UNIVERSITAT POLITÈCNICA DE VALÈNCIA

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA DEL DISEÑO

GRADO EN INGENIERÍA ELECTRÓNICA INDUSTRIAL Y AUTOMÁTICA

TRABAJO FIN DE GRADO

Diseño de un prototipo virtual web para el control de un aeropéndulo

Autor:

Héctor De La Guía González

Tutor:

Antonio González Sorribes

Septiembre 2020

A toda mi familia, en especial a mis padres que me han dado la oportunidad de estudiar esta carrera.

A mis amigos, los de toda la vida y los que me llevo para siempre de esta etapa.

Especialmente a mi abuela Rafa, gracias por apoyarme y ayudarme en todo lo posible durante toda mi vida.

RESUMEN:

En el presente proyecto se lleva a cabo el diseño, modelado y control de un prototipo de un sistema motor – hélice – balancín, así como la implementación de una aplicación web para la visualización y estudio mediante la variación de parámetros del mismo.

Un sistema motor – hélice – balancín, conocido como aeropéndulo, consiste en un sistema que consta con una articulación con un grado de libertad, que, con la aplicación de una fuerza en el extremo del péndulo, realizará un movimiento de giro alrededor del eje horizontal contenido en el mismo, obteniendo un movimiento en el plano vertical.

La fuerza que produce el movimiento viene generada por una hélice conectada a un motor en el extremo del balancín, de modo que, a mayor velocidad de giro, más fuerza de empuje se genera.

El objetivo será realizar un control de dicha fuerza de empuje para la obtención del ángulo deseado entre el balancín y el eje vertical que forma cuando no actúa ninguna fuerza externa sobre él. Para esta tarea se calculará un regulador y se verificará mediante Simulink.

Por otra parte, la aplicación web consistirá en la creación de un sencillo entorno en el cual el usuario pueda modificar los parámetros que conforman el regulador para observar las variaciones en el comportamiento de la planta.

ABSTRACT:

This project consists of the design, model and control of a prototype of a motor – propeller – rocker system, and the subsequent implementation of a web application for its visualization and study by varying its parameters.

A motor – propeller – rocker system, also known as an air pendulum consists of a system with a joint with a degree of freedom, which with the application of a force at the end of the pendulum, will make a turning movement around the contained horizontal axis, in it, obtaining a movement in the vertical plane.

The force that produces the movement ins generated by a propeller connected to a motor at the end of the rocker, so that the higher the turning speed, the more thrust is generated.

The objective will be to control said thrust force to obtain the desired angle between the rocker arm and the vertical axis that it forms when no external force acts on it. For this task a regulator will be calculated and verified using Simulink.

On the other hand, the web application will consist of creating a simple environment in which the user can modify the parameters that make up the regulator to observe the variations in the behavior of the plant.

MEMORIA

HECTOR DE LA GUIA GONZALEZ

INDICE GENERAL

INDICE DE FIGURAS

1. OBJETO

El proyecto que se describe en el presente documento se encuentra ligado al entorno académico, pues forma parte de un proyecto desarrollado por diferentes universidades españolas, entre ellas la UPV, en colaboración con la UNED. El objetivo final de este proyecto es la creación de una serie de diferentes equipos de laboratorio con el fin de poder hacer uso de ellos a distancia, de modo que un alumno pueda hacer uso de un equipo que se encuentre en una universidad distinta a la suya.

Para la realización de dicho proyecto, en primera instancia se van a desarrollar una serie de aplicaciones web, entre las cuales se encuentra la desarrollada en este proyecto.

Concretamente, este caso la aplicación tratará de un sistema de control de posición para un aeropéndulo, con el fin de controlar el ángulo que forma el mismo con su vertical, siendo esta su posición en reposo.

Para su realización, se realizará en primer lugar un modelado del sistema sobre el cual se realizará el diseño de un controlador en base a unos requerimientos para su posterior validación mediante la aplicación web.

2. ESTUDIO DE NECESIDADES, FACTORES A CONSIDERAR

El proyecto se divide esencialmente en dos partes. Por una parte, el modelado del sistema y la implementación de un controlador, y por otra la creación de un entorno web para validar el control mediante una maqueta virtual. Por tanto, el estudio de necesidades puede dividirse también según estas partes.

2.1. Estudio de necesidades para el modelado y control

Para el modelado del sistema, se debe tener en cuenta que este será esencialmente un modelado teórico. Por tanto, además de realizar correctamente el modelo y la linealización del mismo, será importante elegir correctamente las constantes que compondrán el modelo (Momento de inercia, constante de rozamiento, etc.). Para ello, se tomarán como referencia otros trabajos experimentales ya realizados y se será lo más fiel posible a los resultados obtenidos.

Por otra parte, se deberán establecer unos requerimientos a cumplir para realizar en base a ellos el controlador, véanse tiempo de establecimiento, sobreoscilación, error de posición, etc.

2.2. Estudio de necesidades para la aplicación web

A la hora del diseño de la aplicación web, se debe tener en cuenta los elementos de los que tiene que constar la misma. Dichos elementos son los siguientes:

- Visualizador de la maqueta, donde pueda verse la posición del aeropéndulo.
- Elementos que permitan variar los parámetros del controlador (barras deslizantes, etc.)
- Visualizadores de valores (Valor del ángulo formado por el aeropéndulo, de los parámetros del controlador, etc.

3. PLANTEAMIENTO DE SOLUCIONES ALTERNATIVAS Y JUSTIFICACIÓN DE LA SOLUCIÓN ADOPTADA.

El proyecto presenta algunas soluciones alternativas a las elegidas en muchas de las partes en las que se divide.

En primer lugar, para la elección de los parámetros se han tenido en cuenta diferentes estudios, entre los cuales han destacado dos. Los primeros parámetros que se han propuesto han sido los empleados en una práctica del departamento ISA (Ingeniería de Sistemas y Automática) de la universidad de Oviedo, en la que tratan el control de posición de un péndulo simple.

No obstante, finalmente las constantes se han obtenido de un artículo de la revista de investigación ECORFAN. Dicho artículo recibe el nombre de "*Sintonización y comparación de controladores para un aeropéndulo*", desarrollado por Gregorio Castillo, Elda Gómez, Elisa Gonzaga e Iván Reyes.

Se ha optado por elegir este estudio al ser este sobre un aeropéndulo propio, por lo que las constantes serán más precisas que sobre un péndulo simple.

Además, para la obtención del modelo, se puede realizar de dos formas distintas, mediante las fuerzas que actúan en el sistema, o mediante los momentos que estas generan. Se ha realizado por momentos por ser este método más sencillo.

En el apartado del controlador, se ha optado por implementar un PID. Para sintonizar este tipo de controladores hay muchas opciones. En este caso la estrategia seguida ha sido cancelar los polos de la función de transferencia del sistema que determinaban la dinámica del mismo, y establecer otros para que se comporte de la manera deseada.

Por último, se ha optado por usar Easy Java Simulations para el desarrollo de la aplicación web por ser esta una aplicación que presenta unas características muy favorables para el entorno docente. La aplicación cuenta con un sencillo e intuitivo modo de crear interfaces visuales en las que implantar los modelos. Además, al exportar las interfaces web en formato .HTML resulta compatible con la mayoría de sistemas, pudiendo integrarse normalmente en los navegadores de internet.

4. DESCRIPCIÓN DETALLADA DE LA SOLUCIÓN ADOPTADA

4.1. Análisis de la planta.

Un aeropéndulo consiste en un sistema formado esencialmente por dos partes unidas entre sí en un eje que permite el giro de una de ellas. Por tanto, consta de una parte móvil, formada por un sistema de motor + hélice en un extremo y unida a la parte fija mediante el eje de giro en el otro extremo. De este modo, el sistema se comporta de forma similar a un péndulo simple, con la única diferencia de que la posición depende del sistema motor + hélice de la parte móvil.

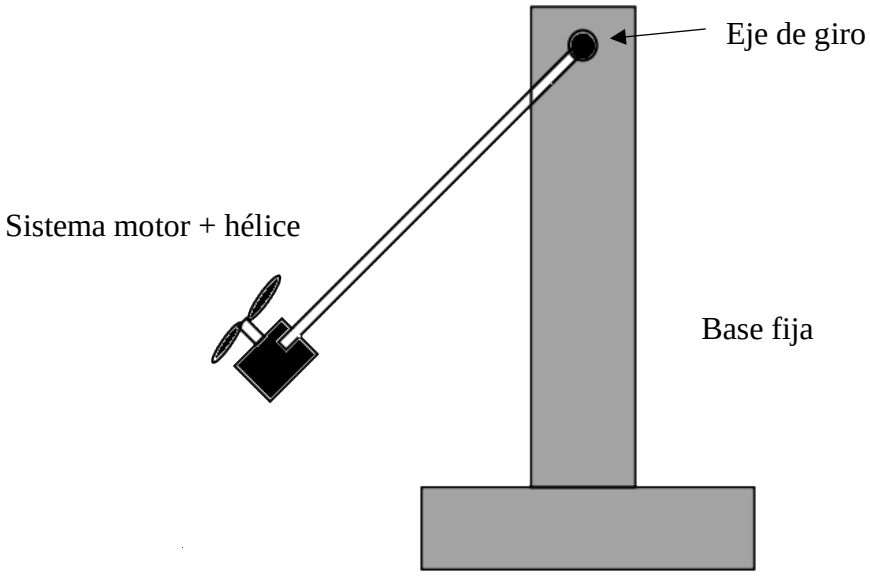


Figura 1: Representación de la planta del aeropéndulo

Como se ha comentado antes, para obtener el modelo del sistema, se ha partido de determinar un ángulo como punto de equilibrio. En dicho punto de equilibrio, por definición física el sumatorio tanto de fuerzas como de momentos es nulo. Sabiendo esto, se realiza un análisis en base al Teorema del Momento Angular a partir del cual, mediante desarrollo en series de Taylor y pasando al dominio de Laplace se obtendrá la función de transferencia del sistema.

En el siguiente apartado, en el que se tratarán los cálculos más detalladamente, se desarrollará el procedimiento seguido.

4.2. Controlador.

Con el objetivo de que la planta del aeropéndulo presente el comportamiento deseado, implantaremos un controlador que modifique su función de transferencia entrada/salida y cumpla así las especificaciones tanto dinámicas como estáticas deseadas.

Existen dos procesos distintos para el proceso de obtención del controlador o regulador, el método de síntesis y el método de análisis. En este caso se usará el método de síntesis, que consiste en calcular directamente el regulador necesario a partir de la función de transferencia de la planta y de la función de transferencia deseada.

El tipo de regulador empleado será un regulador PID, el tipo de controlador más empleado en la industria para todo tipo de procesos.

4.2.1. Regulador PID

Un regulador PID es un mecanismo de control simultaneo por realimentación muy usado en el ámbito industrial. Se basa en calcular la desviación entre un valor medido y un valor deseado.

Recibe el nombre de PID por las tres posibles formas de actuación que puede aplicar, conocidas también como acciones básicas de control. Estas son, respectivamente, proporcional, integral y derivada.

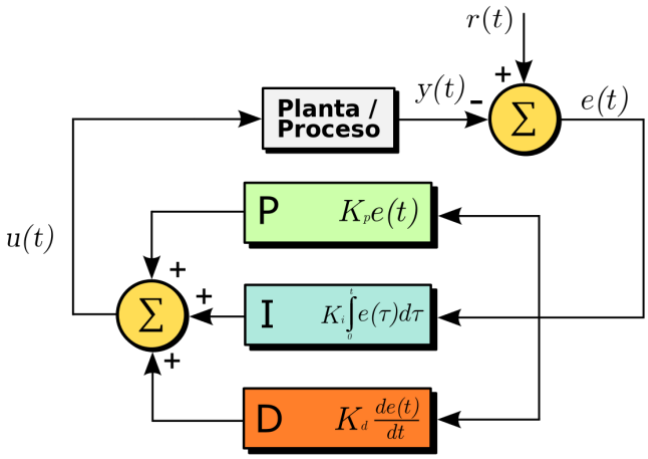


Figura 2: Estructura de un controlador PID

- Acción proporcional:

Recibe este nombre porque su valor es proporcional a la señal de error en cada instante de tiempo. De este modo, cuando el error es grande la acción es grande y cuando el error es pequeño la acción es pequeña. El parámetro que caracteriza la acción proporcional es la ganancia proporcional, normalmente expresada como K_p. Dicha constante es la que multiplica al error para obtener la acción de control, de la siguiente manera:

$$u_P(t) = K_P \cdot e(t)$$

Normalmente, si la ganancia proporcional es pequeña el sistema presentará respuestas lentas y sin oscilaciones. En cambio, si esta es grande presentará respuestas rápidas y con tendencia a la inestabilidad.

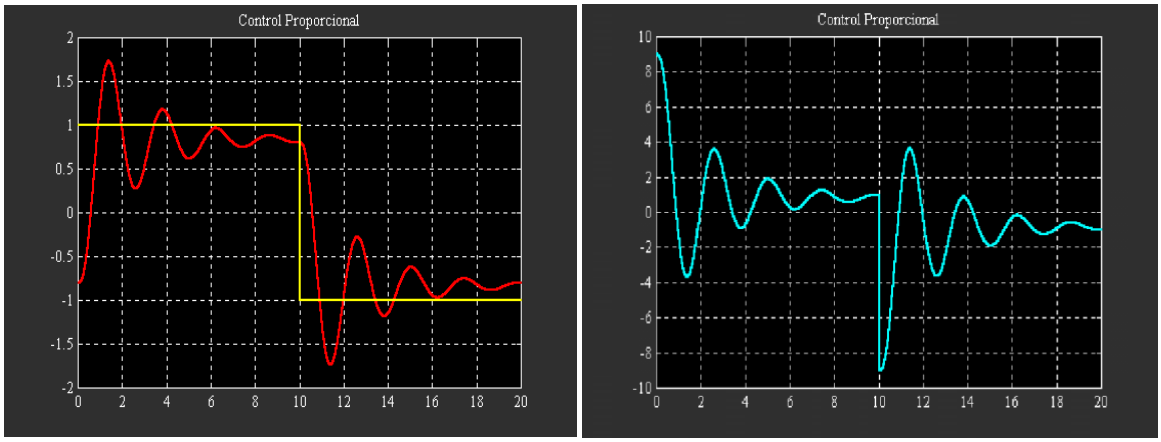


Figura 3: Control Proporcional

- Acción integral:

La primera limitación de la acción proporcional es que no tiene en cuenta la duración del error. Por tanto, al acercarse la variable medida al valor deseado, el error será pequeño y por tanto la acción proporcional también. Esto puede llevar a que la acción proporcional no sea capaz de modificar la variable de interés y por tanto no alcance el valor deseado.

En cambio, la acción integral viene determinada por el error acumulado. La acción integral recibe este nombre porque es proporcional a la integral de la señal de error. El parámetro que determina el valor de la acción integral en el controlador es la ganancia integral K_I:

$$u_I(t) = K_I \cdot \int_{\tau=0}^{t} e(\tau)d\tau$$

La acción integral es pequeña cuando hay poco error acumulado y aumenta a medida que pasa el tiempo sin eliminar el error, hasta que el error deja de existir. Por ello, se puede decir que la acción integral es un control de pasado, ya que no tiene en cuenta el valor instantáneo del error si no la acumulación del mismo en el tiempo.

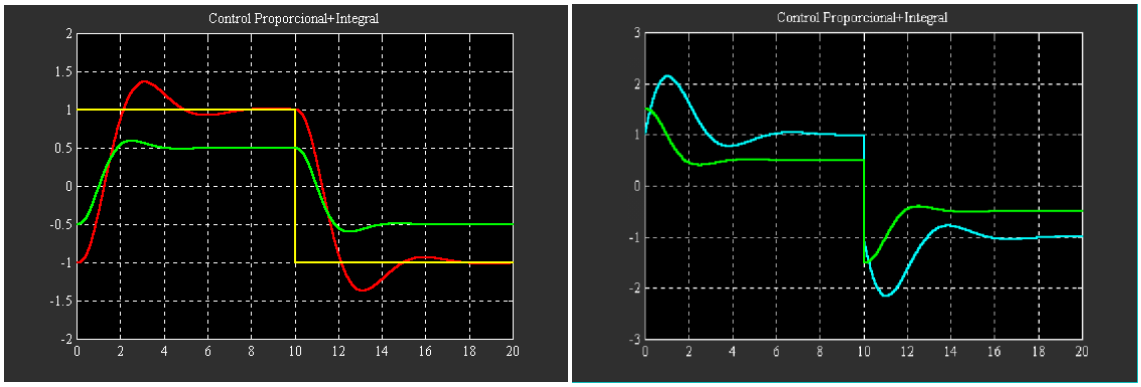


Figura 4: Control Proporcional + Integral

- Acción derivada

La segunda limitación del control proporcional es que no tiene en cuenta la tendencia del error, es decir, el error se trata de igual forma si varía lentamente que si lo hace con rapidez. El control proporcional solo puede corregir los errores después de que se hayan producido. Mediante la acción derivada, se tiene en cuenta la tendencia del error y se puede actuar sobre la planta para corregir los errores antes de que se produzcan.

La acción derivada es proporcional a la derivada del error, siendo la ganancia derivada K_D, el parámetro que determina el calor de la acción derivada.

$$u_D(t) = K_D \frac{d\,e(t)}{dt}$$

La derivada de la señal de error será pequeña cuando el sistema evoluciona lentamente, por tanto, la acción derivada será insignificante. En cambio, cuando la derivada del error es grande significa que el sistema evoluciona rápidamente y el valor de la acción derivada será significativo.

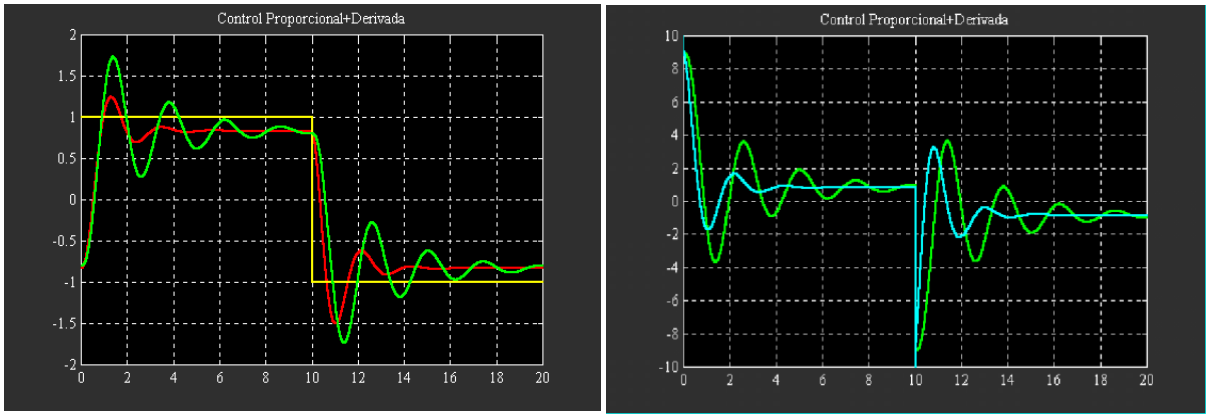


Figura 5: Control Proporcional + Derivada

4.3. Validación mediante MATLAB – Simulink.

MATLAB es un sistema de cómputo numérico que cuenta con su propio lenguaje de programación de alto nivel. Permite una sencilla visualización y procesado de señales, así como la simulación y aplicación de sistemas de control. Se trata de un software muy utilizado en universidades y centro de investigación.

El paquete MATLAB incluye también la herramienta Simulink. Dicha herramienta es una plataforma de simulación de diagramas de bloque usada para modelar y analizar sistemas. Permite la creación de modelos en los cuales simular sistemas lineales y no lineales, en tiempo continuo o discreto, etc., mediante diagramas de bloques y una interfaz gráfica que hace su uso muy intuitivo.

4.3.1. Versión de MATLAB utilizada

En la etapa de validación y simulación de los modelos se ha empleado la versión MATLAB R2017B. En la siguiente figura se muestra la versión de MATLAB utilizada, así como su herramienta Simulink con uno de los modelos estudiados y las librerías que ofrece esta herramienta.

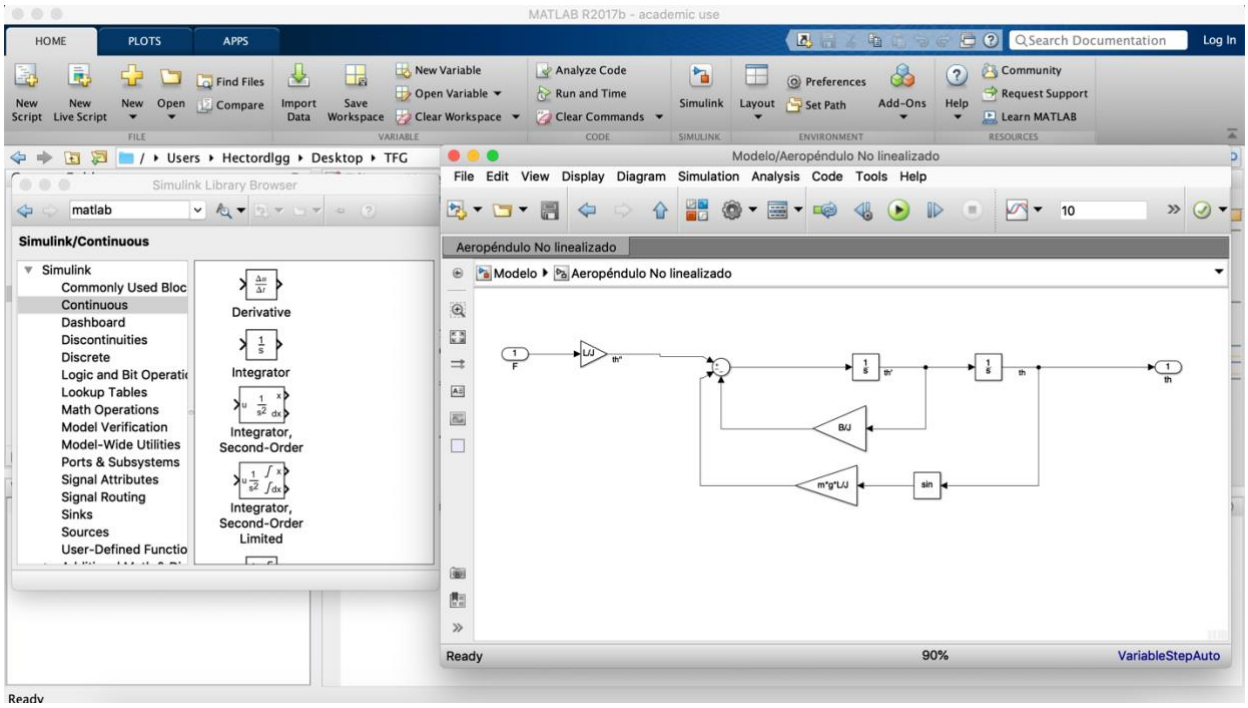


Figura 6: Versión de MATLAB usada en el proyecto.

4.3.2. Modelo Simulink

Como ya se ha comentado anteriormente, la herramienta Simulink de MATLAB nos permite crear modelos a partir de los diferentes bloques de los que disponemos en sus librerías.

Concretamente, en este proyecto la funcionalidad de estos modelos será la comparación entre el modelo linealizado obtenido teóricamente, con un modelo no lineal que construiremos a partir de los bloques comentados.

Además, servirá para validar el regulador calculado, comprobando que se cumplan las especificaciones estáticas y dinámicas propuestas, y hacer una comparación de las respuestas obtenidas mediante el modelo no lineal y el linealizado.

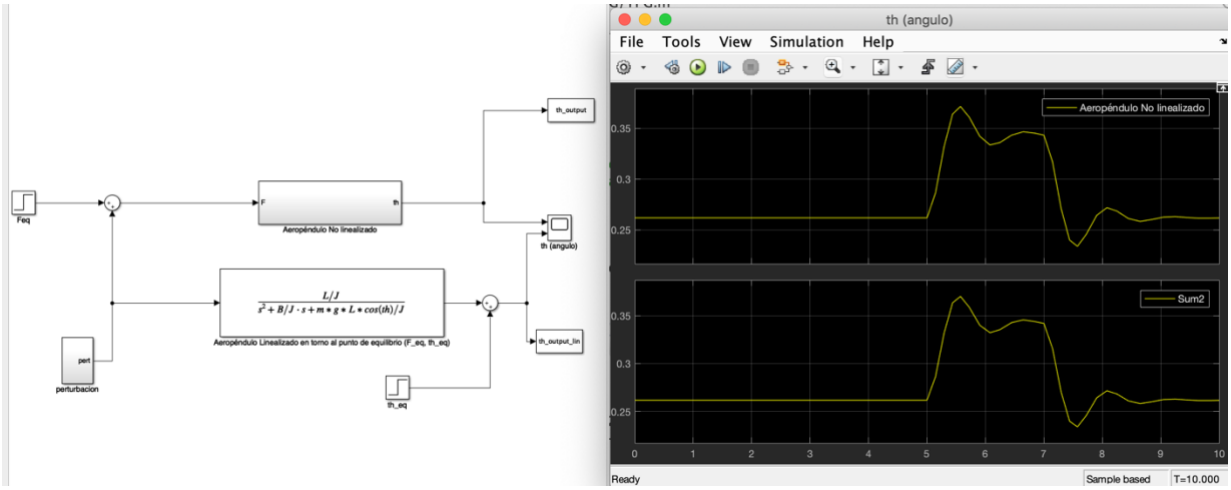


Figura 7: Comparación de modelos Lineal y No Lineal en Simulink

4.4. Desarrollo de prototipo virtual

El objetivo de una aplicación web sobre el control del aeropéndulo consiste en crear un entorno sencillo e intuitivo para que el usuario pueda interactuar fácilmente con él. Para ello, la herramienta usada será Easy Java Simulations.

4.4.1. Easy Java Simulations:

Easy Java Simulations es una herramienta de software de código abierto basada en Java. Su principal función es la realización de simulaciones por computador discretas, es decir programas cuyo objetivo es reproducir fenómenos físicos naturales a partir de una serie de algoritmos dados.

La principal ventaja de esta aplicación es que las simulaciones están creadas en formato JavaScript. Esto supone una ventaja al estar las máquinas virtuales Java presentes en muchas plataformas diferentes. Además, el programa está diseñado para que el formato en el que se guarda no incluya solo el código para la simulación, si no otros elementos como la introducción html, que permite integrarse en la mayoría de navegadores de internet.

4.4.2. Componentes de Easy Java Simulations:

Al ejecutar Easy Java Simulations se abren dos ventanas diferentes. La primera es la Consola de EjsS, que contiene opciones para variar el aspecto gráfico, cambiar el espacio de trabajo, opciones para cargar actualizaciones, etc. También contiene un pulsador para ejecutar la interfaz de Easy Java Simulations (segunda ventana comentada).

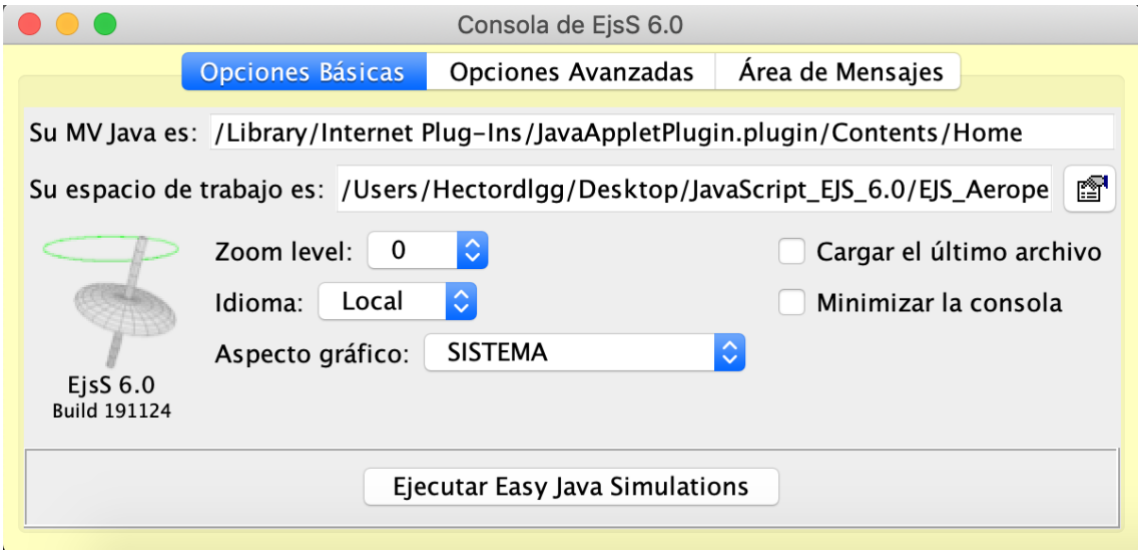


Figura 8: Consola de EjsS 6.0

La segunda ventana que se abre es la interfaz sobre la cual trabajará el usuario. Esta contiene tres partes principales diferenciadas, Descripción, Modelo y HtmlView. Además, cuenta con otros elementos como una barra de herramientas con opciones típicas como guardar, ejecutar, etc., y un panel en los que se mostrarán mensajes como errores y la correcta compilación del programa.

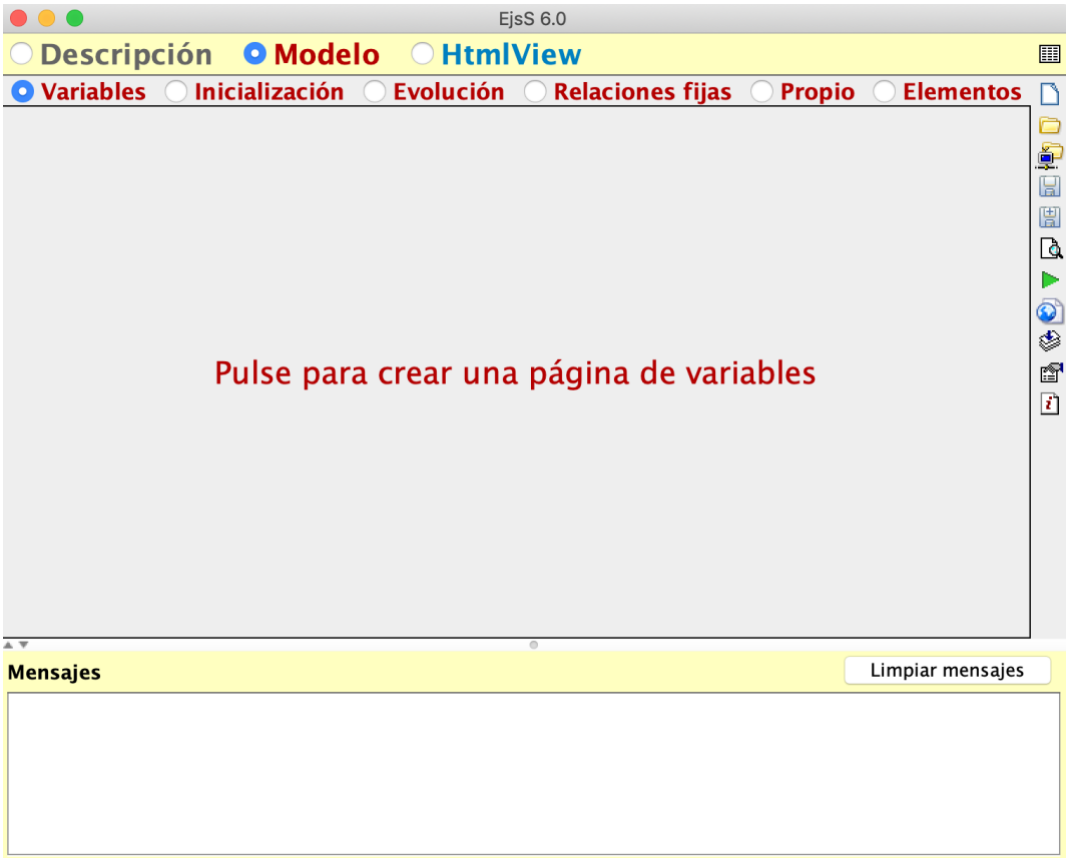


Figura 9: Interfaz de EjsS 6.0

- **Descripción**: esta opción se usa para editar el código HTML con el que trabaja la simulación. Esto permite crear páginas previas a la ejecución de la simulación, a modo de páginas web. En este caso no se usará esta opción.

- **Modelo**: a través de esta opción se elaborará el modelo matemático de la simulación. Esto incluirá inicialización de variables, declaración de ecuación diferencial del modelo, implementación del PID digital, etc.

  Esta opción contiene varias ventanas diferenciadas, estas son: Variables, Inicialización, Evolución, Relaciones fijas, Propio y Elementos.

  'Variables' tiene como función la declaración de todas las variables que se van a utilizar en el proyecto. Presenta una sencilla forma de introducirlas, en forma de tabla con diferentes columnas en las que se introducirá su nombre, el tipo de la variable (*boolean, int, double, string* u *object*), el valor inicial, y la dimensión en el caso de que se trate de una variable de forma matricial.

  'Inicialización' sirve para inicializaciones más complejas, las cuales no pueden llevarse a cabo en 'Variables', por ejemplo, cuando se han de dar diferentes valores a los elementos de un vector.

  'Evolución' contiene las ecuaciones que marcan la variación del modelo en función del tiempo.

  Inicialmente está dividida en tres partes, una vertical donde seleccionamos los parámetros de la evolución, y dos pulsadores para crear páginas de código y páginas de ecuaciones diferenciales ordinarias (EDO) respectivamente.

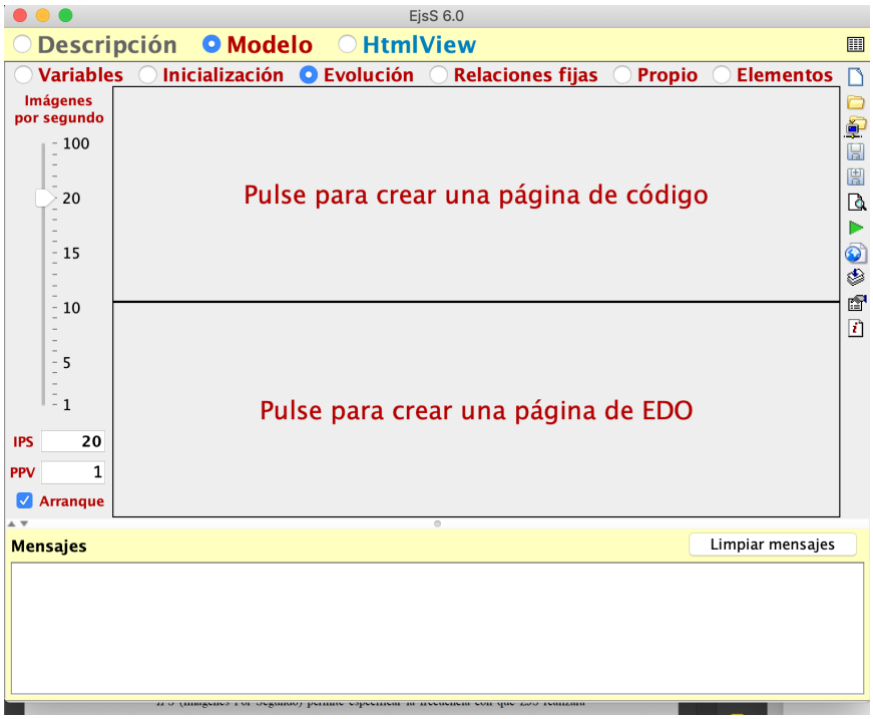


Figura 10: Panel evolución del modelo (EJS)

Cuando se vaya a trabajar en forma discreta, se empleará una página de código donde se escribirán las ecuaciones en lenguaje Java.

Si en cambio se va a trabajar en forma continua se emplea una página EDO. En las páginas EDO se deben introducir ecuaciones diferenciales explícitas de primer orden. En ocasiones será necesaria la elección de variables auxiliares para poder proceder de esta manera, pero eso no supone ningún problema.

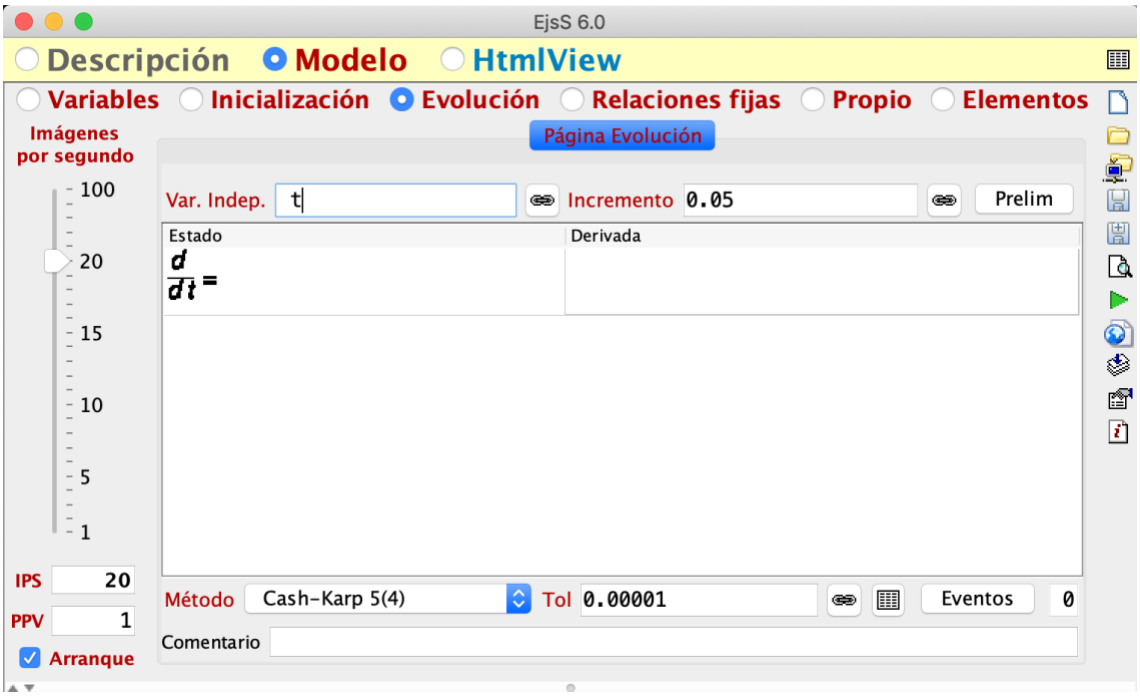


Figura 11: Página EDO en EJS

Se debe tener en cuenta que si se desea una simulación se debe establecer una relación en esta página de modo que el incremento de tiempo venga dado por la inversa del número de imágenes por segundo establecido.

'Relaciones fijas' se usa cuando se desea establecer el valor para una variable que no sea dependiente del tiempo. En ellas se establecen relaciones independientes a la evolución temporal.

'Código propio': este panel tiene la funcionalidad de insertar fragmentos de código Java para simplificar los códigos de otros paneles o para añadir nuevas funcionalidades. Esto se desarrollará a modo de declaración de funciones propias, que solo actuarán en el caso de ser llamadas desde otro punto de la interfaz.

- **HTMLView**: En esta opción se desarrolla todo el entorno gráfico de una manera sencilla e intuitiva. EJS cuenta con una librería con los objetos que se pueden añadir al entorno, incluyendo paneles de dibujo, la posibilidad de incluir imágenes, la representación de funciones, etc. Además, permite la opción de asociar las propiedades de dichos elementos con las variables declaradas anteriormente, facilitando así la interacción.

5. JUSTIFICACIÓN DETALLADA DE LA SOLUCIÓN ADOPTADA

5.1. Modelado matemático de la planta.

Antes de comenzar con el control de la planta, se ha de conocer su comportamiento dinámico. Para ello, en este apartado se aborda el modelado matemático del sistema, con el fin de obtener una ecuación que nos permita relacionar la salida del sistema, en este caso el ángulo formado con la vertical, con la entrada, que en este caso será la fuerza que genera el sistema motor – hélice.

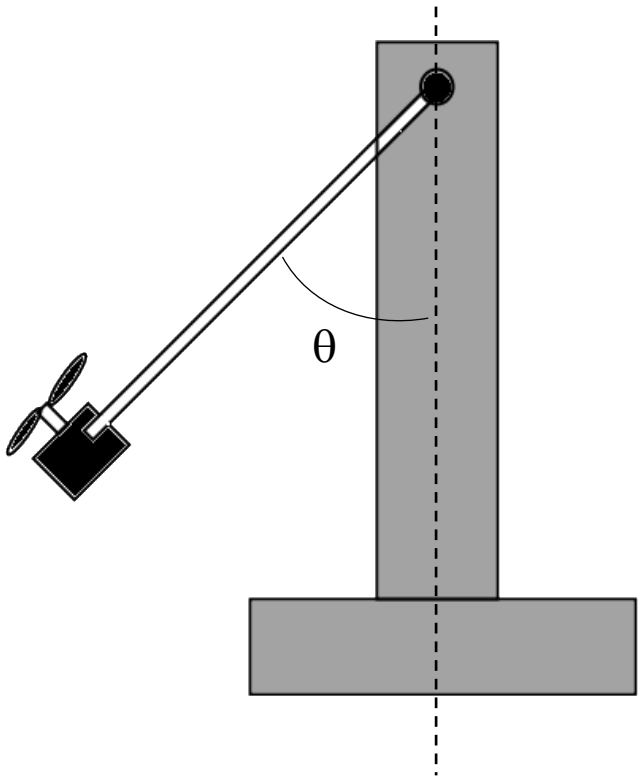


Figura 12: Esquema del sistema a analizar

- Fuerzas implicadas:

Las fuerzas que actúan sobre el sistema son las siguientes:

- Fuerza generada por la hélice ($\overrightarrow{F_h}$)
- Componente $\overrightarrow{P_x}$ del peso del sistema ($\overrightarrow{P_x} = P \cdot sin\ \theta$)
- Componente $\overrightarrow{P_y}$ del peso del sistema ($\overrightarrow{P_y} = P \cdot cos\ \theta$)

Además, se muestra a continuación una representación de las fuerzas sobre el sistema. Se puede observar que, al ser este un modelado teórico, se ha considerado que las fuerzas tanto del peso como la fuerza de rozamiento se aplican en el extremo del péndulo en lugar de en el centro de gravedad para hacer más sencillos los cálculos

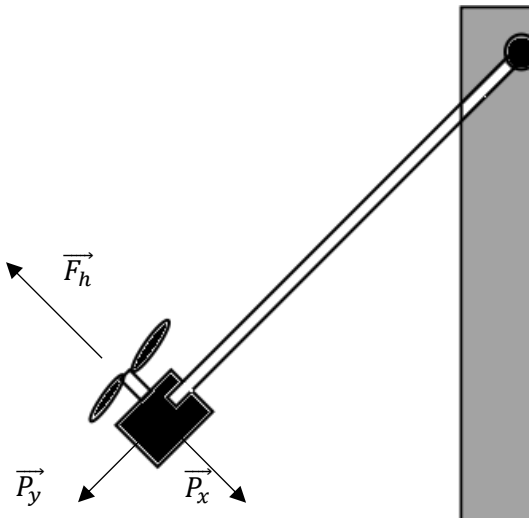


Figura 13: Representación de las fuerzas que actúan en el sistema

- Teorema del Momento angular

En este apartado, se desarrollará el cálculo a partir del Teorema del Momento Angular o Momento Cinético con el fin de obtener la función de transferencia del sistema que relacione la fuerza ejercida por la hélice con la posición del mismo.

Según su definición, el momento cinético en un punto cualquiera de un sistema indeformable se expresa como:

$$\overrightarrow{\Gamma_0} = I_0 \overrightarrow{\omega_{21}} + M\overrightarrow{OG} \wedge \overrightarrow{v_{21}^o}$$

Siendo:

- $\overrightarrow{\omega_{21}}$: Velocidad angular alrededor del eje de giro
- I_0: Momento de inercia del sistema respecto al eje de giro
- M: Masa
- $\overrightarrow{OG}$: Vector posición del centro de gravedad
- $\overrightarrow{v_{21}^o}$: Velocidad del eje respecto al sistema de referencia

Como para todo el modelado se va a tomar como referencia el eje de giro del péndulo, lo más sencillo es establecer ahí un sistema de referencia XYZ con el eje X hacia dentro.

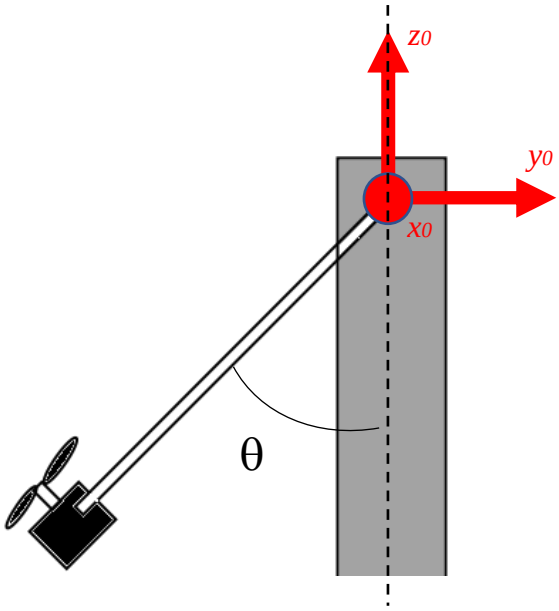


Figura 14: Representación del eje de referencia

De este modo, la velocidad del punto en el que se sitúa el eje con respecto al sistema de referencia será nula, por lo que la ecuación queda:

$$\overrightarrow{\Gamma_0} = I\overrightarrow{\omega_{21}} = I\frac{d\theta}{dt}\vec{x}$$

Si derivamos esta expresión obtenemos:

$$\frac{d\overrightarrow{\Gamma_0}}{dt} = I\frac{d^2\vec{\theta}}{dt^2}\vec{x}$$

Por otra parte, el Teorema del momento cinético nos dice que:

$$\frac{d\overrightarrow{\Gamma_0}}{dt} = \sum_i \vec{r} \times \overrightarrow{F_i}$$

De este modo, podemos igualar ambas expresiones para obtener una única ecuación. Además, añadiremos un término que exprese la fuerza de rozamiento del eje, que contará con una constante de rozamiento y será proporcional a la velocidad angular.

$$I\frac{d^2\vec{\theta}}{dt^2} = \sum_i \vec{r} \times \overrightarrow{F_i} - c\frac{d\theta}{dt}$$

Para terminar el modelado, desarrollamos el sumatorio teniendo en cuenta todas las fuerzas que actuarán, en este caso $\overrightarrow{F_h}$ y $\overrightarrow{P_x}$. La componente del peso $\overrightarrow{P_y}$ no afecta al momento al actuar en la misma dirección de la barra del péndulo.

$$I\frac{d^2\vec{\theta}}{dt^2} = -P_x \cdot L - c\frac{d\theta}{dt} + L \cdot F_h$$

Donde L es la longitud del péndulo.

Desarrollando la expresión del peso, la ecuación queda:

$$I\frac{d^2\vec{\theta}}{dt^2} = -mgL\sin\theta - c\frac{d\theta}{dt} + LF_h$$

O expresada de manera matricial:

$$\begin{pmatrix}\dot{\theta}\\ \ddot{\theta}\end{pmatrix} = \begin{pmatrix}0 & 1\\ \frac{-g\sin\theta}{L\theta} & \frac{-c}{I_0}\end{pmatrix}\begin{pmatrix}\theta\\ \dot{\theta}\end{pmatrix} + \begin{pmatrix}0\\ \frac{1}{mL}\end{pmatrix}F$$

Donde *I* es el momento de inercia ($I = mL^2$), *m* es la masa, *g* es la aceleración de la gravedad y *c* es la constante de rozamiento.

De este modo, conseguimos obtener la ecuación que relaciona la fuerza de empuje de la hélice (entrada del sistema) con el ángulo que forma el péndulo con la vertical (salida del sistema):

$$I\ddot{\theta}(t) + c\dot{\theta}(t) + mgL\sin\theta - LF_h(t) = 0$$

A partir de esta, realizaremos una linealización entorno a una serie de puntos de equilibrio para obtener una función de transferencia G(s) que represente el comportamiento del sistema.

Figura 15: Entrada y salida de la función de transferencia del sistema

5.2. Linealización de la función de transferencia

Para abordar la linealización del sistema, en primer lugar, tenemos que:

$$f(\ddot{\theta}, \dot{\theta}, \theta, F_h) = I\ddot{\theta} + c\dot{\theta} + mgL\sin\theta - LF_h$$

Se sabe que cuando el sistema se encuentra en estado estacionario, o lo que es lo mismo en un punto de equilibrio, la función queda:

$$f(0{,}0, \theta_{eq}, F_{eq}) = mgL\sin\theta_{eq} - LF_{eq} = 0$$

De donde se obtiene que:

$$F_{eq} = mg\sin\theta_{eq}$$

A partir de aquí se realiza un desarrollo en series de Taylor de la función f en torno a un punto de equilibrio ($(\ddot{\theta} = 0, \dot{\theta} = 0, \theta = \theta_{eq}, F_h = F_{eq})$.

$$f(\ddot{\theta}, \dot{\theta}, \theta, F_h) = f(0{,}0, \theta_{eq}, F_{eq}) + \frac{\partial f}{\partial \ddot{\theta}}\Big]_{\ddot{\theta}=0} \Delta\ddot{\theta} + \frac{\partial f}{\partial \dot{\theta}}\Big]_{\dot{\theta}=0} \Delta\dot{\theta} + \frac{\partial f}{\partial \dot{\theta}}\Big]_{\theta=\theta_{eq}} \Delta\theta - \frac{\partial f}{\partial F_h}\Big]_{F_h=F_{eq}} \Delta F$$

Donde $f(\ddot{\theta}, \dot{\theta}, \theta, F_h)$, como se ha explicado anteriormente, es nulo, $\Delta\theta = \theta - \theta_{eq}$ y $\Delta F = F_h - F_{eq}$.

Desarrollando las correspondientes derivadas expresadas en la ecuación anterior, la ecuación linealizada queda:

$$f(\Delta\ddot{\theta}, \Delta\dot{\theta}, \Delta\theta, \Delta F) = I\Delta\ddot{\theta} + c\Delta\dot{\theta} + mgL\cos\theta_{eq}\,\Delta\theta - L\Delta F$$

Se observa que a esta ecuación se le puede aplicar la transformada de Laplace, y obtener así la función de transferencia G(s) que relacione la entrada y la salida del sistema ($G(s) = \frac{\Delta F}{\Delta\theta}$).

$$I(\Delta\theta(s)s^2 - \Delta\dot{\theta}(0) - \Delta\theta(0)s) + c(\Delta\theta(s)s - \Delta\theta(0)) + mgL\cos\theta_{eq}\,\Delta\theta(s) - L\Delta F(s)$$

$$G(s) = \frac{\Delta\theta}{\Delta F} = \frac{L}{Is^2 + cs + mgL\cos\theta_{eq}}$$

5.3. Validación del modelo con Simulink

Una vez obtenida la función de transferencia, el siguiente paso es validar el modelo linealizado comparándolo con un modelo no lineal que obtendremos en Simulink mediante una serie de bloques de sus bibliotecas.

El objetivo es comprobar que se obtienen unas respuestas similares en ambos modelos.

El primer paso para realizar esta comprobación es construir el modelo no lineal. Para ello partimos de la ecuación del sistema:

$$f(\ddot{\theta}, \dot{\theta}, \theta, F_h) = I\ddot{\theta} + c\dot{\theta} + mgL\sin\theta - LF_h = \ddot{\theta} + \frac{c}{I}\dot{\theta} + \frac{mgL\sin\theta}{I} - \frac{L}{I}F_h$$

Sabiendo que la entrada del sistema es la Fuerza F_h y la salida el ángulo θ, construimos el sistema para que se cumpla la ecuación.

El modelo queda:

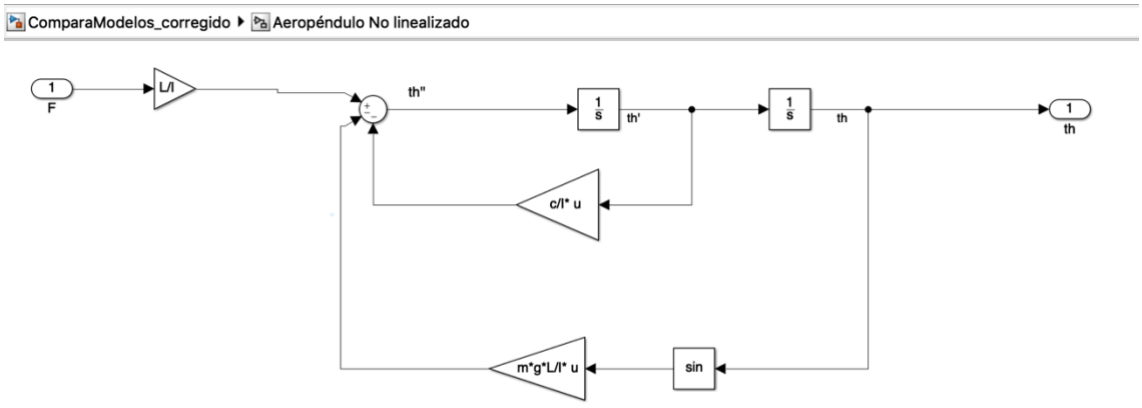


Figura 16: Modelo no lineal del aeropéndulo

Cabe decir que el bloque $\frac{1}{s}$ representa un integrador en el dominio de Laplace. De ahí que se use este bloque para ir integrando el ángulo (que en el modelo Simulink está expresado como “th”) hasta obtener la salida.

El siguiente paso es construir un bloque donde representaremos la función de transferencia G(s) linealizada obtenida anteriormente, y comprobar que efectivamente ambos sistemas se comportan de manera similar ante los mismos estímulos, que en este caso serán perturbaciones.

Para incluir perturbaciones al sistema, se incluye un bloque que representa la suma de dos entradas con forma de escalón, que se aplican en diferente tiempo.

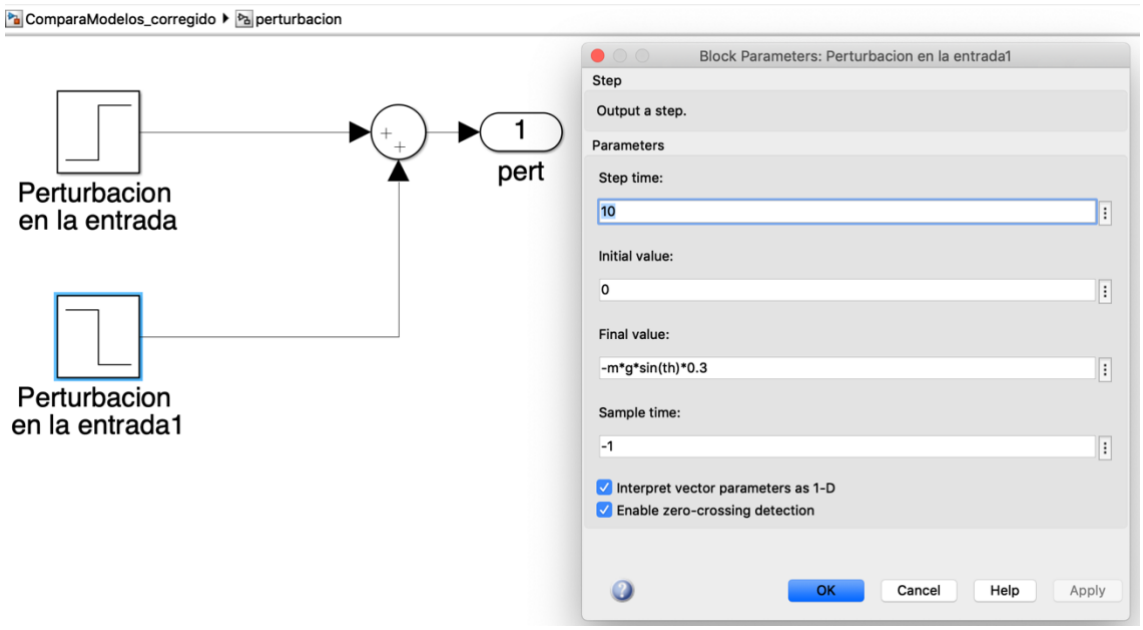


Figura 17: Representación de las perturbaciones incluidas.

Finalmente, el esquema mediante el cual compararemos las salidas de ambos sistemas es el siguiente:

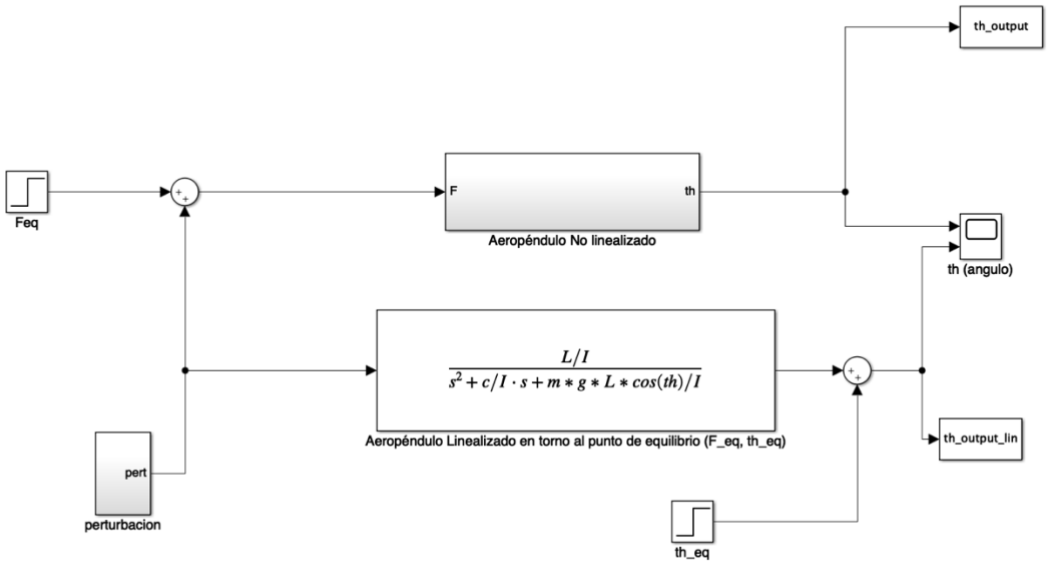


Figura 18: Comparación de modelos Lineal y No Lineal en Simulink

Si observamos las respuestas obtenidas mediante el bloque de tipo Scope (th(angulo)), podemos comprobar que se presentan respuestas muy similares ante las perturbaciones.

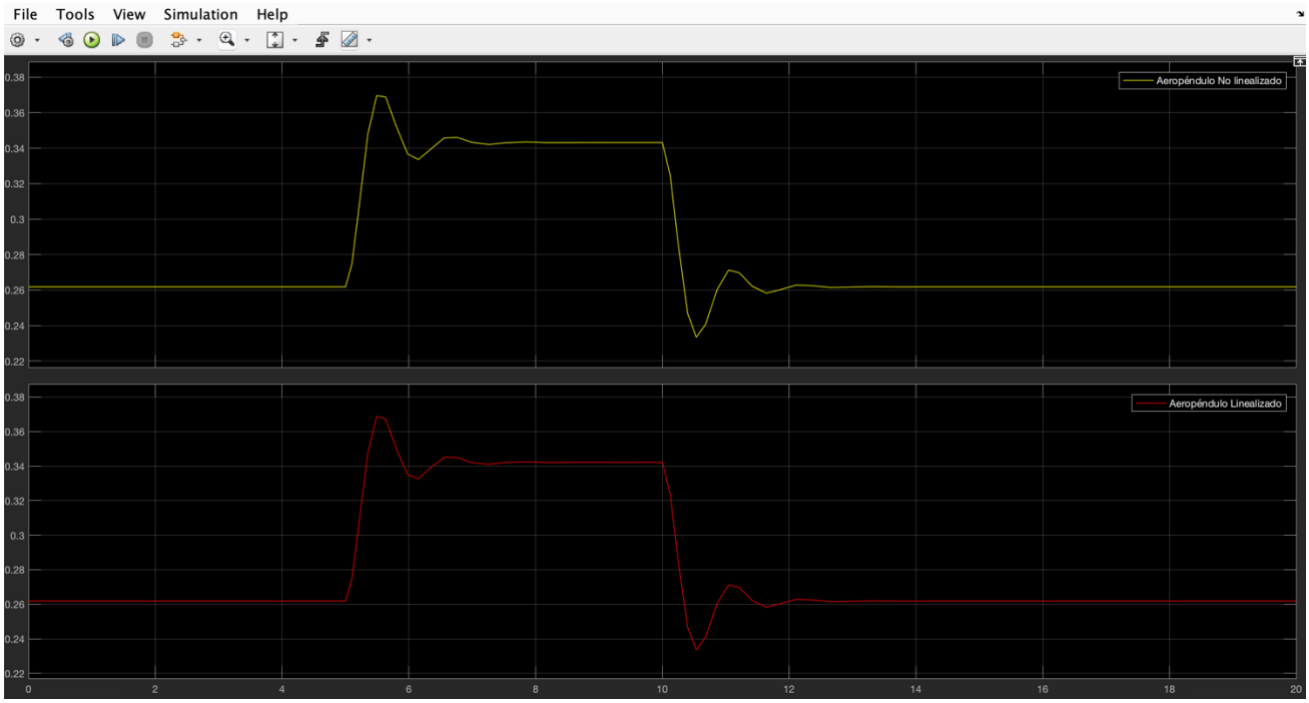


Figura 19: Comparación de las respuestas de ambos modelos

De la respuesta podemos observar que, al aplicar una fuerza permanente en el tiempo, que en este caso está aplicada como una entrada en forma de escalón (perturbaciones), el ángulo θ queda en una nueva posición, hasta que se aplica la segunda perturbación.

El objetivo a cumplir con el diseño del regulador es que el ángulo θ formado por el sistema se mantenga en el deseado ante la aparición de perturbaciones.

5.4. Diseño del regulador PID

En este apartado se aborda el diseño del regulador PID mediante el cual se pretende controlar el comportamiento de la planta.

Por una parte, se pretende que cumpla una serie de requerimientos dinámicos, que se enumeran a continuación:

- T_e < 10 segundos
- Sin sobreoscilación.
- Error de posición nulo

Además, con el regulador conseguiremos que el sistema se autorregule con el fin de solventar el efecto de posibles perturbaciones en régimen estacionario.

Para ello, se realizará un desarrollo teórico para obtener las constantes del regulador y posteriormente se realizará una validación mediante MATLAB – Simulink similar al realizado anteriormente en la comparación de modelos.

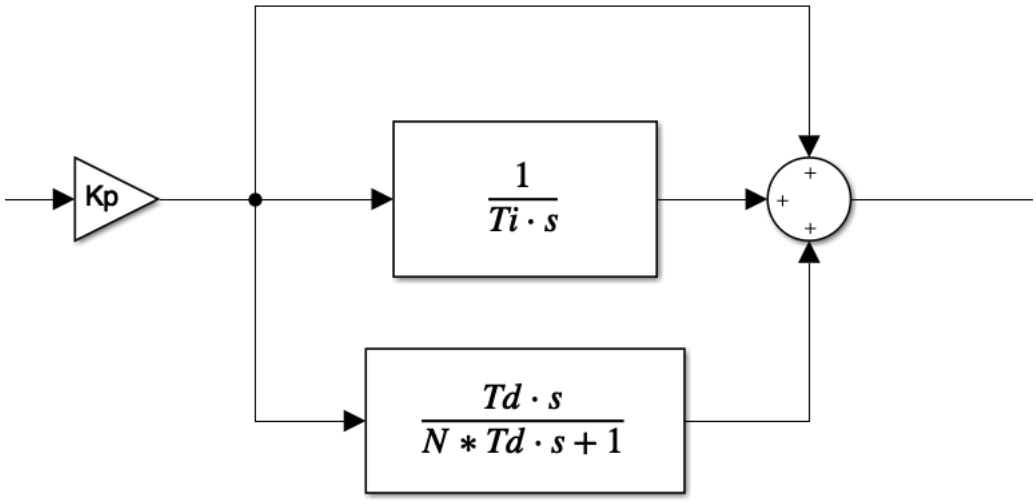


Figura 20: Controlador PID implantado en Simulink

La base que se seguirá para el diseño matemático del regulador será la de buscar la cancelación de la dinámica natural del aeropéndulo, que presenta un comportamiento oscilatorio, para establecer la dinámica deseada.

Esto se consigue con un estudio del conjunto de polos que presenta la función de transferencia linealizada G(s), los cuales son los responsables de la dinámica del sistema.

Como se ha comentado, para el desarrollo del regulador nos ayudaremos de MATLAB. El primer paso es escribir el código mediante el cual declaramos las constantes del sistema y la función de transferencia G(s).

```
s=tf('s');

L=0.26;
c=0.009;
g=9.81;
m=0.035;
I=m*L^2;
th_deg=15; %Angulo th en grados
th=th_deg*pi/180;%Conversión a radianes

Feq=m*g*sin(th);


G=(L/I)/(s^2+(c/I)*s+(m*g*L*cos(th)/I))
```

Figura 21: Código MATLAB para declarar G(s)

A partir de la declaración de la función de transferencia tenemos varios comandos que nos resultarán útiles:

- step(G): mediante este comando observamos la respuesta del sistema ante escalón y verificamos su comportamiento oscilatorio.

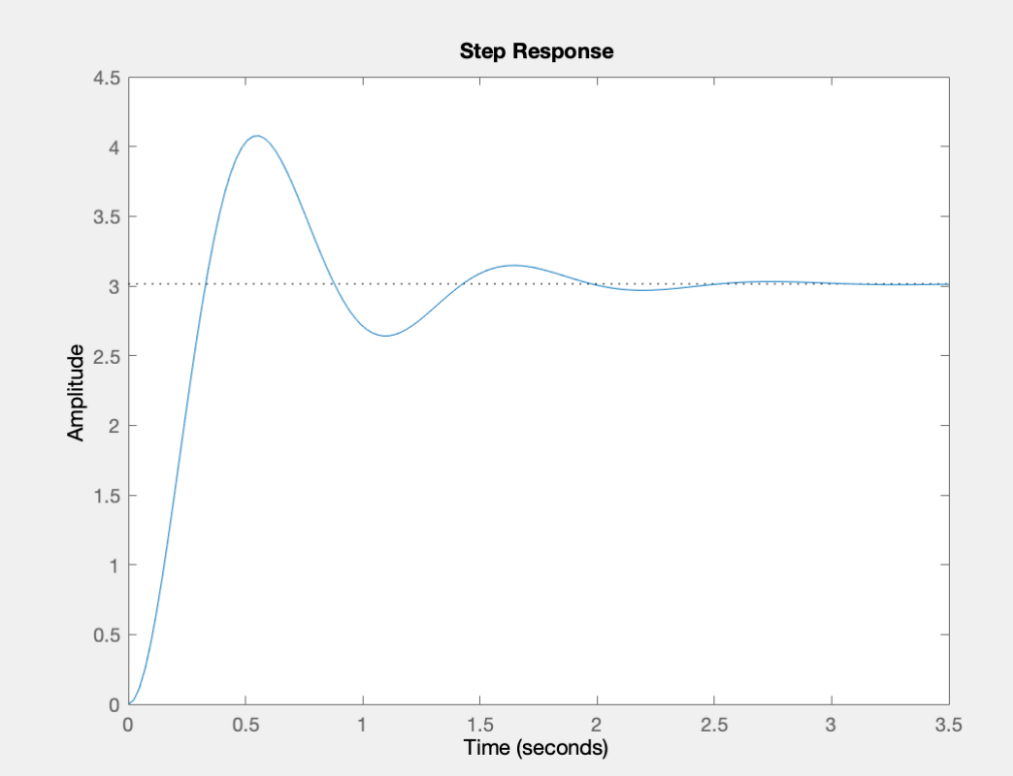


Figura 22: Respuesta de G(s) ante escalón

- pzmap(G): mediante este comando observamos el mapa de polos y ceros del sistema.

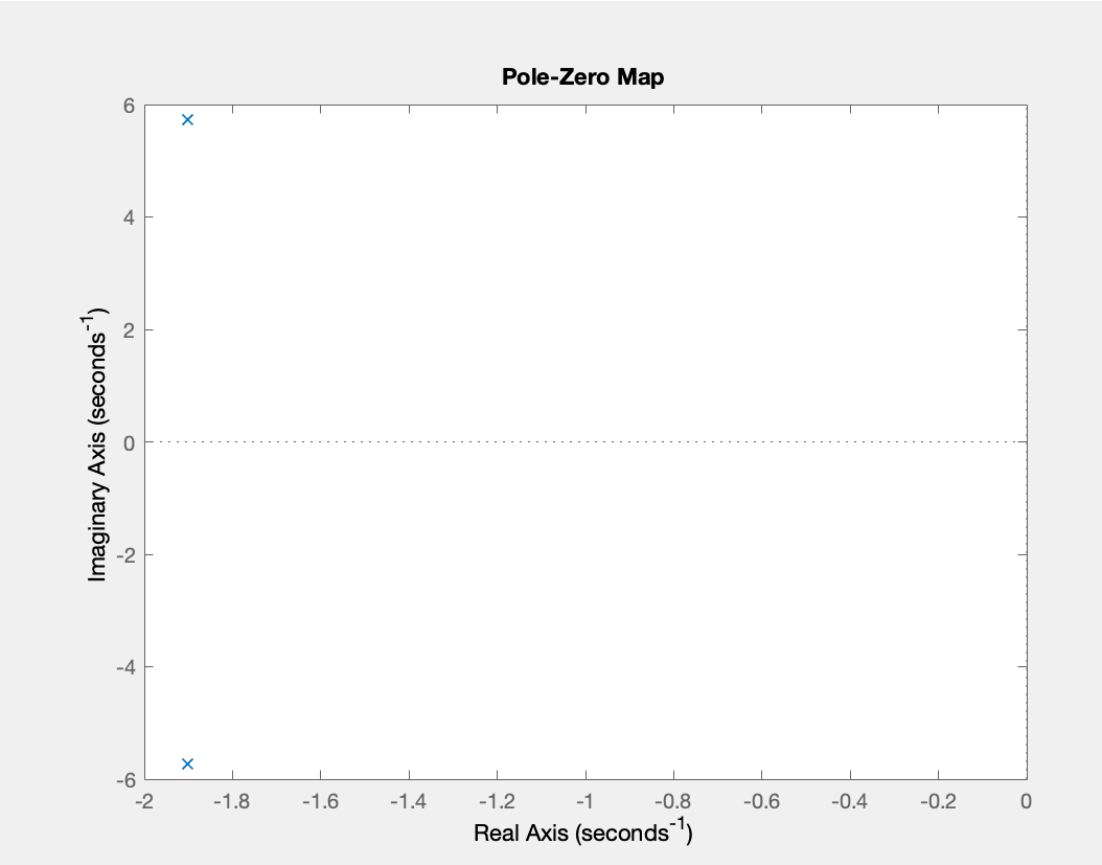


Figura 23: Mapa de polos y ceros de G(s)

Podemos observar que presenta dos polos complejos conjugados, lo que explica el comportamiento oscilatorio del sistema.

- pole(G): Este comando nos devuelve el valor exacto de la posición de los polos, o lo que es lo mismo, las raíces del denominador.

```
Command Window
>> pole(G)

ans =

  -1.9019 + 5.7295i
  -1.9019 - 5.7295i
```

Figura 24: Obtención de polos del sistema en MATLAB

Como se puede observar, el ángulo que puede formar el aeropéndulo abarca un rango considerablemente grande. Al depender la función de transferencia de dicho ángulo, y por tanto la dinámica, no bastará con un solo regulador para cumplir correctamente las especificaciones en todo el rango.

Para solventar este problema, se crearán tres reguladores distintos, que se aplicarán según el rango en el que se encuentre el ángulo de equilibrio deseado. Se creará el regulador para el ángulo intermedio de cada uno de los rangos. Dichos rangos serán:

- 0º < θ < 30º: Se diseñará el regulador para $\theta = 15º$
- 30º < θ < 65º: Se diseñará el regulador para $\theta = 45º$
- 60º < θ < 90º: Se diseñará el regulador para $\theta = 75º$

El proceso es similar para todos los reguladores. A continuación, se explicarán los pasos seguidos para su obtención en el caso de $\theta = 15º$.

En primer lugar, tenemos que:

$$G(s) = \frac{\frac{L}{I}}{s^2 + \frac{c}{I}s + mgL\cos\theta_{eq}} = \frac{K_g}{(s+\alpha+j\omega)(s+\alpha-j\omega)}$$

Sabiendo esto, proponemos un regulador con la siguiente estructura:

$$R(s) = \frac{K_r(s+\alpha+j\omega)(s+\alpha-j\omega)}{s(s+\beta)}$$

Donde Kr y β son los parámetros a diseñar.

La función de transferencia en lazo cerrado queda como:

$$G_{cl}(s) = \frac{R(s)G(s)}{1+R(s)G(s)} = \frac{K_g K_r}{s^2 + \beta s + K_g K_r}$$

Esta función corresponde con una función de transferencia de un sistema de segundo orden de la forma:

$$G_{cl}(s) = \frac{K\omega_n^2}{s^2 + 2\omega_n\zeta s + \omega_n^2}$$

Donde ω_n es la frecuencia natural no amortiguada, y ζ es el factor de amortiguamiento.
Del factor de amortiguamiento sabemos que:

- $0 < \zeta < 1$: Sistema subamortiguado
- $\zeta = 1$: Sistema críticamente amortiguado
- $\zeta > 1$: Sistema sobreamortiguado

Sabiendo esto, si elegimos $\beta = 2\sqrt{K_g K_r}$ obtenemos un sistema de segundo orden críticamente amortiguado. La función de transferencia en lazo cerrado queda:

$$G_{cl}(s) = \frac{K_g K_r}{\left(s + \sqrt{K_g K_r}\right)^2}$$

Una vez desarrollado esto, pasamos a los cálculos numéricos.

Para el caso de $\theta = 15º$:

$$G(s) = \frac{109.9}{s^2 + 3.804s + 36.45}$$

$$R(s) = \frac{K_r(s^2+3.804s+36.45)}{s(s+\beta)}$$

Para cumplir la especificación del tiempo de establecimiento, sabemos que:

$$t_e = \frac{4}{\sigma} < 10\ seg$$

$$\sigma = \sqrt{K_g K_r} = 0.4$$

$$K_r = \frac{0.4^2}{109.9} = 0.001456$$

De este modo, tenemos una primera versión del regulador que queda como:

$$R(s) = \frac{0.001456(s^2 + 3.804s + 36.45)}{s(s + 0.8)}$$

Por otro lado, conocemos la función de transferencia teórica de un regulador PID:

$$C(s) = K_p(1 + \frac{1}{\tau_i s} + \tau_d s)$$

No obstante, en la práctica nos encontramos que esta función de transferencia es irrealizable, ya que, al desarrollar los términos se obtiene una función con mayor grado en el numerador que en el denominador.

Por eso se modifica la ecuación para conseguir un PID realizable:

$$C(s) = K_p(1 + \frac{1}{\tau_i s} + \frac{\tau_d s}{1 + \alpha\tau_d s})$$

Donde α debe tener un valor pequeño (en este caso tendrá un valor de 0.1).

Desarrollando queda como:

$$C(s) = \frac{(\alpha K_p \tau_i \tau_d + K_p \tau_d \tau_i)s^2 + (K_p \tau_i + \alpha K_p \tau_d)s + K_p}{\alpha\tau_d \tau_i s^2 + \tau_i s}$$

Igualamos los términos de ambos numeradores, pues serán estos los que realizarán la función de cancelación de los polos del proceso. Posteriormente comprobaremos si se cumplen las especificaciones de diseño.

Igualando los coeficientes de ambas funciones de transferencia, obtenemos el siguiente sistema de ecuaciones.

$$\begin{cases} 0.001456 = \alpha K_p \tau_i \tau_d + K_p \tau_d \tau_i \\ 0.001456 \cdot 3.804 = K_p \tau_i + \alpha K_p \tau_d \\ 0.001456 \cdot 36.54 = K_p \end{cases}$$

Sabiendo que $\alpha = 0.1$, podemos resolver el sistema de una manera muy sencilla.

Los resultados obtenidos son:

- $K_p = 0.05307$
- $\tau_i = 0.03709$
- $\tau_d = 0.06732$

La forma de proceder para los otros dos rangos de ángulo es similar, donde solo cambiaría la función de transferencia G(s) de la planta. A continuación, se muestra una tabla con los resultados obtenidos para cada uno de los reguladores.

Rango	K_p	τ_i	τ_d
$0º < \theta < 30º$	0.05307	0.03709	0.6732
$30º < \theta < 60º$	0.03885	0.11226	0.3034
$60º < \theta < 90º$	0.01145	0.11702	0.2758

Figura 25: Tabla de parámetros de los diferentes PIDs diseñados

Para validar los resultados obtenidos, añadimos un fragmento de código en el script de MATLAB creado anteriormente, en el cual declararemos el regulador PID diseñado y aplicaremos una entrada en forma de escalón al lazo cerrado del sistema. De este modo comprobamos que se cumplen las especificaciones.

TFG.m*

```
if (th_deg>0)&&(th_deg<30)

        Kp=0.05307;
        Ti=0.03709;
        Td=0.6732;
        N=0.1;

else if (th_deg>30)&&(th_deg<60)

        Kp=0.03885;
        Ti=0.11226;
        Td=0.3034;
        N=0.1;

else if (th_deg>60)&&(th_deg<90)

        Kp=0.01145;
        Ti=0.11702;
        Td=0.2758;
        N=0.1;

    end
    end
end

C=Kp*(1+(1/(Ti*s))+(Td*s/(1+N*Td*s)))

H=(C*G)/(1+C*G);

step(H)
```

Figura 26: Fragmento de código para implantar PID en MATLAB

Observando los resultados que nos brinda el comando step(H), se puede ver que se cumplen las especificaciones.

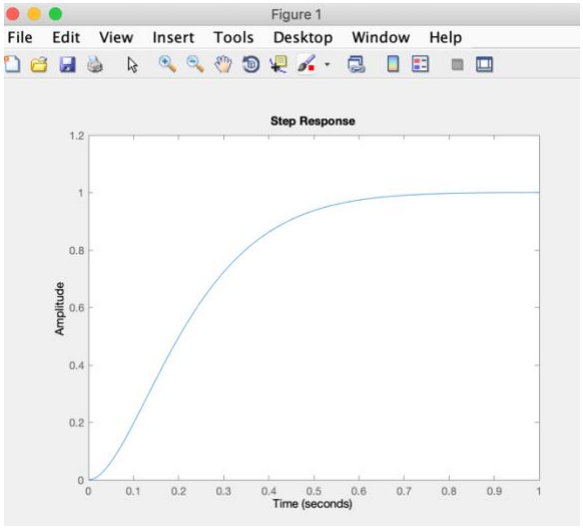


Figura 27: Respuesta del sistema en lazo cerrado ante escalón unitario ($\theta = 15º$)

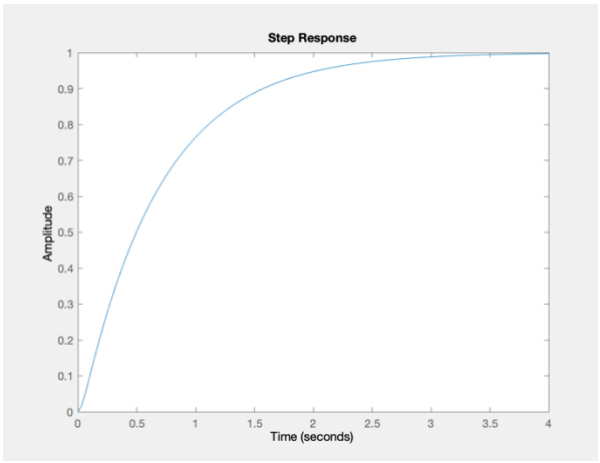


Figura 28: Respuesta del sistema en lazo cerrado ante escalón unitario ($\theta = 45º$)

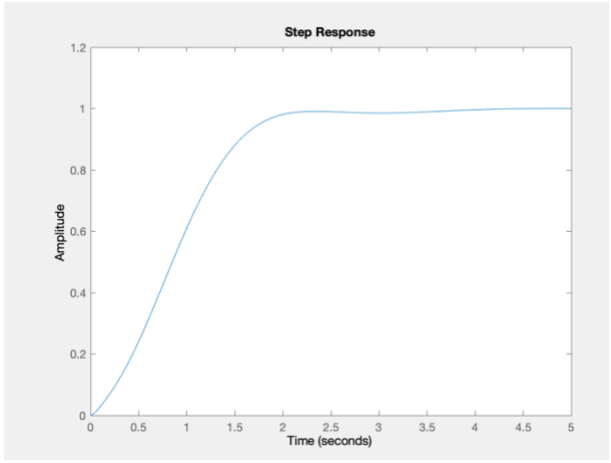


Figura 29: Respuesta del sistema en lazo cerrado ante escalón unitario ($\theta = 75º$)

Habiendo observado que se cumplen las especificaciones dinámicas ante una entrada de escalón unitario, se trabajará para implantar el regulador en Simulink. De este modo, al igual que antes, compararemos sus efectos en el modelo linealizado y en el no lineal ante perturbaciones, esperando obtener resultados similares. Para ello, se seguirá la estructura general de este tipo de reguladores, es decir, se añade el conjunto de bloques que componen el PID en serie con la función de transferencia de la planta y se realimenta.

El esquema completo queda de la siguiente manera:

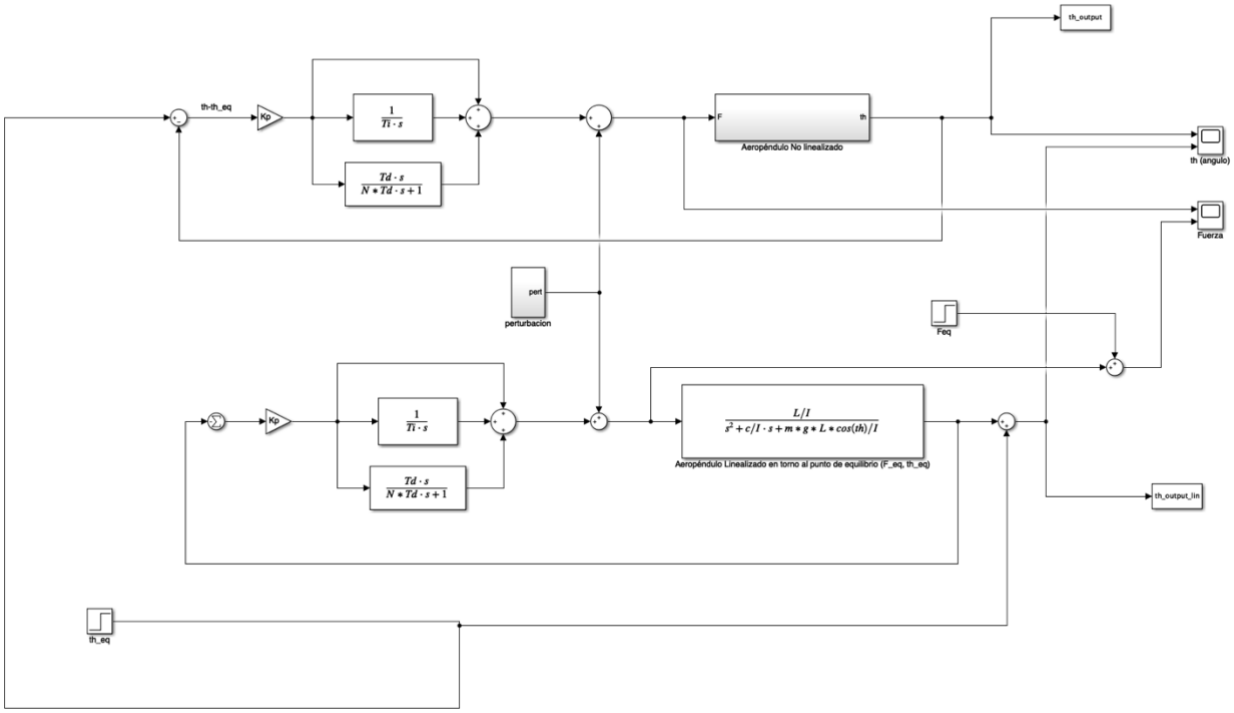


Figura 30: Comparación de modelos con reguladores.

En este caso se cuenta con dos bloques de tipo scope, que permiten observar gráficamente la salida del sistema obtenida durante el tiempo de simulación. En el primero de ellos se podrá observar el ángulo θ de interés, y en el segundo se podrá observar la fuerza aplicada por el sistema motor + hélice, que viene dada de la suma de la fuerza de equilibrio y la acción de control que se aplica sobre la planta.

Al igual que en el modelado, se espera obtener unos resultados similares en ambos casos.

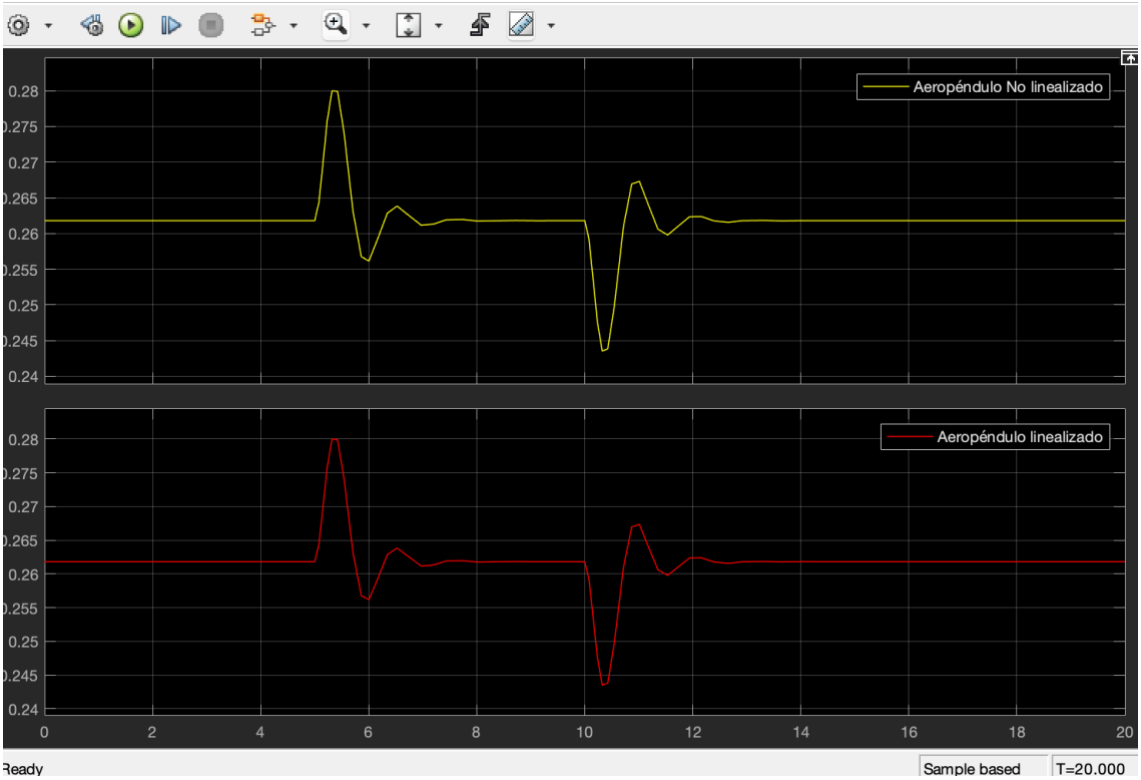


Figura 31: Respuesta de sistemas lineal y no lineal con regulador PID

Se pueden observar unos resultados muy similares, solventando las perturbaciones y siguiendo la referencia correctamente en ambos casos, por lo que el resultado es el esperado.

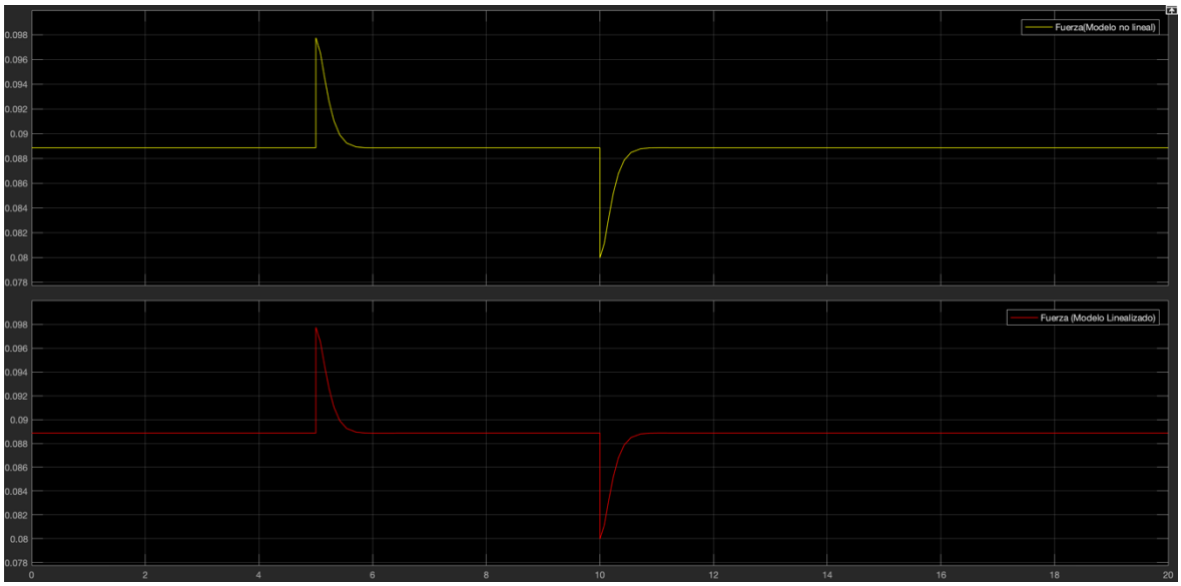


Figura 32: Representación de la fuerza aplicada en sistemas lineal y no lineal

5.5. Creación de prototipo virtual

Una vez modelado el sistema y calculado el regulador que se va a implantar, se procede a desarrollar a través del software Easy Java Simulations.

5.5.1. Inicialización de variables

El primer paso será realizar el trabajo necesario en la ventana 'Modelo'. En primer lugar, se realiza la declaración de todas las variables que intervendrán en el proceso.

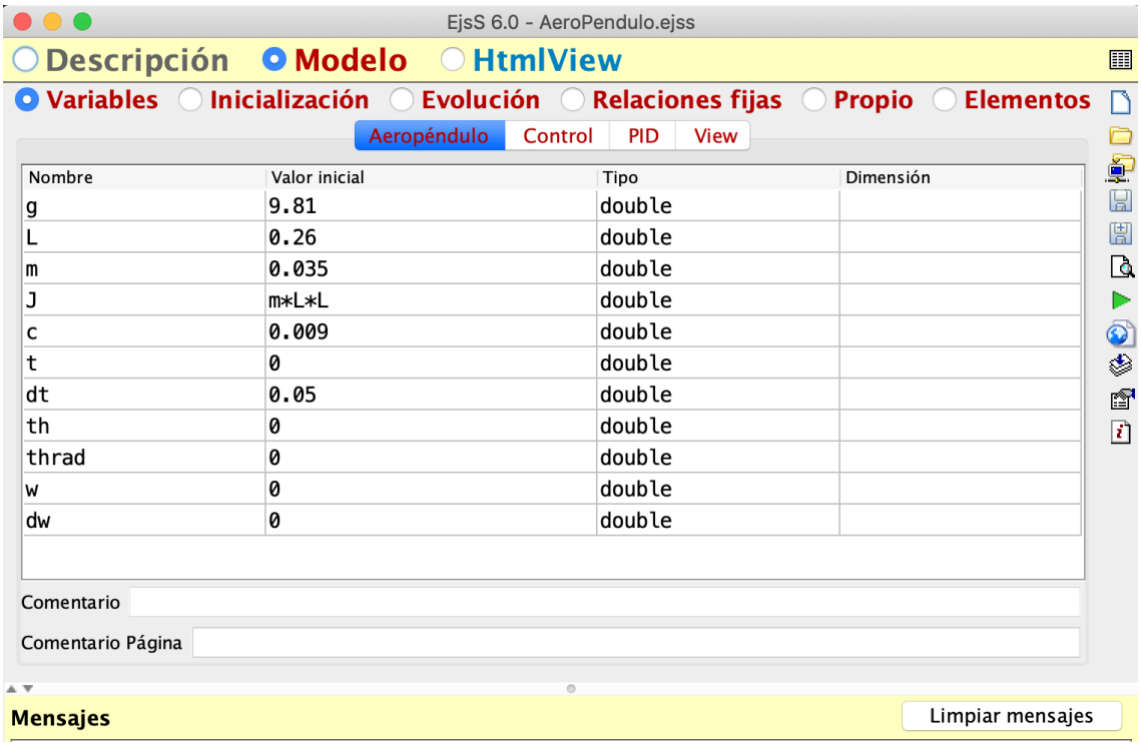


Figura 33: Inicialización de las variables del Aeropéndulo.

Como se puede observar en la *Figura 32*, la declaración de las variables se divide en cuatro secciones.

En la ventana 'Aeropéndulo' se declaran todas las constantes físicas del aeropéndulo, así como la variable temporal y el correspondiente incremento en base al cuál evolucionará el sistema.

En la ventana 'Control', como su nombre indica, se declaran las variables relacionadas con el control del sistema. Estas son referencias, acción de control, valores de saturación, etc.

En la ventana 'PID' se declaran las constantes del PID, así como las variables que serán necesarias para su implementación posterior.

En la ventana 'View' se declaran algunas variables que mejorarán la visualización posterior.

5.5.2. Definición del comportamiento del sistema.

El comportamiento del sistema será definido mediante una pestaña EDO, en la cual se introducirá la ecuación diferencial que define la dinámica del aeropéndulo.

Antes de esto, en la pestaña 'Propio' se creará una sub-ventana ('Air Pendulum Dynamic') en la cual se define la ecuación diferencial, devolviendo el valor de la aceleración angular.

Para ello se utilizan las variables thrad (ángulo θ), w (velocidad angular $\frac{d\theta}{dt}$), dw (aceleración angular $\frac{d^2\theta}{dt^2}$) y u (acción de control).

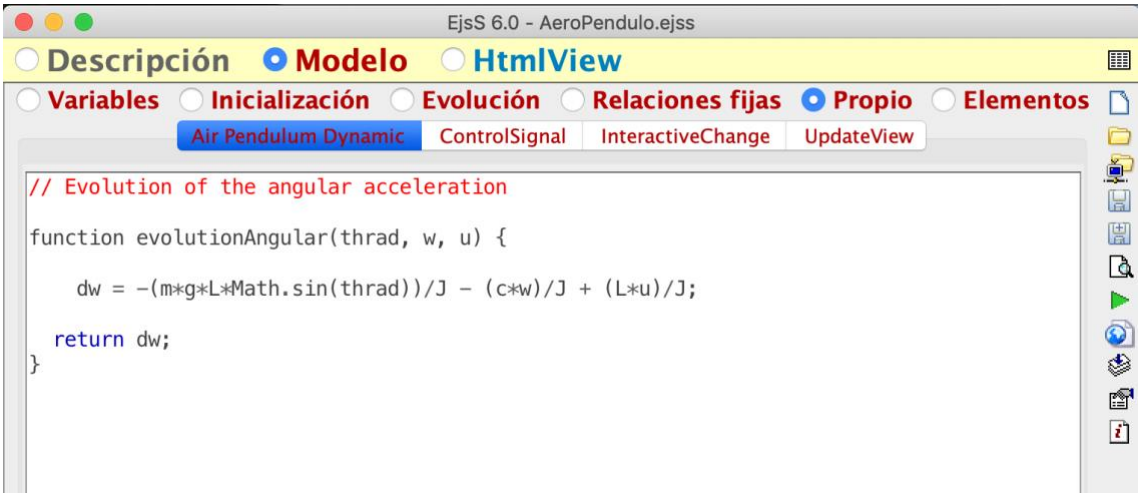


Figura 34: Función que define la dinámica del Aeropéndulo en EJS

Una vez definida esta función la incluimos en una página de EDO de la siguiente manera.

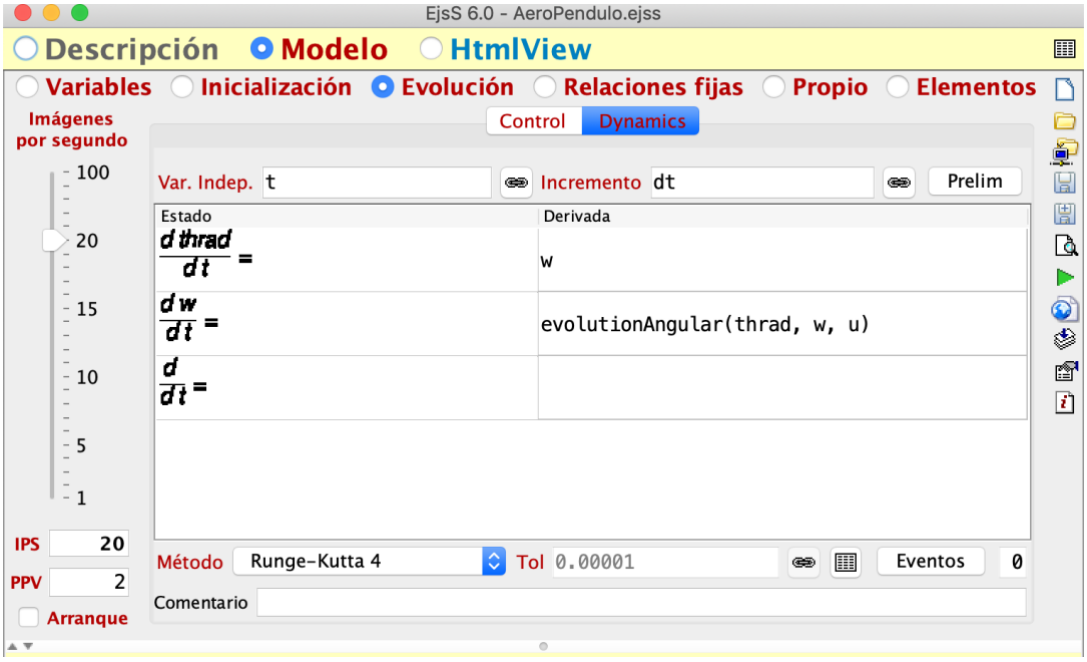
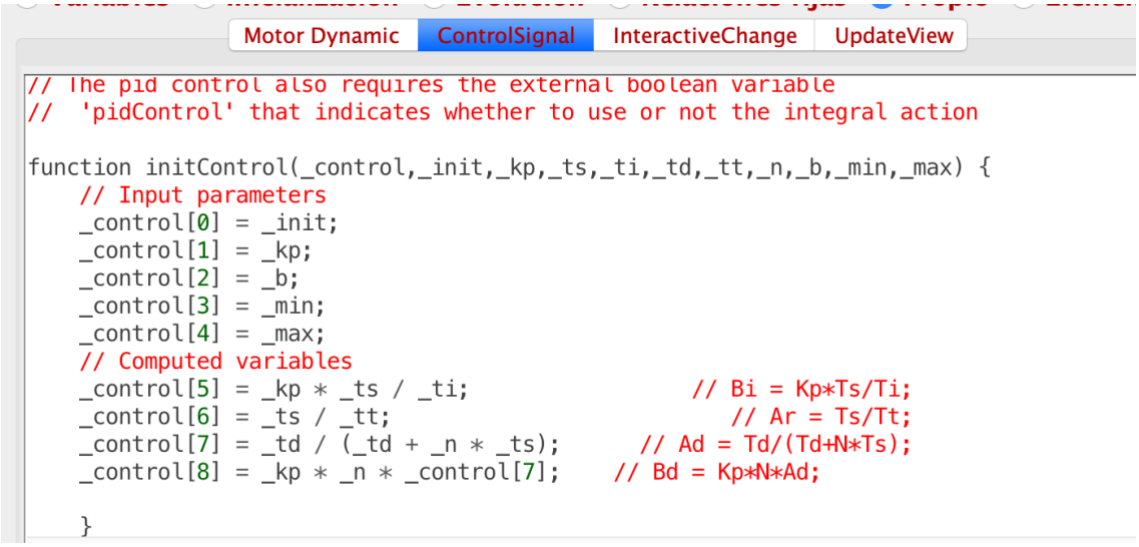


Figura 35: Definición de la EDO en EJS

5.5.3. Implementación del regulador PID.

Una vez definida la dinámica del sistema, procedemos a la implementación del PID. Este será el paso más complejo a realizar, por ello se crearán diferentes funciones que harán más simple esta tarea.

Dichas funciones serán definidas en la sub-ventana 'ControlSignal' de 'Propio'. La primera función será la que inicializará el control, definiendo todas las variables que actuarán en el PID.




```
// The pid control also requires the external boolean variable
//  'pidControl' that indicates whether to use or not the integral action

function initControl(_control,_init,_kp,_ts,_ti,_td,_tt,_n,_b,_min,_max) {
    // Input parameters
    _control[0] = _init;
    _control[1] = _kp;
    _control[2] = _b;
    _control[3] = _min;
    _control[4] = _max;
    // Computed variables
    _control[5] = _kp * _ts / _ti;                  // Bi = Kp*Ts/Ti;
    _control[6] = _ts / _tt;                           // Ar = Ts/Tt;
    _control[7] = _td / (_td + _n * _ts);       // Ad = Td/(Td+N*Ts);
    _control[8] = _kp * _n * _control[7];     // Bd = Kp*N*Ad;

    }
```

Figura 36: Función que inicializa el control en EJS

Seguidamente a esto, se implementa la función que implementa el PID discreto en sí.

```
function controlSignal(_control,_value,_ref,Iaction_ant) {
    // Proportional action : Kp*(b*ref - value)
    P = _control[1]*(_control[2]*_ref - _value);

    // Derivative action : D = Ad*D - Bd*(h-h_old);
    _control[10] = _control[7]*_control[10] - _control[8]*(_value - _control[0]);

    // pControl is a global variable that, when false, forces to ignore the integral action
    if (Ti==0){
        output = P + _control[10]; // P + D
    }else{
        output = P + _control[9] + _control[10]; // P + I + D
    }
    // Saturation of control signal
    if (output<_control[3])  outputSat = _control[3];
    else if (output>_control[4])  outputSat = _control[4];
    else outputSat = output;

    if (control === 0){
        _control[9]=Iaction_ant;
    } else {
        // Update integral action  :  I = I + Bi*(Hr-h) + Ar*(v-u)
        _control[9] = _control[9] + _control[5] * (_ref - _value) + _control[6]*(outputSat-output);
    }
    // keep the value for the next time
    _control[0] = _value;
    return outputSat;
}
```

Figura 37: Código de implementación de PID en EJS

Por último, para implementar el regulador PID sobre la dinámica del proceso, se ha de volver a la ventana 'Evolución' y crear una pestaña de código donde definiremos la acción de control u como la salida de la función controlSignal() declarada anteriormente.

5.5.4. Creación de la interfaz visual.

A la hora de crear la interfaz visual, EJS trabaja principalmente por diferentes paneles en los cuales se introducirán los elementos deseados. En este caso se incluirán un panel principal, que contendrá a su vez dos paneles, uno para incluir unos menús con diferentes opciones, como el selector de control manual o automático, y uno visual, donde se representará el aeropéndulo y las gráficas de evolución del ángulo que forma y de la fuerza aplicada.

Por otra parte, se creará el panel de controles, donde se incluirán elementos que permitan modificar parámetros como el ángulo de referencia o la fuerza aplicada, o los parámetros del controlador.

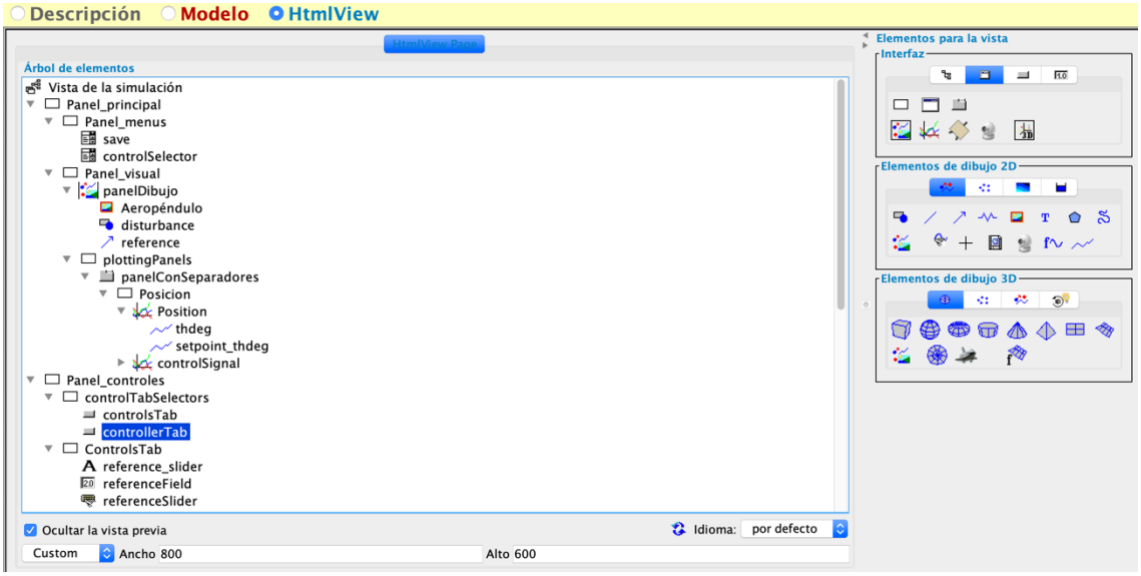


Figura 38: Árbol de elementos de la interfaz visual EJS

El panel de controles tiene dos elementos principales: las dos barras de tipo 'slider' mediante las cuales se pueden variar el ángulo de referencia y la fuerza aplicada, y las casillas en las cuales se introducen los valores de las constantes que componen el regulador PID.

Además, las barras también incluirán una casilla para dar la opción de introducir directamente el valor deseado.

Todas estas opciones tendrán implícitas un fragmento de código en común, que hará que el proceso se resetee al cambiar el valor de estos, reiniciando así la función initControl(). Dicha tarea la cumple otra tarea declarada previamente llamada resetControls.

El fragmento de código mencionado se ha de incluir en la función OnRelease, como se muestra en la *Figura 38.*

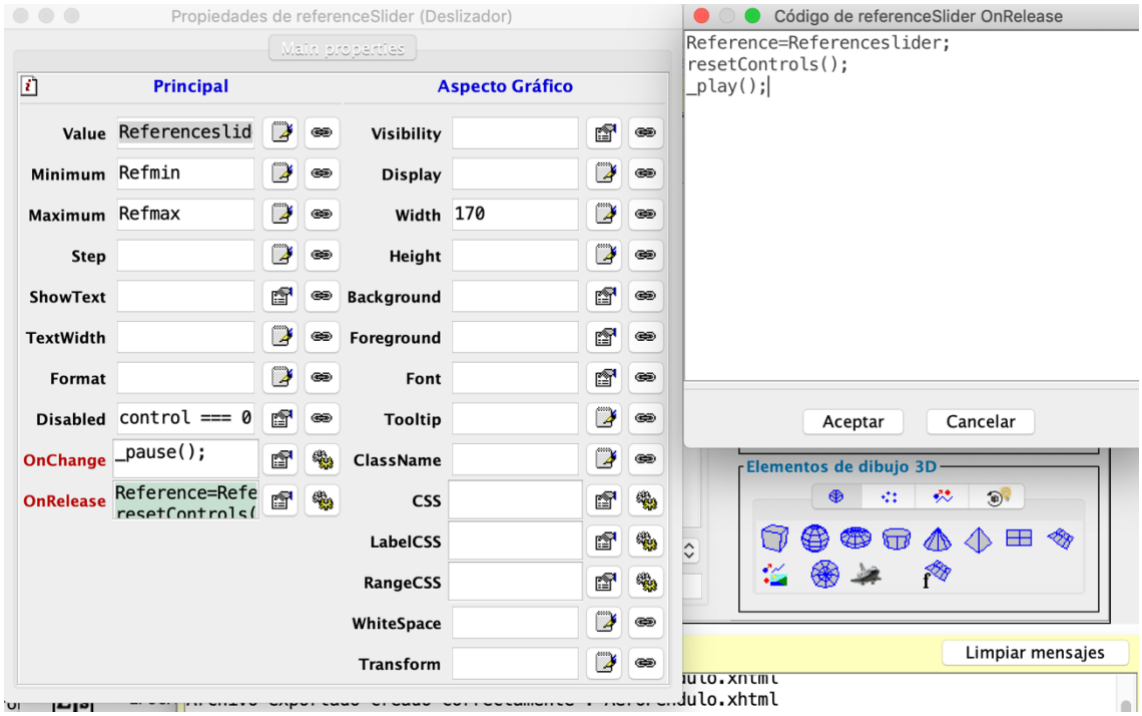


Figura 39: Código para cambiar parámetros incluido en la interfaz de EJS

Por otra parte, el panel visual tiene dos partes principales diferenciadas. En primer lugar, está la figura del aeropéndulo. Para incluirla, lo primero que se debe hacer es obtener externamente una imagen cuadrada en la cual el eje del péndulo se sitúe justamente en el medio, ya que será el punto sobre el cual gire la imagen.

Seguidamente, se incluye un panel de dibujo, y dentro de él una imagen. Dentro de las opciones del elemento 'imagen' hay que realizar varias operaciones. En primer lugar, en la opción 'Transformation' se debe incluir la variable que describe el ángulo formado, en este caso thrad. Por otra parte, hay que incluir la imagen cuadrada mencionada en ImageURL. Cabe destacar que la imagen debe estar en el mismo espacio de trabajo.

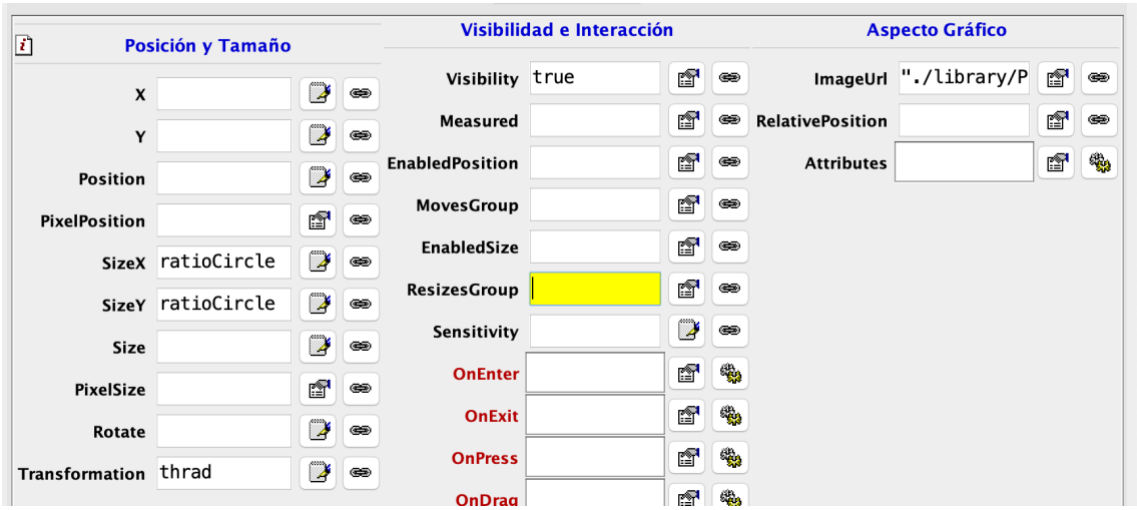


Figura 40: Configuración del elemento 'imagen' de la interfaz EJS

Por último, queda añadir dos paneles con ejes, y representar en uno de ellos ángulo formado por el péndulo, y en el otro la fuerza aplicada.

Finalmente, el prototipo web para la simulación del aeropéndulo queda de la siguiente manera:

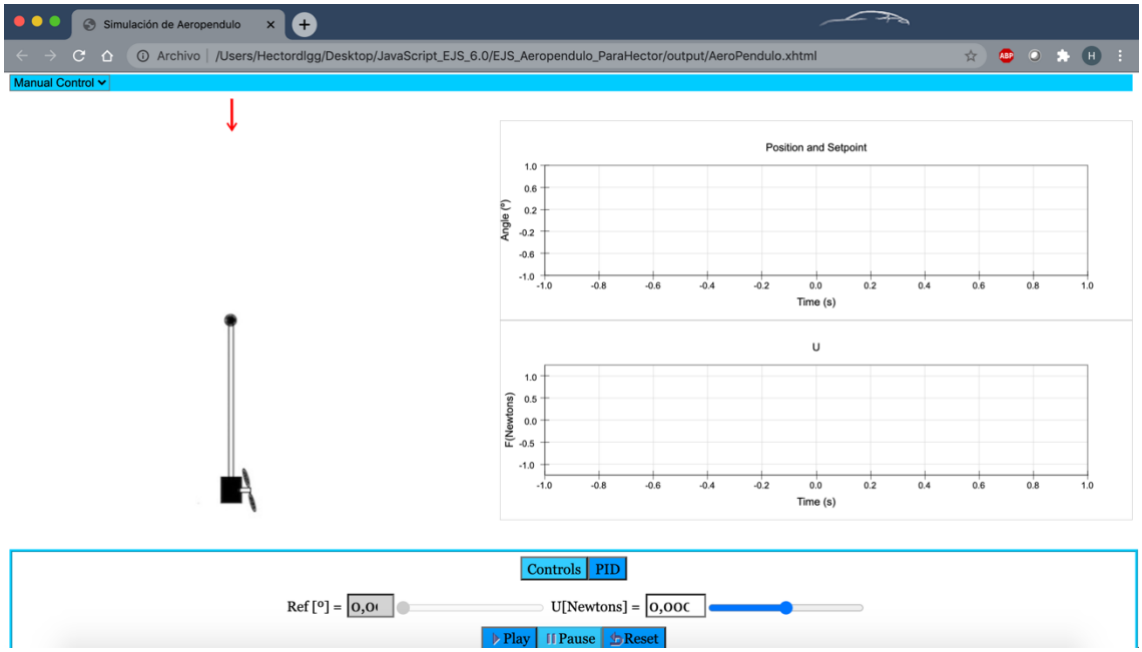


Figura 41: Interfaz web creada con EJS

5.5.5. Comprobación de resultados.

Acabado el desarrollo de la aplicación en Easy Java Simulations, el último paso a realizar es comprobar que las respuestas dadas por la simulación corresponden a las esperadas. Estas deben ser similares a las obtenidas teóricamente y visualizadas anteriormente mediante MATLAB – Simulink, cumpliendo las especificaciones dinámicas y siguiendo correctamente la referencia.

Para ello iniciaremos la aplicación e introduciremos los valores calculados del PID para cada uno de los puntos de equilibrio calculados. Además, podemos comprobar con otros puntos de equilibrio para verificar que se obtiene una respuesta adecuada en todo el rango de ángulos establecido para el regulador.

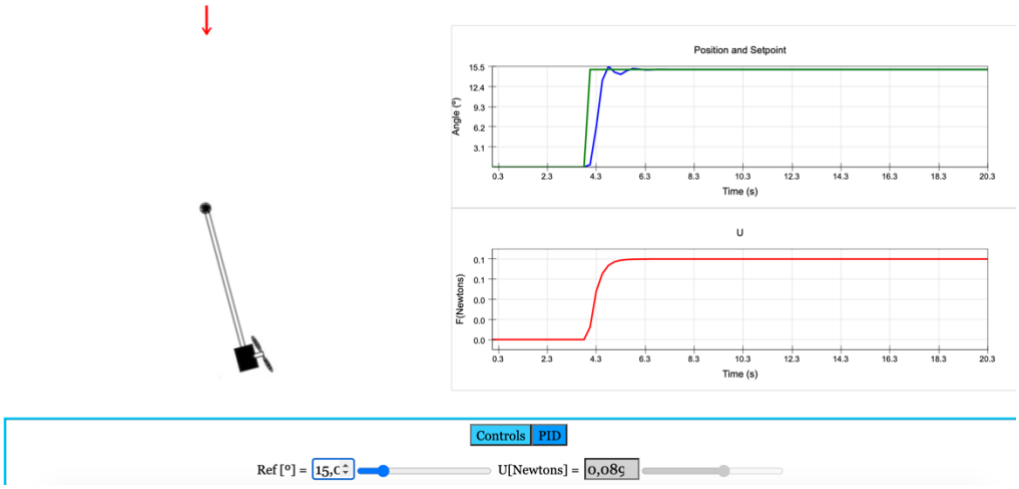


Figura 42: Simulación para $\theta_{eq} = 15º$

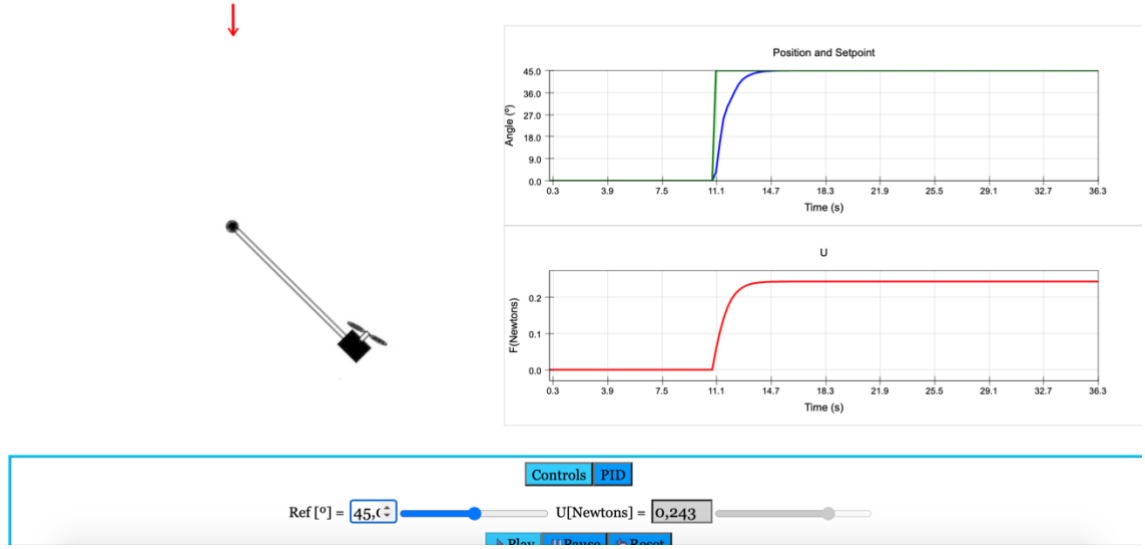


Figura 43: Simulación para $\theta_{eq} = 45º$

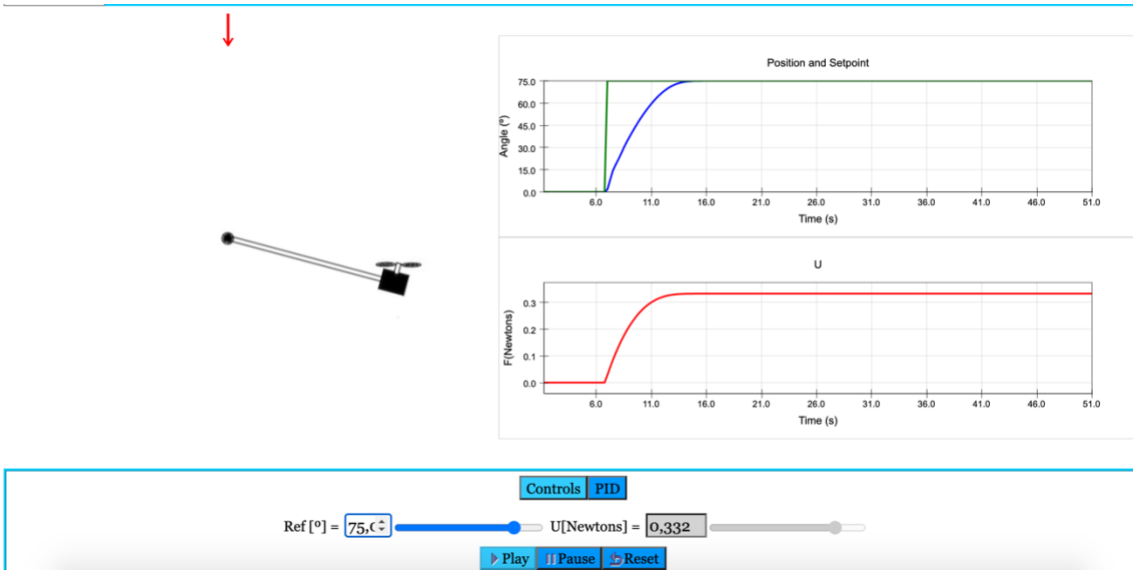


Figura 44: Simulación para $\theta_{eq} = 75º$

Se puede comprobar que las respuestas presentan el comportamiento esperado, cumpliendo las especificaciones estáticas y dinámicas. La única diferencia que se puede apreciar es una pequeña sobreoscilación en el caso de $\theta_{eq} = 15º$. Esto se puede explicar por un fallo al acercarse demasiado el tiempo de muestreo al tiempo de establecimiento.

6. CONCLUSIONES Y VALORACIÓN PERSONAL

Tras finalizar el desarrollo del entorno virtual se puede afirmar que se han cumplido los objetivos de realizar el modelado, control y desarrollo del entorno para un aeropéndulo.

La realización de este proyecto ha servido para profundizar y aplicar los conocimientos sobre modelado y control estudiados durante el grado de ingeniería electrónica industrial y automática. Además, ha servido para conocer e introducirme en la herramienta Easy Java Simulations, la cual, tras trabajar con ella, considero una opción muy interesante para la visualización de sistemas físicos en entornos web.

El desarrollo de aplicaciones web de este tipo para la validación de estrategias de control resulta muy interesante de cara a su uso en el entorno docente, pues permite al alumnado visualizar una representación del sistema físico en el que están trabajando, y no solo una respuesta a modo de gráfica.

Mencionar también por tanto el gran interés que supone el proyecto UNILabs, desarrollado la UNED junto con otras universidades españolas para el desarrollo tanto de laboratorios virtuales como laboratorios remotos, que con las circunstancias actuales de pandemia adquiere una posible alternativa muy interesante para estudiantes.

BIBLIOGRAFÍA

[1] Control Automático. Tiempo Continuo y Tiempo Discreto.
Julián J. Salt Llobregat. Ángel Cuenca Lacruz. Vicente Casanova Calvo. Antonio Correcher Salvador.

[2] Sintonización y comparación de controladores para un aeropéndulo.
Gregorio Castillo. Elda Gómez. Elisa Gonzaga. Iván Reyes.

[3] Mechatronic Aeropendulum: Demonstration of Linear and Nonlinear Feedback Control Principles With MATLAB/Simulink Real-Time Windows Target.
Eniko T. Enikov. Giampiero Campa

[4] Easy Java Simulations. The Manual.
Francisco Esquembre

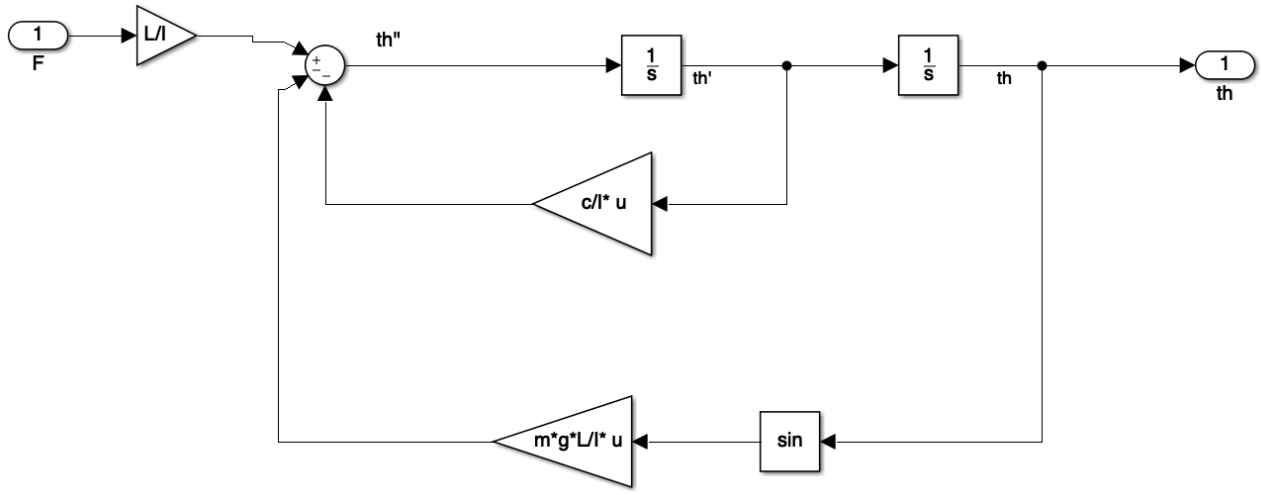
[5] https://fem.um.es/Ejs/

[6] http://isa.uniovi.es/~idiaz/ADSTel/Practicas/ModeladoPendulo.html

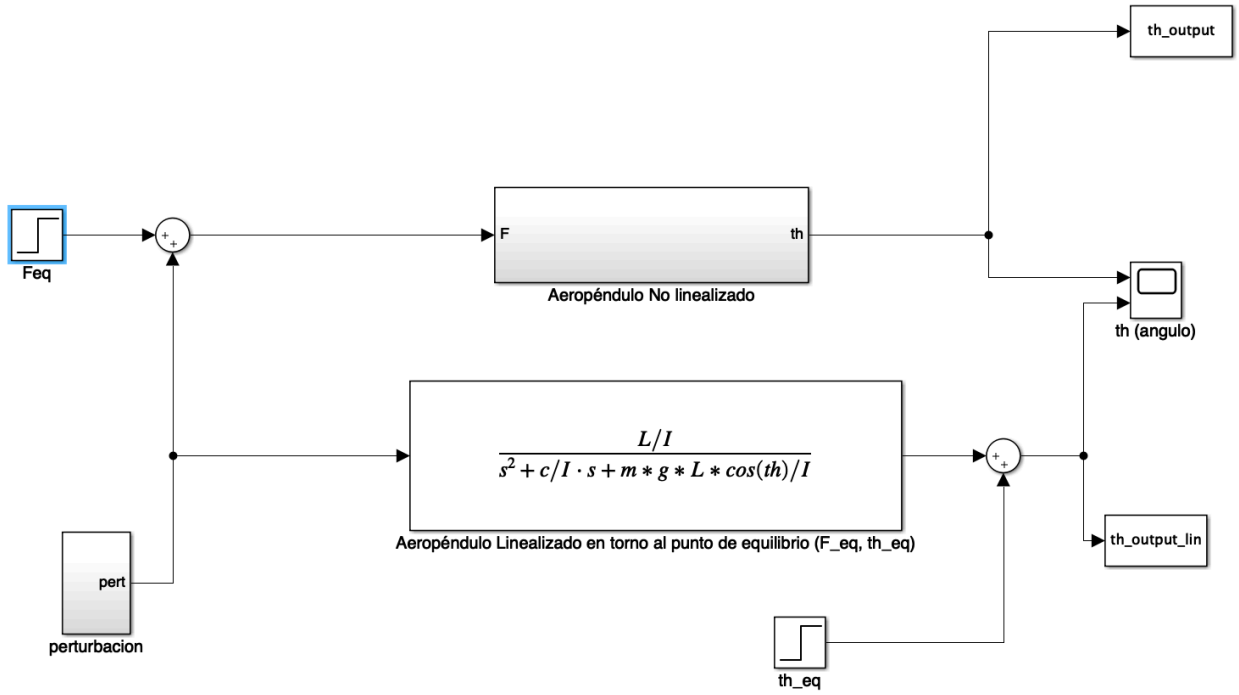
[7] https://es.mathworks.com/help/control/ug/discrete-time-proportional-integral-derivative-pid-controller.html

PLANOS

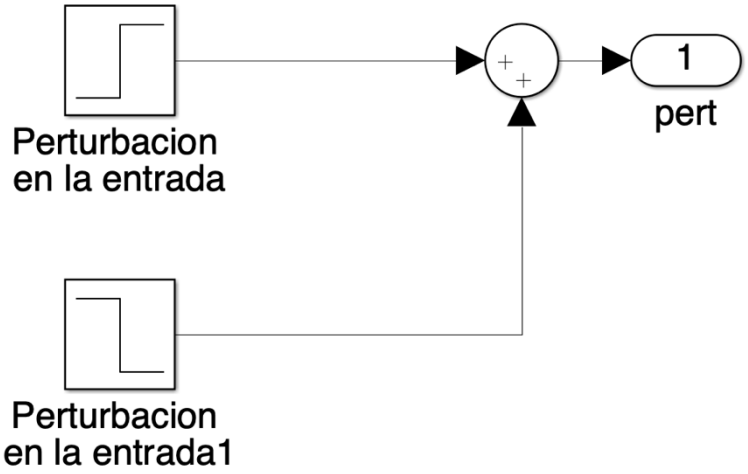
HECTOR DE LA GUIA GONZALEZ



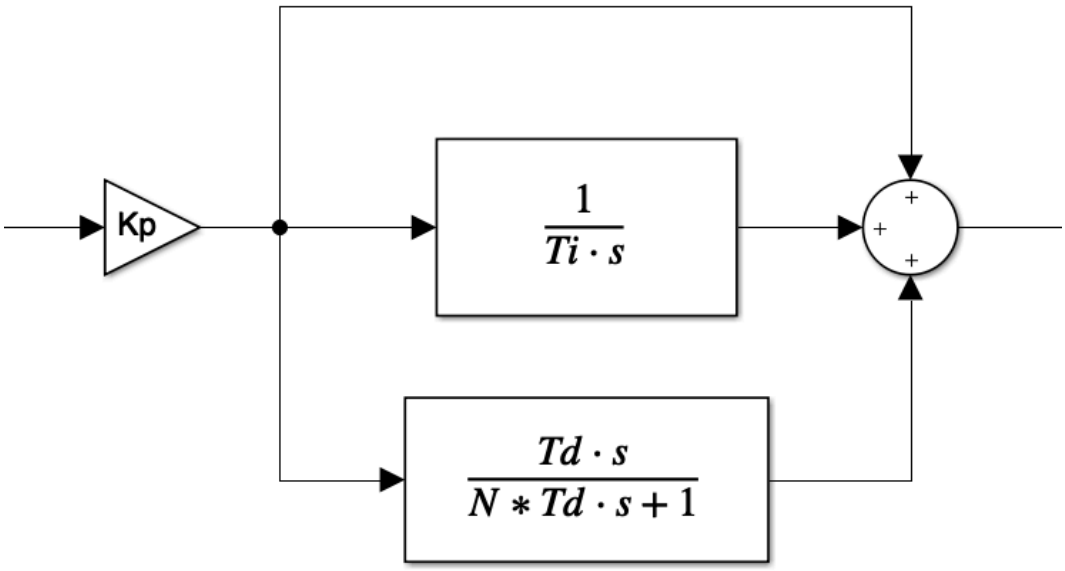
Autor			Fecha	
De La Guía González, Héctor	UPV	ETSID - DISA	08/09/2020	
Diseño de prototipo virtual web para control de un aeropéndulo		Modelado y comparación de modelos		
		Esquema Simulink del modelo no lineal	Escala 1/1	Nº plano 1



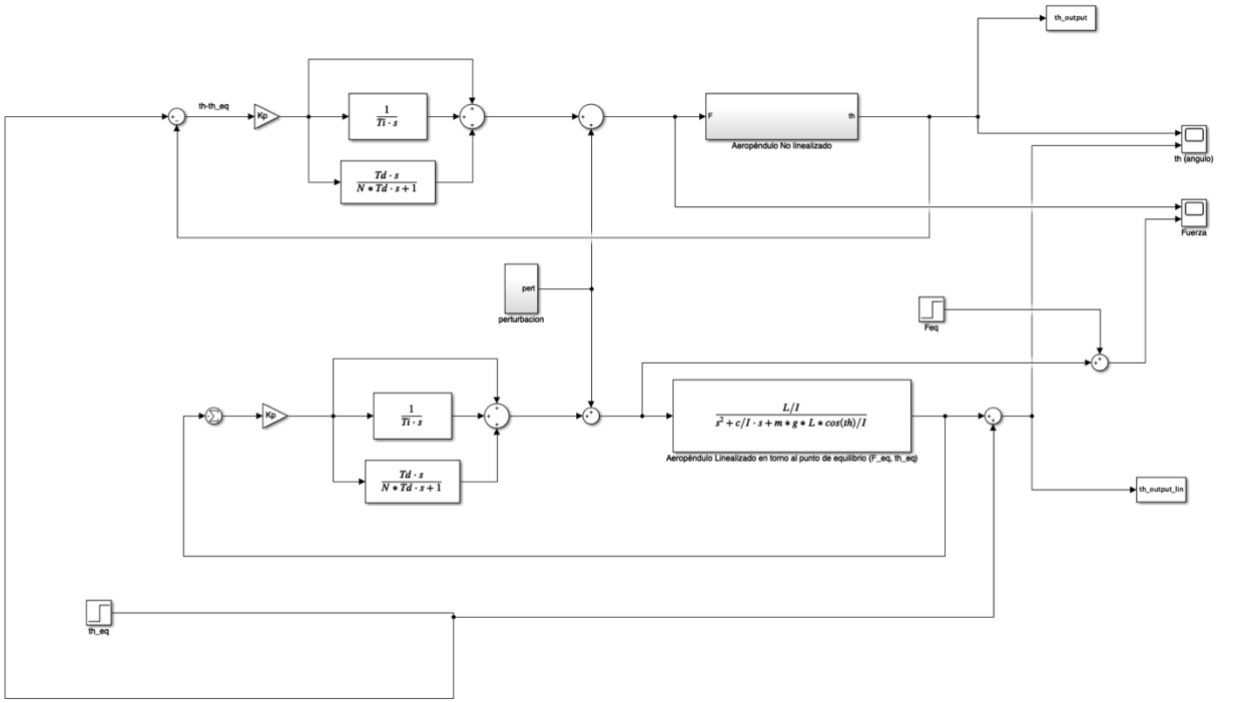
Autor			Fecha
De La Guía González, Héctor	UPV	ETSID	08/09/2020
Diseño de prototipo virtual web para control de un aeropéndulo		Modelado y comparación de modelos	
		Esquema Simulink para comparación de modelos	Escala: 1/1 — Nº plano: 2



Autor			Fecha	
De La Guía González, Héctor	UPV	ETSID	08/09/2020	
Diseño de prototipo virtual web para control de un aeropéndulo		Modelado y comparación de modelos		
		Esquema Simulink para introducción de perturbaciones	Escala 1/1	Nº plano 3



Autor			Fecha
De La Guía González, Héctor	UPV	ETSID	08/09/2020
Diseño de prototipo virtual web para control de un aeropéndulo		Diseño de PID y comparación de sus efectos en modelos lineal y no lineal	
		Esquema Simulink para introducción de regulador PID	Escala: 1/1 — Nº plano: 4



Autor			Fecha
De La Guía González, Héctor	UPV	ETSID	08/09/2020
Diseño de prototipo virtual web para control de un aeropéndulo		Diseño de PID y comparación de sus efectos en modelos lineal y no lineal	
		Esquema Simulink para comparación de modelos con PID	Escala 1/1 / Nº plano 5

PLIEGO DE CONDICIONES

[Subtítulo del documento]

HECTOR DE LA GUIA GONZALEZ

En el presente pliego de condiciones se declaran las extensiones del contrato que deben existir entre contratista y propiedad en la ejecución del trabajo de fin de grado.

1. Condiciones de los materiales

En el documento presente se desarrolla el contenido explicativo del proceso seguido para el modelado, control y simulación de un aeropéndulo, diseñando un control de modo que sea capaz de realizar el seguimiento de referencias que en este caso vendrán dadas a modo de ángulos de equilibrio.

Toda la documentación generada por escrito en este documento, así como los esquemas de control, archivos MATLAB, o cualquiera de los archivos generados mediante Easy Java Simulations (extensión ejss o .html) podrán ser accesibles para cualquiera que lo desee, tanto para su uso particular como para posteriores ampliaciones o variaciones sobre el trabajo. La regulación de este trabajo queda por tanto en disposición de la UPV para su distribución a través de cualquiera de sus plataformas.

El contenido generado en el proyecto a parte del presente documente es el siguiente:

- Archivo con extensión .m de MATLAB para la declaración de variables.
- Modelos Simulink para comparación de respuestas en los diferentes modelos generados.
- Prototipo web generado en Easy Java Simulations para la representación del proceso y la visualización gráfica de la respuesta obtenida.

Las condiciones de trabajo de los diferentes softwares utilizados durante el proyecto, son las siguientes:

- MATLAB – Simulink: Ha sido empleada una licencia individual de uso académico, pudiendo ser esta activada por el mismo usuario hasta en cuatro equipos diferentes.
- Easy Java Simulations: Ha sido empleada la versión EJSs 6.0, disponible para su descarga libre.

PRESUPUESTO

HECTOR DE LA GUIA GONZALEZ

El coste económico de este proyecto se puede dividir en tres partes principales, los costes materiales, los costes de software y los costes de mano de obra.

1. Costes materiales

El principal soporte material empleado en el proyecto ha sido un ordenador, que se ha usado para llevar a cabo todas las tareas que han compuesto el proyecto, tanto la realización de la memoria como el modelado, control y posterior simulación del aeropéndulo.

El ordenador utilizado en este caso ha sido un ordenador portátil modelo MacBook Pro 13'' Retina, que cuenta con un procesador Intel Core i5 de 2,7 GHz, memoria RAM de 8 GB DDR3, tarjeta gráfica Intel Iris Graphics 6100 de 1536 MB y 250 GB de almacenamiento.

En la siguiente tabla se encuentra el coste proporcional del ordenador, asumiendo que el periodo de utilización del mismo es de 5 años y que ha sido utilizado para la realización del proyecto durante 4 meses.

PRESUPUESTO DE COSTES MATERIALES					
Código	Ud	Descripción	Cantidad	Precio	Total
A1	mes	Ordenador portátil modelo MacBook Pro 13'' Retina, que cuenta con un procesador Intel Core i5 de 2,7 GHz, memoria RAM de 8 GB DDR3, tarjeta gráfica Intel Iris Graphics 6100 de 1536 MB y 250 GB de almacenamiento.	4	25 €	100,00 €

2. Costes de software

Como se explica en la memoria, las dos herramientas de software usadas durante la realización del proyecto han sido MATLAB, y Easy Java Simulations. Easy Java Simulatios es una aplicación de software libre, por lo que su utilización ha tenido un coste nulo.

Con respecto a MATLAB, se opta por la licencia anual para educación de la cual se expresa el coste proporcional.

PRESUPUESTO DE SOFTWARE					
Código	Ud	Descripción	Cantidad	Precio	Total
B1	mes	Licencia de MathWorks sobre MATLAB Educational	4	20,83 €	83,32 €
B2	mes	Easy Java Simulations	4	0	0 €
		Subtotal			83,32 €

3. Costes de mano de obra

La mano de obra será la de un graduado en Ingeniería Electrónica Industrial y Automática, titulación que da el rango de Ingeniero Técnico Industrial.

PRESUPUESTO DE MANO DE OBRA					
Código	Ud	Descripción	Cantidad	Precio	Total
C1	h	Ingeniero técnico industrial	300	22 €	6.600,00 €

4. Resumen del presupuesto

RESUMEN DE PRESUPUESTO	
Descripción	Total
Costes materiales	100,00 €
Costes de software	83,32 €
Costes de mano de obra	6.600,00 €
TOTAL	**6.783,32 €**