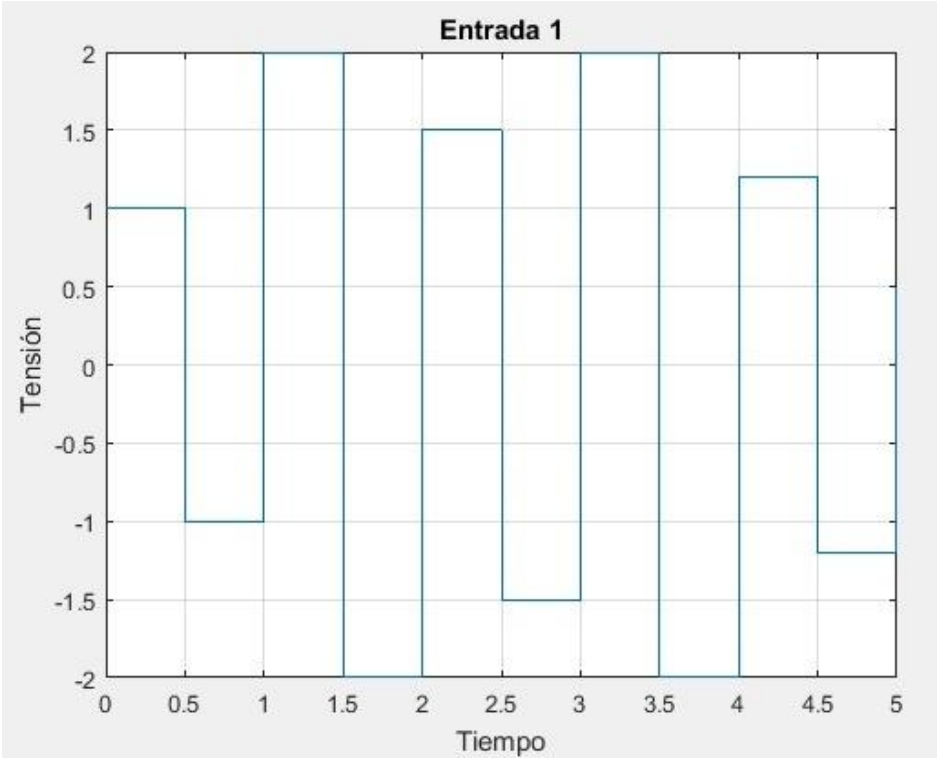

Figura 3-8. Tensión de entrada prueba 1.

Como resultado de la entrada anterior tenemos la respuesta en el ángulo del brazo, *phy*, que se puede apreciar en la figura 3-9, como observación, darnos cuenta que parte desde un valor distinto de cero, ya que se ha dejado que se aplique la entrada desde una posición cualquiera del brazo del péndulo. Este punto de partida deberá de tenerse en cuenta a hora de identificar el sistema.

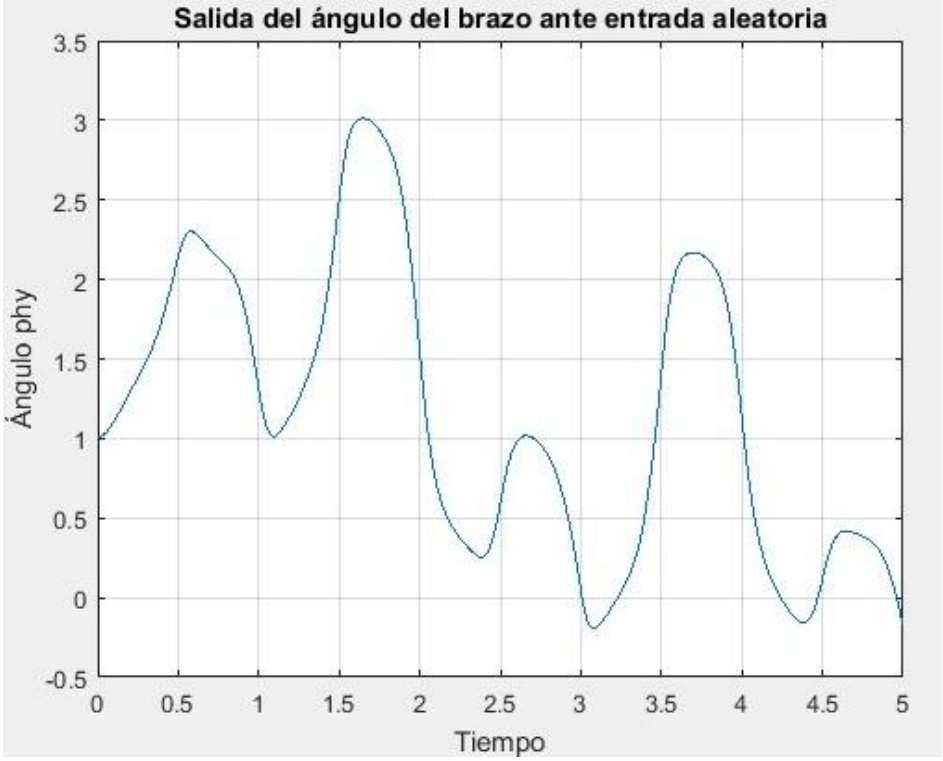

Figura 3-9. Ángulo del brazo del péndulo ante la entrada aleatoria.

Por otro lado en la figura 3-10 se puede ver el ángulo del péndulo obtenido ante la respuesta aleatoria, este parte desde cero, que es la posición vertical hacia abajo del mismo, lo que en nuestro modelo matemático sería $\theta = \pi$. Será importante tener esto en cuenta de cara a la identificación.

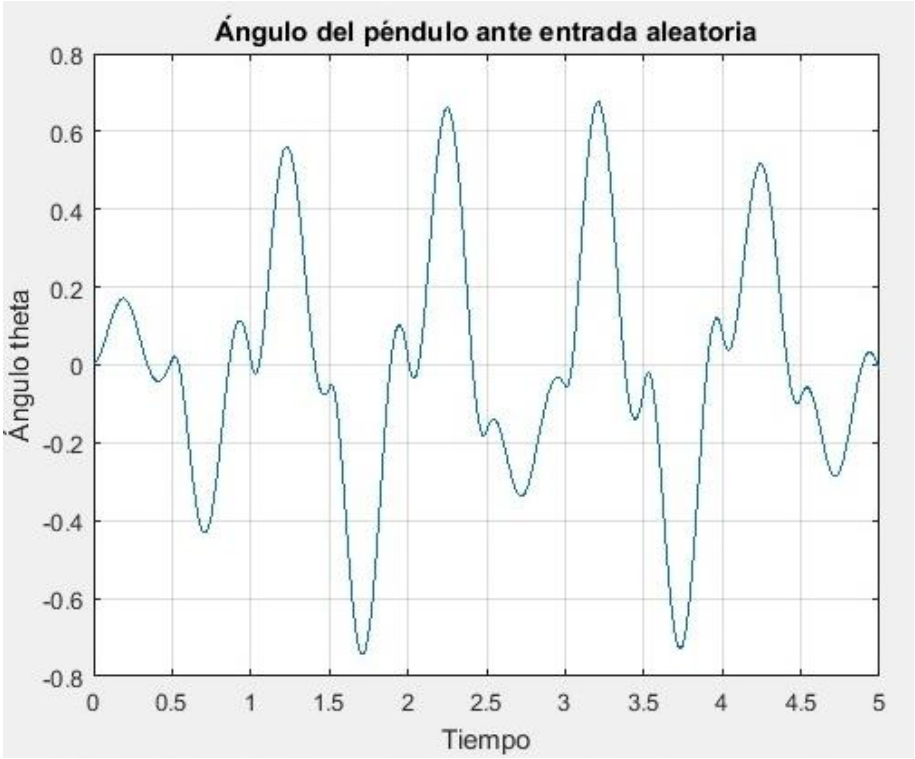

Figura 3-10. Ángulo del péndulo ante la entrada aleatoria.

La otra entrada que se ha probado en el laboratorio es una entrada de tensión sinusoidal con amplitud igual a medio voltio y frecuencia 1 rad/s.

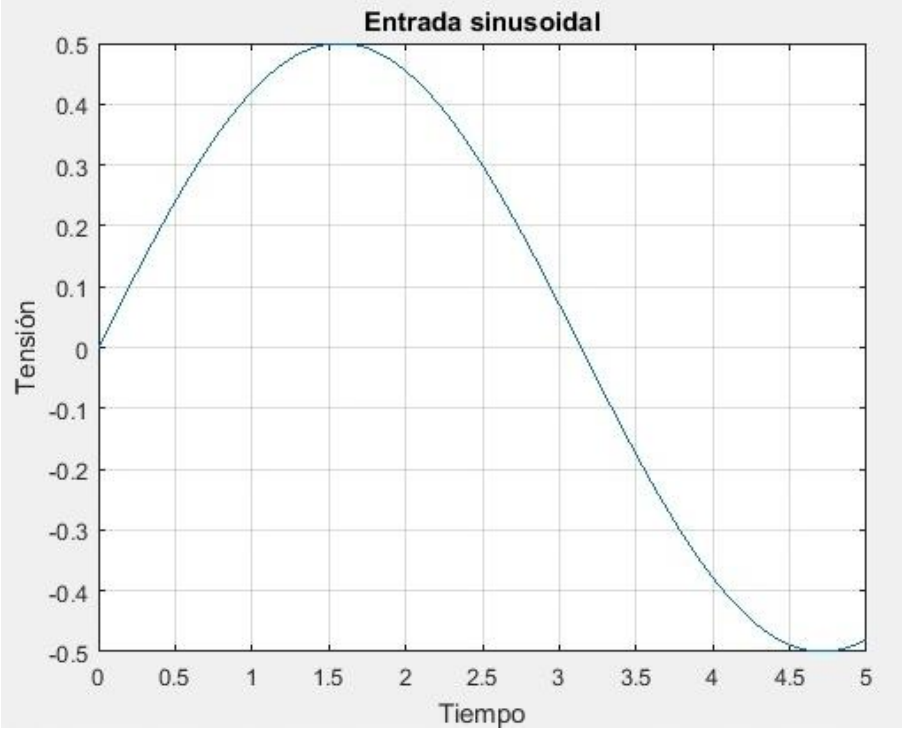

Figura 3-11. Tensión de entrada prueba 2.

Ante esta entrada obtenemos una respuesta en el ángulo del brazo como se puede apreciar en la siguiente figura.

Figura 3-12. Ángulo del brazo del péndulo ante la entrada sinusoidal.

Por su parte el péndulo presenta la siguiente respuesta,

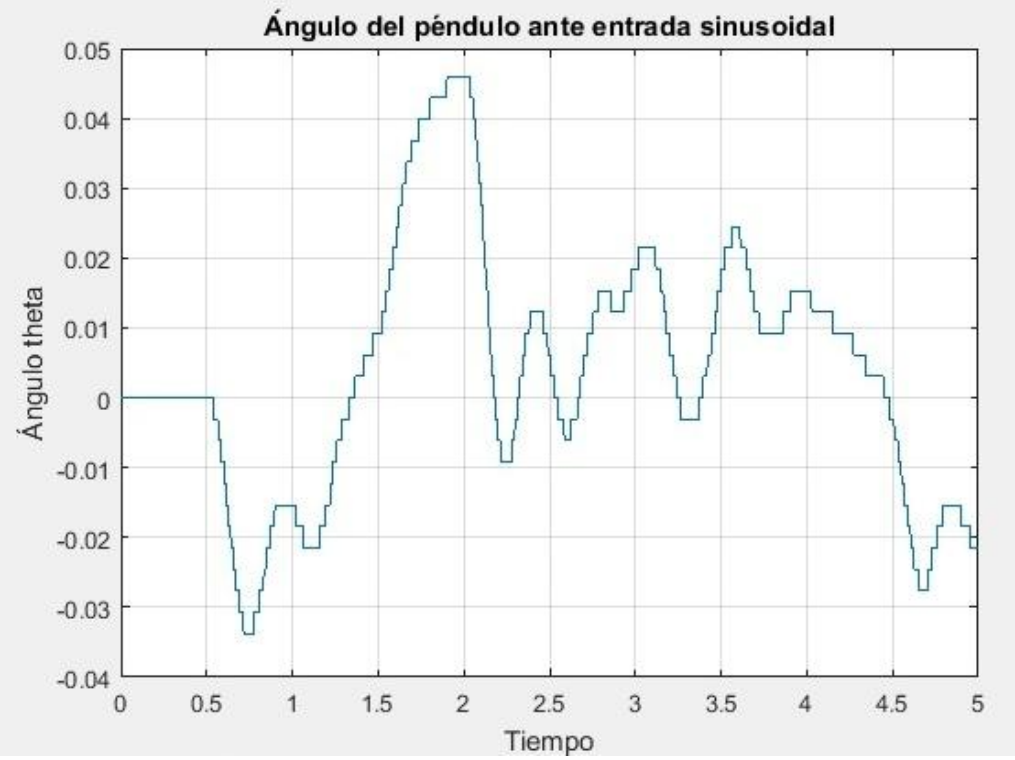

Figura 3-13. Ángulo del péndulo ante la entrada sinusoidal.

Se puede ver la señal del ángulo *theta* como una serie de escalones debido a que realmente el ángulo del péndulo apenas varía y está muy cercano a cero, lo que provoca que podamos apreciar la resolución del *encoder*. Esto se debe a que el brazo se mueve muy despacio, ya que la entrada está comprendida entre -0.5 y 0.5 voltios, y es suave debido a que es sinusoidal.

3.1.2 Identificación a un paso o en bucle abierto

Este método de identificación consiste en ir probando con todas las combinaciones posibles de los parámetros a identificar del péndulo, estos son, la longitud del brazo [r], la mitad de la longitud del péndulo [l], la masa del péndulo [m], la ganancia del motor [Km], el parámetro de fricción del péndulo [cp] y la fricción viscosa del brazo [ca], hasta que la respuesta del modelo matemático se parezca a la real. Ciertamente se trata de un método de fuerza bruta.

El fabricante del péndulo nos proporciona valores de los parámetros a identificar, lo que hacemos es tomar variaciones de ± X% de estos valores y calcular el $\sum e^2$, siendo e el error entre la respuesta real del ángulo *theta* (o *phy*) y el mismo ángulo del modelo matemático, para todas las combinaciones de los parámetros tomando como válidos aquellos que dan un menor sumatorio del error cuadrático. Este es de hecho el método de los mínimos cuadrados (nombre dado por el matemático Adrien-Marie Legendre) creado por Gauss.

Se ha desarrollado un programa en *Matlab* que realiza este conjunto de operaciones, a continuación se explica su funcionamiento.

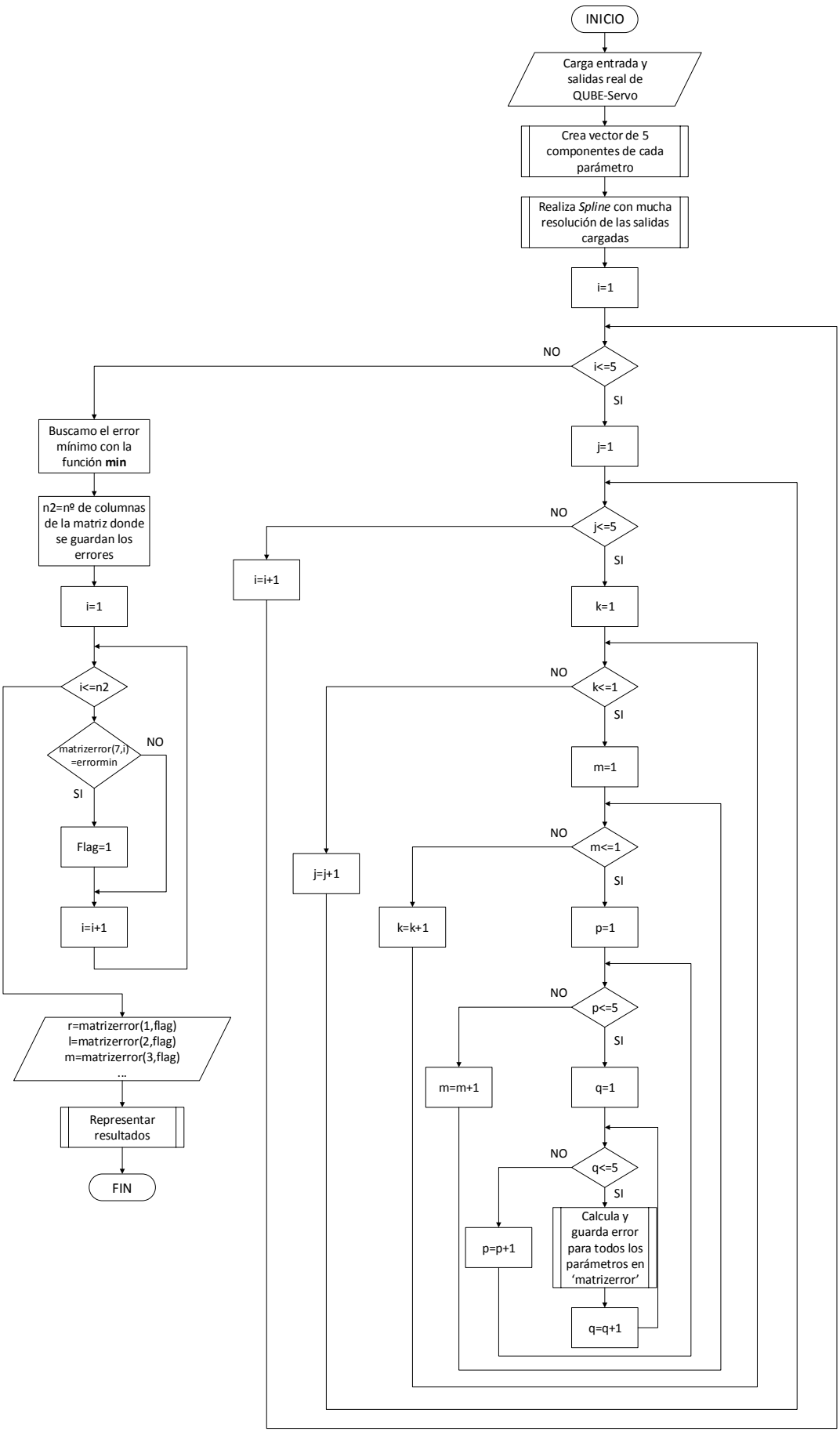

Figura 3-14. Diagrama de flujo (I).

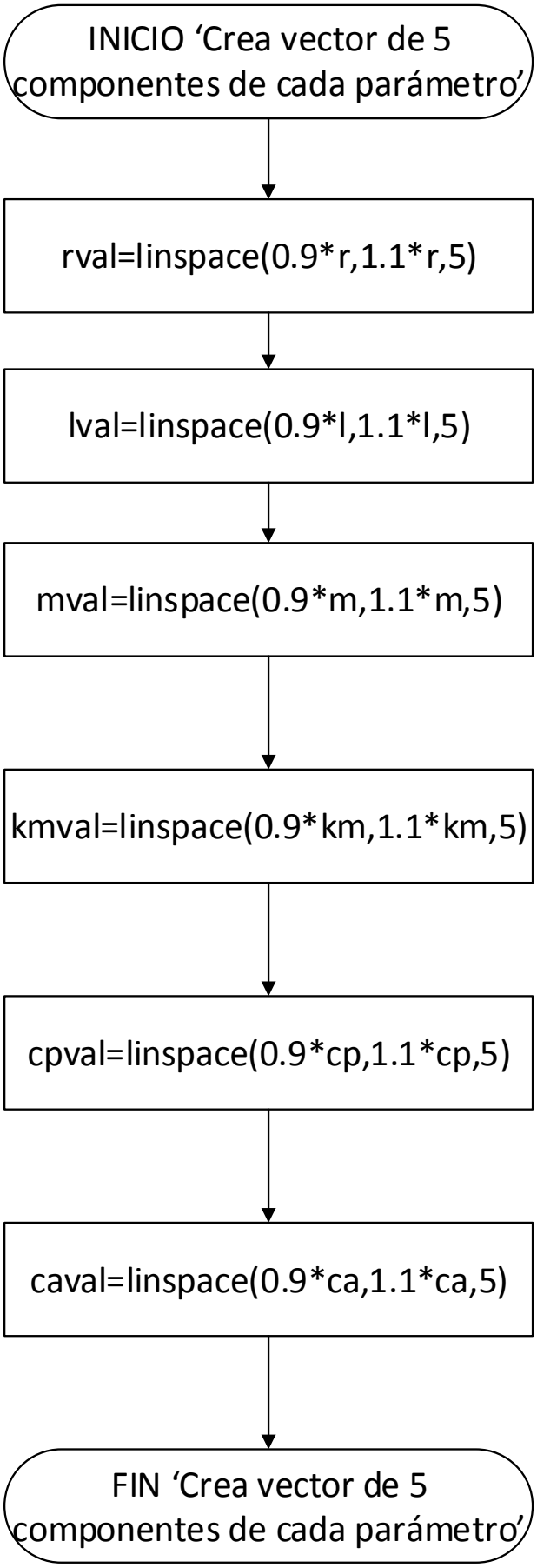

Figura 3-15. Diagrama de flujo (II).

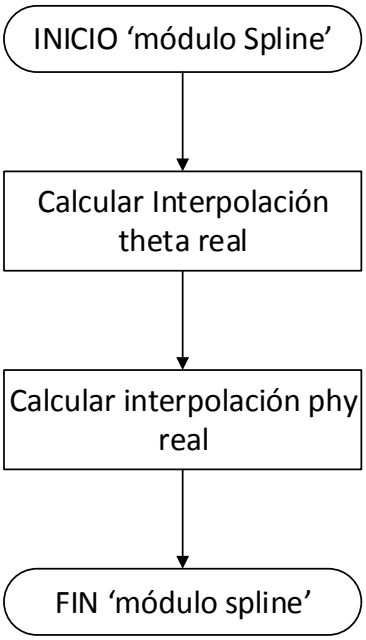

Figura 3-16. Diagrama de flujo (III).

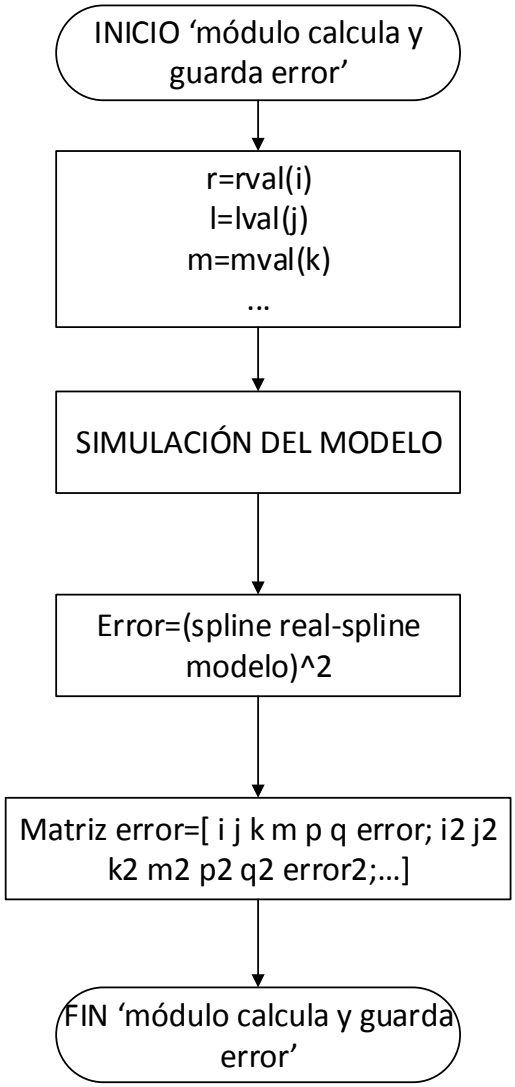

Figura 3-17. Diagrama de flujo (IV).

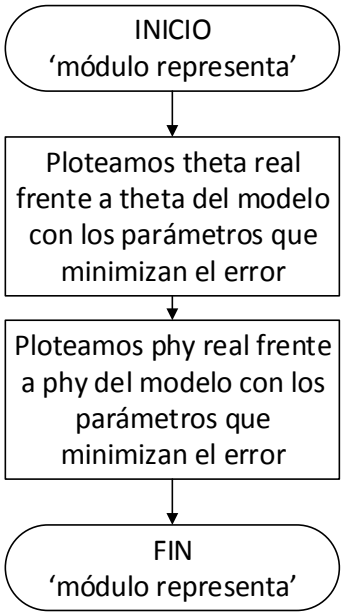

Figura 3-18. Diagrama de flujo (V).

Para cargar los parámetros reales, la entrada de tensión y las salidas *theta* y *phy*, usamos el comando *load*. En el primer módulo se hace uso de la función *linspace*, que proporciona un vector de cinco componentes para cada parámetro entre un menos diez por ciento y más diez por ciento del valor teórico propuesto por el fabricante.

En el segundo módulo se usa el comando *spline*, que permite una interpolación cuadrática entre puntos consecutivos de los valores de los ángulos del péndulo y del brazo reales, pero, ¿por qué hacemos esto?, el motivo es los ángulos reales que cargamos y las que se generan en el modelo matemático no tienen la misma resolución, es decir, la enésima componente del vector donde se guarda el ángulo real y la misma componente del vector del ángulo generado por el modelo matemático no tienen lugar al mismo tiempo, luego no se puede calcular el error como simplemente la diferencia entre ambos vectores si no hacemos antes una interpolación. A continuación se intenta clarificar esta idea, supongamos que tenemos la siguientes respuestas real y obtenida por el modelo matemático,

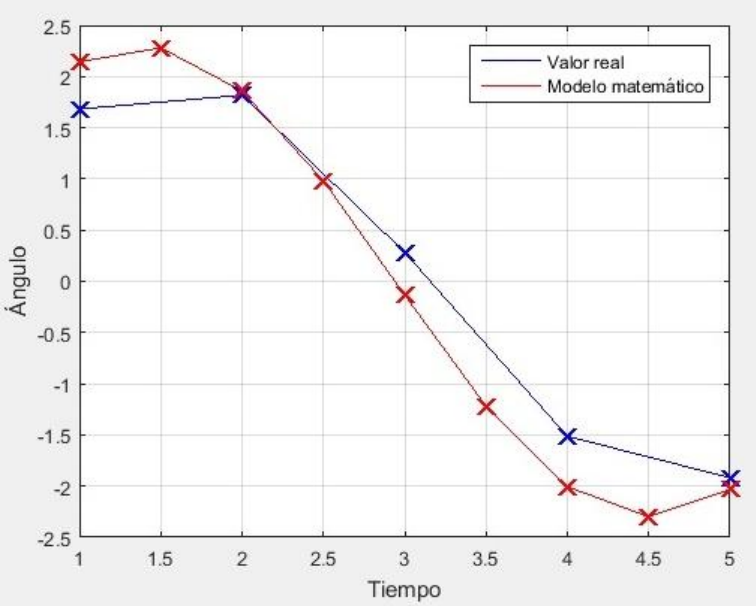

Figura 3-19. Uso de *splines* (I).

Como se puede apreciar tenemos más muestras, más resolución, en la respuesta del modelo matemático que en la respuesta real obtenida, una solución para poder calcular el error, es decir, la diferencia entre la señal real y la del modelo, de la que solo tenemos las muestras discretas, es calcular a través de una interpolación puntos intermedios del modelo real, esto es lo que se aprecia a continuación,

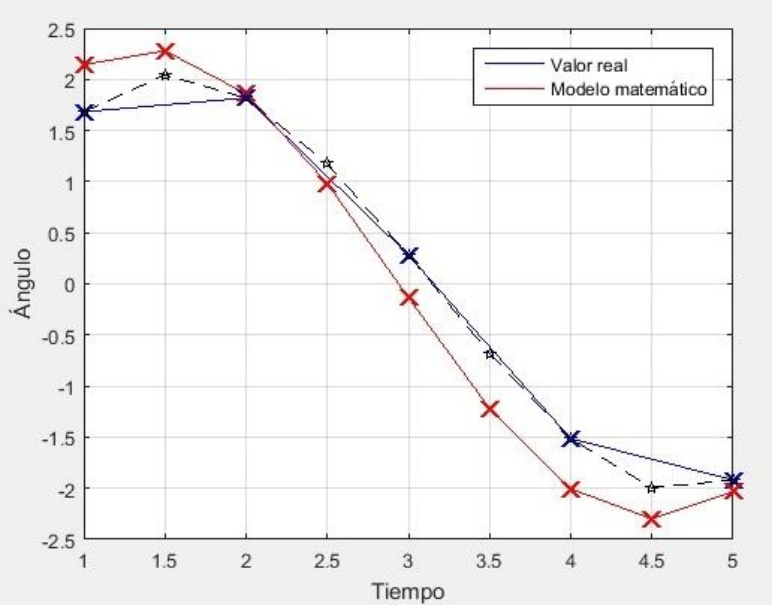

Figura 3-20. Uso de *splines* (II).

Como se puede ver, ahora se dispone de muestras intermedias en la señal con menor resolución para poder calcular el error como la diferencia entre unos valores y otros. *Grosso modo* esto es lo que se hace en el programa, pero teniendo en cuenta que realmente tenemos muchas muestras por segundo tanto en el modelo real como en el modelo matemático, así al interpolar las señales con aún mayor resolución, el error introducido por la interpolación, ante el posible valor real, es mínimo.

Luego entramos en una serie de bucles que lo que hacen es calcular el error cuadrático entre ambas señales para todos los valores de cada parámetro, como son 5 valores por cada parámetro y 6 parámetro en total, se realizarán las operaciones de dentro de los bucles un total de 5^6=15625 veces, este es el principal motivo por el que el programa tarda un tiempo considerable en ejecutarse, ya que debe ejecutar *Simulink* dicho número de veces.

Las operaciones que se realizan dentro de los bucles no son otras más que ejecutar el modelo matemático de *Simulink* con los parámetros que correspondan a través de la instrucción *sim* y luego se calcula el error cuadrático en *theta* y *phy*. Para cada iteración guardamos los valores de los parámetros y el error obtenido en una matriz.

Tras salir de los bucles, vamos a la matriz de error que se creó y se busca a través de la función *min* cual es el error mínimo cometido y el valor de los parámetros que corresponden a ese error mínimo.

En último lugar, el programa simula de nuevo el modelo para los parámetros que producen un error mínimo y representa los resultados comparándolos con los resultados reales.

Tras ejecutar el programa iterativamente con los parámetros que van proporcionando menor error en cada ocasión llegamos a los siguientes parámetros:

r=0.0619 m, l=0.0859 m, m=0.0238 kg, Km=0.0320 V/(rad/s), Cp=0.7895 y Ca=1.0526

Con estos parámetros obtenemos los resultados que se muestran a continuación.

En primer lugar, para la entrada sinusoidal hemos obtenido la siguiente identificación.

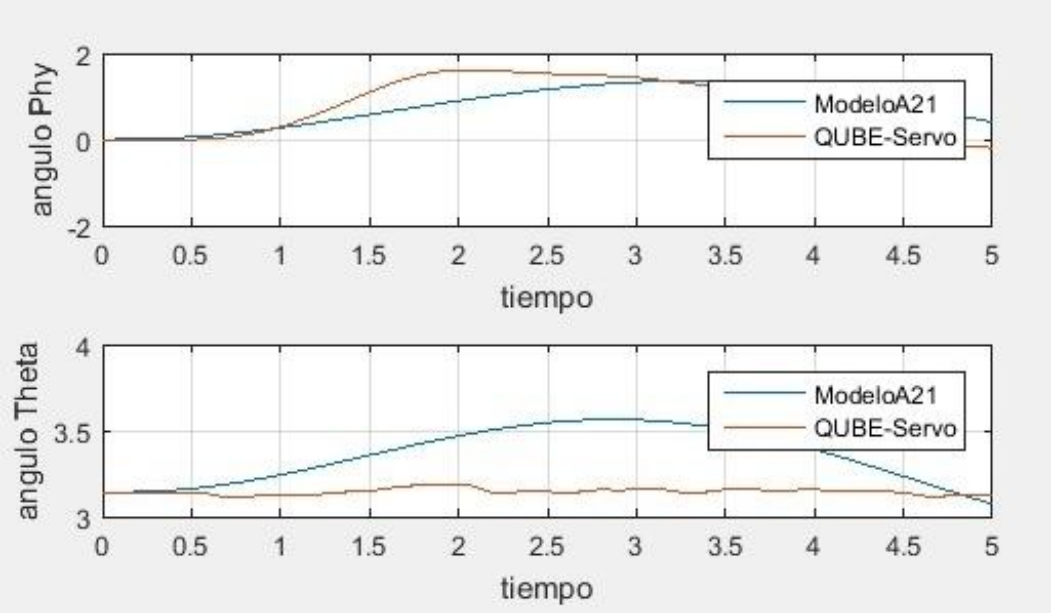

Figura 3-21. Comparación entre *QUBE-Servo* y modelo A21 frente a entrada sinusoidal.

Con estos parámetros, tenemos la siguiente respuesta entre el modelo matemático y el real para el ángulo *phy* ante la entrada aleatoria.

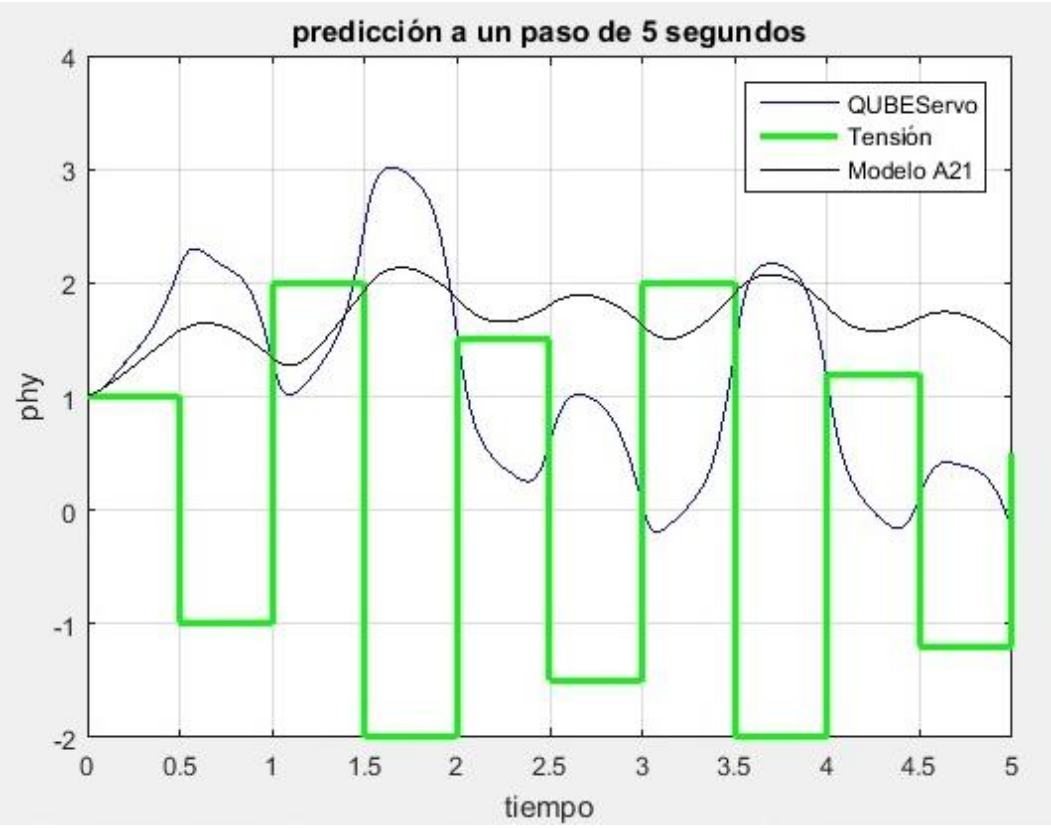

Figura 3-22. Comparación entre *QUBE-Servo* y modelo A21 frente a entrada aleatoria.

Como se puede apreciar la identificación no es buena, esto se debe principalmente a que identificamos los parámetros en bucle abierto, es decir, si la identificación es mala y la respuesta del sistema real y del modelo divergen, dentro de un tiempo no estás considerando únicamente el error entre un modelo y otro sino que también estas considerando un error extra debido a que ambas respuestas parten de puntos distintos debido a que vienen divergiendo desde atrás en el tiempo.

En el caso del ángulo *phy* ante la entrada aleatoria la identificación es igualmente mala, aunque se consigue que ambas señales no estén desfasadas, lo cual es lógico.

El programa se encuentra en el anexo al final del documento.

3.1.3 Identificación a N pasos o en bucle cerrado

Como dijimos en el apartado anterior, al identificar en bucle abierto estamos incluyendo un error extra debido a que las señales vienen divergiendo desde atrás en el tiempo y aunque, de repente, el sistema real se comportara igual que el modelo matemático, seguiríamos teniendo un error (que no debería de existir) que vamos sumando debido a que ambos modelos parten desde puntos distintos a causa de los errores anteriores.

Para evitar esto se ha creado un nuevo programa que trata de identificar el sistema a N pasos, es decir, corrigiendo su estado respecto al estado real del *QUBE-Servo* cada un paso de N segundos. De esta forma vamos viendo durante cuánto tiempo el modelo matemático tiene validez y tiene un comportamiento similar al modelo real.

Como se verá a continuación, para ello debemos coger las cuatro variables del espacio de estados y tomar el valor adecuado de las mismas como condiciones iniciales cada periodo de tiempo N para que cada dicho periodo de tiempo, el sistema real y el modelo matemático partan desde el mismo punto. Para poder introducir desde fuera las condiciones iniciales de posición y velocidad de *phy* y *theta* se han añadido estas variables al sistema de *Simulink*.

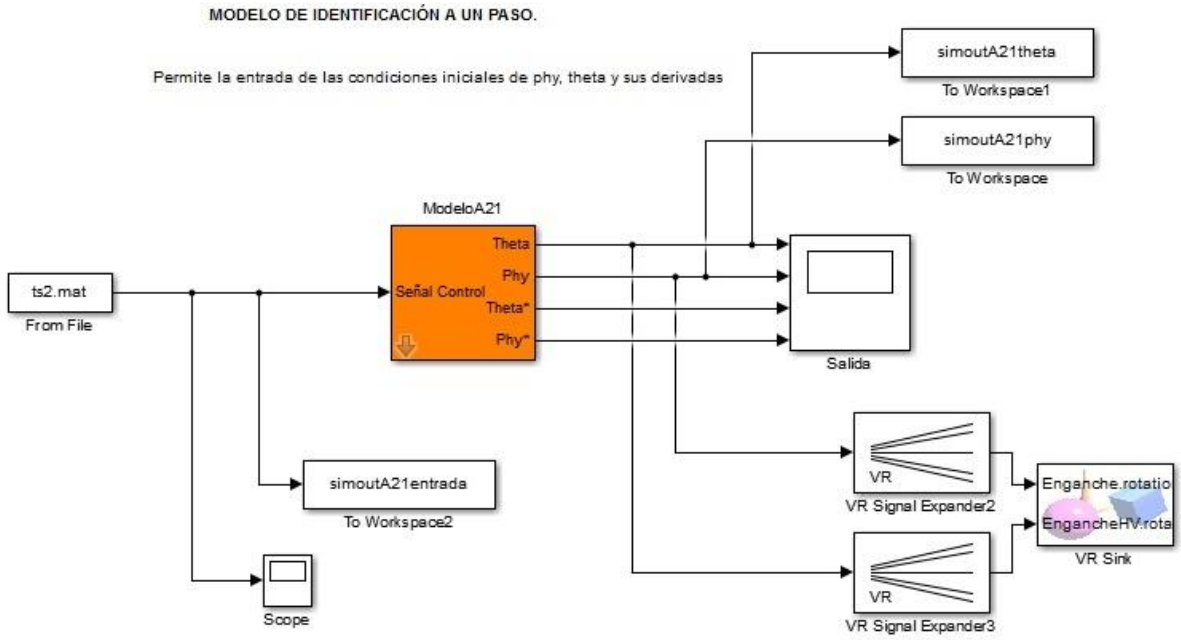

Figura 3-23. Modelo en *Simulink* del péndulo que se cargará desde el programa (I).

En este modelo se pueden introducir desde el exterior, el *workspace*, las condiciones iniciales de posición y

velocidad.

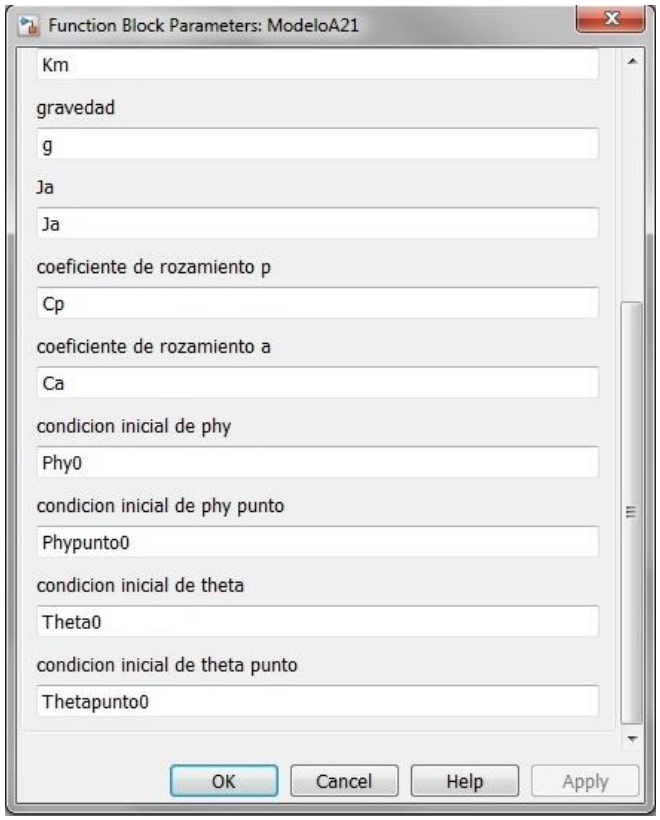

Figura 3-24. Modelo en *Simulink* del péndulo que se cargará desde el programa (II).

A continuación se explicará el programa de identificación a N pasos, pero previamente hay que explicar cómo se crea la entrada que se usará en el sistema, en concreto la entrada aleatoria.

Se creó un vector aleatorio de tensiones que iban desde -2 hasta 2 voltios, cada uno de estos valores se mantenían durante medio segundo, ya que la entrada al sistema real era una *Repeating Sequence Stair* con un *Sample Time* de medio segundo. Pero al guardar la entrada y cargarla lo que tenemos son solo los puntos de tensión sin ser mantenidos en el tiempo, esto es lo que se muestra a continuación.

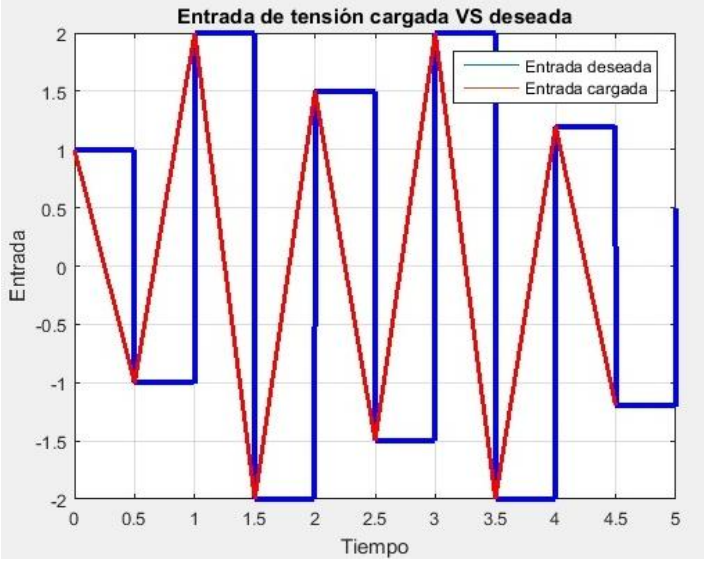

Figura 3-25. Tensión cargada VS deseada.

Hay que manipular la entrada para que sea la que realmente queremos, para ello hacemos un breve programa que crea un vector que a cada medio segundo la asigna 250 valores cuyo valor es el que corresponde de la entrada cargada. Así como la identificación es durante cinco segundos, tendremos un vector de 5x2x250 = 2500 valores que si representan la entrada deseada, llamamos b a este vector.

```
a=load('entrada_pendulo_aleatoria.mat')
b=[];
for i=1:30
    unos=ones(1,250);
    b=[b,a(i)*unos];
end
```

Para acabar de completar la entrada ejecutamos el siguiente modelo en *Simulink.*

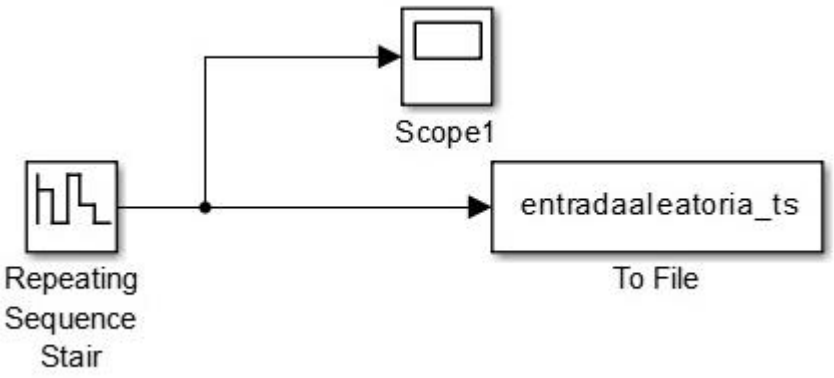

Figura 3-26. Creación de la entrada aleatoria.

En la secuencia repetitiva introducimos el vector b creado anteriormente, con un *Sample Time* de 0.002 segundos, ya que tenemos 2500 valores para 5 segundos (5/2500=0.002, un valor cada 0.002 segundos). *To File* permite guardar un archivo con un nombre dado y en un formato determinado, el archivo se ha llamado *entradaaleatoria_ts* y se ha guardado en formato *Timeserie.* Una vez que se ejecuta el modelo ya tenemos la entrada correcta creada y además en formato *Timeserie* que nos permitirá manipular fácilmente dicha entrada, ya que se debe ir introduciendo 'a pasos'.

A continuación se explica a través de diagramas de flujo el funcionamiento del programa.

En primer lugar pedimos el número de pasos del sistema a través del comando *input,* luego se calculan las velocidades de los ángulos *theta* y *phy* (esto se explica en un módulo distinto) y se interpolan todas las variables del espacio de estado para obtener su valor en cada momento de tiempo t= k*paso con k=0, 1, 2, 3, 4, 5 ya que tendremos que tomar esos valores como condiciones iniciales para la simulación de cada tramo de 'paso' segundos. Luego cargamos los parámetros del modelo y finalmente simulamos paso a paso, como realizar dicha simulación se explica en un módulo diferente.

En la figura 3-28 se puede apreciar lo que hacemos al interpolar, que no es otra cosa más que coger los valores de *theta, phy* y sus derivadas en ciertos puntos y que serán consideradas las condiciones iniciales de la simulación en cada intervalo de N segundos.

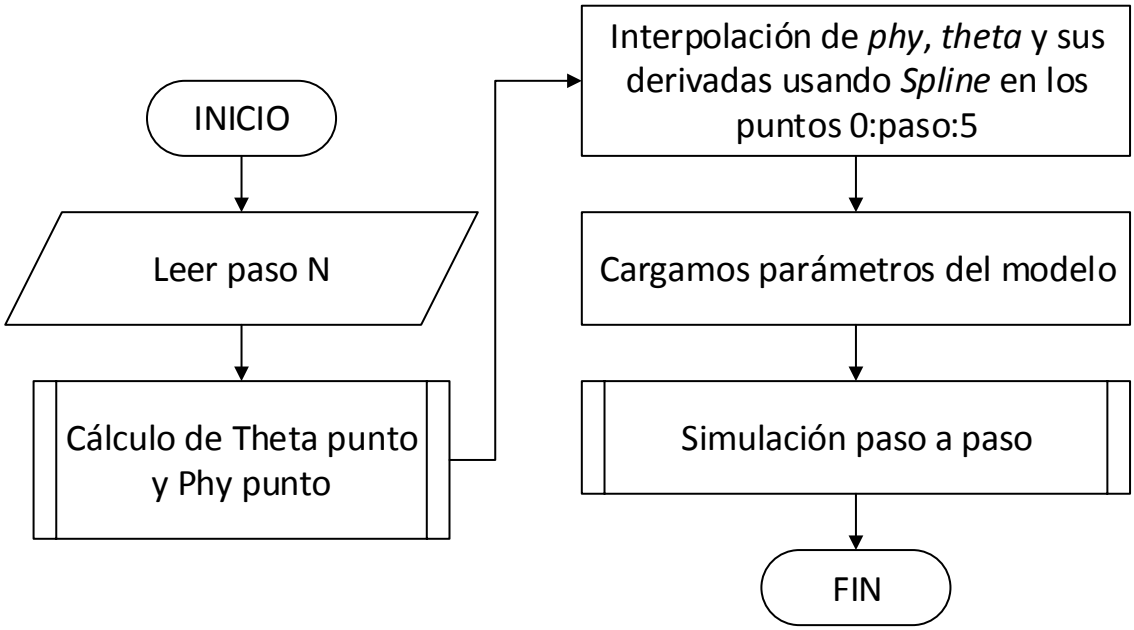

Figura 3-27. Diagrama de flujo (VI).

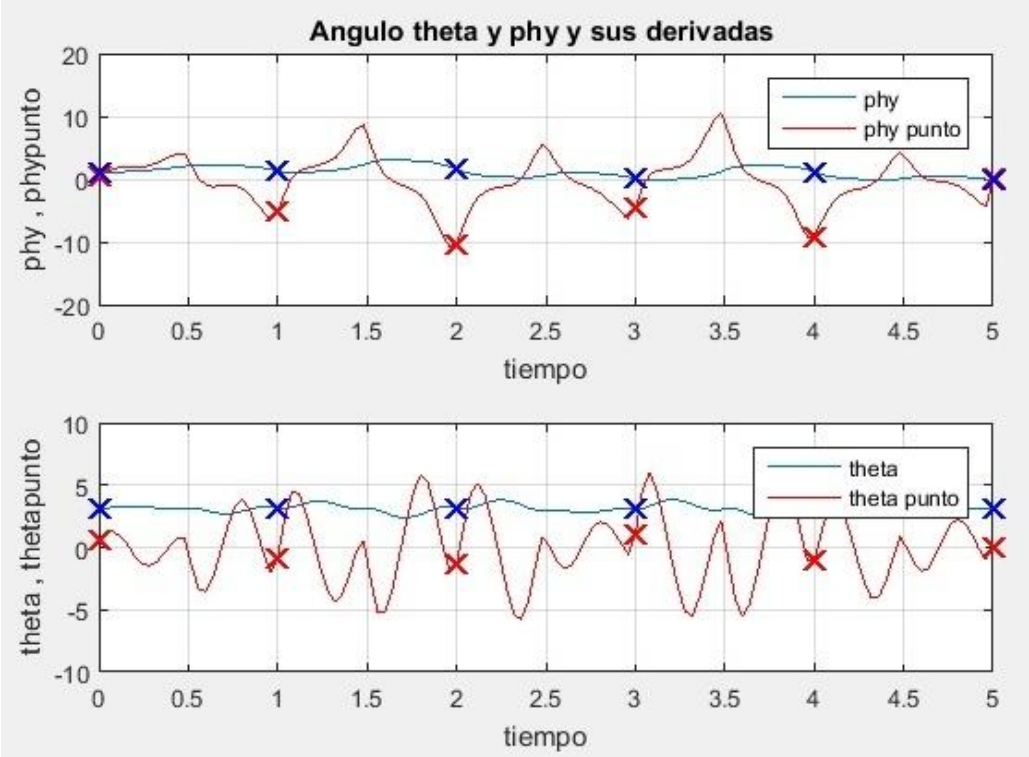

Figura 3-28. Interpolación de las variables del espacio de estados cada 'paso' segundos.

Esto que se ha explicado es el funcionamiento general del programa cuyo diagrama de flujo se puede apreciar en la figura 3-27. A continuación se explicarán detalladamente los dos módulos que se han comentado anteriormente.

Para obtener las derivadas de los ángulos, tal y como se aprecia en el diagrama de flujo de la figura 3-29, en primer lugar cargamos los ángulos reales que se obtuvieron del laboratorio y que son de tipo *Timeserie*. Hemos tomado Td=0.04 como tiempo de derivación, como ya sabemos la derivada de una función en un punto 'a' es,

$$\lim_{h\to 0} \frac{f(a+h)-f(a)}{h}$$

o lo que es lo mismo, la pendiente de la recta tangente en el punto 'a', numéricamente lo que hacemos es sustituir h → 0 por un valor de h muy pequeño pero no cero, que he llamado tiempo derivativo (*Td*).

Para obtener el valor de los ángulos cada *Td* segundos volvemos a hacer uso de los *splines* y luego vamos recorriendo los valores de dichos ángulos cada *Td* segundos guardando en un vector los resultados de la operación *(ángulo(i+1)-angulo(i))/Td* , finalmente se representan los resultados.

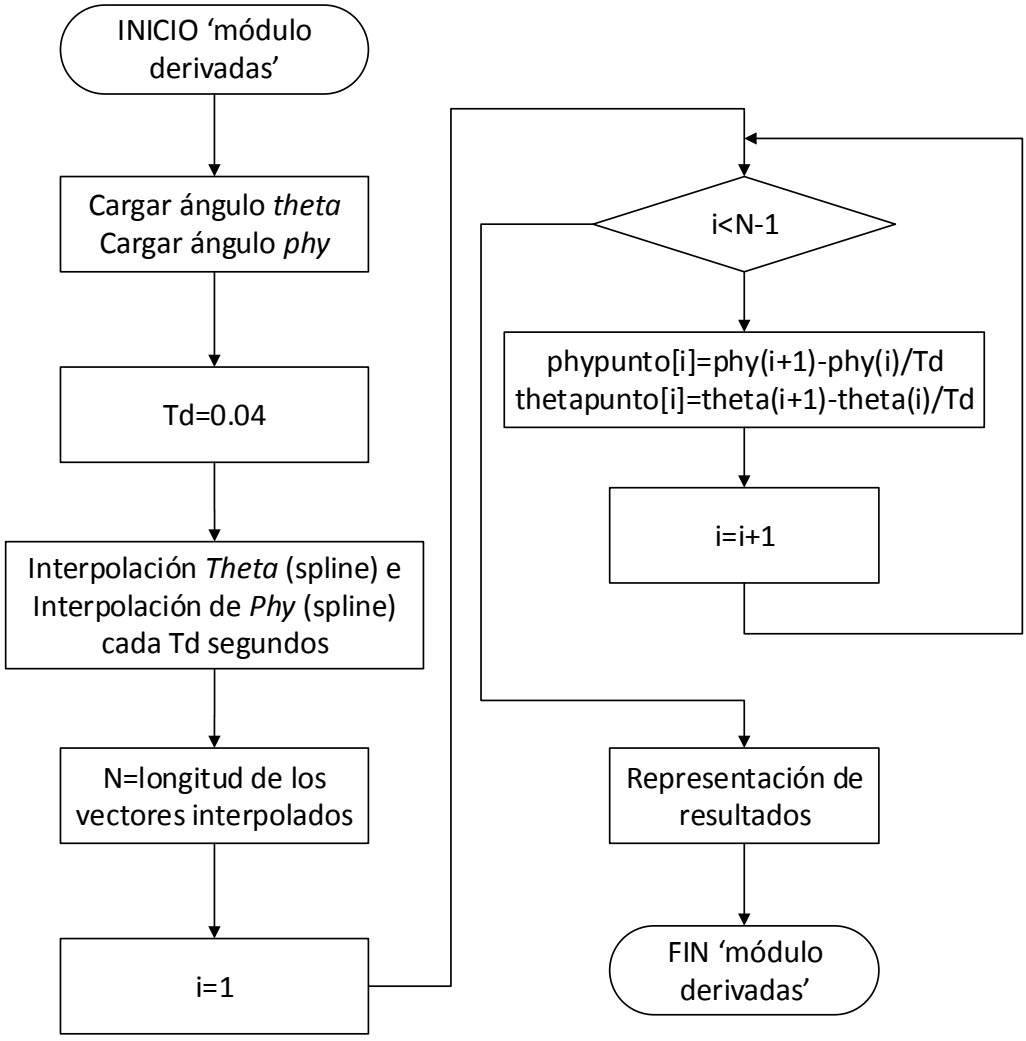

Figura 3-29. Diagrama de flujo (VII).

El resultado de este módulo es la representación de los ángulos reales del péndulo y del brazo y sus derivadas, es en definitiva lo que se representa en la figura 3-28.

A continuación se va a explicar el módulo que simula 'paso a paso' cuyo diagrama de flujo se puede apreciar en la figura 3-30. En primer lugar se carga la entrada cuya creación se explica al comienzo de este apartado, y nos metemos en un bucle que se ejecutará '5/ paso' veces, cada una de estas veces o tramos duran N segundos, en cada ocasión se toman como condiciones iniciales las del péndulo real en el instante de tiempo que corresponde al comienzo de cada tramo. Dentro del bucle lo primero es saber desde que tiempo hasta que tiempo de los 5 segundos totales dura el tramo en cuestión, que dependerá de la pasada del bucle por la que vayamos. Se usa *getsampleusingtime*, que es un comando de *Matlab* para *Timeseries* que permite tomar los datos que van desde un tiempo inicial a otro tiempo final y así tenemos una entrada que dura N segundos, pero que tenemos que trasladar en el tiempo para que en la simulación dicha entrada empiece a partir de cero, para

ello usamos *setuniformtime*, que es otro comando de *Matlab* que permite cambiar el tiempo de una *timeserie* desde un punto de tiempo inicial, que será cero, con un intervalo determinado que es 0.002 segundos en nuestro caso, el tiempo que se comentó anteriormente que va entre dato y dato. Luego es muy importante guardar esta entrada para poder ejecutar el modelo en *Simulink*, para ello usamos *save ('ts2.mat','ts2','-v7.3')*, es de gran importancia guardar en la versión 7.3. Ahora solo nos falta tomar las condiciones iniciales del brazo y del péndulo que corresponden al tramo que estamos simulando y finalmente simular con el comando *sim* durante un tiempo 'paso'. Finalmente se representan los resultados.

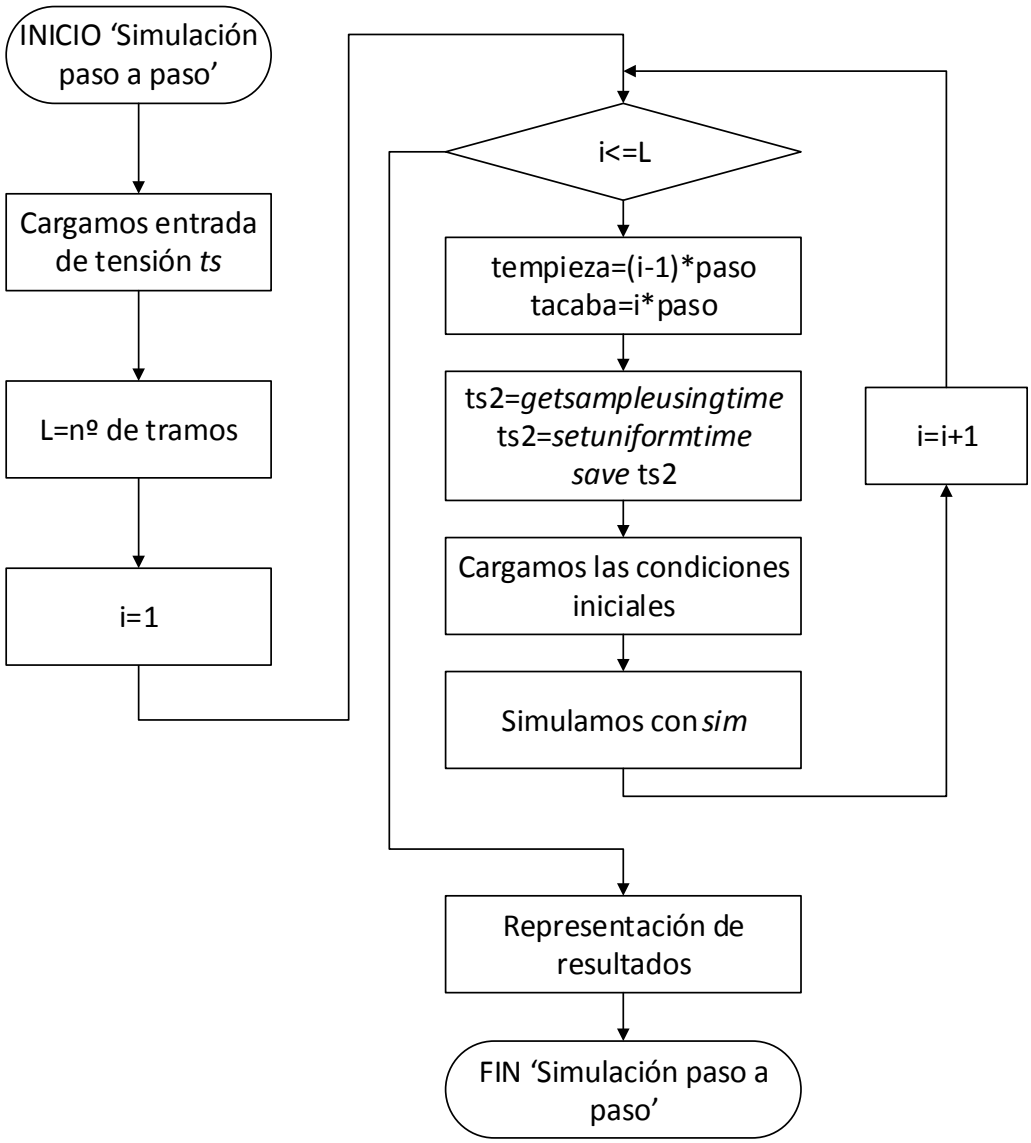

Figura 3-30. Diagrama de flujo (VIII).

Tras simular con diversos valores para los parámetros del péndulo, aquellos que proporcionan menor error cuadrático son los mismos que obtuvimos en la identificación en bucle abierto, es decir,

Tabla 3-1. Parámetros identificados.

r (m)	l(m)	m (kg)	Km (V/(rad/s))	Cp	Ca

0.0619	0.0859	0.0238	0.0320	0.7895	1.0526

Con dichos parámetros vamos a representar los resultados obtenidos para distintos pasos para los dos tipos de señal. Si el paso fuera de cinco segundos tendríamos el mismo resultado que en la identificación en bucle abierto.

En la figura 3-31 se muestra la respuesta ante la entrada aleatoria con un paso de 1 segundo, como se puede apreciar la respuesta entre el modelo matemático y el real van divergiendo, pero cada un segundo vuelve a tomar las condiciones iniciales del péndulo real. En la figura 3-32 se ve el ángulo del brazo con mayor detalle.

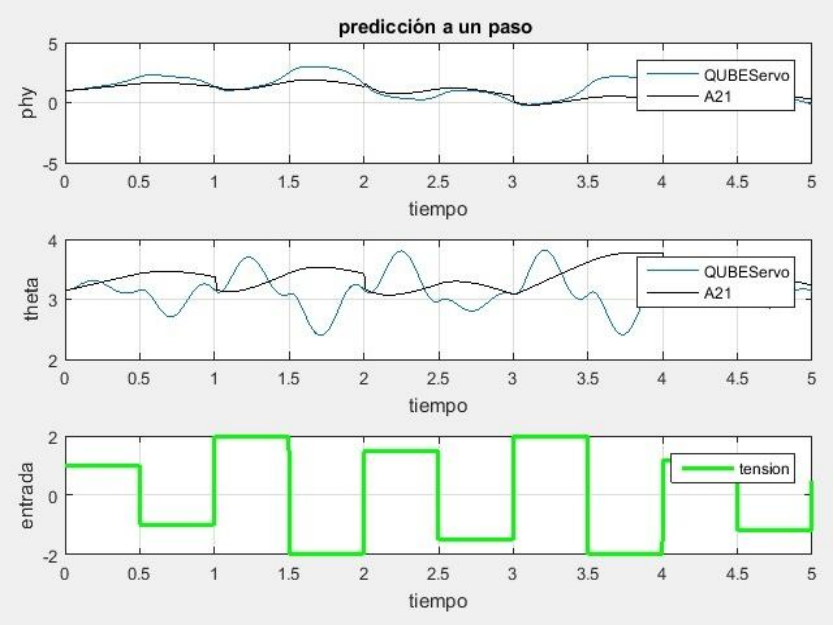

Figura 3-31. Respuesta a entrada aleatoria con paso 1 segundo (I).

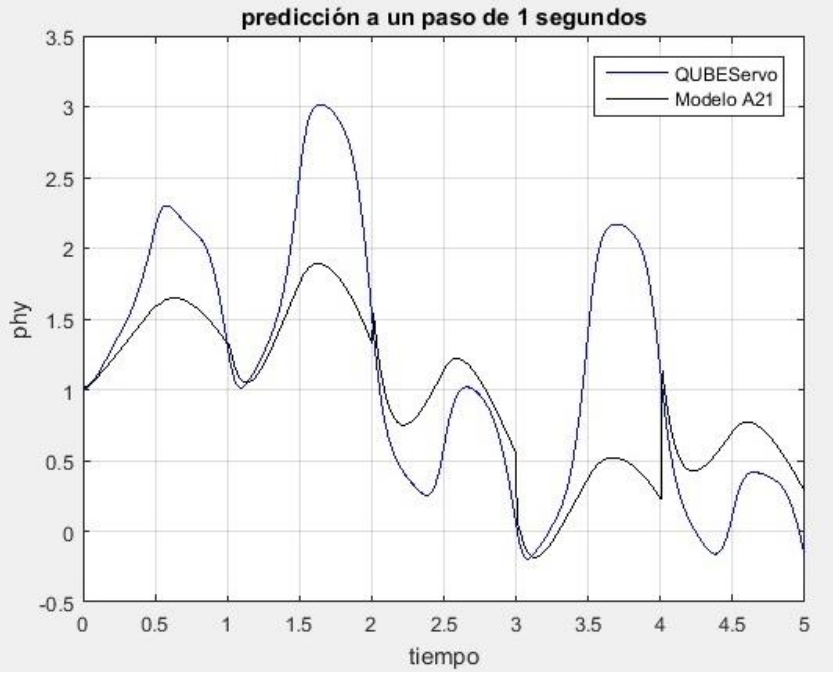

Figura 3-32. Respuesta a entrada aleatoria con paso 1 segundo (II).

Como se puede apreciar hay una gran diferencia entre el modelo real y el modelo matemático e incluso con un paso de un segundo se puede apreciar bastante error. Si disminuimos el paso a 0.2 segundos se puede ver que el error disminuye considerablemente, es decir, tenemos una respuesta aceptable durante un breve periodo de tiempo. La respuesta a un paso de 0.2 segundos es la que se muestra en las figuras 3-33 y 3-34.

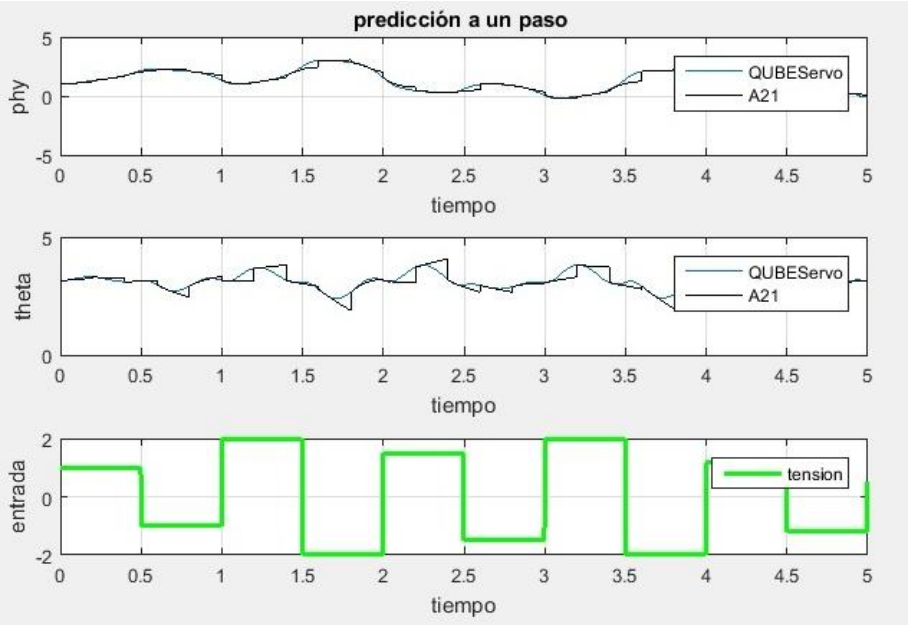

Figura 3-33. Respuesta a entrada aleatoria con paso 0.2 segundos (I).

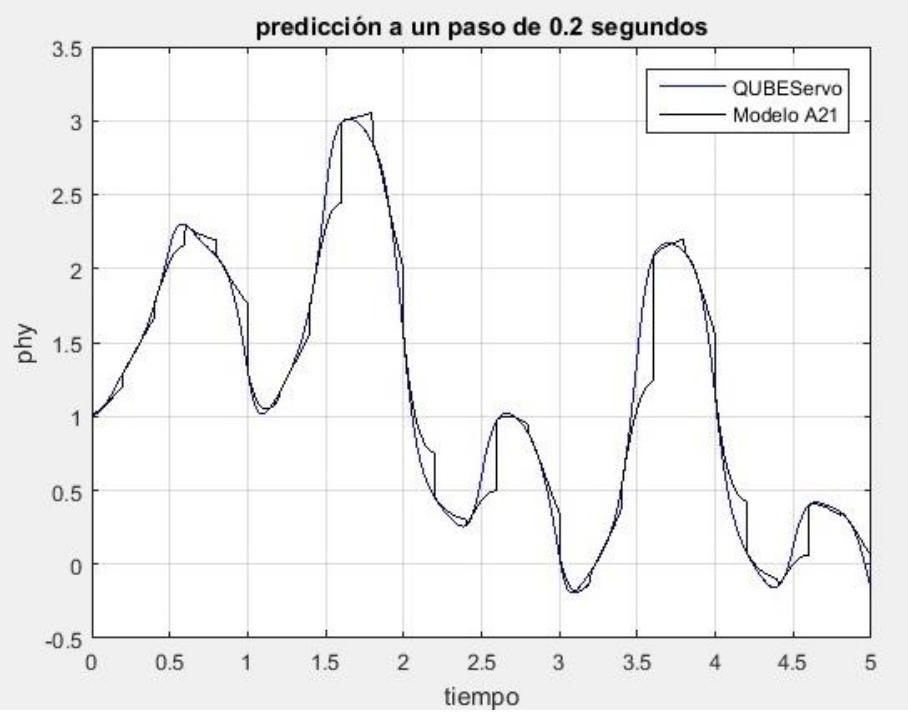

Figura 3-34. Respuesta a entrada aleatoria con paso de 0.2 segundos (II).

A continuación se muestra la respuesta del sistema si la entrada es sinusoidal y no aleatoria, en las figuras 3-35 y 3-36 se ve el resultado para el paso igual a un segundo.

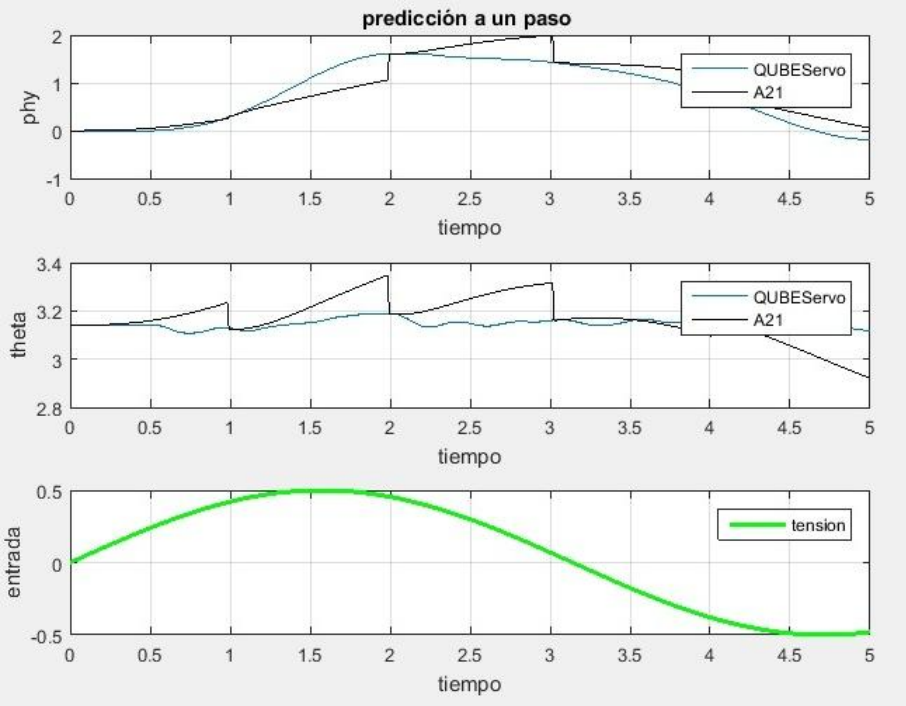

Figura 3-35. Respuesta a entrada sinusoidal con paso de 1 segundo (I).

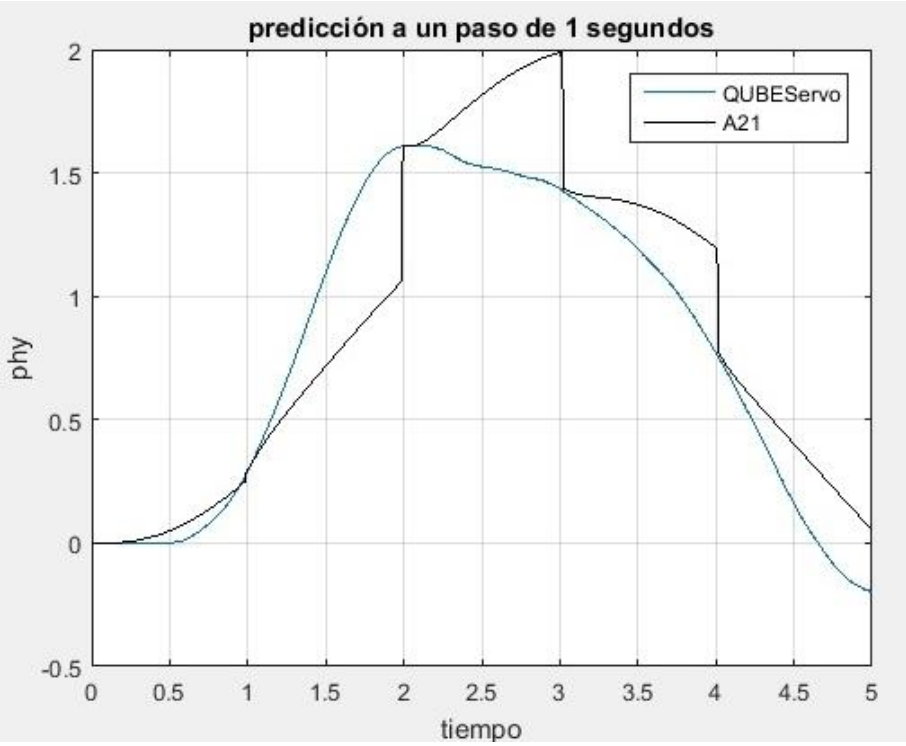

Figura 3-36. Respuesta a entrada sinusoidal con paso de 1 segundo (II).

Como se puede apreciar el error llega a ser bastante grande en un segundo, así que probamos con pasos más pequeños para ver si entonces la respuesta es válida. En las figuras 3-37 y 3-38 se puede ver la respuesta real y la del modelo A21 para un paso de 0.2 segundos.

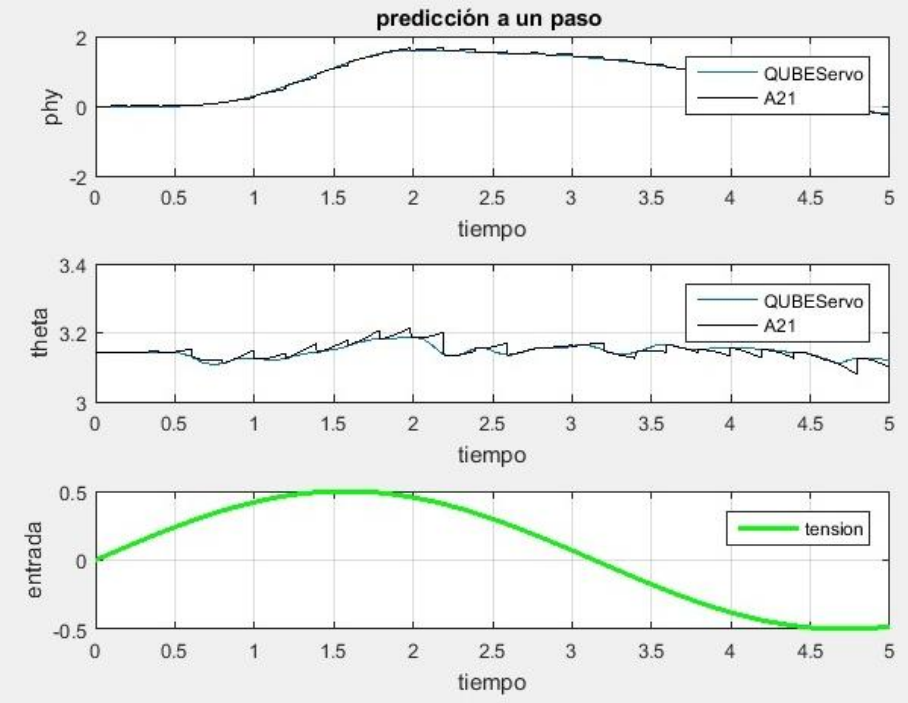

Figura 3-37. Respuesta a entrada sinusoidal con paso de 0.2 segundos (I).

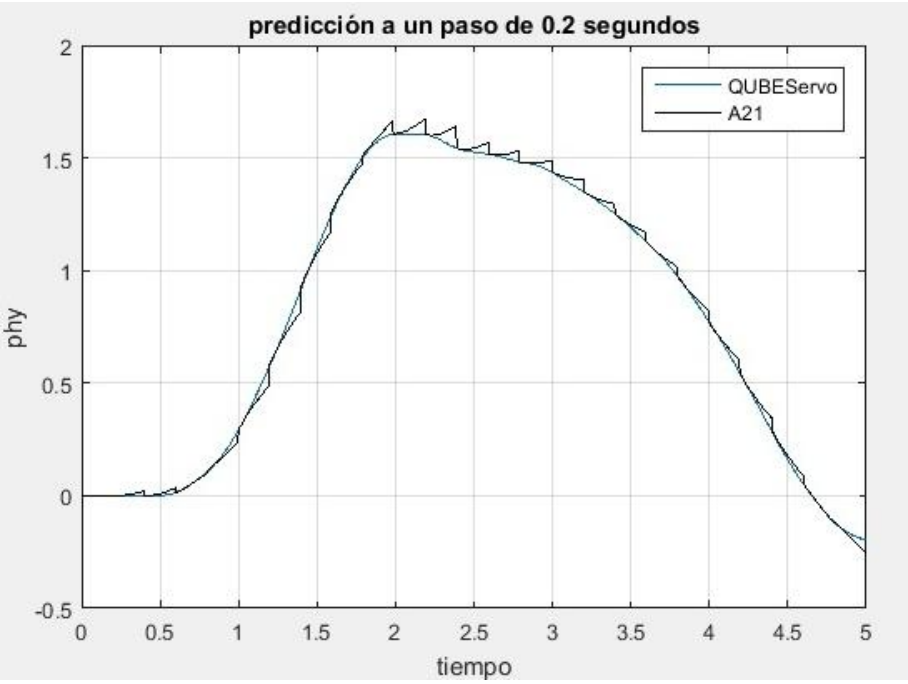

Figura 3-38. Respuesta a entrada sinusoidal con paso 0.2 segundos (II).

Incluso en un tiempo de 0.2 segundos, aunque no diverjan demasiado ambas respuestas, hay un error apreciable.

Se puede concluir que el modelo matemático quizás no contiene toda la dinámica del péndulo real, ya que dicho modelo, incluso el que se ha encontrado que comete menor error cuadrático, solo es válido para periodos de tiempo muy pequeños tales como 0.1 o 0.2 segundos, cuando aumentamos mas el tiempo de identificación ambas respuestas se muestran bastante distintas. A pesar de que la identificación del péndulo solo es válida y no comete demasiado error en intervalos de tiempo pequeñito, en apartados posteriores se hallará un

controlador LQR para estos parámetros y se verá si es capaz de controlar el péndulo.

El código del programa de identificación en bucle cerrado o a N pasos se adjunta en los anexos de este documento.

3.1.4 Identificación mediante ingeniería inversa

En la documentación del péndulo *QUBE-Servo* se proporciona información sobre las matrices A y B del control LQR, estas matrices dependen de los parámetros del sistema, se puede tratar de identificar a los parámetros a partir de dichas matrices. Las matrices proporcionadas se pueden apreciar en la figura 3-39.

```
A =

         0         0    1.0000         0
         0         0         0    1.0000
         0  149.2751   -0.0104         0
         0  261.6091   -0.0103         0


B =

         0
         0
   49.7275
   49.1493
```

Figura 3-39. Matrices A y B proporcionadas por el fabricante.

En el apartado 2.3.2 llegamos a que las matrices A y B de nuestro modelo matemático son,

$$A = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & \frac{-\alpha}{\beta-\alpha^2} & \frac{-C_a}{\beta-\alpha^2} & 0 \\ 0 & \frac{\beta}{\beta-\alpha^2} & 0 & \frac{-C_p}{\beta-\alpha^2} \end{bmatrix}$$

$$B = \begin{bmatrix} 0 \\ 0 \\ \frac{\gamma}{\beta-\alpha^2} \\ \frac{-\gamma\alpha}{\beta-\alpha^2} \end{bmatrix}$$

Donde,

$$\alpha = 3r/7l$$

$$\beta = \frac{m\,r^2 + J_a}{J_p + m\,l^2}$$

$$\gamma = \frac{K_m}{m\,g\,l}$$

Donde *Jp* es la inercia del péndulo,

$$J_p = \frac{4}{3}\, m\, l^2$$

Como se puede apreciar, las componentes A (4,3) y A (4,4) están aparentemente intercambiadas, esto puede deberse a que el modelo matemático que se ha usado para este trabajo no es el mismo que ha usado el fabricante del péndulo, quizás nuestro modelo matemático es demasiado simplificado para representar la compleja dinámica del péndulo, por ello a la hora de identificar el péndulo, dicha identificación solo es válida durante breves periodos de tiempo.

Aunque los modelos fueran algo distintos se podrían tratar de identificar con *fsolve*, una función de *Matlab* que nos permite resolver sistemas de ecuaciones no lineales, pero la diferencia entre los modelos es tal que se puede apreciar a simple vista que no existe solución posible, ya que *alpha* y *beta* deben ser positivos, y si comparamos A (3,2) y A (4,2) de ambas matrices A, nos damos cuenta que dichos valores de la matriz A de nuestro modelo matemático deberían de ser forzosamente de signo contrario, algo que vemos que no ocurre en la A usada por el fabricante, luego no existe forma posible de identificar los parámetros a partir del modelo matemático que se ha usado en este trabajo. En trabajos futuros se podría tratar de hallar otros modelos matemáticos para nuestro sistema.

3.2 Pruebas de control en el péndulo de QUBE-Servo

En los apartados 3.2.2 y 3.2.3 se han obtenido unos parámetros del sistema a partir de la identificación que surge de comparar la respuesta real del sistema con nuestro modelo matemático ante diversas entradas. Estos parámetros son aquellos que han proporcionado un mínimo error cuadrático entre ambas respuestas.

En este apartado se va a obtener un controlador LQR a partir de esos parámetros, se va a comprobar cómo es capaz de controlar en nuestro modelo matemático y posteriormente se comprobará si es capaz de controlar en el péndulo real, lo que dependerá de que tan buena sea la identificación que hemos obtenido.

3.2.1 Obtención de un controlador LQR

Para obtener la ganancia óptima del controlador LQR, previamente hemos de linealizar las ecuaciones de movimiento del mismo. Como ya se mencionó anteriormente, el péndulo real y el péndulo del modelo matemático no tienen el mismo sistema de referencia para la coordenada del péndulo y esto es algo que debe ser considerado, pero como el control LQR funciona solo cerca del punto de funcionamiento, obtenemos lo mismo en ambos sistemas de referencia.

Considerando las variables con un subíndice 2 como aquellas del péndulo real, con el ángulo $\theta_2 = 0$ cuando el péndulo está vertical hacia abajo, y sin subíndice para el modelo matemático, con θ=0 en la posición vertical hacia arriba del péndulo, ambas definidas en el sentido de la regla de la mano derecha, tenemos,

$$\theta_2 = \theta + \pi \quad \dot{\theta}_2 = \dot{\theta} \quad \ddot{\theta}_2 = \ddot{\theta}$$

$$\varphi_2 = \varphi \quad \dot{\varphi}_2 = \dot{\varphi} \quad \ddot{\varphi}_2 = \ddot{\varphi}$$

Usando el cambio anterior y linealizando llegamos a las siguientes ecuaciones,

$$(\beta - \alpha^2)\ddot{\theta}_2 = \beta(\theta_2 - \pi) - \gamma\alpha u \; - \; C_p\dot{\theta}_2$$

$$(\beta - \alpha^2)\ddot{\varphi}_2 = -\alpha(\theta_2 - \pi) - \gamma u \; - \; C_a\dot{\varphi}_2$$

Como queremos controlar entorno a $\theta_2 = \pi$, es como si θ fuera la variable incremental, expresándolo en función de θ llegamos a la misma expresión que en el apartado 2.3.3,

$$(\beta - \alpha^2)\ddot{\theta} = \beta\theta - \gamma\alpha u \; - \; C_p\dot{\theta}$$

$$(\beta - \alpha^2)\ddot{\varphi} = -\alpha\theta - \gamma u \; - \; C_a\dot{\varphi}$$

Escribiendo la ecuación en espacio de estados y usando los parámetros de la identificación se obtienen las matrices A y B.

$$A = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & -2.4288 & -8.2789 & 0 \\ 0 & 1.75 & 0 & -6.2069 \end{bmatrix}$$

$$B = \begin{bmatrix} 0 \\ 0 \\ 12.5623 \\ -3.8792 \end{bmatrix}$$

A continuación hay que seleccionar una matriz Q y otra R, en este caso se han seleccionado las siguientes,

$$Q = \begin{bmatrix} 5 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 20 \end{bmatrix}$$

R=1

Usando esos parámetros se obtiene la siguiente ganancia óptima tras usar el comando *lqr*,

$$K = [-2.2361 \quad -12.5120 \quad 0.0261 \quad -4.2876]$$

Si usamos esta ganancia, figura 3-40, sobre el modelo matemático se obtiene la respuesta que se puede apreciar en la figura 3-41 y 3-42.

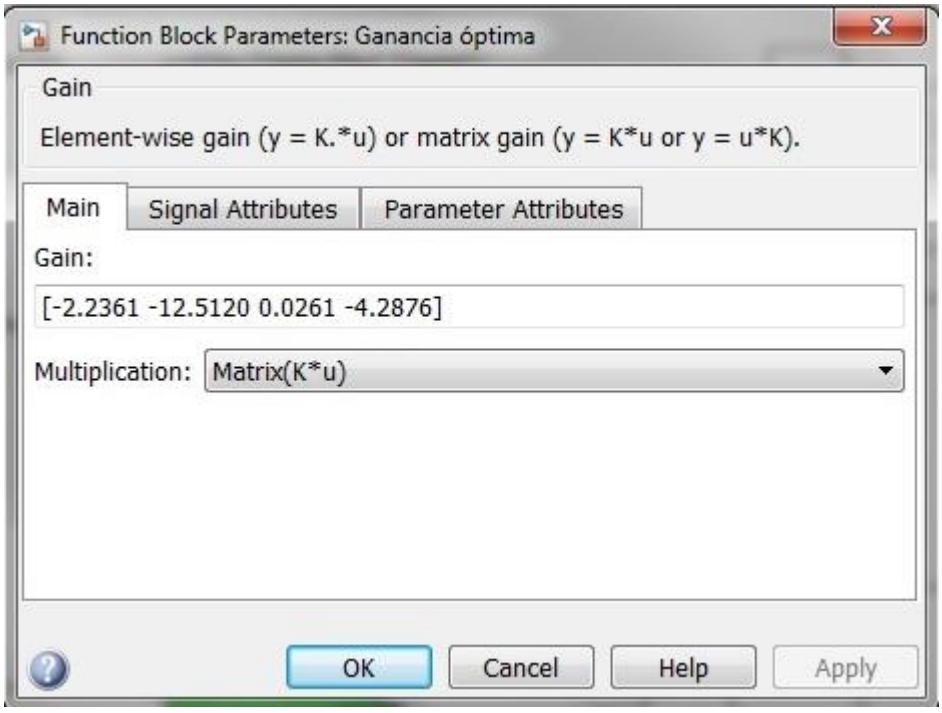

Figura 3-40. Ganancia óptima para control LQR.

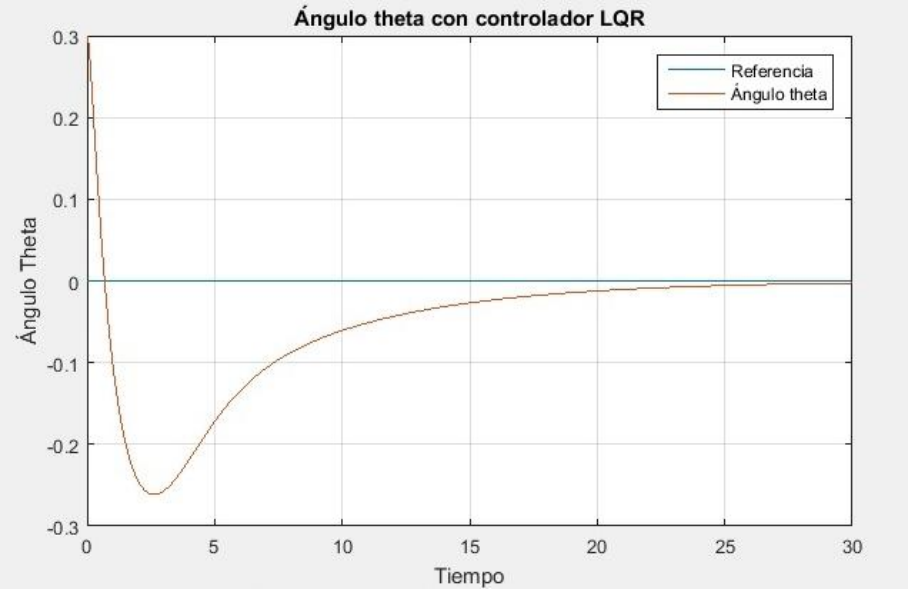

Figura 3-41. Ángulo theta con control LQR en modelo matemático con los parámetros identificados.

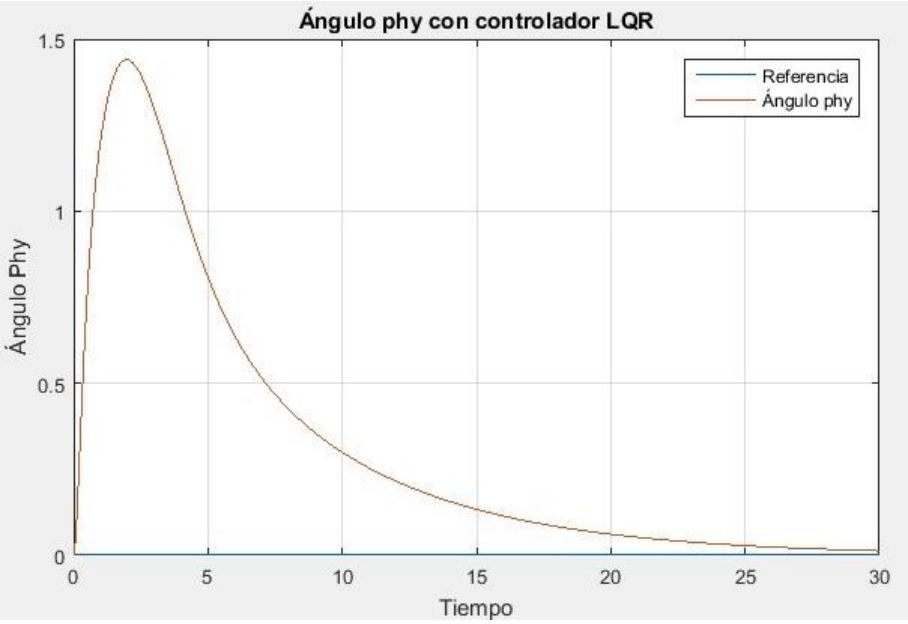

Figura 3-42. Ángulo phy con control LQR en modelo matemático con los parámetros identificados.

3.2.2 Resultados del control

Se usa ahora el controlador obtenido en el apartado anterior sobre el péndulo real de *QUBE-Servo* para ver si se consigue controlar correctamente.

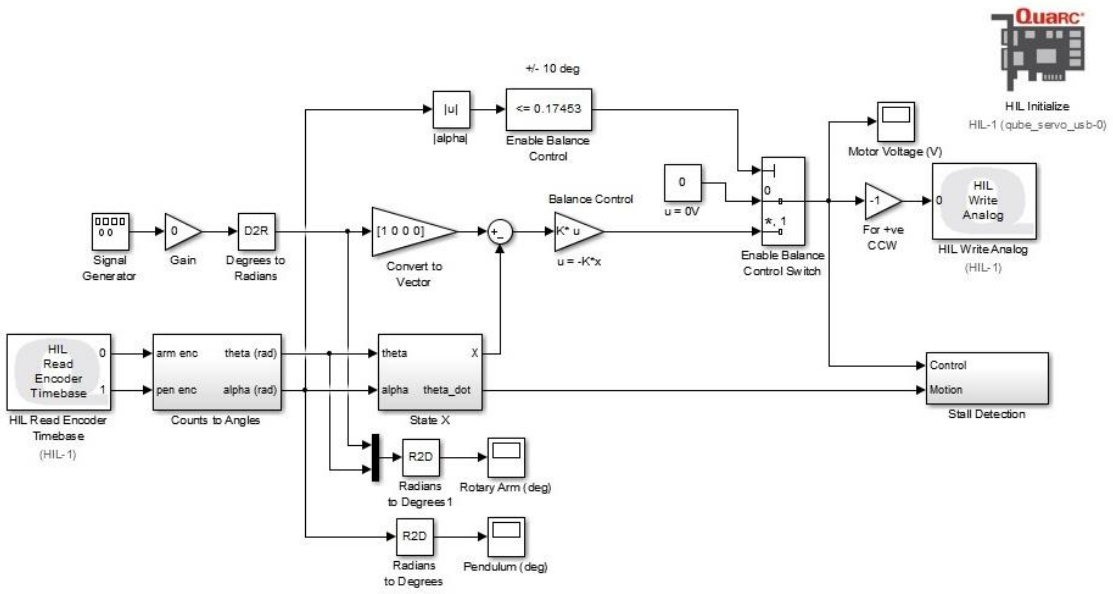

Figura 3-43. Control LQR en el *QUBE-Servo*.

Se consigue que el péndulo esté controlado durante un segundo aproximadamente, aunque termina cayendo. Esto es coherente con la identificación obtenida, ya que como se pudo observar la identificación era buena durante un corto periodo de tiempo a partir del cual el comportamiento del modelo real y el modelo matemático identificado divergían considerablemente, debido principalmente a la complejidad del sistema a identificar, cuya dinámica no parece estar completamente recogida en el modelo matemático usado.

A continuación se muestra el resultado obtenido, se puede ver que controla aproximadamente bien durante un breve periodo de tiempo pero termina cayendo, debido a que la identificación no es completamente satisfactoria debido al modelo del sistema.

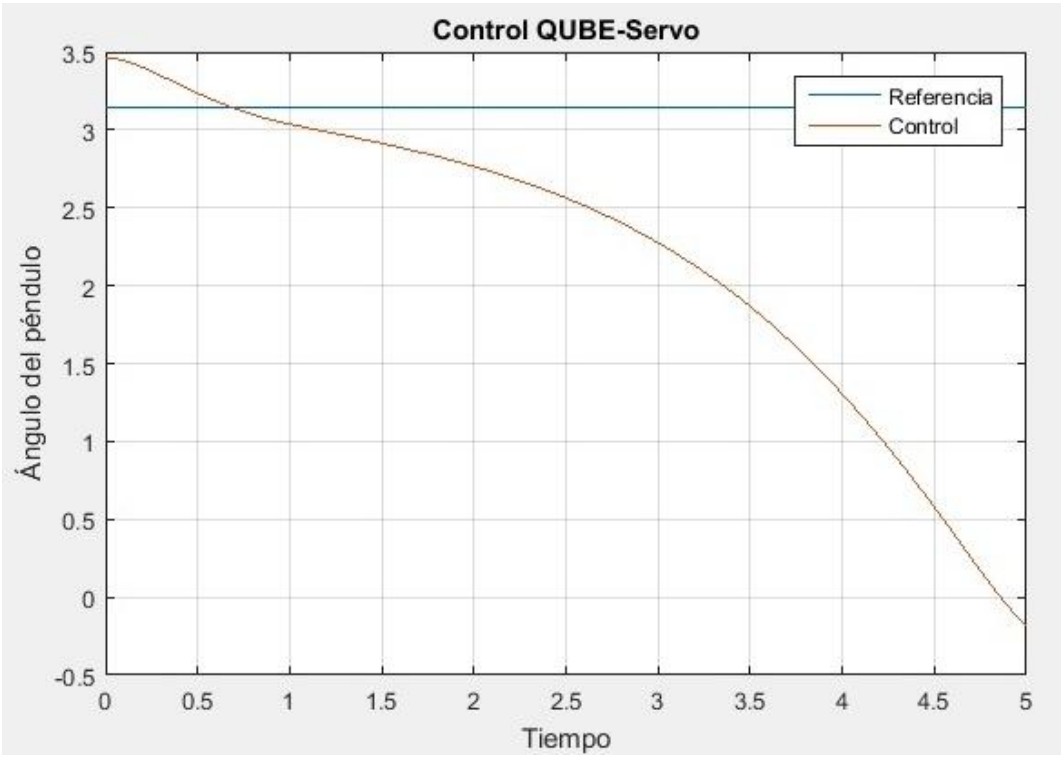

Figura 3-44. Prueba en péndulo real.

4 CONCLUSIONES Y TRABAJOS FUTUROS

No es el conocimiento, sino el acto de aprendizaje, y no la posesión, sino el acto de llegar allí, lo que concede el mayor disfrute.

- Carl Friedrich Gauss -

En este trabajo se propone en primer lugar un modelo matemático que representa de una manera coherente el comportamiento del péndulo de Furuta. Dicho modelo se ha desarrollado en *Simulink*, permitiéndose que cualquier persona que quiera hacer prácticas de control con un sistema complejo y con dinámicas muy rápidas pueda usar directamente el modelo para poner a prueba sus conocimientos. Además se ha creado un modelo en tres dimensiones del péndulo, que una vez conectado a *Simulink* proporcionará una importante ayuda visual a la hora de ver el comportamiento del sistema.

Se han desarrollado diversas técnicas de control para poder controlar el sistema y también una técnica de control híbrido que lleva el sistema cerca de la zona de funcionamiento interviniendo luego otra técnica de control distinta.

Se puede decir que los controladores proporcionan un resultado francamente bueno, esto añadido al modelo tridimensional nos permite ver de una forma amigable el comportamiento del péndulo.

La identificación de sistemas es una parte esencial, que puede llegar a ser muy compleja, en la ingeniería de control. En este trabajo se ha intentado identificar el péndulo de Furuta, un sistema fuertemente no lineal y que además contiene dos parámetros de fricción viscosa que son realmente muy complejos de identificar. Para identificar el sistema se ha realizado una comparación entre el sistema real y el modelo matemático para una misma entrada, intentando obtener qué parámetros del modelo matemático proporcionan menor diferencia cuadrática entre la respuesta de ambos sistemas.

Tras diversas técnicas de identificación se han obtenido unos parámetros que permiten modelar el comportamiento del sistema con cierta validez pero durante tiempos 0,1 ó 0,2 segundos. Una validez durante tan breve periodo de tiempo puede deberse principalmente a que el modelo matemático que se contempla no contiene toda la dinámica del péndulo. Con los parámetros identificados se ha determinado una ganancia óptima y se ha probado sobre el modelo matemático, con un excelente resultado, y sobre el modelo real consiguiéndose que controle como se pudo apreciar.

En trabajos futuros se podría tratar de obtener un modelo matemático más complejo que describiera mejor la dinámica del sistema, de esta forma al tratar de identificar el péndulo real quizás se podrían obtener unos resultados más satisfactorios. También podría ser interesante tratar de construir un péndulo que podríamos identificar y tratar de controlar a través de una placa tipo *arduino*, ya que no se necesitarían demasiados materiales y el péndulo de *QUBE-Servo* tiene un precio relativamente alto.

5 ANEXOS

En el presente anexo se incluyen los códigos de los dos programas de identificación que se usaron en los apartados 3.2.2 y 3.2.3.

Tabla 5-1.Identificación a un paso.

Programa de identificación a un paso.

```
% Identificador de parámetros :
% Identificación de parámetros del péndulo educacional QUBE-Servo. Trabajo
% Final de Grado: Control y Simulación del Péndulo de Furuta. Autor: Carlos
% Regalo Núñez.

%Se van a identificar los parámetros r~0.085, l~0.129/2, m~0.024, Km~0.0365,
%Cp~0.5 y Ca~0.5 del péndulo de Furuta. Probaremos valores del +-10% del
%valor supuesto para los 4 primeros parámetros y del +300% para los dos
%últimos.

% Cargamos los parámetros REALES de los archivos correspondientes que
% dependerá de si probamos con la entrada seno o aleatoria:


load('entrada_pendulo_seno.mat');
load('phy_angulo_brazo_seno.mat');
load('theta_angulo_pendulo_seno.mat');

%Valores entorno a los que probamos:
r=0.0688;l=0.0781;m=0.0251;Km=0.0356;Cp=8;Ca=1.7188;
% Valores de los parámetros que vamos a probar:
r_val=linspace(0.9*r,1.1*r,5);
l_val=linspace(0.9*l,1.1*l,5);
m_val=linspace(0.9*m,1.1*m,5);
Km_val=linspace(0.9*Km,1.1*Km,5);
Cp_val=linspace(0.5*Cp,1.5*Cp,5);
Ca_val=linspace(0.5*Ca,1.5*Ca,5);
g=9.8; Ja=0;

%Cálculo del error para cada conjunto de valores:
vectorvalores=[];pasada=0;
resolucion=0.005; % Resolucion para sacar el error.
intervalodeseado=[0:resolucion:5];
phyrealinterpolado=spline(phy_angulo_brazo.time,phy_angulo_brazo.signals.valu
es,intervalodeseado);
thetarealinterpolado=spline(theta_angulo_pendulo.time,theta_angulo_pendulo.si
gnals.values,intervalodeseado);
for i=1:5
    for j=1:5
        for k=1:5
            for n=1:5
                for p=1:5
                    for q=1:5
                        r=r_val(i); l=l_val(j); m=m_val(k);
                        Km=Km_val(n); Cp=Cp_val(p); Ca=Ca_val(q);
                        %Simulamos con esos valores:
                        [t,x]=sim('TFG_14PENDULOENMASCARADO.slx',5);
                        %Interpolacion de las salidas:
```

```
simoutA21phyinterpolado=spline(t,simoutA21phy,intervalodeseado);

simoutA21thetainterpolado=spline(t,simoutA21theta,intervalodeseado);
                        % Cálculo de los errores al cuadrado:
                        errorphy=sum((phyrealinterpolado-
simoutA21phyinterpolado).^2);
                        errortheta=sum((thetarealinterpolado-
simoutA21thetainterpolado).^2);
                        error=errorphy+errortheta;
                        %Guardamos error y valores para dicho error:
                        errorypos=[i;j;k;n;p;q;error];
                        vectorvalores=[vectorvalores,errorypos];
                        pasada=pasada+1;
                        disp(pasada);
                    end
                end
            end
        end
    end
end

% Buscar que valor ofrece el minimo error:
errormin=min(vectorvalores(7,:));
[m2,n2]=size(vectorvalores);
for i=1:n2
    if vectorvalores(7,i)==errormin
        flag=i;
    end
end

r=r_val(vectorvalores(1,flag));
l=l_val(vectorvalores(2,flag));
m=m_val(vectorvalores(3,flag));
Km=Km_val(vectorvalores(4,flag));
Cp=Cp_val(vectorvalores(5,flag));
Ca=Ca_val(vectorvalores(6,flag));
disp(r),disp(l),disp(m),disp(Km),disp(Cp),disp(Ca),disp(errormin);

% Representacion del resultado:
[t,x]=sim('TFG_14PENDULOENMASCARADO.slx',5);% Ejecutamos con los valores
deseados

%Representación de Phy
figure(1);
subplot(3,1,1);
plot(t,simoutA21phy);
grid;
hold on;
plot(phy_angulo_brazo.time,phy_angulo_brazo.signals.values);
legend('ModeloA21','QUBE-Servo');
xlabel('tiempo');ylabel('angulo Phy');

% Representación de theta
subplot(3,1,2);
plot(t,simoutA21theta);
grid;
hold on;
plot(theta_angulo_pendulo.time,theta_angulo_pendulo.signals.values+pi);
legend('ModeloA21','QUBE-Servo');
xlabel('tiempo');ylabel('angulo Theta');
```

```
%Representación de la entrada
subplot(3,1,3);
plot(t,simoutA21entrada,'r--');
grid;
hold on;
plot(entrada_pendulo.time,entrada_pendulo.signals.values,'b');
legend('ModeloA21','QUBE-Servo');
xlabel('tiempo');ylabel('entrada U');

%Figura con valores de interpolacion:
figure(2);
subplot(2,1,1);
intervalodeseado=[0:resolucion:5];
simoutA21phyinterpolado=spline(t,simoutA21phy,intervalodeseado);
plot(intervalodeseado,simoutA21phyinterpolado,'or',t,simoutA21phy,'r');grid;
hold on;
plot(intervalodeseado,phyrealinterpolado,'ob',phy_angulo_brazo.time,phy_angul
o_brazo.signals.values,'b');grid;
grid;
xlabel('tiempo');ylabel('angulo phy');
legend('','Modelo A21','','QUBE-Servo');
title(['El error es: ',num2str(errormin),' ,con resolucion: ',
num2str(resolucion)]);%Titulo con el error.
subplot(2,1,2);
intervalodeseado=[0:resolucion:5];
simoutA21thetainterpolado=spline(t,simoutA21theta,intervalodeseado);
plot(intervalodeseado,simoutA21thetainterpolado,'or',t,simoutA21theta,'r');gr
id;
hold on;
plot(intervalodeseado,thetarealinterpolado+pi,'ob',theta_angulo_pendulo.time,
theta_angulo_pendulo.signals.values+pi,'b');grid;
grid;
xlabel('tiempo');ylabel('angulo theta');
legend('','Modelo A21','','QUBE-Servo');
```

Tabla 5-2. Identificación a N pasos.

Programa de identificación a N pasos.

```
%% Identificación usando error de predicción a un paso ó en bucle
% cerrado. Autor.: Carlos Regalo.
close all; clear all; clc;
%Tiempo de identificación:
%--------------------------------------------------------------------------
paso=input('Introduce el paso de identificación:');
%--------------------------------------------------------------------------
%Obtencion de la velocidad Theta punto y Phy punto.
load('theta_angulo_pendulo_aleatoria_ts.mat');theta_angulo_pendulo_aleatoria_
ts.Data=theta_angulo_pendulo_aleatoria_ts.Data+pi;
load('phy_angulo_brazo_aleatoria_ts.mat');
Td=0.04;%Tiempo de derivacion.
phyrealinterpolado=spline(phy_angulo_brazo_aleatoria_ts.Time,phy_angulo_brazo
_aleatoria_ts.Data,0:Td:5);
thetarealinterpolado=spline(theta_angulo_pendulo_aleatoria_ts.Time,theta_angu
lo_pendulo_aleatoria_ts.Data,0:Td:5);
N=length(phyrealinterpolado);
thetapunto=[];phypunto=[];
```

```
for i=1:N-1
    phypunto=[phypunto,((phyrealinterpolado(i+1)-phyrealinterpolado(i))/Td)];
    thetapunto=[thetapunto,((thetarealinterpolado(i+1)-
thetarealinterpolado(i))/Td)];
end
figure(1);%Posiciones y velocidades de QUBE-Servo.
subplot(2,1,1);
plot(phy_angulo_brazo_aleatoria_ts.Time,phy_angulo_brazo_aleatoria_ts.Data);g
rid;hold on;
plot(0:Td:5,[phypunto,0],'r');
title(['Angulo theta y phy y sus derivadas ']);
xlabel('tiempo');ylabel('phy , phypunto');
legend('phy','phy punto');
subplot(2,1,2);
plot(theta_angulo_pendulo_aleatoria_ts.Time,theta_angulo_pendulo_aleatoria_ts
.Data);grid;hold on;
plot(0:Td:5,[thetapunto,0],'r');
xlabel('tiempo');ylabel('theta , thetapunto');
legend('theta','theta punto');
%%
%Ahora hay que cargar el nuevo modelo Simulink, que cada periodo de tiempo
%'paso' debe partir desde las condiciones reales de péndulo QUBE en ese
%momento, para ello el tiempo de simulación del nuevo modelo debe ser igual
%al 'paso'.

% 1) Interpolación, usando spline, de phy, theta y sus derivadas en un
% intervalo 'paso'
phyrealinterpolado=spline(phy_angulo_brazo_aleatoria_ts.Time,phy_angulo_brazo
_aleatoria_ts.Data,0:paso:5);
thetarealinterpolado=spline(theta_angulo_pendulo_aleatoria_ts.Time,theta_angu
lo_pendulo_aleatoria_ts.Data,0:paso:5);
phypuntorealinterpolado=spline(0:Td:5,[phypunto,0],0:paso:5);
thetapuntorealinterpolado=spline(0:Td:5,[thetapunto,0],0:paso:5);
% 2) Simular el modelo durante el tiempo de 'paso':
%parámetros de partida.
r=0.0619;l=0.0859;m=0.0238;Km=0.032;Cp=0.7895;Ca=1.0526;g=9.8; Ja=0;

% Bucle que ejecute la simulacion para cada tramo de 'paso'
L=length(phyrealinterpolado)-1;% Numero de tramos son el número de puntos-1.
phyA21=[phyrealinterpolado(1)];
thetaA21=[thetarealinterpolado(1)];

load('entradaaleatoria_ts.mat')%Creada previamente.
ts=timeseries;
ts=ans;
clear ans;

phyA21bis=[];
thetaA21bis=[];
ts_completo_tiempo=[];
ts_completo_datos=[];
for i=1:L
    empieza=(i-1)*paso;
    acaba=i*paso;
    ts2=getsampleusingtime(ts,empieza,acaba);
    ts_completo_datos=[ts_completo_datos;ts2.Data];
    ts_completo_tiempo=[ts_completo_tiempo;ts2.Time];
    ts2=setuniformtime(ts2,'StartTime',0,'Interval',0.002);
    save ('ts2.mat','ts2','-v7.3');
```

```
    Theta0=thetarealinterpolado(i);Thetapunto0=thetapuntorealinterpolado(i);
    Phy0=phyrealinterpolado(i);Phypunto0=phypuntorealinterpolado(i);
    [t,x]=sim('TFG_16IdentificacionAUnPaso.slx',paso);
    %hemos de guardar el ultimo valor de phy y theta:
    phyA21=[phyA21,simoutA21phy(length(simoutA21phy))];
    phyA21bis=[phyA21bis;simoutA21phy];
    thetaA21bis=[thetaA21bis;simoutA21theta];
    thetaA21=[thetaA21,simoutA21theta(length(simoutA21theta))];
end


%Representación de resultados:
figure(2);
subplot(3,1,1);
plot(phy_angulo_brazo_aleatoria_ts.Time,phy_angulo_brazo_aleatoria_ts.Data);h
old on;
L2=length(phyA21bis);
plot(0:5/(L2-1):5,phyA21bis,'k','LineWidth',1);grid;
legend('QUBEServo','A21');xlabel('tiempo');ylabel('phy');
title(['predicción a un paso ']);
subplot(3,1,2);
plot(theta_angulo_pendulo_aleatoria_ts.Time,theta_angulo_pendulo_aleatoria_ts
.Data);hold on;
L2=length(thetaA21bis);grid;
plot(0:5/(L2-1):5,thetaA21bis,'k','LineWidth',1);
legend('QUBEServo','A21');xlabel('tiempo');ylabel('theta');
subplot(3,1,3);
plot(ts_completo_tiempo,ts_completo_datos,'g','LineWidth',2);grid;
legend('tension');xlabel('tiempo');ylabel('entrada');

figure(3);
plot(phy_angulo_brazo_aleatoria_ts.Time,phy_angulo_brazo_aleatoria_ts.Data,'b
');hold on;grid;
L2=length(phyA21bis);
plot(0:5/(L2-1):5,phyA21bis,'k','LineWidth',1);
legend('QUBEServo','Modelo A21');xlabel('tiempo');ylabel('phy');
title(['predicción a un paso de ',num2str(paso),' segundos']);
```

6 REFERENCIAS

Tesis doctoral. Bifurcaciones en sistemas de control no lineales. Daniel Juan Pagano. Sevilla, junio de 1999.

Furuta, K., Yamakita, M. and Kobayashi, S. (1992) "Swing-up control of inverted pendulum using pseudostate feedback", Journal of Systems and Control Engineering, 206(6), 263-269.