UNIVERSIDAD CARLOS III DE MADRID
ESCUELA POLITÉCNICA SUPERIOR
DPTO. DE INGENIERÍA DE SISTEMAS Y AUTOMÁTICA

DEPARTAMENTO DE INGENIERÍA DE SISTEMAS Y AUTOMÁTICA

IMPLEMENTACIÓN DE CONTROLADORES PID CON EASY JAVA SIMULATIONS PARA APLICACIONES WEB DOCENTES

PROYECTO FIN DE CARRERA
INGENIERÍA TÉCNICA INDUSTRIAL: ELECTRÓNICA INDUSTRIAL

AUTOR: **SARA ANTÓN MONTERO**
TUTOR: **RAMÓN BARBER CASTAÑO**

JULIO 2010

Agradecimientos

Muchas veces he pensado en este momento como en algo inalcanzable, ficticio; algo que parecía que nunca iba a llegar.

Me emociono escribiendo porque han sido muchos años, con momentos más bonitos y otros más difíciles, pero siempre cargados de esfuerzo y de ganas, y al final consiguiendo superar siempre las metas propuestas, incluso llegando a sacar matrícula de honor en expresión gráfica, mi peor pesadilla…

En primer lugar quiero dar las gracias a los profesores que me han ayudado de manera desmesurada cuando he acudido en busca de su ayuda; entre ellos quiero destacar a Juan Manuel Martínez Tarifa, Roberto Martínez Bejarano y Ramiro Díez Zaera.
También quiero dar las gracias a Ramón Barber, mi tutor de proyecto, admirable por su elevado grado docente y su sencillez.

El mejor recuerdo que me llevo de la universidad sois vosotros: Isa, Andrea y Wei. Gracias por todas las risas compartidas.

Gracias Carol y Sandra porque sois las mejores; por sacarme de casa cuando me encerraba en mi mundo y por todos los viajes, momentos y buenas charlas que hemos tenido.

Gracias a ti Yago. Eres una persona genial.

Gracias Chus, Isa y Toño por apoyarme siempre, siempre escrito en mayúsculas, por haberme animado como vosotros sólo sabéis; y lo peor de todo, por aguantar mis impertinencias en épocas de exámen. Os quiero mucho.

Y el GRACIAS más grande se lo debo a mis padres. De vosotros he aprendido lo que es el sacrificio, la honradez y la humildad. Gracias por ser las mejores personas que han existido en este mundo.

Gracias.

Índice general

Índice de figuras

Capítulo 1

Introducción

1.1 MOTIVACIÓN

El objeto de este proyecto es demostrar al lector la eficacia y la flexibilidad de la aplicación informática *Easy Java Simulations (EJS)*, una herramienta empleada en el ámbito técnico para diseñar simulaciones por ordenador.

Las simulaciones son el ingrediente fundamental de la implantación de los laboratorios virtuales, presenciales o remotos, y su correcto diseño puede aportar la calidad necesaria a estos laboratorios para que puedan desplazar o descargar significativamente de trabajo a sus homólogos reales. Un programa que permita crear estas simulaciones con la mayor rigurosidad científica, presentarlas con un interfaz amigable para operadores y/o alumnos e integrarlas en los sistemas informáticos más empleados, siendo la herramienta necesaria para aportar esa calidad buscada [9].

A su vez los laboratorios virtuales cuentan con una serie de considerables ventajas respecto a los reales: son mucho más económicos, son versátiles, no requieren mantenimiento, pueden ser configurados para emplearse desde el hogar y a cualquier hora, muchos usuarios y pueden usarlos simultáneamente [10]. El correcto aprovechamiento de dichas ventajas debería permitir implantar plenamente estos laboratorios en los planes de estudios de los centros docentes y en la formación de personal de las empresas, de forma que pudiesen suplir las carencias y dificultades que inevitablemente plantean las prácticas presenciales en laboratorios reales.

Las teorías del conocimiento constructivo, la tendencia al aprendizaje activo y el Plan de Bolonia crean un contexto en el que la participación del alumno y el trabajo práctico deben primar cada vez más en la enseñanza. En este ambiente el empleo de *EJS* y sistemas similares pueden ser una herramienta muy útil para adaptarse a las nuevas necesidades y potenciar estas facetas del aprendizaje.

Así, con este proyecto se espera demostrar, de forma práctica y teórica, que *EJS* permite crear simulaciones que pueden sustituir y/o complementar a los ejercicios prácticos tradicionales con una serie de ventajas que no van en detrimento de la formación que los estudiantes necesitan.

En [2] se comenzó con el estudio de *EJS* y se implementaron unos primeros sistemas sencillos.

1.2 OBJETIVOS DEL PROYECTO

El principal fin de este proyecto es desarrollar prácticas de la asignatura Señales y Sistemas basándose en simulaciones sobre plataformas informáticas. Con dichas prácticas se pretende la correcta comprensión de los sistemas y fenómenos emulados por ordenador sin que el estudiante tenga que acudir físicamente al laboratorio del centro docente.

Las simulaciones que se desarrollan son: el control del llenado de un depósito, el control del movimiento de un avión y el control del desplazamiento del émbolo en un cilindro hidráulico. Sólo en el primer caso se estudia también cómo trabaja el sistema no linealizado.

Los objetivos pueden concretarse en:

- Estudio de *EJS* como herramienta de diseño de Sistemas Virtuales.
- Desarrollo de tres sistemas con *EJS*:
  - un depósito controlado
  - un avión controlado
  - un cilindro hidráulico controlado
- Desarrollo de tres casos prácticos:
  - En el sistema del depósito se compara el modelo real y el lineal, ambos controlados, y se estudia el comportamiento de los diferentes reguladores.
  - Para el sistema del avión se estudia el sistema lineal junto con las posibles opciones para poder controlarlo.
  - En el caso del cilindro hidráulico se comparan los distintos reguladores para el sistema lineal, analizando su señal de control.

Las simulaciones creadas están enfocadas a ofrecer la máxima interactividad al alumno y junto al interfaz gráfico presentan todos los datos de la evolución que este puede requerir. A su vez es necesario que el alumno realice un estudio previo para el total aprovechamiento de su trabajo.

1.3 PARTES DEL DOCUMENTO

Este Proyecto Final de Carrera se divide en ocho capítulos, detallados en el índice. A continuación se realiza una breve descripción de cada uno de ellos, con excepción del primero, para poder dar una visión panorámica de la estructura del trabajo.

El *capítulo 2* desglosa todas las partes de *EJS* de interés para sus usuarios. Se detallan las posibilidades que el programa ofrece, así como la forma de emplearlas, para crear una simulación desde el planteamiento matemático al diseño del interfaz.

En el *capítulo 3* se hace una introducción al control: partiendo de la definición de los elementos básicos de un sistema de control, pasando por la respuesta estática y dinámica de los sistemas de control y describiendo los posibles tipos de control que actúan para corregir el error de un sistema y hacer que mejore su respuesta.

El *capítulo 4* muestra al detalle el modo en que se elabora la simulación del control del llenado de un depósito, comentando cada paso tanto del modelo real como del lineal.

En el *capítulo 5* se describe paso a paso el modo en que se lleva a cabo la simulación, explicando cada fase, del control del movimiento de un avión.

El *capítulo 6* presenta la confección de la simulación del control del desplazamiento del pistón de un cilindro hidráulico, ofrecida punto a punto.

El *capítulo 7* muestra los resultados de la sistemática modificación de las variables de los tres sistemas y plasma las distintas respuestas para cada uno de ellos. Una vez estudiados los resultados aporta una serie de conclusiones para explicar su funcionamiento.

El *capítulo 8* contiene las conclusiones del proyecto en todo lo referente a *EJS* y su validez y potencial para el desarrollo de simulaciones; y aporta nuevas líneas de investigación para futuros proyectos.

Por último se incluye la bibliografía, donde se enumeran las publicaciones y páginas Web empleadas en el proyecto o de interés en relación con su temática.

Capítulo 2

Easy Java Simulations (EJS)

2.1. INTRODUCCIÓN

El contenido de este capítulo se basa en la lectura de [1].

2.1.1. ¿Por qué *EASY JAVA SIMULATIONS*?

El motivo que ha llevado a elegir *Easy Java Simulations* para realizar este trabajo es porque es una herramienta que presenta las características idóneas para el entorno docente. Dicho entorno tiene una serie de exigencias que pueden concretarse en las siguientes:

- Necesidad de acceso desde ordenadores particulares alejados del equipo que aloje a la aplicación.
- Compatibilidad con la multitud de configuraciones que los ordenadores personales pueden tener.
- Posibilidad de empleo simultáneo por parte de múltiples usuarios.
- Versatilidad.
- Facilidad de empleo.

Al ser una aplicación basada en Java, *EJS* es compatible con los sistemas operativos más empleados. Esto permite que los equipos informáticos que se emplean en nuestro contexto (ordenadores personales compatibles) puedan emplearla, sin importar si se trata de terminales del centro de estudios o de terminales domésticos de los alumnos.

Java puede integrarse en la mayoría de los navegadores de Internet, de forma que los desarrollos basados en ella pueden ser ejecutados por cualquier equipo con una configuración compatible con dichos navegadores, disfrutando además de todas las posibilidades que una página web puede ofrecer. No solo eso, el hecho de que *EJS* funcione a través de un navegador hace muy sencillo ubicar nuestro trabajo en un servidor y hacer que los usuarios accedan a él a través de un sitio web sin necesidad de albergar en sus ordenadores la aplicación en concreto. Esta infraestructura es sencilla de establecer, tanto a través de redes locales como a través de Internet. Asimismo, la arquitectura "cliente-servidor" permite que todos los usuarios que lo deseen trabajen al mismo tiempo con la aplicación sin interferirse mutuamente.

EJS facilita enormemente, para los fines que nos interesan en este proyecto, el empleo de los ya de por sí populares códigos Java. Además la vista que el usuario tiene de las simulaciones desarrolladas puede ser muy sencilla e intuitiva. Lo mismo puede decirse de los

navegadores web y los procedimientos para emplear Internet o una red de área local para acceder remotamente a información. Estas características, junto a las previamente señaladas, no solo garantizan la versatilidad y la sencillez de las simulaciones sino también su reducido coste económico.

2.1.2. ¿Cómo funciona *EASY JAVA SIMULATIONS*?

Una simulación por ordenador es un programa informático que trata de emular un sistema real a través de la visualización de los diferentes estados que éste puede presentar. Cada uno de estos estados está descrito por los valores que toman un conjunto de variables. Dichos valores pueden venir determinados por el tiempo, en función de un algoritmo que se repite continuamente, o por los valores de otras variables, dependiendo de las ecuaciones que las relacionan y de la interacción de los usuarios.

Las emulaciones de fenómenos reales son programas en sí mismos. Es más: muchos sistemas reales, para ser simulados con rigurosidad, pueden requerir de desarrollos muy complejos. *EJS* facilita la confección de estas simulaciones, sin embargo presenta una característica (aparte de las previamente mentadas) que lo diferencia de la mayoría de programas con el mismo objetivo: *EJS* está creado para que las simulaciones las diseñen profesores y alumnos, con fines de investigación y pedagógicos, de forma que vuelca su esfuerzo en facilitar su empleo a usuarios que no son expertos en informática. Es responsabilidad de estos últimos definir las variables y las ecuaciones adecuadas y la herramienta las transforma en las instrucciones Java correspondientes y las aplica siguiendo siempre una misma rutina. Así se permite a los diseñadores hacer hincapié en la parte científica y en los verdaderos objetivos de la simulación. Otro tanto sucede con los interfaces de los diseños *EJS*, sencillos de diseñar y de emplear, que ofrecen las mismas posibilidades que cualquier otra aplicación Java.

Solo si se quieren implementar opciones complejas al margen del desarrollo de la simulación es necesario un conocimiento de instrucciones Java complejas. En cualquier caso la sencillez y la interactividad, bazas principales de *EJS*, en ningún momento van en detrimento de la validez técnica de la simulación.

2.1.3. Arrancando *EASY JAVA SIMULATIONS*

Al ejecutar *EJS* emergen dos ventanas: la consola de EJS y la interfaz de usuario.

La primera ventana presenta opciones de configuración del programa: aspecto visual, directorios empleados, mensajes sobre el funcionamiento de la aplicación, etc. Dado que estas opciones son de un menor interés al estudiar el proceso de diseño de una simulación, remito al lector interesado a [2].

La segunda ventana emergente se estudia en profundidad en el siguiente apartado.

2.1.3.1. Interfaz del usuario

Es esta ventana la que emplea el usuario para diseñar su simulación *(ver figura 1).*

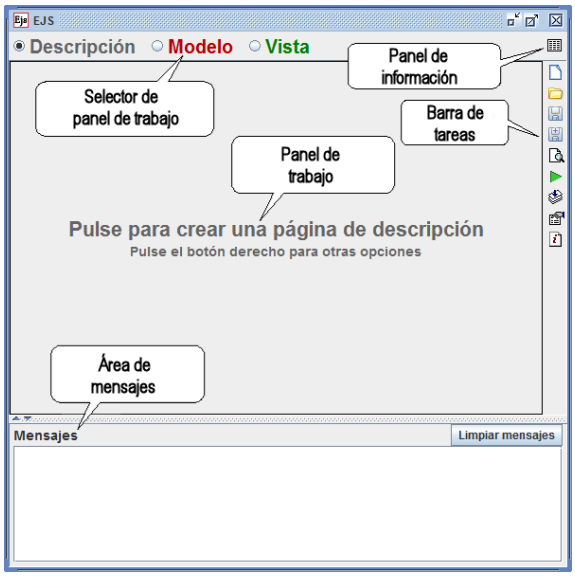

Figura 1: Ventana inicial del interfaz de usuario

La barra de tareas de la derecha tiene una lista de iconos que permiten hacer gestiones sobre las simulaciones con las que se está trabajando: iniciar una nueva, abrir y guardar una simulación, buscar en el código Java, configurar *EJS* o mostrar el menú de ayuda de *EJS*. De la misma forma ejecutan una simulación, empaquetan una o más simulaciones en un fichero JAR o las exportan como objetos para páginas web. En el área de mensajes se muestra la información que el programa tenga a bien proporcionar acerca del desarrollo del diseño o de la ejecución de

la simulación.

La parte central y más importante es el panel de trabajo, en la que se desarrollan todas las tareas de creación de la simulación. Puede mostrarse, según marquemos en el selector, en tres formas distintas: *descripción, modelo y vista*.

Descripción se emplea para crear y editar código HTML adjunto a la simulación. Estas instrucciones, escritas en el lenguaje empleado para la elaboración de páginas web, permiten introducir elementos o páginas previas que el usuario ve antes de ejecutar la simulación o que se integran en un sitio web junto a ésta si es el caso. Esto hace posible adjuntar a las simulaciones instrucciones previas, explicaciones, cuestionarios y multitud de opciones más. Así se contribuye a tener simulaciones más amistosas y completas.

Los paneles *modelo* y *vista* por su importancia son estudiados en los siguientes apartados.

2.2. MODELO

Desde este panel se elabora la parte matemática de nuestra simulación y son los datos y algoritmos que en él se introducen los que el computador emplea a la orden del usuario. Tiene cinco subpaneles: *Variables, inicialización, evolución, relaciones fijas* y *propio*.

Al entrar en cada subpanel por primera vez se genera automáticamente una "página", un elemento que sirve para organizar en grupos la información que corresponda a cada apartado. *EJS* permite crear varias páginas en un mismo panel con una finalidad puramente organizativa de agrupar variables o ecuaciones con fines similares. EJS procesa estas páginas de igual manera; siempre de izquierda a derecha.

En cualquier momento pueden crearse nuevas páginas, eliminarse, activarse o desactivarse, modificarse, etc.

En los siguientes apartados se explica como funciona cada subpanel por separado y los pasos a seguir para crear el modelo de una simulación.

2.2.1. Variables

El primer paso en la elaboración del modelo es declarar las variables. Éstas necesitan: un nombre válido, declarar de que tipo son, un valor inicial y una dimensión si se trata de un vector o una matriz. Además en la parte inferior del editor existe un campo en el que se puede incluir una breve descripción de la variable.

2.2.1.1. Nomenclatura

A lo largo del proceso de confección se da nombre a multitud de elementos (variables, páginas, métodos propios y elementos de la vista) y para evitar conflictos entre ellos y con los procesos de *EJS* deben seguirse las siguientes normas:

- En el modelo cada variable o método propio debe tener un nombre único. Lo mismo sucede con la vista de la simulación y sus elementos.
- Los nombres de los elementos han de ser series de caracteres alfanuméricos que empiecen con un carácter alfabético. No existe una longitud límite y se puede emplear el símbolo "_".
- Se recomienda que el primer carácter de todos los nombres del modelo se escriba en minúsculas y el primero de los elementos de la vista en mayúsculas.
- Es recomendable poner nombres descriptivos que no contengan espacios en blanco.
- No se pueden emplear nombres empezando con "_".

Además Java y *EJS* se reservan el empleo de los siguientes nombres: *boolean, break, byte, case, catch, char, continue, default, do, double, else, float, for, getSimulation, getView, if, initialize, instanceof, int, long, Object, reset, return, short, step, String, switch, synchronized, throws, try, update, while.*

No es necesario tener en cuenta estas convenciones al dar nombre a las páginas de los subpaneles.

2.2.1.2. Tipos de variables

Para hacer un uso óptimo de sus capacidades el ordenador debe saber con que clase de variables va a trabajar, para poder decidir cuanta memoria adjudicarles y como trabajar con ellas. Java permite definir un gran número de tipos, pero *EJS* sólo trabaja con las siguientes clases de variables:

- *Boolean*: para valores tipo verdadero o falso (*true* y *false* respectivamente).
- *Int*: para valores enteros.
- *Double*: para números reales.
- *String*: para trabajar con caracteres y textos.
- *Object*: permite introducir "objetos" desarrollados con Java que requieren de conocimientos avanzados en este programa para elaborarse.

2.2.1.3. Valor inicial

El valor que en el instante inicial de la simulación tiene una variable puede ser una constante o una expresión sencilla que definiremos en la correspondiente cuadrícula de este subpanel. Si no se especifica nada en esta cuadrícula *EJS* marca un valor por defecto (0 si es una variable numérica). En el subpanel *Inicialización* o a través de las ecuaciones de ligadura también pueden determinarse los valores iniciales de las variables.

2.2.2. Inicialización

Para inicializaciones complejas, como cuando se debe dar un valor propio a cada elemento de una matriz por ejemplo, la cuadrícula de *Valor inicial* del subpanel *Variables* no es suficiente.

En este subpanel se pueden crear tantas páginas como se quiera, en las que se escribirán las instrucciones necesarias para dar valores iniciales a las variables deseadas. Estas instrucciones serán las primeras en ejecutarse cuando se arranque la simulación.

2.2.3. Evolución

En la *figura 2* se observa que este subpanel es más complejo que los anteriores. En él se escriben las normas que marcan como varía el estado del sistema a simular en función del tiempo: Las ecuaciones de evolución.

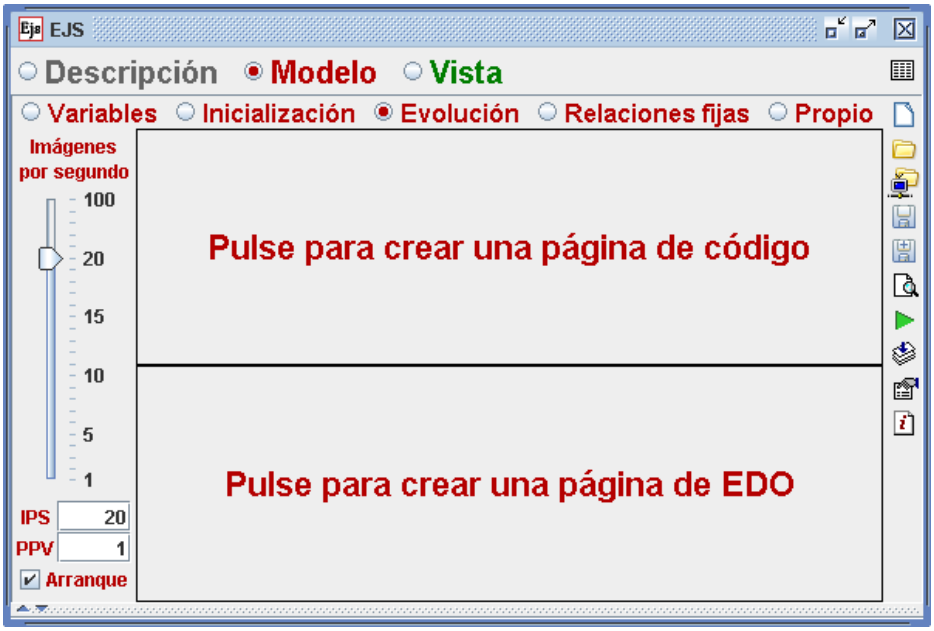

Figura 2: Subpanel *evolución*

El Subpanel está dividido en tres partes: dos botones para crear páginas y una parte vertical situada a la izquierda de éstos en la que podemos especificar los parámetros de la evolución.

IPS (Imágenes Por Segundo) permite especificar la frecuencia con que *EJS* realizará todos los pasos de la simulación y puede modificarse desde el deslizador o tecleando un valor en la celda *IPS*. Una cadencia demasiado elevada dificulta localizar un instante concreto y puede sobrecargar el equipo, normalmente con un *IPS* de valor comprendido entre 10 y 20 es suficiente. *PPV* (Pasos Por Visualización) marca el número de veces que el ciclo de la simulación se repite antes de que se modifique la vista. Si lo que se busca es una aproximación en tiempo real de lo que tarda en ejecutarse la simulación se tiene que poner un *IPS* de valor 10 junto con un valor asignado a la variable diferencial de tiempo de 0,1 segundos.

Al crear las páginas en las que se escriben las ecuaciones se tiene que decidir entre páginas para código ordinario (*Pulse para crear una página de código*) o páginas para ecuaciones diferenciales (P*ulse para crear una página de EDO*). Si el sistema trabaja en intervalos discretos de tiempo los cálculos del programa son sencillos para el ordenador y puede definirlos directamente el usuario en una página de código normal. Sin embargo si la simulación trabaja en el tiempo de forma continua los cálculos son mucho más complicados para la maquina. Las ecuaciones diferenciales deben convertirse en ecuaciones discretas para facilitar el trabajo al ordenador. *EJS* puede realizar automáticamente esta traducción si así se desea.

Para trabajar con el tiempo de forma discreta se crean "páginas de código" y allí se escriben las ecuaciones adaptadas a Java, teniendo en cuenta todas las consideraciones de carácter general que se han indicado en apartados anteriores.

Si se trabaja con tiempo continuo deben elegirse "páginas EDO". Las particularidades de trabajar con estas ecuaciones se detallan en el siguiente apartado.

2.2.3.1. Ecuaciones diferenciales ordinarias (EDO)

Las ecuaciones diferenciales ordinarias a introducir en *EJS* deben ser explícitas de primer orden. Esta limitación no supone ningún impedimento, sin embargo requiere en ocasiones de una correcta elección de variables, de variables auxiliares y/o de un trabajo previo sobre los sistemas de ecuaciones. También hay que tener en cuenta que todas las ecuaciones diferenciales que compartan una misma variable independiente deben escribirse en la misma página.

Para introducir en *EJS* un sistema de ecuaciones diferenciales en primer lugar hay que declarar cuál es la variable independiente (normalmente es el tiempo). A continuación hay que indicar el "paso de integración" que se quiere utilizar. El "paso de integración" marca el aumento del valor de la variable independiente al final del proceso matemático que se da en cada paso de la evolución de la simulación.

Por último, para escribir las instrucciones propiamente dichas se llega a una tabla en la que cada fila corresponde a una ecuación. En una de las cuadrículas de la fila se indica la variable que se quiere derivar y en la cuadrícula contigua se plasma su expresión matemática.

Cuando se trabaja con expresiones muy complejas pueden emplearse "métodos Java propios", instrucciones predefinidas de Java, para hacerlas más sencillas y breves. En cualquier caso, si se emplean estos "métodos", debe tenerse en cuanta que sus parámetros (si los hay) deben estar correctamente referenciados a las variables o podrían producirse errores.

2.2.4. Relaciones fijas

El valor de cada variable no depende únicamente del tiempo: las modificaciones en otras variables también pueden afectarle. Estos lazos son lo que se denomina "relaciones fijas". Estas relaciones, independientes de las ecuaciones de evolución y del tiempo, están plasmadas en las ecuaciones de ligadura.

No siempre es sencillo discernir entre ecuaciones de ligadura y de evolución, pero es importante para la simulación distinguir los dos tipos ya que cada uno de ellos se aplicará en momentos distintos y con distinta prioridad. Por ejemplo: las relaciones fijas deben tener efectos prácticos incluso cuando la simulación está detenida (en el caso hipotético de que el usuario alterara manualmente una variable mientras la evolución está pausada) y en cambio la evolución no.

Este subpanel funciona de forma análoga a las páginas de código del subpanel *Evolución*: se escriben las ecuaciones de ligadura del modelo transcritas a lenguaje Java.

2.2.5. Código propio

El objeto de este subpanel es insertar métodos Java en la simulación, bien para simplificar y abreviar los códigos o bien para añadir funcionalidades soportadas por Java al producto final.

Pese a que el funcionamiento del subpanel es parecido al de los subpaneles *inicialización* y *relaciones fijas*, los métodos que aquí se introducen tienen varias peculiaridades:

- Los métodos Java aquí implantados solo son ejecutados si son invocados explícitamente desde otro punto de la simulación.

- Puede emplearse cualquier método válido en Java, teniendo una mayor libertad para la creación de código.
- Estos métodos necesitan de unos datos para coordinarse con la simulación:
    - El nombre. Solo hay que tener en cuenta las limitaciones que se describen en el punto "Nomenclatura" del apartado "Variables".
    - La accesibilidad. Permite seleccionar desde que partes de la simulación (vista, modelo e incluso externamente desde JavaScript) se puede acceder a los métodos. Si se declara el método como público, con la palabra clave *public*, se puede acceder a él desde todas partes; por el contrario si al método tan sólo se puede acceder desde el modelo, se asignará la palabra clave *private* o no se asignará nada.
    - El tipo de valor de salida. Debe especificarse si el método devuelve algún valor o no, y en el caso de que lo haga qué clase de valor Java es. Si no devuelve ningún valor, el método funciona como subrutina, debe indicarse con el tipo especial *void*. Si devuelve un valor el método actuando como función debe finalizar con la sentencia *return*.
    - Los parámetros. Debe escribirse una lista de las variables locales con las que trabajará el método (puede no hacer uso de ninguna). Esta lista debe escribirse dentro de los paréntesis que siguen al nombre del método. Siempre que se invoque este método deberán proporcionársele las variables, en número y tipo, para las que se ha configurado.

Existe una librería de métodos Java específicos de EJS ("métodos predefinidos"), orientados principalmente a controlar el desarrollo de la simulación (ver libro [1], páginas 79-82). Java además cuenta con una librería de métodos enfocados a problemas matemáticos, que pueden ser empleados en EJS para simplificar los códigos de las simulaciones (ver libro [1], páginas 82-83).

2.3. VISTA

Una vez desarrollado el modelo hay que crear la vista de éste: el interfaz que presentará los datos de forma visual al usuario y permitirá a este último interactuar con la simulación. En un entorno pedagógico este aspecto de la simulación cobra gran relevancia, siendo además primordiales la sencillez y el uso intuitivo.

Programar elementos gráficos e interactivos requiere de conocimientos informáticos avanzados, pero de nuevo *EJS* simplifica enormemente las tareas puramente informáticas y facilita el trabajo del diseñador. El programa cuenta con una completa librería de objetos y códigos creados para la visualización de datos y procesos científicos, de forma que el diseñador solo tiene que elegir el que más le convenga, ubicarlo y relacionarlo con los correspondientes elementos del modelo.

2.3.1. Estructura de la vista

Se entiende por *contenedor* el elemento gráfico que pueden albergar a otros elementos en su zona de la pantalla; así se denomina padre al elemento contenedor e hijo al elemento contenido. Debido a que un contenedor puede ser a la vez padre e hijo, se puede crear una estructura de elementos gráficos en forma de árbol, a cuya raíz se la denomina *Vista de la simulación*. El primer hijo de la raíz es la *ventana principal*, que es la que aparece al generar la simulación.

En la *figura 3* hay una imagen del panel *Vista* que ilustra la arquitectura lógica de la vista de una simulación.

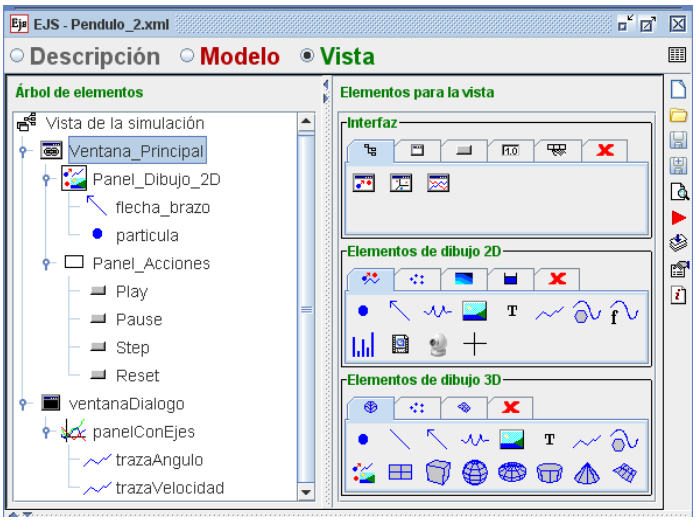

Figura 3: Panel *vista*

Existen dos clases de contenedores:

- Contenedores para elementos básicos y otros contenedores: la unión de éstos con los elementos básicos proporcionan al usuario la capacidad de controlar la simulación y modificar el valor de las variables del modelo.
- Contenedores para elementos de dibujo: junto con los elementos de dibujo se crean gráficos animados que permiten visualizar los distintos estados del modelo.

La propiedad más importante de los contenedores es la distribución. Cuando queremos añadir un elemento a un contenedor, éste lo ubicará gráficamente según el espacio que tenga, lo que solicite el elemento insertado y la política de distribución con que el contenedor haya sido configurado. Las políticas de distribución que los contenedores de *EJS* emplean son las siguientes:

- Flujo: distribuye los elementos en una fila de izquierda a derecha, como si de palabras de un texto se trataran.
- Márgenes (o border): establece a los elementos en cinco posiciones (arriba, abajo, izquierda, derecha y centro), entre las que éstos pueden elegir.
- Rejilla: sitúa los elementos en una tabla de cuadrículas idénticas, con tantas filas y columnas como se desee.
- Caja horizontal: funciona como una rejilla de una sola fila, pero no es necesario que todos los elementos tengan el mismo tamaño.
- Caja vertical: funciona como una rejilla de una sola columna, sin que todos los elementos tengan que tener las mismas dimensiones.

Los contenedores de elementos de dibujo funcionan como un sistema de ejes cartesianos en el que cada elemento de dibujo indica las coordenadas de la posición que quiere ocupar.

Esta manera de distribuir los elementos evita que se descoloquen o pierdan su posición relativa cuando cambiamos el tamaño de la vista.

2.3.2. Elementos gráficos

Para insertar un elemento gráfico hay que seguir los siguientes pasos:

1. Elección del tipo de elemento a crear.

2. Elección del contenedor que albergará al elemento.
3. Elección de un nombre para el elemento.
4. Asociación entre las propiedades del elemento y las variables de la simulación.

Existen dos familias de elementos gráficos:

- Los elementos básicos. Son los que realizan las tareas de visualizar y editar variables del modelo, invocar acciones de control y decorar la vista. Pueden ser botones, campos de edición, deslizadores, desplegables, etc. Son imprescindibles para interactuar con la simulación.
- Los elementos de dibujo. Se caracterizan porque crean gráficos animados que permiten visualizar dinámicamente los estados del modelo. Pueden tener dos o tres dimensiones y adoptan multitud de formas, sencillas y complejas. Son este tipo de elementos los que presentan de forma intuitiva y agradable los valores del modelo al usuario. Los elementos de dibuja a su vez se agrupan en elementos 2-D y elementos 3-D.

2.3.3. Asociación entre variables y propiedades

Conseguir que los elementos gráficos cambien su forma, su tamaño, su posición, su inclinación o los caracteres que muestran según los valores que adopten las variables del modelo es la razón de ser del interfaz gráfico de la simulación. Para ello hay que establecer vínculos entre los elementos gráficos y el modelo. Este es un proceso complicado a nivel informático, pero *EJS* lo realiza automáticamente si le damos unas sencillas instrucciones y además los objetos que utiliza están especialmente diseñados para este tipo de tareas.

Cada elemento gráfico tiene un menú *Propiedades* en el que se pueden determinar muchas de sus características, entre las cuales está la de asociar alguna de las propiedades del elemento a la variable del modelo que se desee. Este vínculo se establece en ambos sentidos: si la variable cambia con la evolución del modelo, el elemento de la vista refleja el cambio de forma instantánea; de la misma forma que si el usuario modifica alguna propiedad de un elemento de la vista, la variable del modelo recibe el nuevo valor.

Capítulo 3

Reguladores automáticos

3.1 INTRODUCCIÓN AL CONTROL

3.1.1 Clasificación de los sistemas de control

Se entiende por sistema de control el conjunto de elementos relacionados entre sí capaces de realizar una función deseada. Así podemos distinguir dos tipos de sistemas de control:

-Sistema de control de lazo abierto: En este tipo de sistemas la señal de salida no afecta a la señal de entrada [6].

-Sistema de control de lazo cerrado: Al contrario que en lazo abierto existe una realimentación de la señal de salida, de manera que ésta ejerce un efecto sobre la acción de control [6]. En la *figura 4* se puede apreciar un lazo cerrado de control.

Como ejemplo para su entendimiento se opta por un sistema de aire acondicionado. Se programa la temperatura idónea y por medio de un termostato cuando la temperatura difiere de la deseada, se modifica la temperatura hasta alcanzar el valor programado.

Para el caso de lazo abierto el sistema no estaría dotado de termostato, mientras que en lazo cerrado sí.

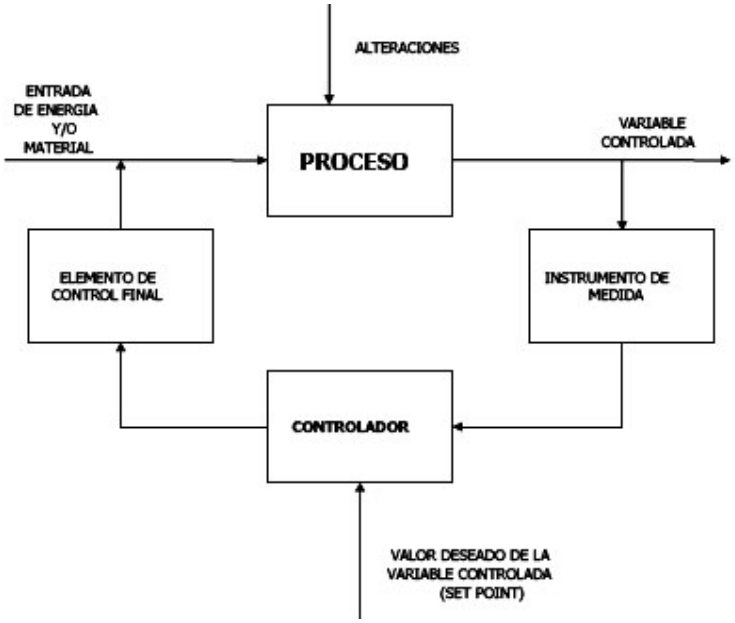

Figura 4: Lazo cerrado de control

3.1.2 Sistema de control automático

Hay que destacar que un sistema de control automático ("sistema que es capaz de regular el funcionamiento de una planta por sí mismo, de modo óptimo, corrigiendo las perturbaciones que se presentan") siempre exige un lazo cerrado de acción, ya que el controlador automático utiliza la diferencia entre al valor deseado (conocido como valor de consigna, de referencia o set-point) y la señal medida para obtener lo deseado [8,6].

En la *figura 5* se procede a detallar el funcionamiento de un sistema automático de control mediante la presentación un diagrama de bloques de control en lazo cerrado con la descripción de funcionamiento de cada elemento.

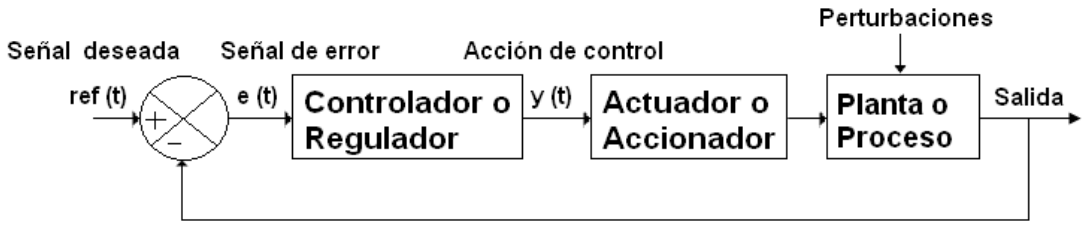

Figura 5: Sistema automático de control

Planta: Sistema sobre el que hay que actuar.

- Entrada: Estímulo procedente de una fuente de energía externa que se le aplica a un sistema de control.
- ref(t): Señal de referencia, como se ha dicho anteriormente es el valor que se quiere conseguir.
- Unidad de realimentación: Mide el valor actual de la variable cuyo valor se quiere mantener dentro de unos límites aceptables y lo traslada al comparador [5].
- Comparador: También llamado detector de error. En él se calcula la diferencia entre la señal de referencia y la señal medida, siendo esta diferencia la señal de error e(t).

- Regulador: Condiciona la acción del actuador dependiendo de la señal de error obtenida. La señal de control creada por el regulador se denomina acción de control [11].
- Actuador: Recibe la orden del regulador y actúa sobre los elementos del sistema llevando a cabo la acción de control.
- Salida: Respuesta del sistema de control.

3.1.3 Clasificación de las acciones de control

La manera en que el regulador genera la señal de control es la acción de control. Las acciones de control básicas son las siguientes:

-Control proporcional o tipo P: La señal de error obtenida en el comparador se amplifica y se transmite al actuador [8].

-Control integral o tipo I: En el caso de que la señal de error sea muy pequeña el actuador no actúa porque la señal de control no supera el umbral mínimo necesario; si el actuador deja de funcionar la señal de salida nunca tomará el valor de la de referencia. Si se quiere solucionar este problema se debe realizar una integración de la señal de error para conseguir que la señal de control alcance el umbral necesario de actuación y así el actuador cumpla su función [8].

-Control derivativo o tipo D: Al calcular la derivada del error se obtiene por adelantado cómo va a variar la señal de error y con esta información se puede actuar de manera más rápida contra la perturbación [8].

Los tipos de control expuestos se pueden combinar entre sí para mejorar la respuesta del sistema de control. Esto es lo que se va a describir en los apartados 3.2, 3.3, 3.4 y 3.5. El valor correcto de los ajustes de los diferentes tipos de control depende de las características del proceso.

3.1.4 Respuesta estática y dinámica de un sistema de control automático

Descrito el punto anterior se procede a explicar la respuesta estática y dinámica de un sistema de control.

Cuando se varía la señal de referencia o el sistema de control sufre una perturbación se producen una serie de fenómenos transitorios antes de que las variables del sistema se estabilicen (respuesta permanente del sistema). Este valor estático que toma la salida del sistema se obtiene pasado un periodo de tiempo llamado tiempo de respuesta [5].

Haciendo referencia al ejemplo del sistema de aire acondicionado al cambiar el valor de la temperatura de referencia la respuesta estática sería el valor que toma la temperatura de la habitación al cabo de un tiempo determinado, mientras que la respuesta dinámica transcurre en el tiempo que tarda en llegar al valor estacionario.

Observando la respuesta transitoria se percibe la estabilidad y la rapidez que pueda tener un sistema, con la repuesta permanente la precisión del sistema.

3.2 REGULADORES P

La respuesta proporcional es la base de los reguladores, es decir si la acción derivativa o integral están presentes, también lo va a estar la acción proporcional [6].

La acción de control es proporcional a la señal de error del sistema. Dicho de otra forma la señal de error es amplificada para luego aplicarla al actuador correspondiente.

Llamando u(t) a la acción de control (señal de salida del regulador) y e(t) a la señal de error (señal de entrada al regulador), con un control proporcional se obtiene:

$u(t) = kp.e(t)$, donde kp es la constante o ganancia proporcional.

En este tipo de reguladores matemáticamente si el error es cero la salida del controlador también debería ser cero; pero en la práctica no se cumple [5].

Se procede a ejemplificar la respuesta real observando un caso en el que la señal de error es un escalón unitario. *Ver figura 6.*

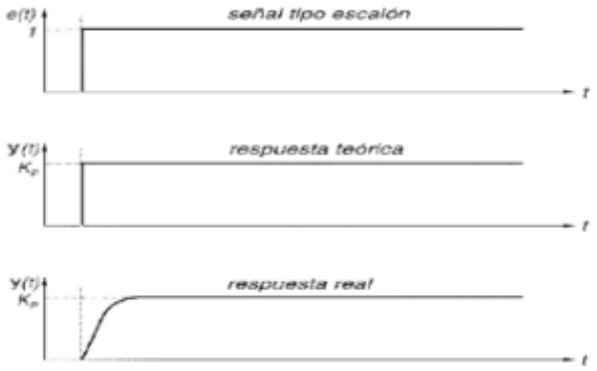

Figura 6: Respuesta de regulador P ante una entrada de tipo escalón unitario

Una consecuencia del regulador P es el offset. Cuando la variable regulada se acerca al valor de consigna la acción proporcional se hace más débil y no vence la tendencia de la variable, alcanzando un reposo antes de lo previsto y por lo tanto manteniendo un error permanente. El regulador P no puede compensar esta señal de error permanente del sistema.

3.3 REGULADORES PD

Como se expuso anteriormente, la acción derivativa anticipa cómo va a variar la señal de error. Por tanto esta acción contrarresta la inercia del proceso, frenándolo cuando evoluciona demasiado rápido y acelerándolo en caso contrario [6].

Cuando se tiene un regulador proporcional y derivativo, la acción de control responde a la siguiente ecuación:

$$y(t) = Kp.e(t) + kd.\frac{de(t)}{dt} = kp.e(t) + kp.Td.\frac{de(t)}{dt}$$

El parámetro Td, tiempo derivativo o de adelanto, controla la acción derivativa del sistema; mientras que kp controla ambas acciones.

Primero decir que el control derivativo únicamente es útil en los casos en los que la señal de error varía en el tiempo de forma continua; es decir, si el error es constante su derivada se anula y por tanto la acción derivativa también lo hace.

En la *figura 7* se expone un ejemplo para su mejor comprensión:

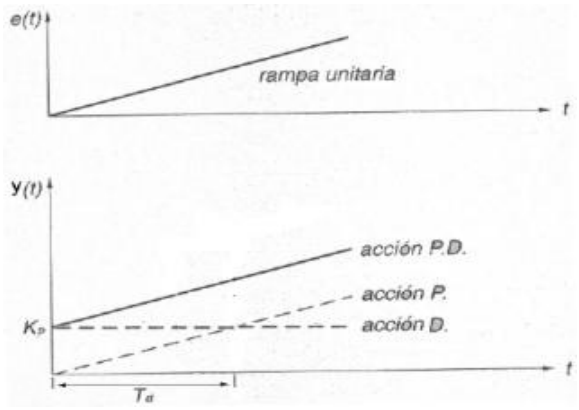

Figura 7: Respuesta de regulador PD al recibir a su entrada una señal de error rampa unitaria

Como se observa en la figura la respuesta del controlador se anticipa a la señal de error, demostrando todo lo expuesto anteriormente.

Al incorporar a un regulador proporcional la acción derivativa mejora de manera considerable la velocidad de respuesta del sistema. El regulador PD se utiliza en sistemas que deben actuar rápidamente, ya que la utilidad de este tipo de controlador reside en aumentar la velocidad de respuesta de un sistema de control.

El control PD no es ampliamente utilizado debido a que no puede compensar las desviaciones permanentes del sistema y por otro lado si hay un exceso de control derivativo puede ser causa de inestabilidad del sistema.

3.4 REGULADORES PI

La acción de control de un regulador PI es la siguiente:

$$y(t) = kp.e(t) + ki.\int_0^t e(t)dt = kp.e(t) + \frac{kp}{Ti}\int_0^t e(t)dt$$

A Ti se le denomina tiempo integral y controla la acción integral (o de reset) del sistema, mientras que kp controla tanto la acción proporcional como la integral. Al ser Ti grande la pendiente de la rampa correspondiente al efecto integral es pequeña y en consecuencia el

efecto de esta acción débil, y viceversa. Cuanta más acción integral exista en el regulador, más rápido cambia la salida en función del tiempo.

En la *figura 8* se expone un ejemplo de respuesta de regulador PI ante una señal de error escalón unitario:

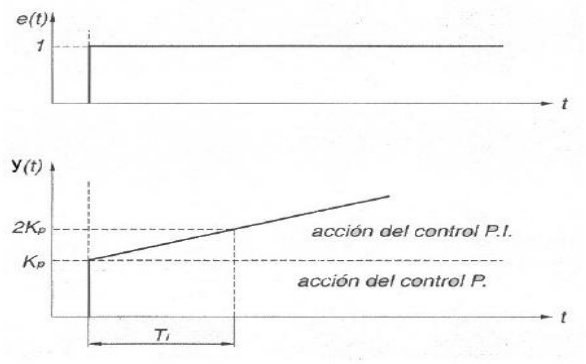

Figura 8: Respuesta de regulador PI ante un escalón unitario

Observando la figura se percibe que la respuesta del controlador PI es la suma de las respuestas de un controlador proporcional y un controlador integral. Esto implica una respuesta casi instantánea (en la realidad no es tan inmediata como aparece en la figura) provocada por la acción proporcional y un posterior control integral que hará que la variable regulada coincida con la señal de referencia. El momento en el cual coincide la variable controlada con la señal de referencia se elimina el posible error remanente del sistema. La función de la acción integral es eliminar el offset [11].

Los reguladores PI son bastante utilizados debido a la sensibilidad al ruido que presenta la acción derivativa.

3.5 REGULADORES PID

La salida de un regulador PID viene dada por la siguiente expresión:

$$y(t) = kp.e(t) + ki.\int_0^t e(t)dt + kd.\frac{de(t)}{dt};$$

También se puede expresar de la siguiente manera:

$$y(t) = kp.e(t) + \frac{kp}{Ti}.\int_0^t e(t)dt + kp.Td.\frac{de(t)}{dt}$$

Los valores de las constantes proporcional, integral y derivativa definen el comportamiento del regulador. El objetivo de los ajustes del PID es conseguir que se corrija de manera eficaz y en el mínimo tiempo los efectos de las perturbaciones.

El PID es el algoritmo de control más ampliamente extendido ya que proporciona buenos resultados en la inmensa mayoría de los casos.

La acción proporcional es la que hace que el regulador PID actúe enérgicamente cuando el error es grande, con esto parece suficiente, pero no es así debido a dos motivos: El primero se debe a la aparición del offset, ya que cuando la variable regulada se acerca al valor de consigna la acción proporcional se debilita y alcanza el reposo antes de lo previsto; el segundo es que si se aumenta en gran medida la acción proporcional para disminuir el error la variable controlada se acercará al valor de consigna demasiado deprisa, provocando la inestabilidad del sistema [4].

Por la primera razón expuesta conviene añadir una acción integral que mantenga una respuesta cuando el error se anula, gracias al error que existió en el pasado. Por la segunda razón se percibe la necesidad de añadir la acción derivativa frena el proceso cuando evoluciona demasiado rápido y lo acelera en caso contrario.

El problema reside en dar valores a las constantes que representan las intensidades con las que actúan las tres acciones. La solución no es inmediata ya que depende de cómo responde el proceso a los esfuerzos que desarrolla el regulador para corregir el error [4].

En los siguientes capítulos la variable asociada a la respuesta del controlador se expresa como u(t) en lugar de y(t).

Capítulo 4
Control de un depósito

4.1 ENUNCIADO DEL PROBLEMA

El sistema a estudiar consiste en un depósito de área constante *A*, el cual almacena líquido, siendo *h(t)* la altura del líquido en el depósito. La entrada de líquido tiene un caudal *qe(t)* y un caudal de salida *qs(t). Ver figura 9.*

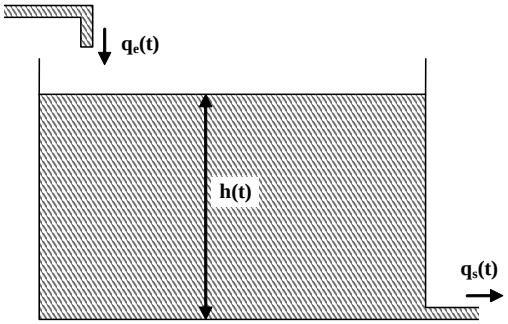

Figura 9: Esquema sistema depósito

Se desea controlar el nivel del líquido en el depósito, manteniéndolo a la altura indicada por la señal de referencia *href*. Para ello se utilizan varios tipos de reguladores: proporcional (P), proporcional derivativo (PD), proporcional integral (PI) y proporcional integral derivativo (PID).

El diagrama de bloques correspondiente al regulador PID que se muestra en la *figura 10* contempla las tres acciones básicas de control. Para el caso del regulador proporcional se anula la acción integral y la acción derivativa, en el regulador PD se anula la acción integral y en el regulador PI se anula la acción derivativa.

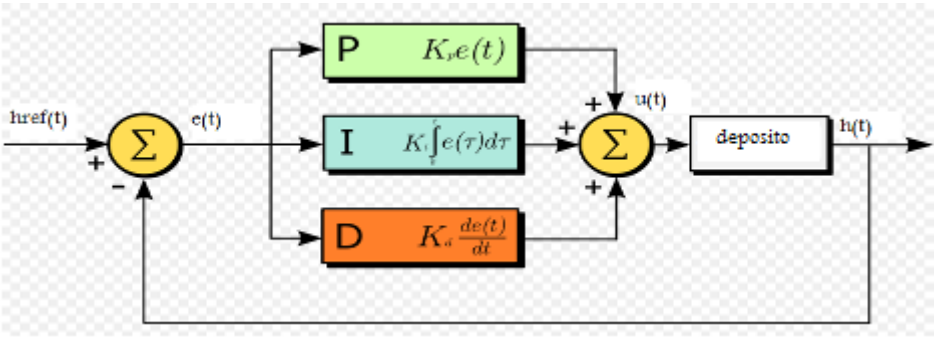

Figura 10: Diagrama de bloques de regulador PID sistema depósito

En primer lugar se estudia el sistema real y posteriormente se trabaja con el modelo linealizado en torno al punto de equilibrio elegido.

Las ecuaciones que modelan el sistema junto con la ecuación necesaria para poder controlarlo son las que se muestran a continuación:

$$q_e(t) - q_s(t) = A \cdot \frac{dh(t)}{dt} \qquad (1)$$

$$q_s(t) = a \cdot \sqrt{2 \cdot g \cdot h(t)} \qquad (2)$$

$$q_e(t) = k \cdot u(t) \qquad (3)$$

La unión de (1), (2) y (3) da lugar a la ecuación siguiente:

$$\frac{dh(t)}{dt} = -\frac{a}{A} \cdot \sqrt{2 \cdot g \cdot h(t)} + \frac{k}{A} \cdot u(t)$$

Cuando se trabaja con el modelo linealizado el sistema permanece estable, y por tanto la altura *h(t)* permanece constante, por lo que su derivada se hace cero; con lo cual tenemos:

$$0 = -\frac{a}{A} \cdot \sqrt{2 \cdot g \cdot h(t)} + \frac{k}{A} \cdot u(t)$$

$$u_0 = \frac{a \cdot \sqrt{2 \cdot g \cdot h_0}}{k}$$

La ecuación con la que obtenemos el error, *e(t)*, entra la señal de salida y la señal de referencia:

$$e(t) = h_{ref}(t) - h(t)$$

Por último la acción de control que actúa sobre el depósito depende del tipo de regulador con el que se controla el sistema:

- Para el regulador proporcional:

  $$u(t) = kp.e(t)$$

- Para el regulador proporcional derivativo:

  $$u(t) = kp.e(t) + kd.\frac{de(t)}{dt}$$

- Para el regulador proporcional integral:

$$u(t) = kp.e(t) + ki.\int_0^t e(t)dt$$

- Para el regulador proporcional integral derivativo:

$$u(t) = kp.e(t) + ki.\int_0^t e(t)dt + kd.\frac{de(t)}{dt}$$

4.2 RESOLUCIÓN DEL SISTEMA: IMPLEMENTACIÓN CON EJS

Ahora se procede a solucionar el problema a través de la herramienta software *EJS*. Para conseguir la simulación hay que contemplar los distintos paneles de trabajo: *descripción, modelo* y *vista*. Se comienza con el panel de trabajo *modelo* (a su vez el panel *modelo* se divide en *variables, inicialización, evolución, relaciones fijas* y *propio*), seguidamente con el panel *vista* y por último con el panel de trabajo *descripción.*

Se simulara el modelo del sistema no linealizado y el del modelo linealizado.

4.2.1 Modelo real

4.2.1.1 Variables

Se declaran las variables que comparten los dos modelos:

- g: Gravedad.
  La gravedad toma el valor 9.8 m/s^2.
- R: Radio de la circunferencia que define el depósito.
  El radio *R* vale 2.95x10^{-2} m.
- r: Radio de la circunferencia que define las tuberías
  El radio *r* vale 1.5x10^{-3} m.
- A: Área del depósito.
  El área del depósito es: A= $\pi \cdot R^2$ (m^2)**.**

- a: Área de las tuberías.

  El área de las tuberías es: a= $\pi \cdot r^2$ (m^2).
- k: Constante.

  La constante *k* tiene un valor de $2.7x10^{-6}$.
- href: Altura del líquido buscada en el depósito.

  La altura que inicialmente se quiere conseguir es 0.5 m.
- t: Tiempo.

  La variable *t* se inicia con el valor cero y con el cambio de este valor iremos viendo cómo evoluciona el sistema en el tiempo.
- dt: Diferencial de tiempo.

  El diferencial de tiempo va a tener el valor 0.1 s. Para valores inferiores el proceso de estabilización del sistema es más lento.
- kp: Constante de proporcionalidad del regulador.

  La variable *kp* se inicia con el valor 100.
- kd: Constante derivativa del regulador.

  La variable *kd* comienza tomando el valor numérico cero.
- ki: Constante de integración del regulador.

  La variable *ki* comienza teniendo el valor cero.

Y añadimos las del modelo no lineal:

- h: Altura del líquido en el depósito.

  A la variable *h* se le da un valor distinto a *href* para observar cómo actúa el regulador ante esta diferencia (la altura tiene que variar para conseguir igualar a la altura de referencia).

  Se inicializa con valor nulo.
- error: Error cometido por el controlador.

  Este valor comienza tomando el valor 0 m.
- derror: La derivada del error.

  Esta variable auxiliar se emplea para calcular la derivada del error; asignándole su valor. Necesaria para la acción derivativa del regulador.

  Su valor inicial es cero.

- int_error: La integral del error.

  Esta variable auxiliar se emplea para calcular la integral del error, necesaria para la acción integral del regulador. Al no admitir *EJS* integrales, se procede a calcular la integral asignando a esta variable la suma acumulativa del error.

  Su valor al inicio es cero.

- u: Acción de control.

  Esta variable se inicializa al valor cero.

Para facilitar la comprensión y el seguimiento del código se van a clasificar las variables en tres grupos; el primer grupo de variables se halla situado dentro de la pestaña *variables_comunes* (*figura 11*) y son por tanto las variables comunes para los dos modelos (linealizado y no linealizado). En el segundo grupo se encuentran las variables específicas del modelo no lineal, en la pestaña *regulador_nolineal* (*figura 12*).

Luego se añade una tercera pestaña (apartado 4.2.2.1), *regulador_lineal*, para el modelo lineal *(ver figuras 11 y 12).*

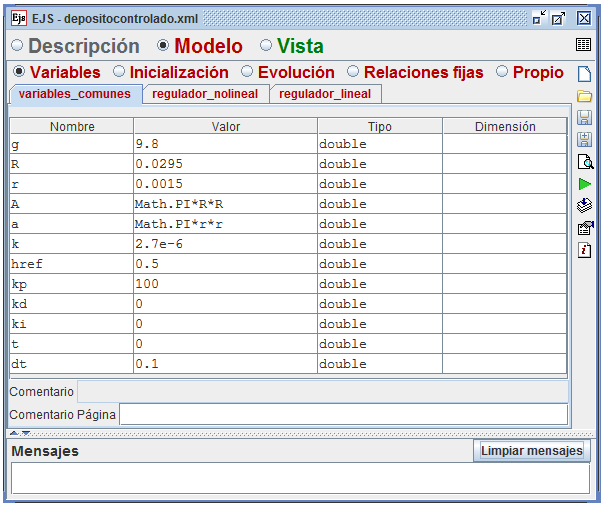

Figura 11: Variables comunes a los modelos real y lineal del depósito

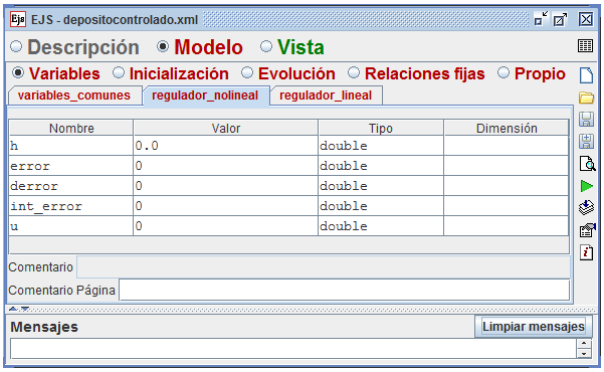

Figura 12: Variables del modelo no lineal del depósito

4.2.1.2 Ecuaciones

Presentadas las variables, se escriben las ecuaciones que van a definir nuestro sistema. Las ecuaciones, transcritas a lenguaje *Java*, las vamos a escribir en el *panel modelo* dentro de la opciones *evolución (figura 13) y relaciones fijas* y *propio (figura 14).*

En la opción *evolución* se encuentran las ecuaciones que se vean afectadas por el paso del tiempo, como la fórmula de la derivada del error y la ecuación que modela el sistema: $\frac{dh(t)}{dt} = -\frac{a}{A} \cdot \sqrt{2 \cdot g \cdot h(t)} + \frac{k}{A} \cdot u(t)$ también aparece aquí al ser una ecuación diferencial; pero al encontrarse *h(t)* en el segundo término de la ecuación se crea una función auxiliar, f1, que calcula además el error, el valor de la variable *int_error* y el parámetro de control. Esta función auxiliar se encuentra en la opción *propio.*

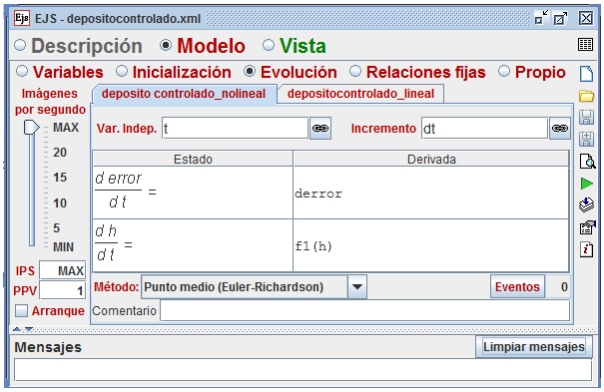

Figura 13: Subpanel *evolución* del modelo real del depósito

Al parámetro *imágenes por segundo (IPS)* se le da el valor máximo para que la simulación sea lo más rápida posible ya que en algunas ocasiones el sistema tarda mucho en estabilizarse.

Si se quisiera aproximar al tiempo real lo que tarda en ejecutarse la simulación lo que se tiene que poner es un valor de *IPS* de 10 junto con un diferencial de tiempo de 0.1 segundos.

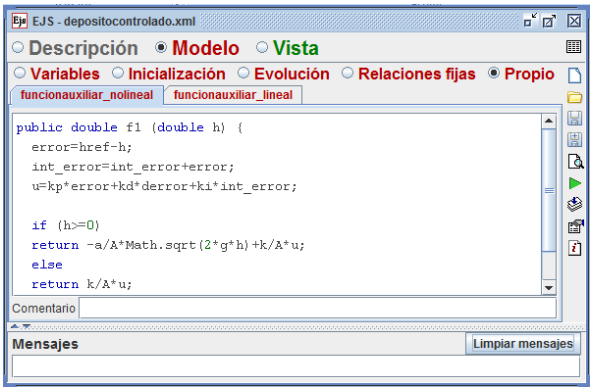

Figura 14: Subpanel *propio* del modelo real del depósito

En la opción *relaciones fijas* aparece el código necesario para que se eliminen las trazas de las gráficas cuando *t* toma el valor 1000 y comiencen de nuevo su aparición desde el origen de coordenadas. Aparte se añaden unas sentencias en las que se establece que si la altura del líquido es menor que cero la altura sea cero, y que si la altura es mayor que uno su valor se iguale a uno; es decir que el controlador solo funcione cuando el caudal de agua se encuentre dentro de las dimensiones del depósito *(ver figura 15).*

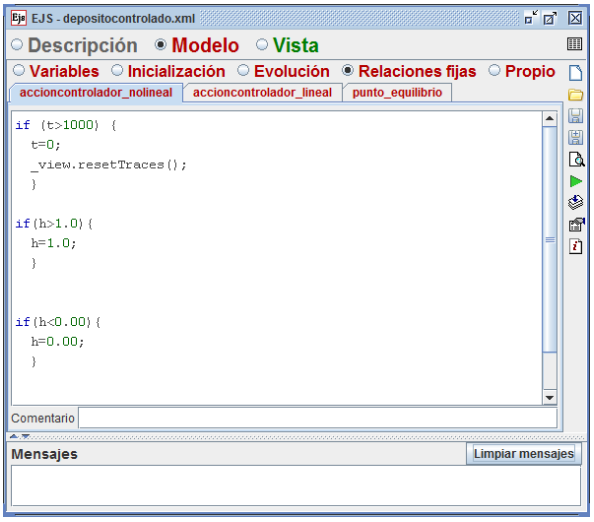

Figura 15: Subpanel *relaciones fijas* del modelo real del depósito

4.2.1.3 Vista

El siguiente paso es centrarse en el panel *vista*. En este apartado se describen todos los elementos gráficos pertenecientes al modelo no linealizado y los elementos de los dos modelos en común.

En primer lugar se coloca la ventana principal donde se incluyen todos los elementos necesarios para la vista. Esta ventana recibe de título *Depósito y* sigue una política de distribución *rejilla* (una fila y tres columnas). Esta distribución se presenta a través de los tres contenedores en los que se desglosa la ventana principal: *no lineal*, *comun* y *lineal (ver figura 16).*

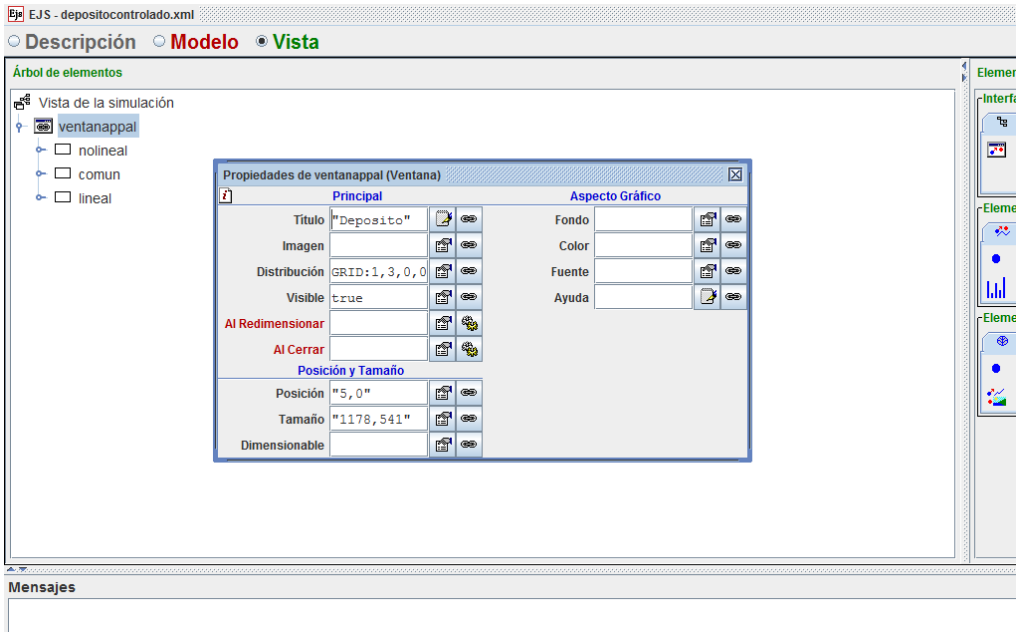

Figura 16: Menú *propiedades* de ventana principal sistema depósito

El contenedor *no lineal* tiene una distribución de *márgenes (*o *border*). En la posición de arriba se localiza el contenedor *panel*, en el centro se encuentra el *panel dibujo* y abajo el contenedor *panel4 (ver figura 17).*

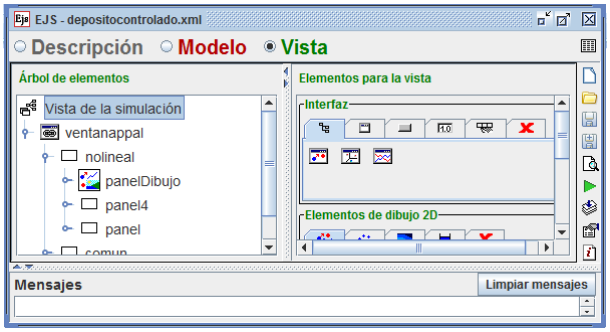

Figura 17: Elementos de contenedor *nolineal*

En el contenedor *panelDibujo* se asignan las variables que se han presentado anteriormente a sus correspondientes elementos gráficos dentro de un entorno que simula el sistema real. También se encuentran un conjunto de textos interactivos que facilitan la visualización del nivel de líquido *(ver figura 18).*

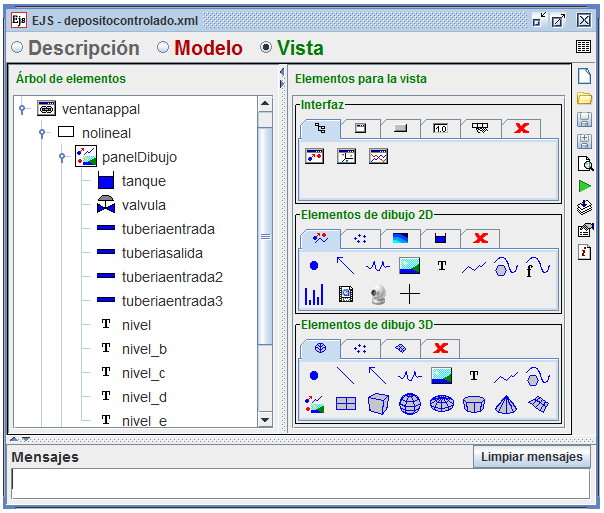

Figura 18: Elementos de contenedor *panelDibujo*

Como elemento se escoge la tubería de salida, *tuberiasalida*, cuyo caudal se presenta en el caso de que *h* sea mayor que cero. Dentro del menú *propiedades* se puede definir el color del líquido, la posición, la orientación que tiene esta tubería dentro de la ventana principal, etc. *(ver figura 19).*

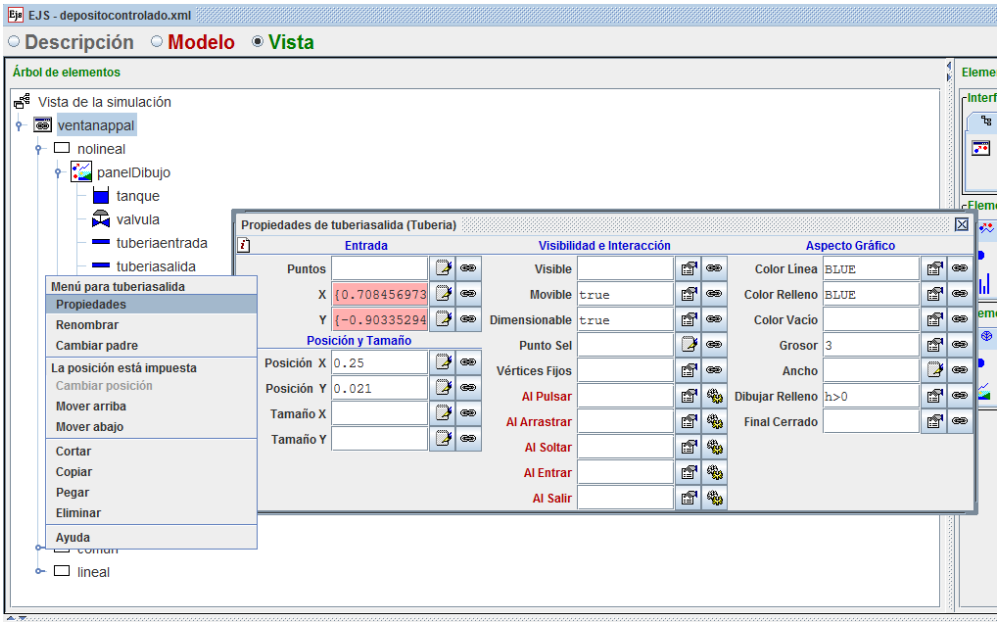

Figura 19: Propiedades de elemento *tuberiasalida*

Como ejemplo de texto interactivo se escoge *nivel_c*, que informa de cuando la altura *h* alcanza un nivel de 0.6 m *(ver figura 20).*

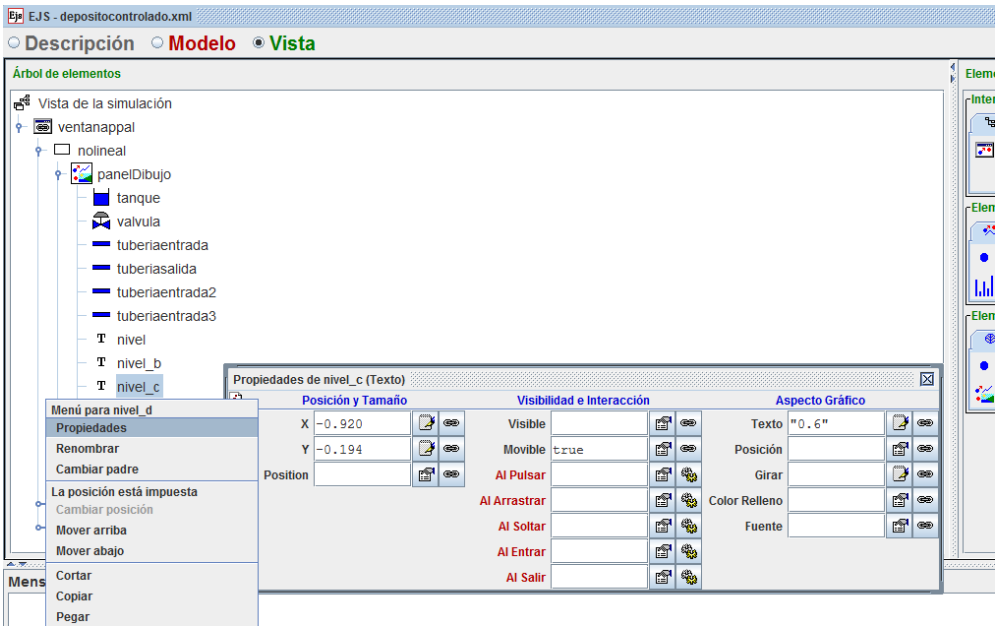

Figura 20: Propiedades de texto interactivo *nivel_c*

En el contenedor *panel4* se encuentra una etiqueta que informa de que el modelo presente es el no lineal. Dentro del menú *propiedades* se puede elegir el estilo, el tamaño, el color de la fuente, etc. *(ver figura 21).*

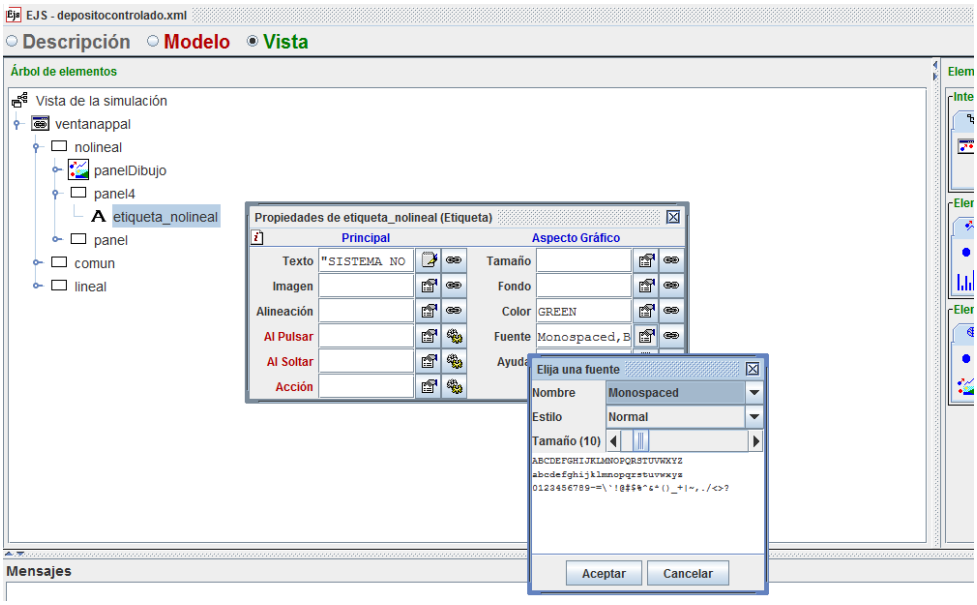

Figura 21: Propiedades de *etiqueta_nolineal*

Dentro del contenedor *panel* se encuentra el campo *h* para mostrar y modificar el valor numérico de la altura del líquido. El valor inicial de *h* es cero y el formato de su presentación es decimal *(ver figura 22).*

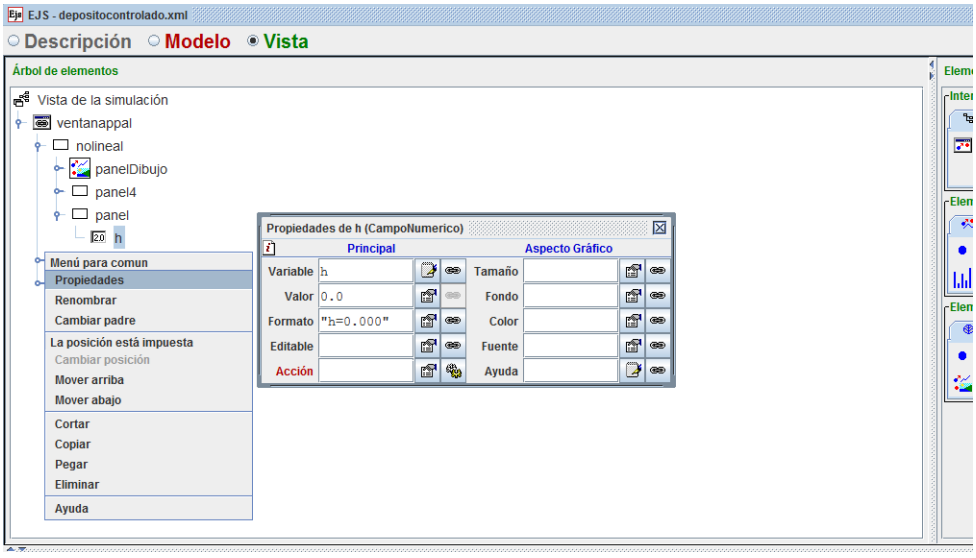

Figura 22: Propiedades de campo *h*

Ahora se pasa a estudiar el contenedor *comun*, sitio donde se encuentran los elementos que comparten el modelo real y el lineal. Dicho contenedor presenta una distribución *VBOX (caja vertical).* En el árbol de elementos, dentro del contenedor *comun*, están los contenedores *panel2*, *panel7*, *panel5* y *PANEL_HREF*; y el *panelConEjes*.

El *panelConEjes* es un contenedor de dibujo 2D con un sistema de ejes cartesianos.

El título que ponemos a este panel es “altura VS tiempo”, el eje x representa el tiempo y el eje y, dependiendo de si se quiere ver el gráfico del sistema linealizado o no linealizado, representa *hl* o *h (ver figura 23).*

Lo que muestra la evolución de la altura (*h* ó *hl*) a lo largo del tiempo (*t*), es la traza (no lineal: *traza* o lineal: *traza_lineal*) que se incluye dentro de *panelConEjes (ver figura 24).*

También se incluye una traza que representa la altura de referencia, que permite ver cómo está funcionando el regulador *(ver figura 24).*

El eje x alberga valores de tiempo comprendidos entre cero y mil segundos, mientras que los valores que se representan en el eje y van de cero a un metro.

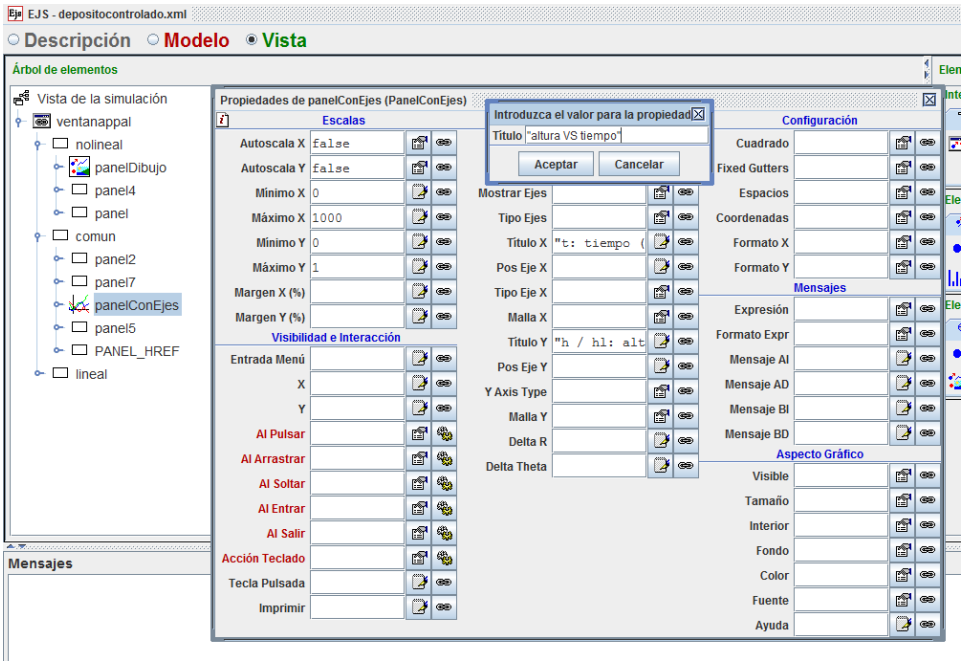

Figura 23: Propiedades de contenedor *panelConEjes*

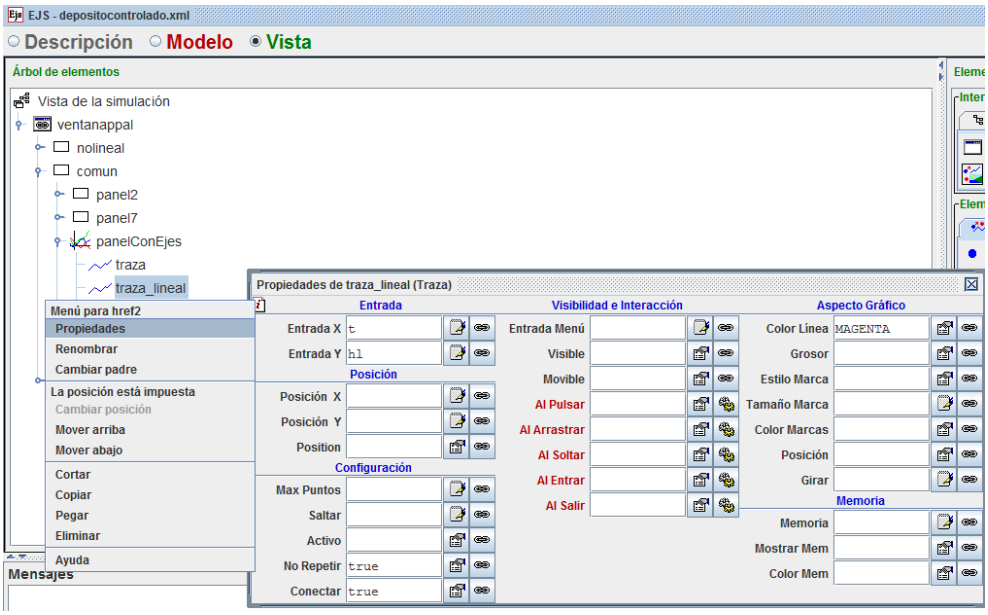

Figura 24: Propiedades de *traza_lineal*

El contenedor *panel2* sigue una política de distribución *rejilla* con tres filas y una columna; en él se encuentran una serie de deslizadores cuyo cometido es variar el valor de *kp*, *kd* y *ki* y con ello estudiar el comportamiento de los distintos reguladores *(ver figura 25).*

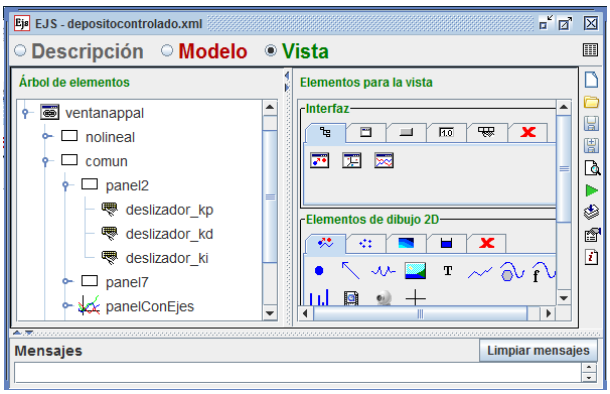

Figura 25: Elementos de *panel2*

El contenedor *panel7* sigue una distribución de *rejilla* con tres filas y una columna; en él se han situado tres campos para mostrar el valor numérico de *kp*, *kd* y *ki* respectivamente *(ver figura 26).*

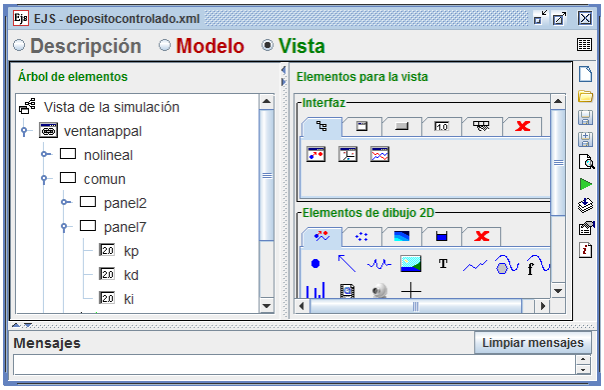

Figura 26: Elementos de *panel7*

En el *panel5* (con una distribución *rejilla* de una fila y tres columnas) se obtiene un panel con tres botones para controlar la ejecución de la simulación *(ver figura 27).*

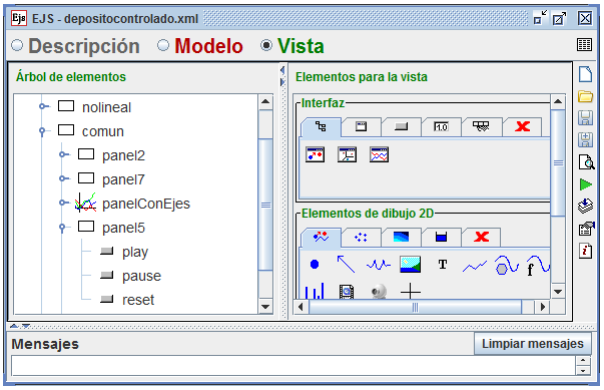

Figura 27: Elementos de *panel5*

Al pulsar el botón *play* se pone en marcha la evolución de la simulación, siempre y cuando la evolución esté parada *(ver figura 28).*

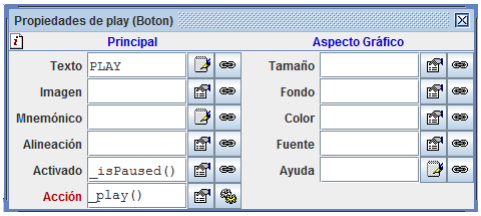

Figura 28: Menú *propiedades* de botón *play*

Cuando se presiona el botón *pause* se detiene la evolución de la simulación cuando la evolución está en marcha *(ver figura 29).*

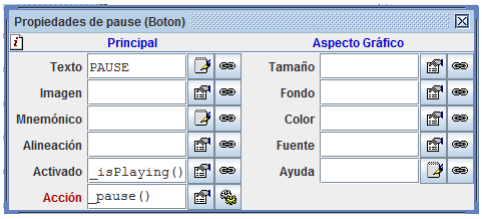

Figura 29: Menú *propiedades* de botón *pause*

Al pulsar *reset* se inicializan las variables al valor en las tablas de variables (apartado *variables* dentro de panel *modelo*) y se limpia la vista *(ver figura 30).*

Figura 30: Menú *propiedades* de botón *reset*

En el contenedor *PANEL_HREF*, con una distribución *rejilla* de dos filas y una columna, se encuentra un deslizador que permite modificar el valor de *href* y un campo para mostrar su valor numérico *(ver figura 31).*

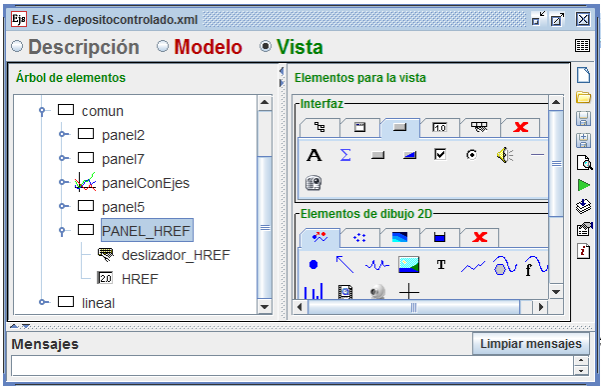

Figura 31: Elementos de *PANEL_HREF*

Con el deslizador al variar la altura de referencia el regulador actúa para alcanzar una altura aproximada a la buscada (la aproximación depende del tipo de regulador y de las constantes asociadas).

Este deslizador está asociado a la variable *href*, inicializada al valor 0.5 m, su formato

de presentación es el decimal, el valor que puede tomar *href* se puede variar de cero a uno con una precisión de centésimas, la orientación del deslizador es horizontal, el color de la fuente con el que se escribe *href* es roja, etc. *(ver figura 32).*

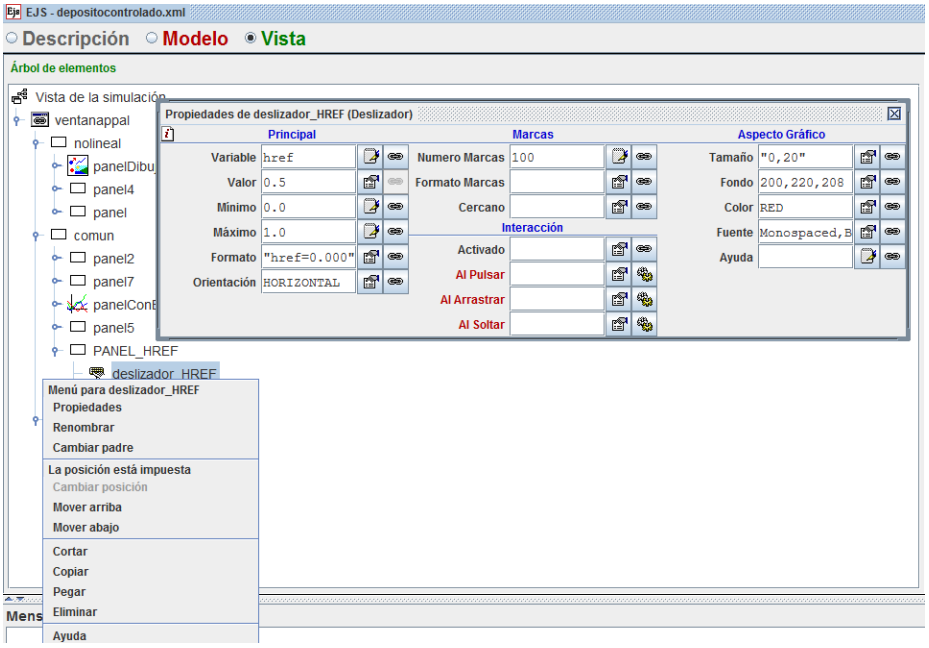

Figura 32: Propiedades de *deslizador_HREF*

4.2.2 Modelo lineal

4.2.2.1 Variables

Se declaran las variables del modelo lineal

- hl: Altura que alcanza el líquido cuando el sistema está linealizado.

  A la variable hl le damos un valor distinto a href y así observar el comportamiento del regulador ante esta diferencia (la altura tiene que variar para conseguir igualar a la altura de referencia).

  Se inicializa con el valor 0 m.
- errorl: Error cometido por el controlador cuando el sistema ha sido linealizado.

  Esta variable se inicializa a un valor de 0 m.
- derrorl: Es la derivada del error en el sistema linealizado.

  Esta variable auxiliar se empleará para calcular la derivada del error, necesaria para la acción derivativa del regulador.

Esta variable comienza tomando el valor cero.

- int_errorl: Es la integral del error en el modelo lineal.
  Esta variable auxiliar se empleará para calcular la integral del error, necesaria para la acción integral del regulador.
  Esta variable se inicializa con un valor cero.
- ul: Acción de control en el sistema linealizado.
  Comienza valiendo cero.
  Las variables que definen el punto de equilibrio son h0 y u0;
- h0: Altura del líquido en equilibrio.
  Al inicio vale 0.3 m.
- u0: Acción de control en equilibrio.
  Comienza tomando el valor cero.

Estas variables se incluyen dentro de la pestaña *regulador_lineal* en el apartado *variables* del panel *modelo (ver figura 33).*

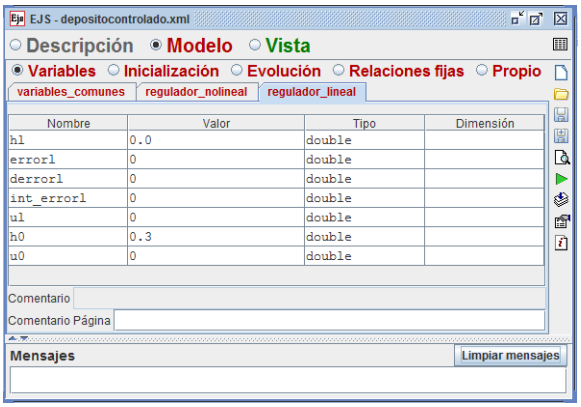

Figura 33: Pestaña *regulador_lineal* del subpanel *variables*

4.2.2.2 Ecuaciones

Definidas las variables, se escriben las ecuaciones que van a definir nuestro sistema. Las ecuaciones, transcritas a lenguaje *Java*, las vamos a escribir en el *panel modelo* dentro de la opciones evolución, relaciones fijas y propio.

En la opción evolución vamos a encontrar las ecuaciones que se vean afectadas por el paso del tiempo; se tiene la fórmula de la derivada del error para el modelo lineal y la ecuación

que modela el sistema lineal, $\frac{dhl(t)}{dt} = -\frac{a}{A} \cdot \frac{g}{\sqrt{2 \cdot g \cdot h0}} \cdot (hl(t) - h0) + \frac{k}{A} \cdot (ul(t) - u0)$, también aparecerá aquí al ser una ecuación diferencial; pero al encontrarse *hl (t)* en el segundo término de la ecuación, se creará una función auxiliar, f2, que calculará además el error en el modelo lineal, el valor de la variable *int_errorl* y el parámetro de control para el sistema linealizado. Esta función auxiliar se encuentra en la opción *propio (ver figuras 34 y 35).*

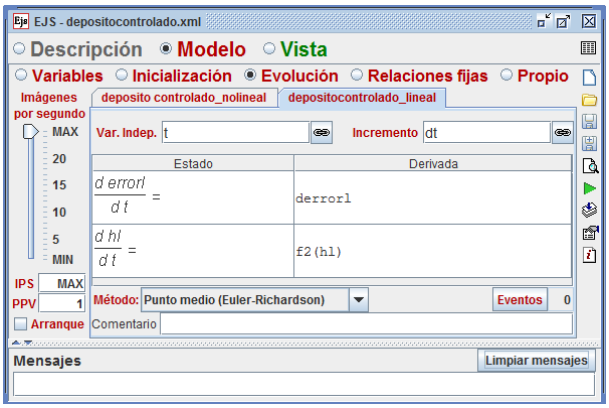

Figura 34: Subpanel *evolución* del modelo lineal del depósito

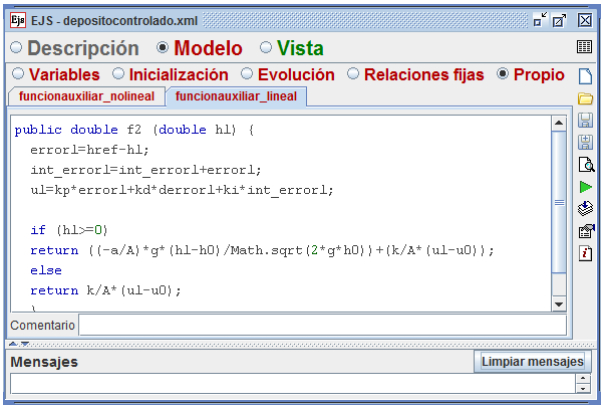

Figura 35: Subpanel *propio* del modelo lineal del depósito

Se añaden unas sentencias en relaciones fijas en las que se establece que si la altura del líquido en el depósito del sistema linealizado es menor que cero la altura *hl* sea cero, y que si la altura es mayor que uno su valor se iguale a uno; es decir que el controlador solo funcione cuando el caudal de agua se encuentre dentro de las dimensiones del depósito *(ver figura 36).*

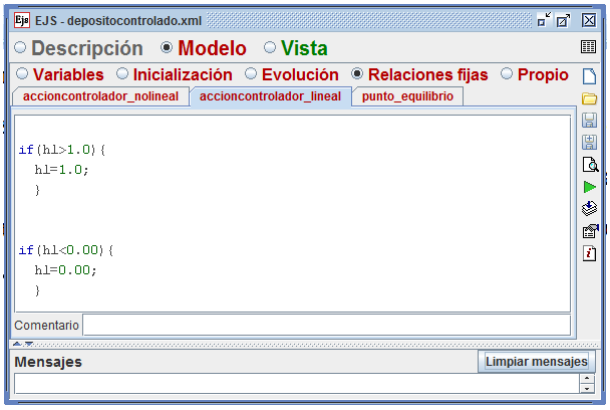

Figura 36: Subpanel *relaciones fijas* del modelo lineal del depósito

En relaciones fijas se incluye la ecuación del punto de equilibrio *(ver figura 37).*

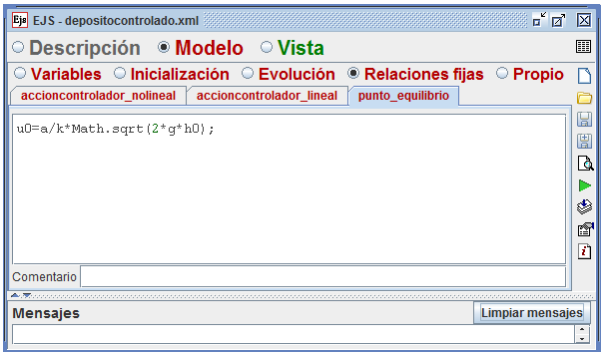

Figura 37: Ecuación del punto de equilibrio en subpanel *relaciones fijas*

4.2.2.3 Vista

Después de declarar las variables y las ecuaciones se pasa a crear la parte del interfaz de usuario del modelo linealizado en el *panel vista.*

Se pasan a describir los elementos gráficos situados dentro del contenedor lineal (ver ubicación en el árbol de elementos en la *figura 38*).

El contenedor *lineal* tiene una distribución de *márgenes (*o *border*).

En la posición de arriba se encuentra el contenedor *panel3*, en el centro se localiza el *panel Dibujo2* y en la parte inferior el contenedor *panel6 (ver figura 38).*

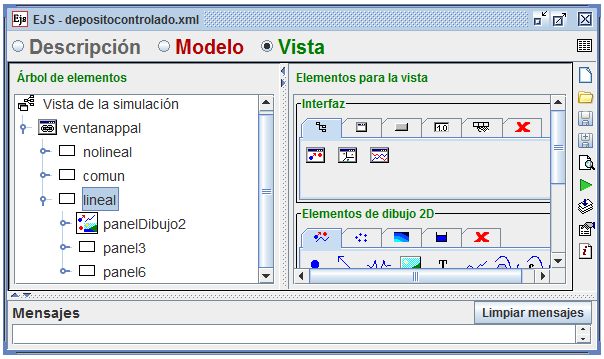

Figura 38: Composición de panel *lineal*

El *panelDibujo2* alberga una serie de elementos gráficos los cuales se van a ligar a las variables correspondientes para poder crear la simulación. En este panel de dibujo también se encuentran un conjunto de textos interactivos que facilitan la visualización del nivel de líquido *(ver figura 39).*

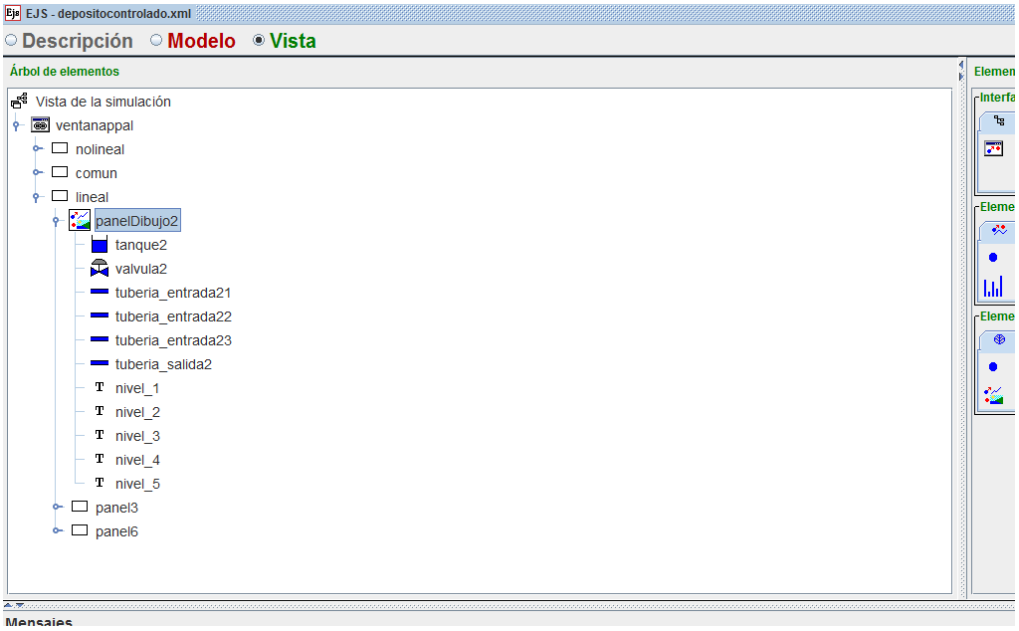

Figura 39: Composición de *panelDibujo2*

Como elemento gráfico se elige *valvula2* a través de la cual aparece caudal en el caso de existir una acción de control por el regulador. En el menú *propiedades* se puede definir el color del líquido, la posición, el tamaño de esta válvula dentro de la ventana principal, etc. *(ver figura 40).*

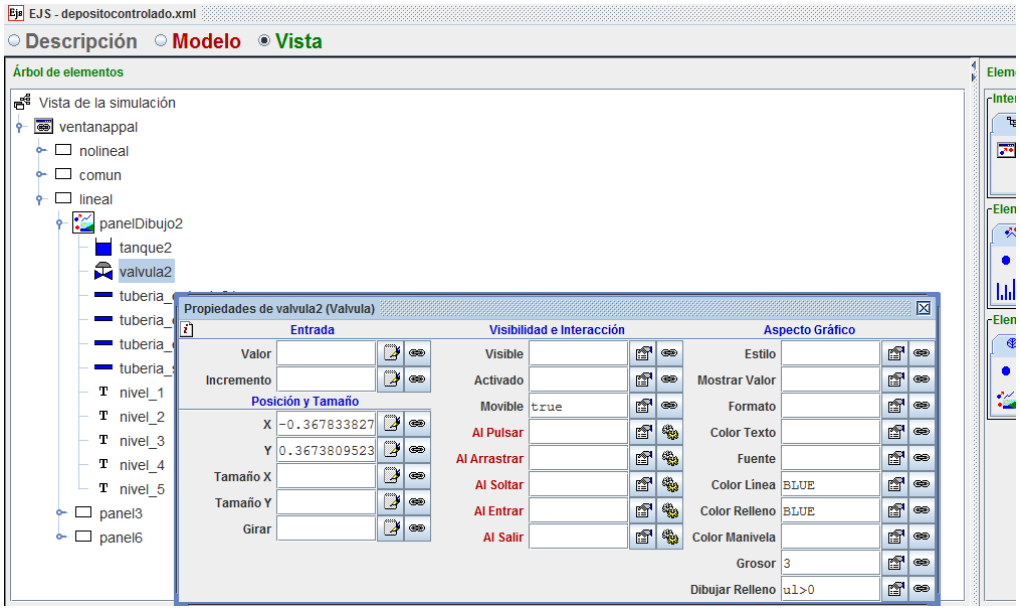

Figura 40: Menú *propiedades* de *valvula2*

Como texto interactivo se escoge *nivel_5*, que indica que la altura del líquido en el depósito linealizado alcanza su nivel máximo *(ver figura 41).*

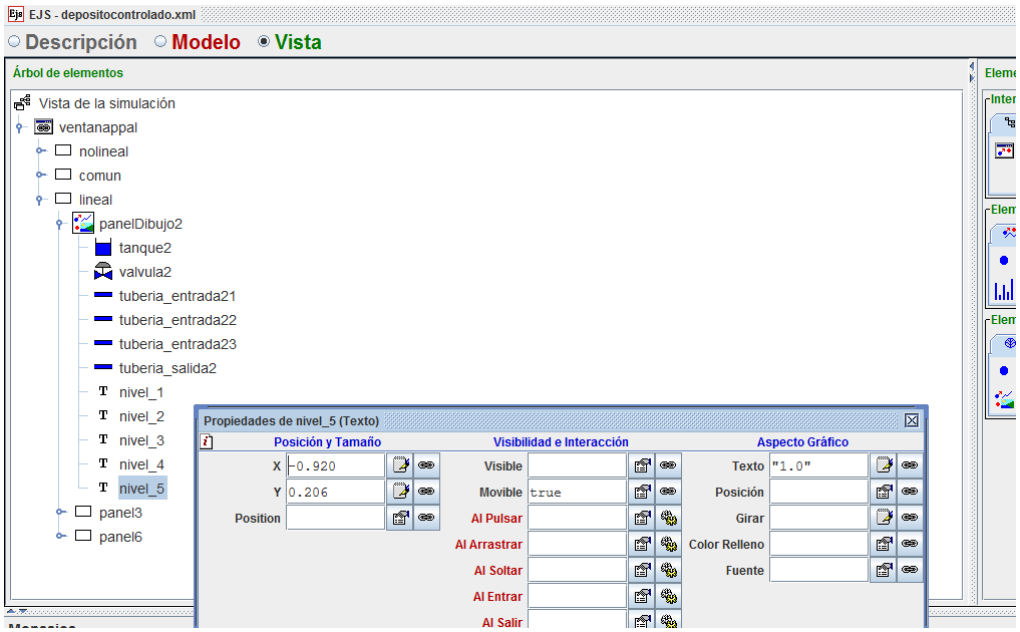

Figura 41: Menú *propiedades* de texto interactivo *nivel 5*

En el contenedor *panel6* hay una etiqueta que informa de que el modelo presente es el linealizado.

Dentro del menú *propiedades* se puede elegir el estilo, el tamaño, el color de la fuente, etc. *(ver figura 42).*

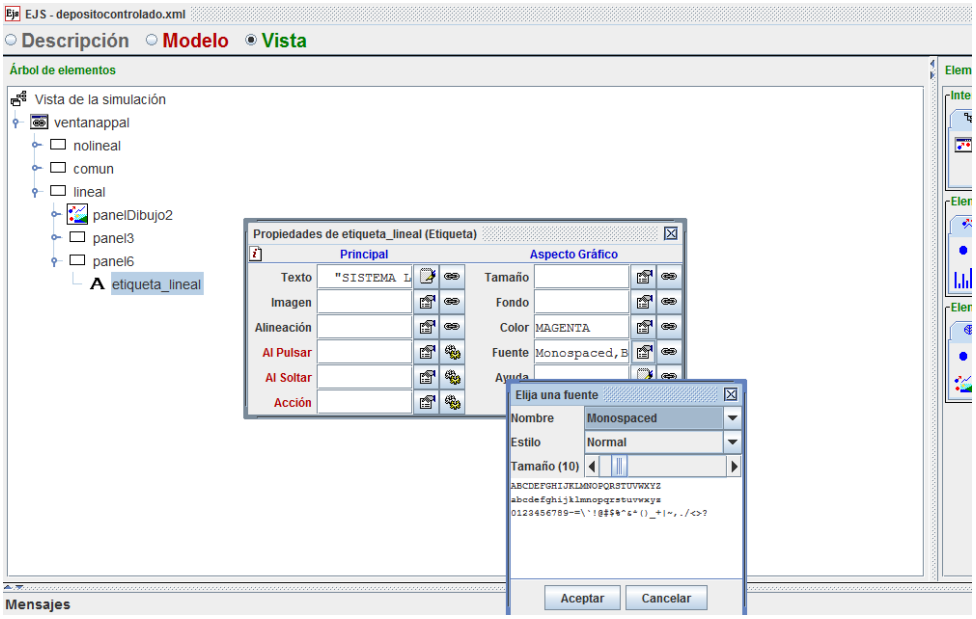

Figura 42: Menú *propiedades* de *etiqueta_lineal*

Dentro del contenedor *panel3* (el cual sigue una política de distribución *HBOX*) se encuentran los campos *hl*, *h0* y *u0;* desde los que se puede mostrar y modificar el valor numérico de la altura del líquido y del punto de equilibrio.

Una vez descritas todas las partes de la interfaz gráfica se pasa a mostrar su visualización en la *figura 43*.

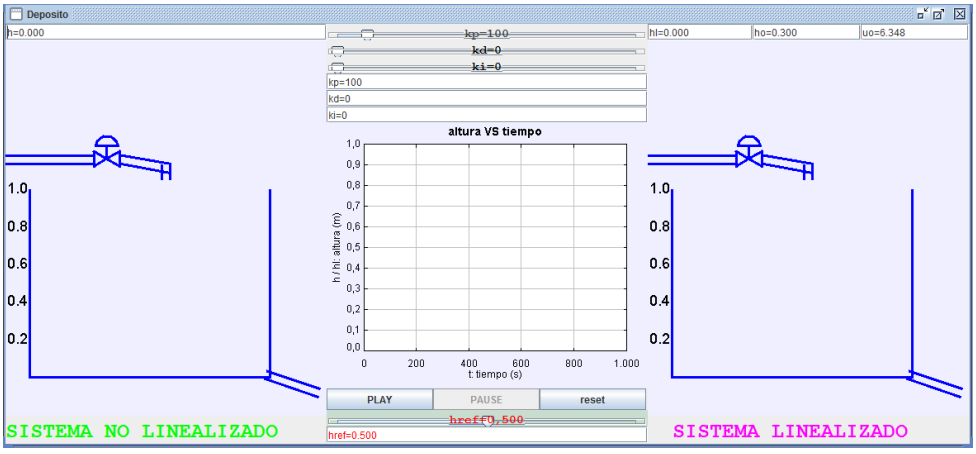

Figura 43: Vista de simulación sistema depósito

4.2.3 Descripción

Como se dijo en el *capítulo 2* el panel *descripción* tiene como finalidad crear y editar código HTML adjunto a la simulación. En este caso se hace una descripción sobre el sistema del depósito. Lo único que hay que tener en cuenta es que la página donde se realiza la descripción tiene que estar habilitada para editarla y que si se quiere insertar algún gráfico se debe extraer del directorio de *EJS*, *workspace*, del subdirectorio *source* (*ver figuras 44 y 45*).

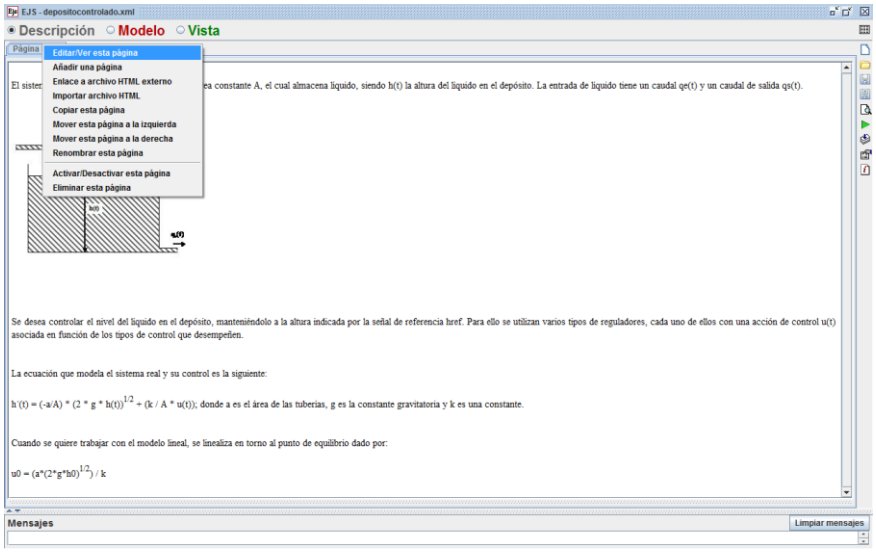

Figura 44: Procedimiento editar en panel *descripción*

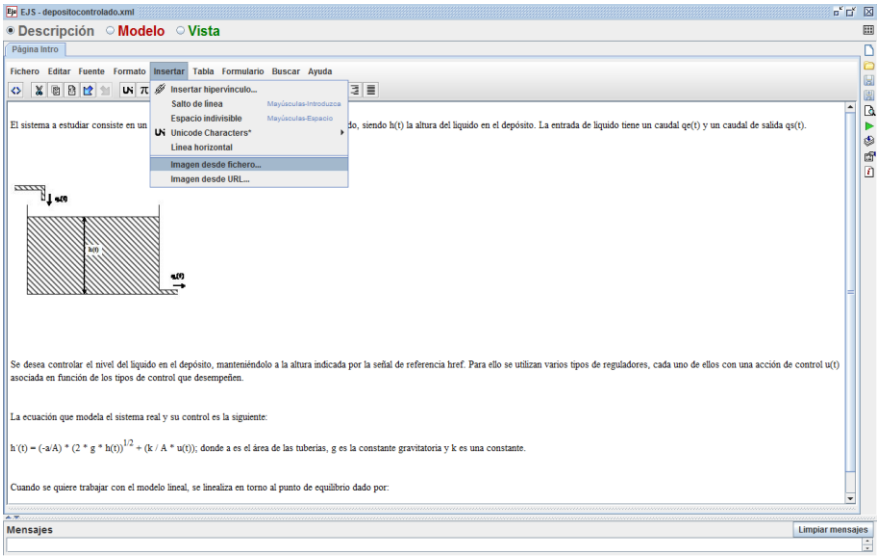

Figura 45: Procedimiento insertar imagen en panel *descripción*

Capítulo 5
Control de un avión

5.1 ENUNCIADO DEL PROBLEMA

El enunciado de este sistema se basa en [3].

Se desea estudiar el sistema consistente en un avión que se posiciona en función del ángulo de la dirección de vuelo relativa a la horizontal $\gamma(t)$, del ángulo del avión respecto al eje horizontal $\theta(t)$, de la desviación respecto al ángulo de ataque $\alpha(t)$ y de la desviación vertical del avión $h(t)$.(*Ver figura 46*).

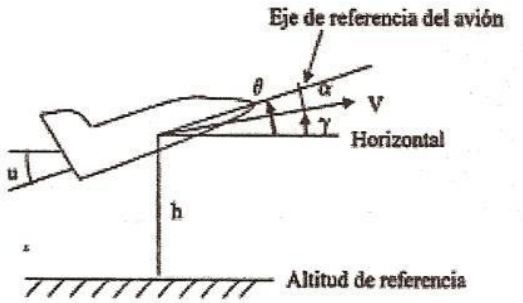

Figura 46: Esquema sistema avión

El análisis de control se basa en el movimiento vertical y de cabeceo del sistema, lo que se quiere es intentar mantener el avión a la altura indicada por *href(t)* y a un ángulo concreto respecto al eje horizontal, $\theta(t)$. Para conseguir este control se hace uso de una serie de reguladores: proporcional (P), proporcional derivativo, proporcional integral (PI) y proporcional integral derivativo (PID).

El diagrama de bloques correspondiente al regulador PID que se muestra en la *figura 47* contempla las tres acciones básicas de control. Para el caso del regulador proporcional se anula la acción integral y la acción derivativa, en el regulador PD se anula la acción integral y en el regulador PI se anula la acción derivativa.

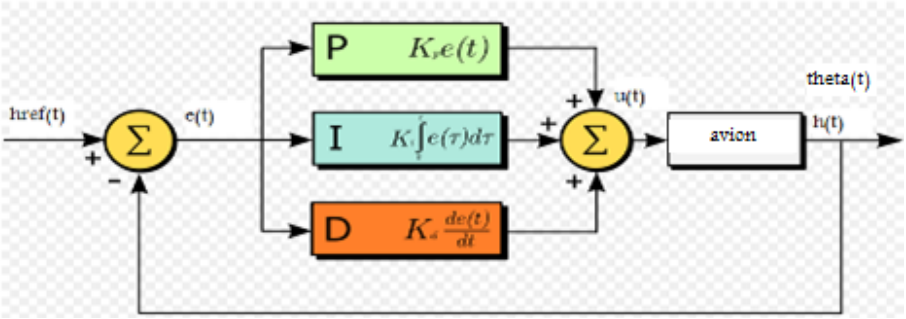

Figura 47: Diagrama de bloques de regulador PID sistema avión

Las ecuaciones que modelan el movimiento vertical y el cabeceo del avión junto con las ecuaciones necesarias para poder controlarlo son las que se muestran a continuación. *V*, Q, τ y ω_0 son constantes del sistema.

$$\tau.\frac{d\gamma(t)}{dt} = \alpha(t)$$

$$\frac{d^2\theta(t)}{dt} = -\omega_0^2.(\alpha(t) - Q.u(t))$$

$$\frac{dh(t)}{dt} = V.\gamma(t)$$

$$\gamma(t) = \theta(t) - \alpha(t)$$

Se van a presentar dos acciones de control: *u1(t)* para regular la altura a la que se encuentra el avión y *u2(t)* para controlar el cabeceo del avión. La acción de control total resultante es *u(t)=u1(t)-u2(t).* La señal de error que aparece en *u1(t)* es igual a la diferencia entre *href(t)* y *h(t),* mientras que la señal de error resultante del control del movimiento de cabeceo es *θ(t).*

La acción de control que actúa sobre el avión depende del tipo de regulador con el que se controla el sistema; a continuación se desglosan los distintos tipos de acciones de control de una forma genérica, u_x; como se ha dicho la acción de control del presente sistema será la resta resultante de las dos acciones de control presentes en el sistema de control.

- Para el regulador proporcional:

$$u_x(t) = kp.e(t)$$

- Para el regulador proporcional derivativo:

$$u_x(t) = kp.e(t) + kd.\frac{de(t)}{dt}$$

- Para el regulador proporcional integral:

$$u_x(t) = kp.e(t) + ki.\int_0^t e(t)dt$$

- Para el regulador proporcional integral derivativo:

$$u_x(t) = kp.e(t) + ki.\int_0^t e(t)dt + kd.\frac{de(t)}{dt}$$

5.2 RESOLUCIÓN DEL PROBLEMA: IMPLEMENTACIÓN CON EJS

La resolución de este problema se consigue a través de *EJS* actuando sobre los distintos paneles de trabajo que proporciona la herramienta software: *descripción, modelo y vista*.

El desarrollo de la creación del modelo para su simulación se inicia en el panel de trabajo *modelo* (a su vez el panel *modelo* se divide en *variables, inicialización, evolución, relaciones fijas* y *propio*), a continuación en el panel *vista* y finalmente en el panel de trabajo *descripción*.

5.2.1 Variables

Se declaran las variables del sistema:

- h: Desplazamiento vertical del avión.
  La variable *h* tiene un valor diferente al de *href* para así poder observar cómo reacciona el regulador ante esta diferencia (la distancia vertical a la que se encuentra el avión ha de variar para intentar igualar a *href*).
  Se inicializa con un valor de 0 u. La altura del avión está expresada en tanto por uno para que la simulación equivalga a la escala real más apropiada.

- theta (θ): Ángulo del avión respecto al eje horizontal, expresado en radianes.
  Comienza tomando un valor de 0 rad.
- thetagra: Esta variable indica lo mismo que *theta* con la diferencia de que se mide en grados.
  Al inicio toma el valor de 0 º.
- d_theta: Se le asigna el valor de la derivada de *theta*.
  Su valor inicial es 0.
- alfa (α): Desviación respecto al ángulo de ataque.
  Su valor es 0 rad.
- gamma (γ): Ángulo de la dirección de vuelo relativa a la horizontal.
  Su valor es 1.57 rad.
- tau (τ): Constante del sistema de valor 1.
- omega_cero (ω_0): constante del sistema de valor 1.
- Q: Constante del sistema de valor 2.
- V: Constante del sistema de valor 4.
- t: Tiempo.
  El progresivo cambio de la variable *t* va ligado a la evolución de la simulación del sistema.
  Se inicia a 0 s.
- dt: Diferencial de tiempo
  El diferencial de tiempo toma el valor 0.1 s. Para valores menores el proceso de estabilización del sistema es más lento.
- href: Desplazamiento vertical deseado. Esta variable puede tomar cualquier valor comprendido entre -1.5 u y 1.5 u, que son los valores que se han impuesto.
  Esta variable comienza la simulación con el valor 0.5 u.
- theta_ref: Ángulo del avión respecto al eje horizontal. Lo que se quiere es que el avión no sufra oscilaciones; es decir, que el valor de este ángulo siempre valga 0 rad.
- thetarefgra: Es el valor de *theta_ref* expresado en grados.
- error: Error cometido por el regulador para el control de *h*.
  Este valor comienza tomando el valor 0 u.
- error2: Error cometido por el regulador para el control de *theta*.
  Su valor inicial es 0 rad.

- d_error: Es la derivada del error cometido en el control de *h*.
  Esta variable auxiliar se emplea para asignarle el valor de la derivada de *error*. Necesaria para la acción derivativa del regulador.
  Su valor inicial es cero.
- d_error2: Es la derivada del error cometido en el control de *theta*.
  Esta variable auxiliar se emplea para calcular la derivada de *error2*; asignándole su valor. Necesaria para la acción derivativa del regulador.
  Comienza valiendo cero.
- int_error: Variable auxiliar empleada para la asignación de la integral del error cometido en el control de *h*. La integral se desarrolla por medio de una suma acumulativa del error.
  Al inicio toma el valor cero.
- int_error2: Variable auxiliar empleada para la asignación de la integral del error cometido en el control de *theta*. La integral es una suma que va acumulando los valores del error.
  Su valor inicial es cero.
- u1: Acción de control.
  Esta variable inicialmente toma el valor cero. Cuando empiece la ejecución de código *u1* toma el valor correspondiente dependiendo del regulador que esté controlando el sistema, con el valor dado por el error en el control de *h*.
- u2: Acción de control.
  El valor inicial asignado a esta variable es cero. Al empezar la ejecución del código *u2* tiene el valor correspondiente dependiendo del regulador que esté controlando el sistema, con el valor dado por el error en el control de *theta.*
- u: Acción de control conjunta del regulador. Es igual a la diferencia entre *u1* y *u2*.
- kp: Constante de proporcionalidad del regulador.
  La variable *kp* se inicia con el valor 1.
- kd: Constante derivativa del regulador.
  La variable *kd* comienza tomando el valor numérico cero.
- ki: Constante de integración del regulador.
  La variable *ki* comienza teniendo el valor 0.001.

En la opción *variables* del panel *modelo* se procede a clasificar las variables en dos grupos; el primer grupo de variables se encuentra situado dentro de la pestaña *AVION (ver figura 48)* y son por tanto las variables que definen el sistema. En el segundo grupo se encuentran las variables relacionadas con el control del sistema y por consiguiente engloba las variables que permiten controlar el sistema. Estas variables están ubicadas dentro de la pestaña *CONTROL (ver figura 49).*

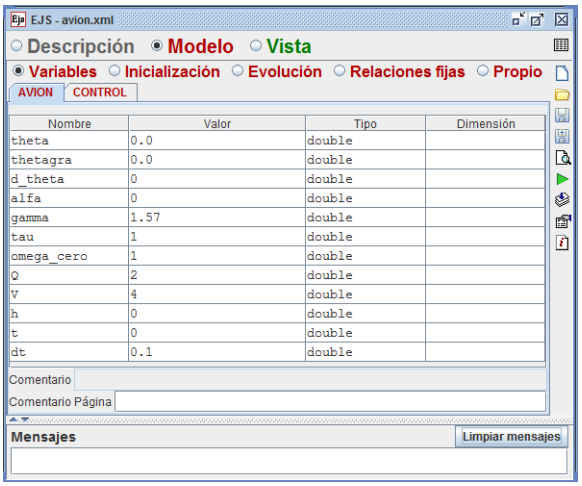

Figura 48: Variables en pestaña *AVION*

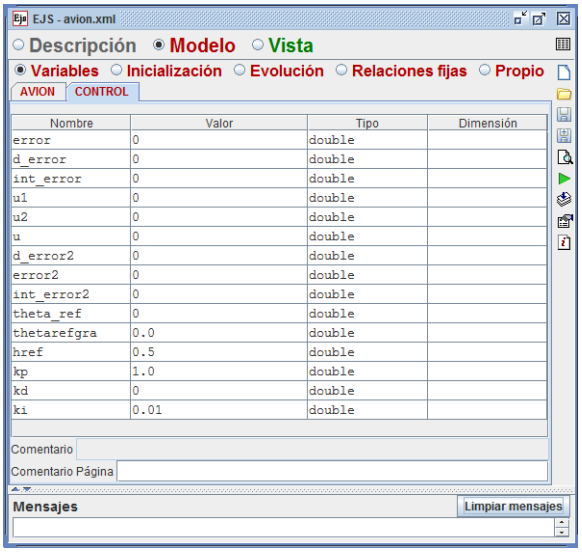

Figura 49: Variables en pestaña *CONTROL*

5.2.2 Ecuaciones

Conocidas las variables, se procede a escribir las ecuaciones que van a definir nuestro sistema controlado. Las ecuaciones, transcritas a lenguaje *Java*, aparecen en el *panel modelo* dentro de las opciones *evolución* y *relaciones fijas*.

En la opción *evolución* aparecen las ecuaciones que muestran la evolución del sistema controlado en función del tiempo. Se encuentra la ecuación de la derivada del error que se comete en el control de *h*, la ecuación de la derivada del error que se comete al regular *theta* y las ecuaciones de movimiento vertical y cabeceo del avión que llevan asociadas variables derivadas *(ver figura 50).*

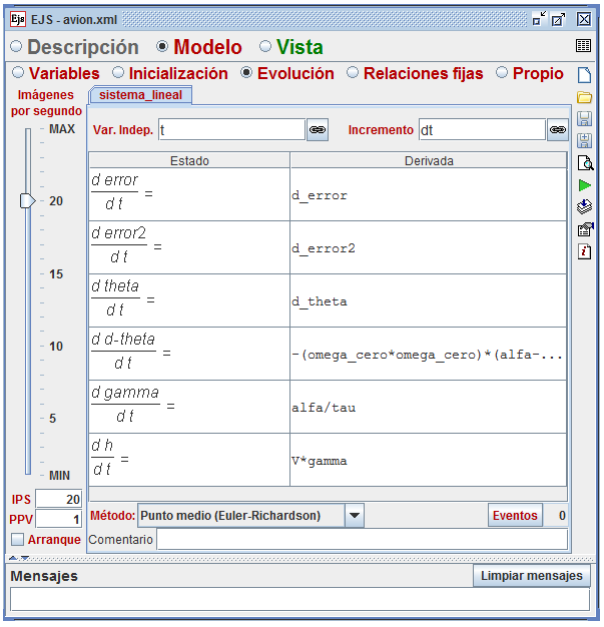

Figura 50: Subpanel e*volución* de sistema avión

Al parámetro *imágenes por segundo (IPS)* se le asigna el valor 20 para que la simulación sea rápida ya que en algunas ocasiones el sistema tarda mucho en estabilizarse.

Si se quisiera aproximar al tiempo real lo que tarda en ejecutarse la simulación lo que se debe poner es un valor de *IPS* de 10, acompañado de un diferencial de tiempo de 0.1 segundos.

En la opción *relaciones fijas* es donde aparece el código que hace que se eliminen las trazas de las gráficas cuando *t* toma el valor 100 y comiencen de nuevo su aparición desde el

origen de coordenadas.

También es en esta opción donde se añaden unas sentencias en las que se acota el ángulo del avión respecto al eje horizontal para que el sistema no se haga inestable, se resetea el valor de *ki* ya que cuando vale cero el sistema se inestabiliza, se calculan los distintos errores cometidos por el regulador, el valor de las variables *int_error* e *int_error2*, el valor de las diferentes acciones de control. Se realiza el cálculo para poder expresar las variables *theta* y *theta_ref* en grados. Por último se añade la única ecuación de movimiento de cabeceo que no lleva ninguna derivada asociada *(ver figura 51).*

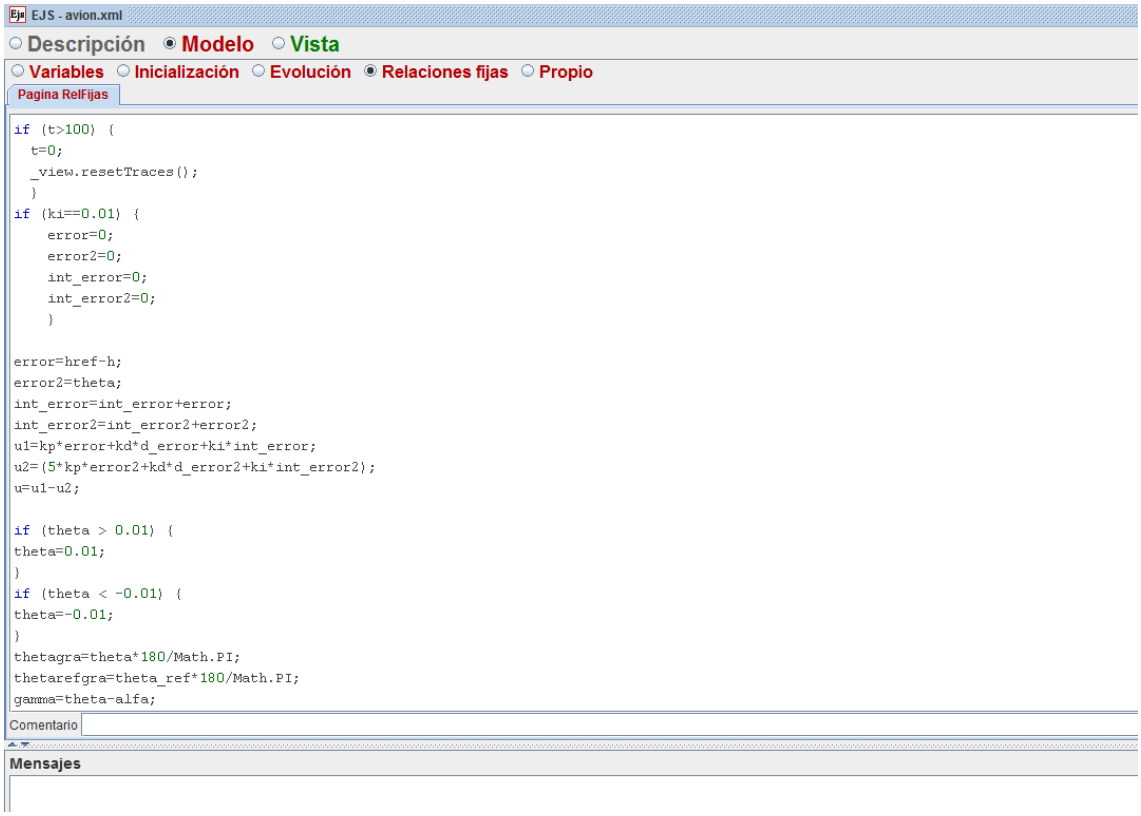

Figura 51: Ecuaciones de ligadura de sistema avión

5.2.3 Vista

A continuación se crea la interfaz gráfica de usuario en el *panel vista*. Al inicio del árbol de elementos se coloca la ventana principal donde se incluyen todos los elementos gráficos

necesarios. Dicha ventana recibe el nombre *avion* y sigue una política de distribución *rejilla*, con una fila y dos columnas. Esta distribución se presenta a través de los dos contenedores cuyo padre es la ventana principal: *zona_avion y graficas (ver figura 52).*

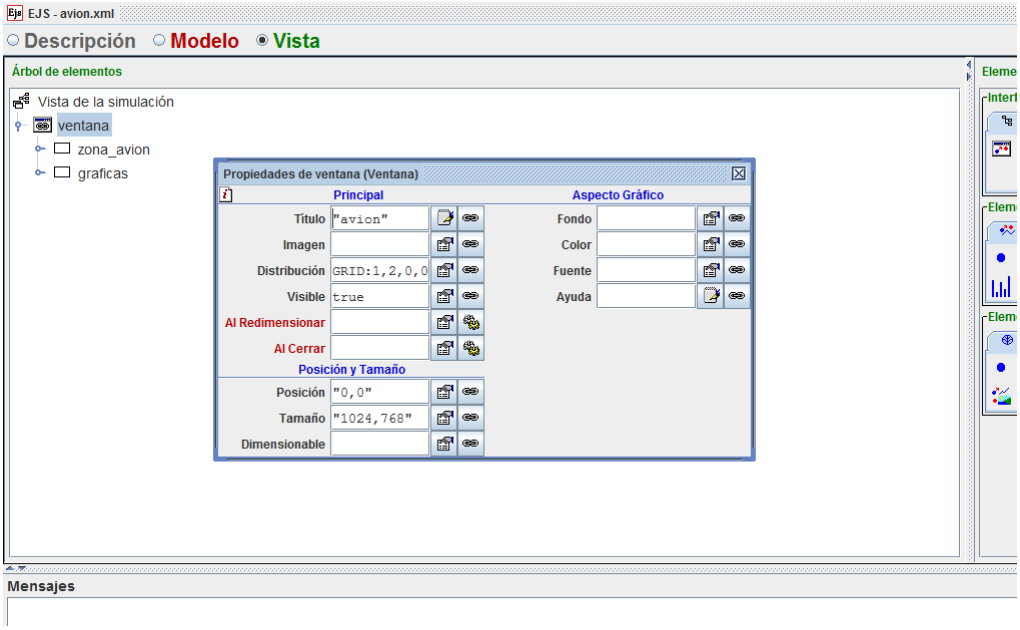

Figura 52: Menú *propiedades* de ventana principal sistema avión

El contenedor *zona_avion* sigue una política de distribución de *márgenes (*o *border*). En la posición de arriba se encuentra el contenedor *panel4*, el hijo que se ubica en el centro es *panelDibujo* y el contenedor *panel* se encuentra en la posición de abajo *(ver figura 53).*

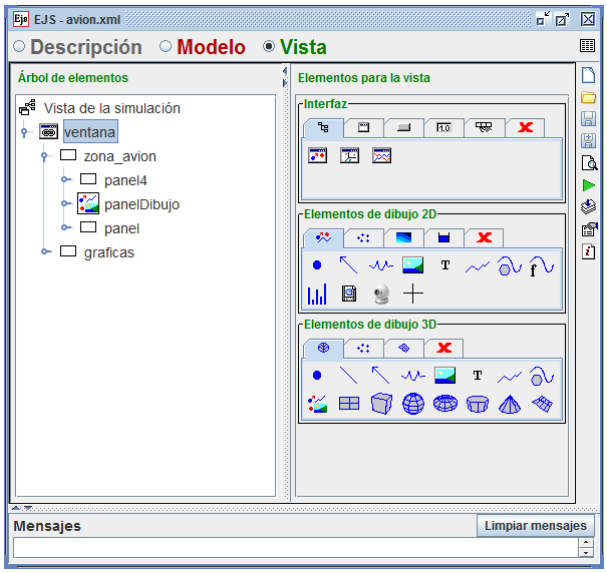

Figura 53: Composición de contenedor *zona_avion*

En el contenedor *panelDibujo* se encuentra la imagen que va a simular el sistema. También se encuentran un conjunto de textos interactivos que facilitan la visualización de la altura a la que se encuentra el avión. La imagen, en formato .jpg, se posicionará verticalmente en función de la variable *h* y el ángulo del avión respecto al eje horizontal viene dado en función de *theta (ver figura 54).*

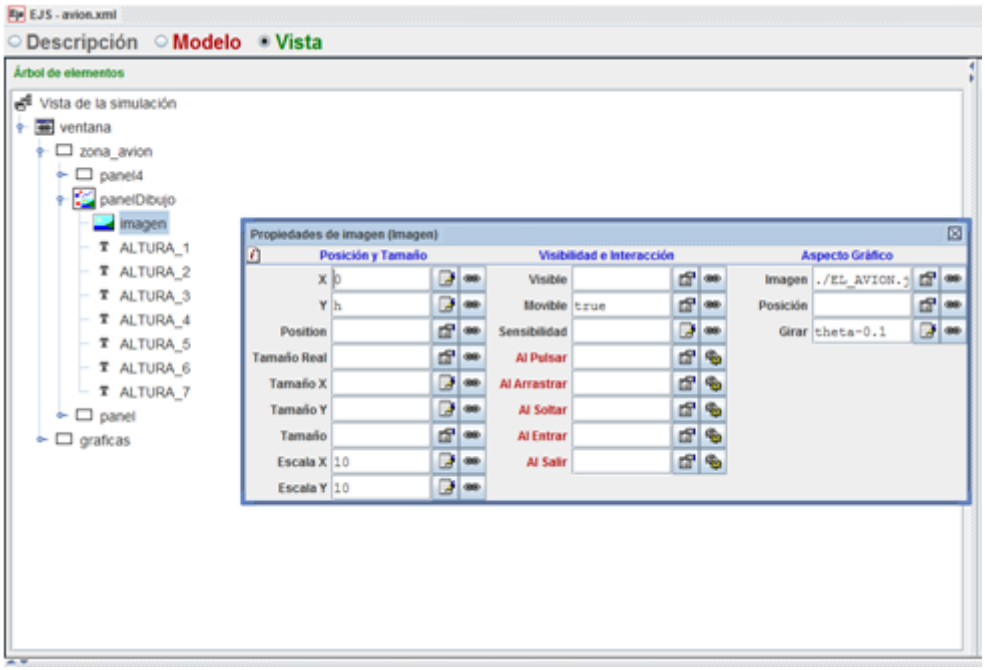

Figura 54: Menú *propiedades* de *imagen*

Como texto interactivo se escoge *ALTURA_5*, que informa de cuando la altura *h* alcanza el valor de 1 m *(ver figura 55).*

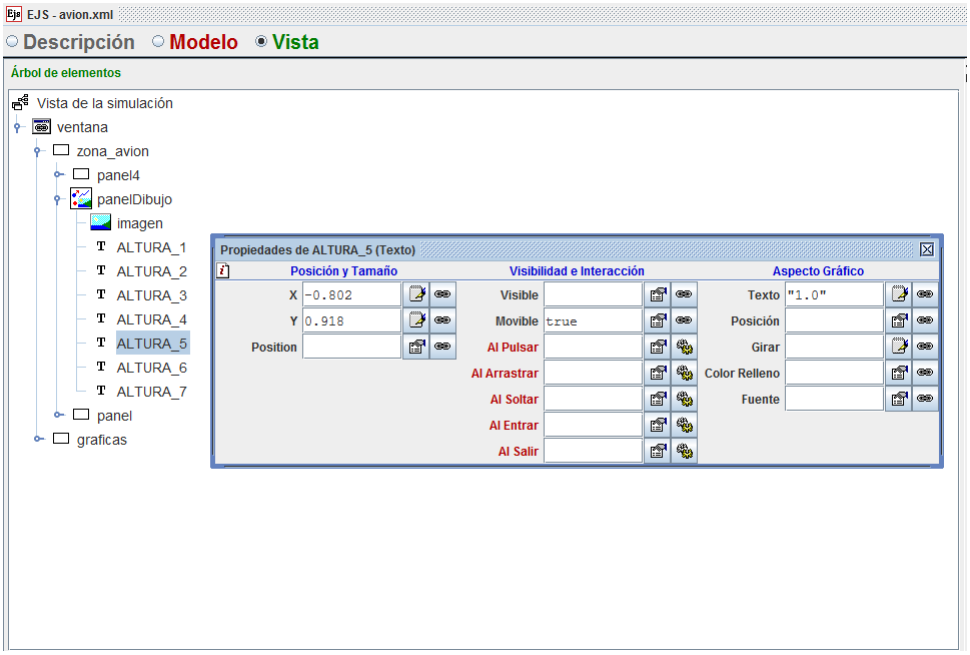

Figura 55: Menú *propiedades* de texto interactivo *ALTURA_5*

El contenedor *panel4* sigue una distribución *border*; este contenedor tiene dos hijos: *panel2* en la posición de arriba y *panel5* en la posición centro *(ver figura 56).*

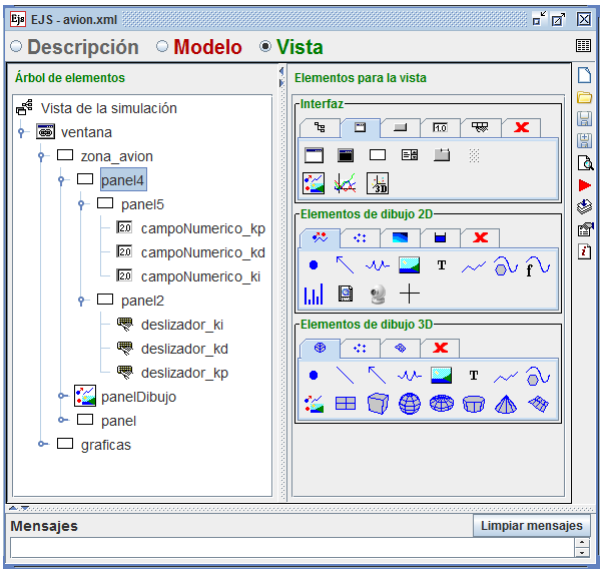

Figura 56: Distribución de contenedor *panel4*

El contenedor *panel2* sigue una distribución *rejilla* de 3 filas y una columna; los elementos básicos contenidos son deslizadores cuyo papel es variar el valor de *kp*, *kd* y *ki* y con ello estudiar el comportamiento de los distintos reguladores. Si se analiza el deslizador que

tiene por nombre *deslizador_ ki* se obtiene que los valores entre los que puede variar *ki* son de 0.01 a 10, el formato de presentación es decimal, la orientación del deslizador es horizontal, el color de la fuente negro, etc. (*ver figura 57*).

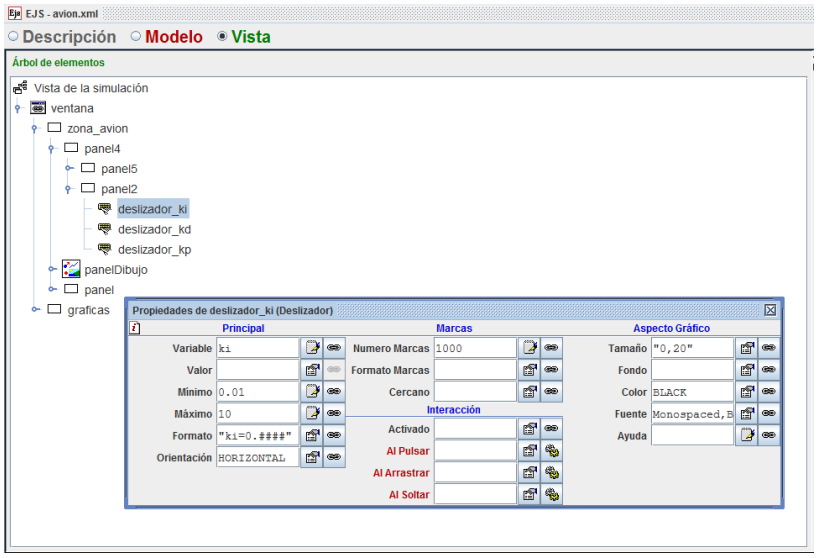

Figura 57: Propiedades de *deslizador_ki*

El contenedor *panel5* se encuentra dividido en tres columnas. Alberga una serie de campos para mostrar y modificar el valor numérico de *kp*, *kd* y *ki*. En estos campos se refleja el cambio que sufren estas variables mediante los deslizadores contenidos en *panel2 (ver figura 58).*

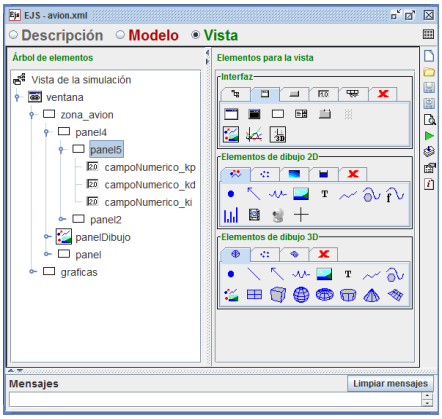

Figura 58: Elementos de contenedor *panel5*

El contenedor *panel* sigue una política de *rejilla* de una sola fila y tres columnas. Contiene un campo que muestra el valor que toma la variable *h*, otro campo que muestra el valor de la variable *thetagra* y una etiqueta que informa que el modelo que se está estudiando es

el lineal.

Los dos campos numéricos albergados en el contenedor panel no son editables; es decir, no se puede modificar desde ellos el valor de la variable asociada a cada campo, únicamente sirven para mostrar su valor. Ver propiedades de *campoNumerico_theta* en *figura 59*.

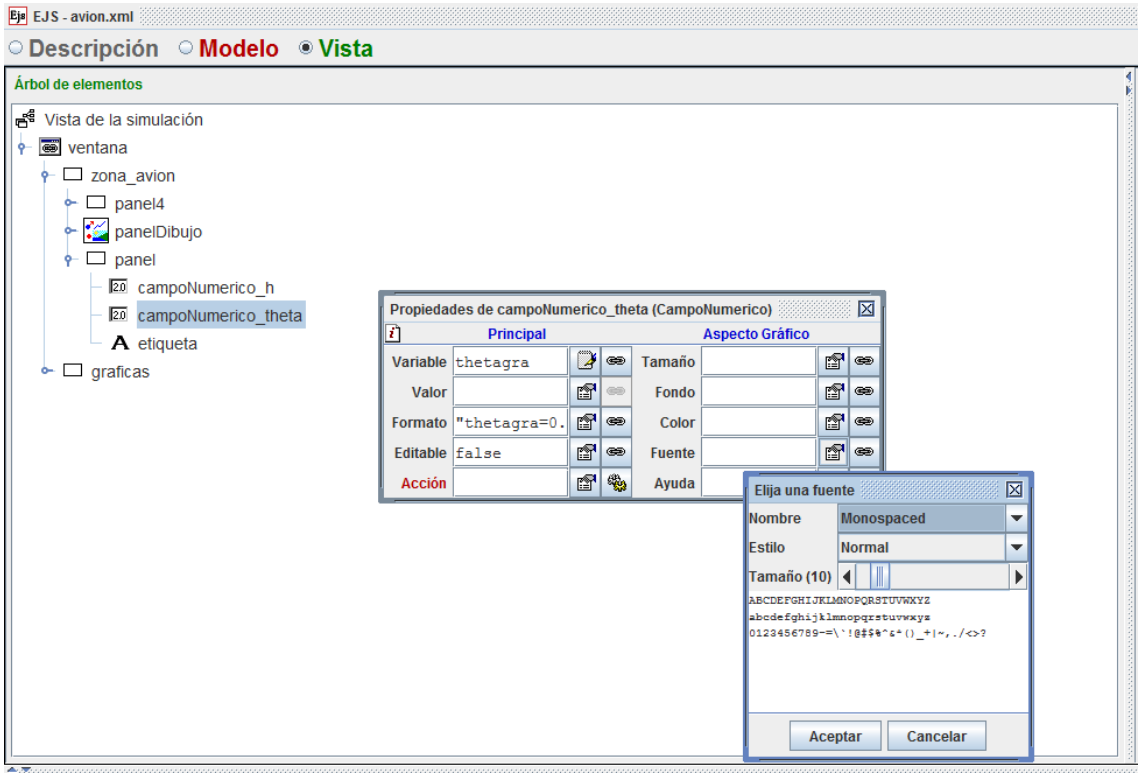

Figura 59: Menú *propiedades* de *campoNumerico_theta*

El siguiente paso es analizar el diseño del contenedor *graficas*. Su política de distribución es *VBOX*. Los hijos de este contenedor son *panel7*, *panelConEjes*, *panelConEjes2* y *panel6 (ver figura 60).*

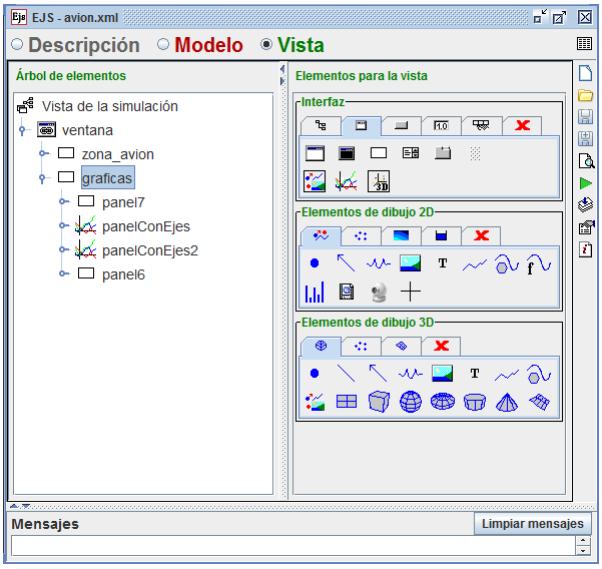

Figura 60: Contenedor *graficas*

El contenedor *panel7*, con una distribución *rejilla* de dos filas y una columna, presenta un deslizador que permite modificar el valor de *href* y un campo para mostrar y/o modificar el valor de *href* (*ver figura 61).*

Al modificar el valor deseado *href* el regulador a través de su acción de control consigue aproximar la altura real a la altura buscada; esta aproximación depende del regulador elegido y de los valores de las constantes asociadas a este regulador.

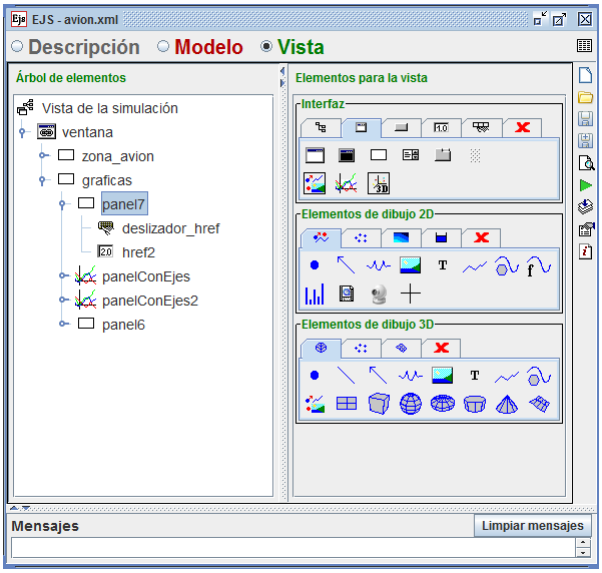

Figura 61: Elementos de contenedor *panel7*

El contenedor *panelConEjes* es un contenedor de dibujo bidimensional que incorpora un sistema de ejes cartesianos.

El título que recibe este panel es "altura VS tiempo", el eje x representa el tiempo y el eje y representa *h* y *href* de manera simultánea, se van a añadir dos trazas para la misma gráfica. Estas trazas son *Altura* y *Altura_ref* *(ver figura 62).* El hecho de añadir la traza con el valor de *href* es para observar cómo está reaccionando el regulador.

El eje x alberga valores de tiempo comprendidos entre cero y cien segundos, mientras que los valores que se representan en el eje y van de -1.1 p. u a 1.1 p.u.

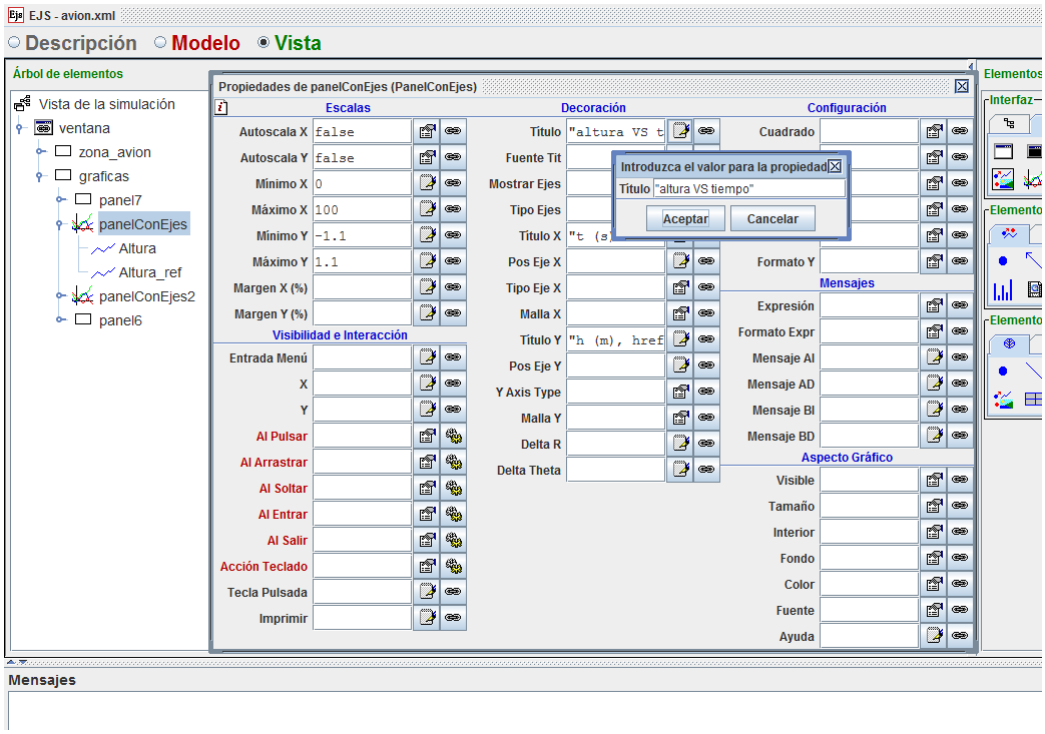

Figura 62: Propiedades de *panelConEjes*

El contenedor *panelConEjes2* alberga la traza *theta* que representa la evolución en el tiempo del ángulo del avión respecto al eje horizontal y la traza *theta_ref* que muestra el valor de *set-point* (*ver figura 63*).

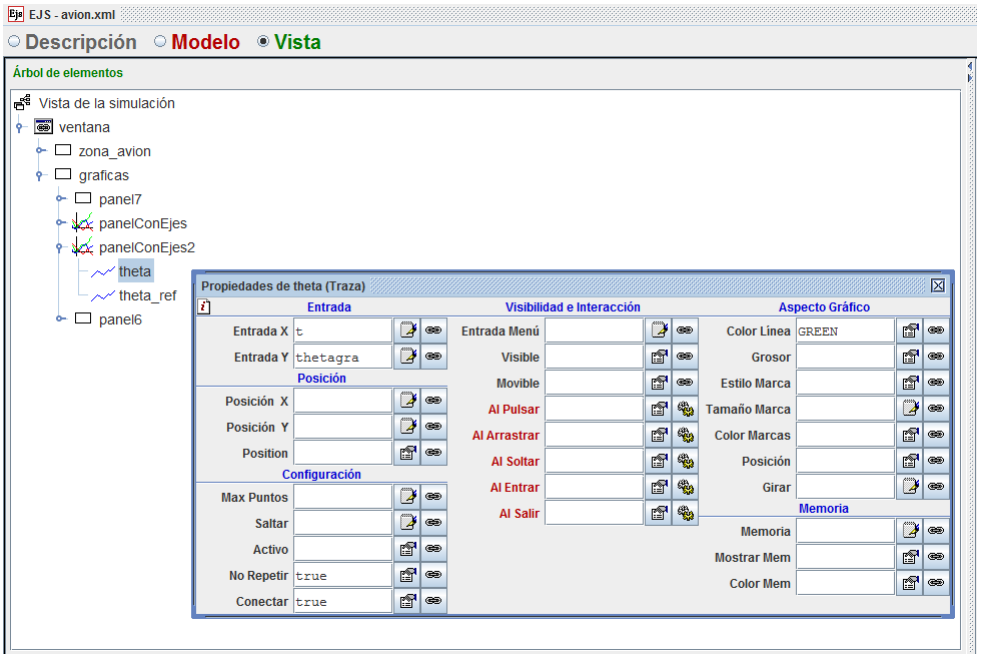

Figura 63: Propiedades de traza *theta*

En el contenedor *panel6*, con distribución *rejilla* de una fila y tres columnas, se implementa un panel de accionamiento con tres botones para controlar la ejecución de la simulación *(ver figura 64)*.

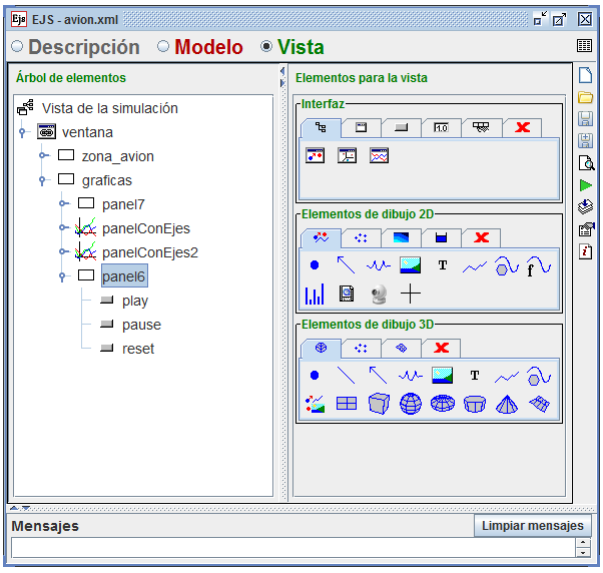

Figura 64: Elementos de contenedor *panel6*

Al presionar el botón *play* se pone en marcha la evolución de la simulación; cuando la evolución esté parada *(ver figura 65)*.

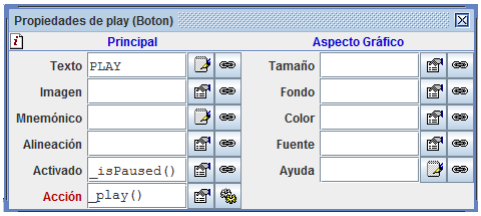

Figura 65: Menú *propiedades* de botón *play*

Cuando se pulsa el botón *pause* se detiene la evolución de la simulación siempre y cuando la evolución está en marcha *(ver figura 66).*

Figura 66: Menú *propiedades* de botón *pause*

Al pulsar *reset* se inicializan las variables a su valor de partida (toman el mismo valor que tienen en el apartado *variables* dentro de panel *modelo*) y la vista vuelve a su estado inicial *(ver figura 67).*

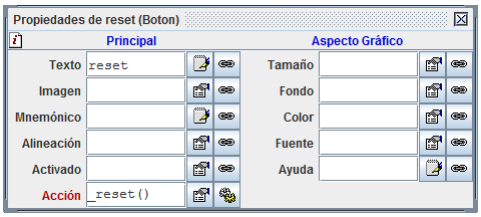

Figura 67: Menú *propiedades* de botón *reset*

Creada la estructura completa de los elementos gráficos en la *figura 68* se muestra cómo queda la interfaz gráfica que aparece cuando se lanza la simulación.

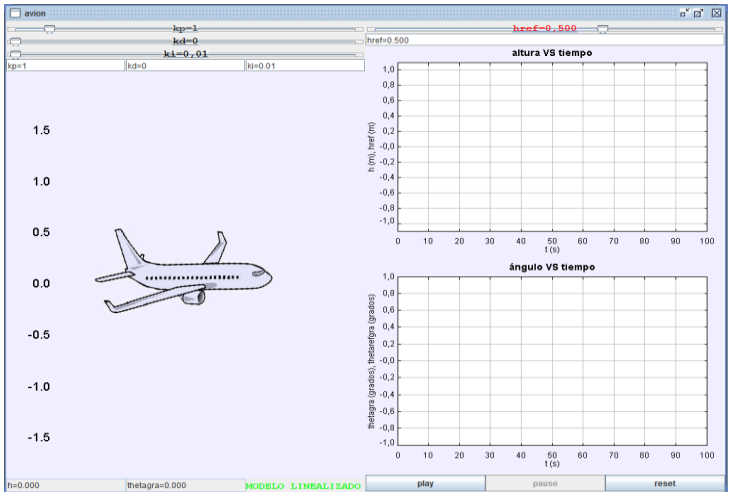

Figura 68: Vista de simulación sistema avión

5.2.4 Descripción

Este panel contiene el descriptivo del sistema del avión, complemento a la simulación en el servidor correspondiente. La *figura 69* muestra el panel *descripción* de *avion.xml.*

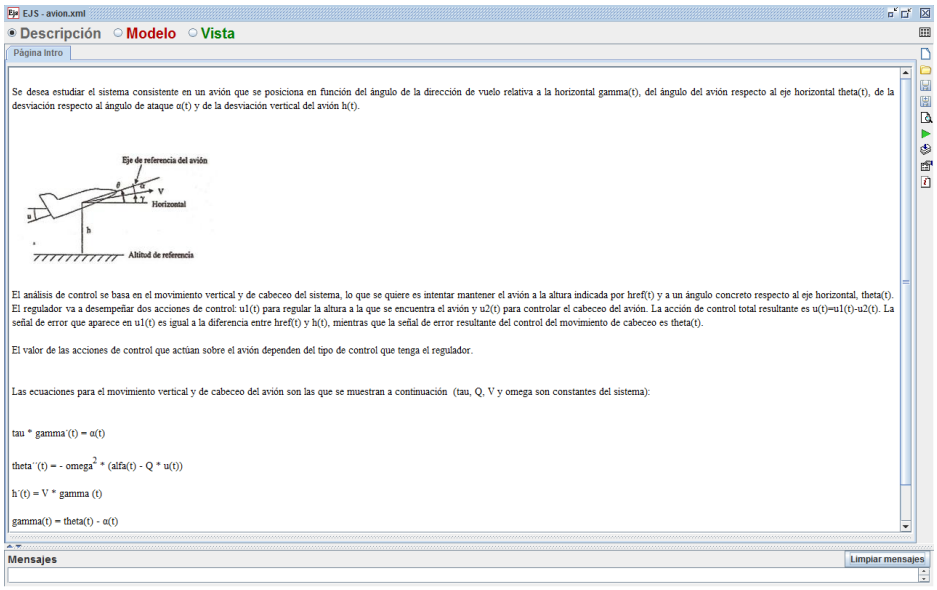

Figura 69: Panel *descripción* sistema avión

Capítulo 6

Control de un cilindro hidráulico

6.1 ENUNCIADO DEL PROBLEMA

El presente sistema consiste en un cilindro hidráulico, donde un fluido ejerce presión actuando sobre el pistón del cilindro y produciendo movimiento; llamamos *x(t)* al desplazamiento del pistón. La fuerza *F(t)* que recibe el pistón es igual a la presión multiplicada por la superficie activa del émbolo (*ver figura 70*).

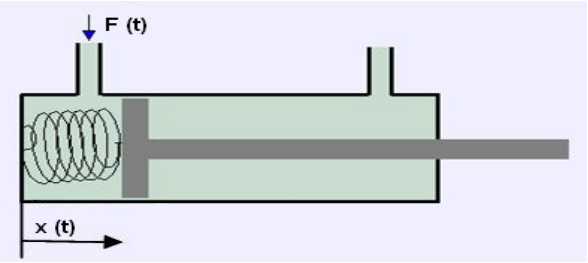

Figura 70: Esquema sistema cilindro hidráulico

Se desea controlar el desplazamiento del émbolo en el cilindro, manteniéndolo en la posición indicada por la señal de referencia *xref*. Con el fin de corregir el error resultante de la comparación entre *x* y *xref* se utilizan varios tipos de reguladores: proporcional (P), proporcional derivativo (PD), proporcional integral (PI) y proporcional integral derivativo (PID).

En la *figura 71* aparece el diagrama de bloques correspondiente al regulador PID. Para el caso del regulador proporcional se anula la acción integral y la acción derivativa, en el regulador PD se anula la acción integral y en el regulador PI se anula la acción derivativa.

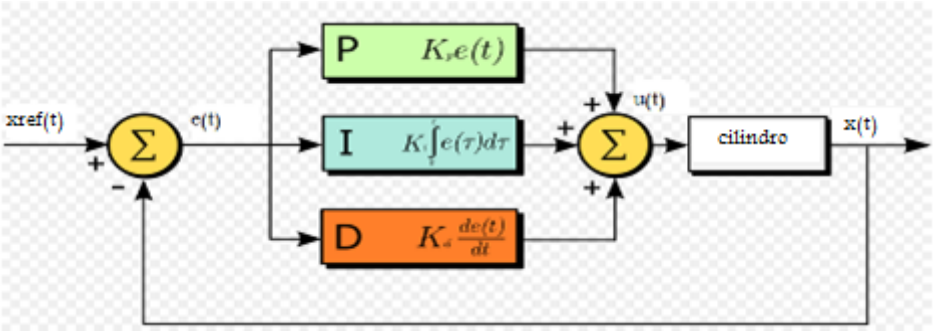

Figura 71: Diagrama de bloques de regulador PID sistema cilindro

La ecuación que modela el sistema junto con las ecuaciones necesarias para poder controlarlo son las que se muestran a continuación:

$$F(t) = k.x(t) + B.v(t) + m.a(t) = u(t)$$

La ecuación con la que obtenemos el error, *e(t)*, entre la señal de salida y la señal de referencia:

$$e(t) = h_{ref}(t) - h(t)$$

La acción de control que actúa sobre el cilindro depende del tipo de regulador con el que se controla el sistema, lo desglosamos a continuación:

- Para el regulador proporcional:

  $$u(t) = kp.e(t)$$

- Para el regulador proporcional derivativo:

  $$u(t) = kp.e(t) + kd.\frac{de(t)}{dt}$$

- Para el regulador proporcional integral:

  $$u(t) = kp.e(t) + ki.\int_0^t e(t)dt$$

- Para el regulador proporcional integral derivativo:

  $$u(t) = kp.e(t) + ki.\int_0^t e(t)dt + kd.\frac{de(t)}{dt}$$

6.2 RESOLUCIÓN DEL PROBLEMA: IMPLEMENTACIÓN CON EJS

Para comenzar a resolver este problema y conseguir la simulación del sistema a través de *EJS* hay que contemplar los distintos paneles de trabajo que proporciona la herramienta: *descripción, modelo y vista*.

La implementación se inicia en el panel de trabajo *modelo* (a su vez el panel *modelo* se divide en *variables, inicialización, evolución, relaciones fijas* y *propio*), a continuación en el panel *vista* y finalmente en el panel de trabajo *descripción*.

6.2.1 Variables

Se declaran las variables del sistema:

- x: Desplazamiento lineal que sufre el émbolo.
  La variable *x* toma un valor diferente de *xref* para poder observar cómo actúa el regulador ante esta diferencia (la posición del pistón tiene que variar para conseguir igualar a *xref*).
  Se inicializa con un valor de 0 m.
- v: Velocidad del pistón en el interior del cilindro.
  Comienza tomando un valor de 0 m/s.
- a: Aceleración que sufre el pistón en el interior del cilindro.
  Al inicio toma el valor 0 m^2/s.
- m: Masa del émbolo.
  Su valor es 10 kg.
- B: Rozamiento viscoso. Esta constante define la resistencia que opone un líquido a fluir cuando se le aplica una fuerza [7].
  Toma el valor de 10 Pa.s.
- k: Constante elástica. Esta variable define el comportamiento elástico del resorte al que va unido el émbolo; es decir es la propiedad mecánica que tiene el resorte de sufrir deformaciones al estar sometido a una fuerza exterior y recuperar su forma inicial cuando desaparece la fuerza.
  Su valor es 1 N/m.
- t: Tiempo.
  La variable *t* se inicia con el valor cero y con el cambio de este valor iremos viendo cómo evoluciona el sistema en el tiempo.
- dt: Diferencial de tiempo.
  El diferencial de tiempo va a tener el valor 0.1 s. Para valores más pequeños el proceso de estabilización del sistema es más lento.
- xref: Desplazamiento del émbolo deseado. Esta variable puede tomar cualquier valor entre cero y un metro, que es lo que se ha fijado.
  Esta variable comienza la simulación con el valor 0.5 m.
- error: Error cometido por el controlador.
  Este valor comienza tomando el valor 0 m.
- derror: La derivada del error.

Esta variable auxiliar se emplea para asignarle la derivada del error, necesaria para la acción derivativa del regulador.

Comienza valiendo cero.

- int_error: La integral del error.

  Esta variable auxiliar se emplea para calcular la integral del error, necesaria para la acción integral del regulador. La integral se desarrolla por medio de una suma que va acumulando el error.

  Comienza valiendo cero.

- u: Acción de control.

  Esta variable inicialmente toma el valor cero. Cuando empiece la ejecución de código *u* tomará el valor correspondiente dependiendo del regulador que esté controlando el sistema, con el valor dado por el error.

- kp: Constante de proporcionalidad del regulador.

  La variable *kp* se inicia con el valor 1.

- kd: Constante derivativa del regulador.

  La variable *kd* comienza tomando el valor numérico cero.

- ki: Constante de integración del regulador.

  La variable *ki* comienza teniendo el valor cero.

Se van a clasificar las variables en dos grupos; el primer grupo de variables está situado dentro de la pestaña *cilindro (ver figura 72)* y son por tanto las variables que definen el sistema a estudiar. En el segundo grupo se encuentran las variables referentes al regulador y por tanto engloba las variables que permiten controlar el sistema. Estas variables están ubicadas dentro de la pestaña *reguladores,* en la opción *variables (ver figura 73).*

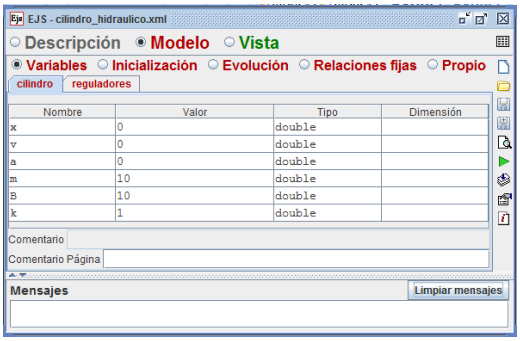

Figura 72: Variables de pestaña *cilindro* sistema cilindro

Figura 73: Variables de pestaña *reguladores* sistema cilindro

6.2.2 Ecuaciones

Una vez presentadas las variables, se escriben las ecuaciones que van a definir nuestro sistema. Las ecuaciones, transcritas a lenguaje *Java*, las vamos a escribir en el *panel modelo* dentro de la opciones *evolución* y *relaciones fijas.*

En la opción *evolución* aparecen las ecuaciones que se vean afectadas por el paso del tiempo; así como la fórmula de la derivada del error para poder aplicar un control derivativo y las ecuaciones que definen la velocidad y la aceleración a las que se va a someter al émbolo *(ver figura 74).*

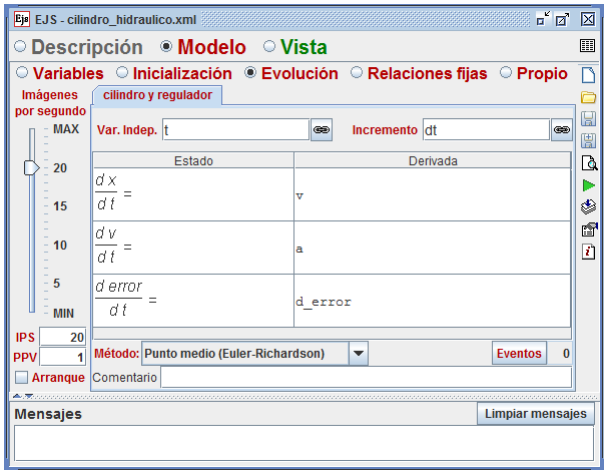

Figura 74: Subpanel *evolución* de sistema cilindro

El parámetro imágenes por segundo (*IPS*) recibe el valor veinte. Se escoge un valor *IPS* de veinte y un diferencial de tiempo de 0.1 segundos para que la simulación sea más rápida de lo que es en tiempo real; si se busca una simulación en tiempo real hay que poner un valor de *IPS* de 10 junto con un diferencial de tiempo de 0.1 segundos.

Se añaden unas sentencias en *relaciones fijas* en las que se establece que si el desplazamiento del émbolo es menor que cero el desplazamiento sea cero, y si el desplazamiento es mayor que uno su valor se iguale a uno; es decir que el controlador solo funcione cuando el émbolo se encuentre dentro de las dimensiones del cilindro.

También es aquí donde aparece el código que hace que desaparezcan las trazas de las gráficas cuando *t* es igual a 100 y se inicie de nuevo su aparición desde el origen de coordenadas.

En este apartado se incluye aparte de la ecuación que modela el sistema, las ecuaciones que calculan el error cometido por el regulador, el valor de la variable *int_error* y el parámetro de control *(ver figura 75).*

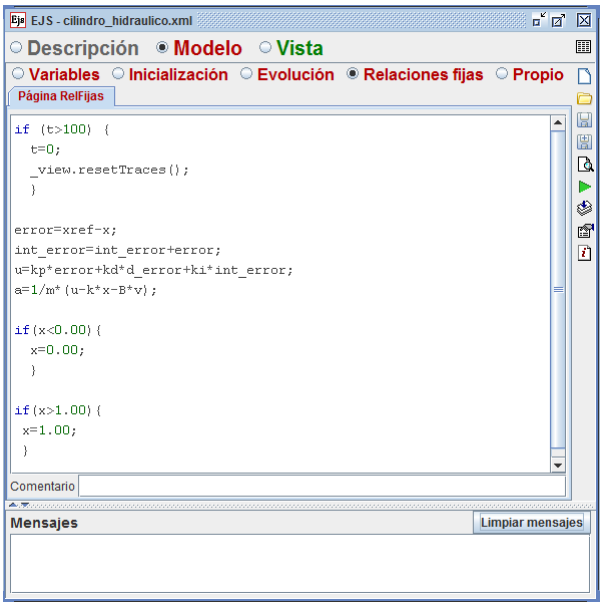

Figura 75: Ecuaciones de ligadura sistema cilindro

6.2.3 Vista

El siguiente punto es crear la interfaz de usuario en el *panel vista*. Al comenzar se coloca la ventana principal donde se incluyen todos los elementos necesarios para la vista. La ventana principal recibe el nombre *Cilindro Hidraulico* y tiene una política de distribución *rejilla*, con una fila y dos columnas. Esta distribución se presenta a través de los dos contenedores en los que se desglosa la ventana principal: *cilindro y reguladores (ver figura 76).*

.

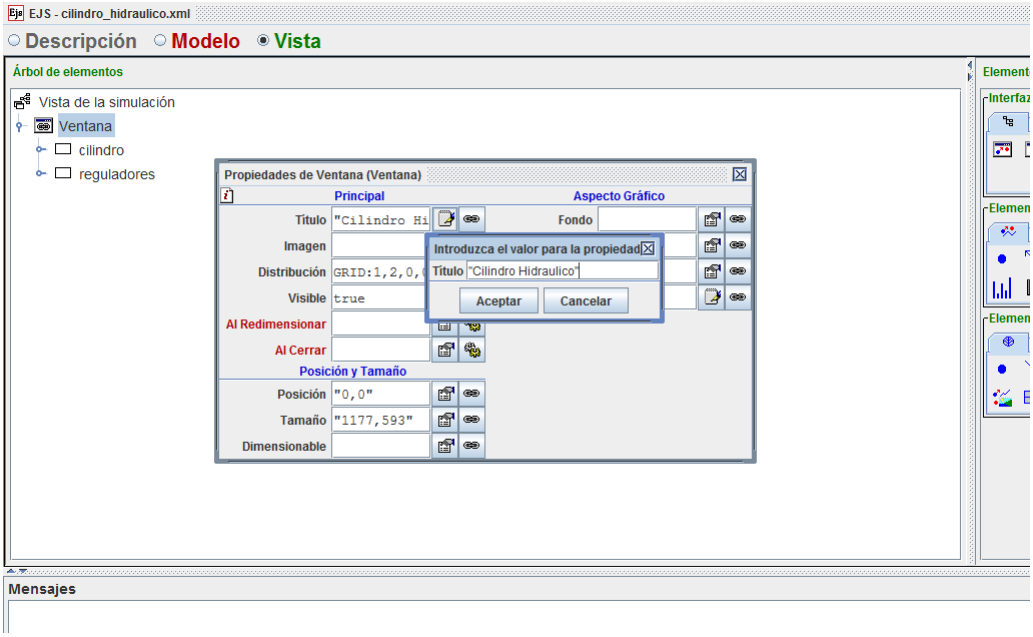

Figura 76: Menú *propiedades* de ventana principal sistema cilindro

El contenedor *cilindro* tiene una distribución de *márgenes (*o *border*). En la posición de arriba se localiza el contenedor *panel*, el hijo que se encuentra en el centro es *panelDibujo* y el contenedor *panel2* se encuentra en la posición de abajo *(ver figura 77).*

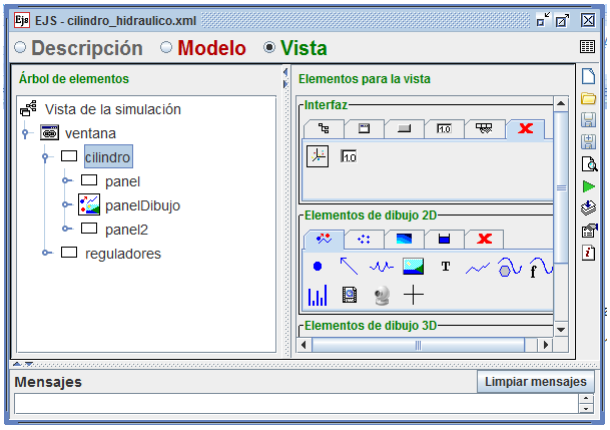

Figura 77: Composición contenedor *cilindro*

En el *panel Dibujo* aparecen los elementos gráficos que van a simular el sistema físico. También se encuentran un conjunto de textos interactivos que facilitan la visualización del nivel de líquido *(ver figura 78).*

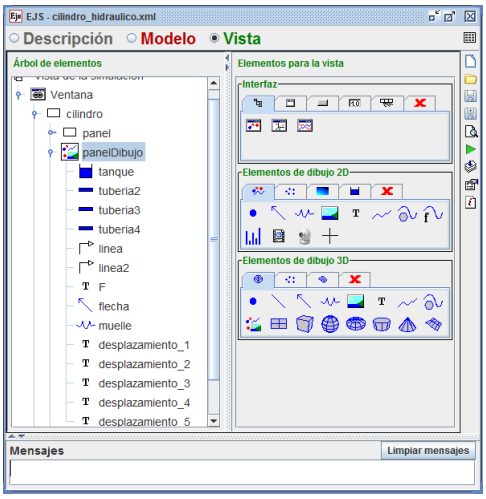

Figura 78: Elementos de contenedor *panelDibujo*

Las estructuras del cilindro y del émbolo se han diseñado a través de una serie de tuberías y un tanque. Si uno se fija en la componente vertical del émbolo correspondiente al elemento gráfico *tuberia3*, en el menú *propiedades* se puede observar lo siguiente: su posición coincide con el desplazamiento *x*, está definida la orientación de la tubería dentro de la ventana principal, su color de línea y de relleno es gris, etc. (*ver figura 79*).

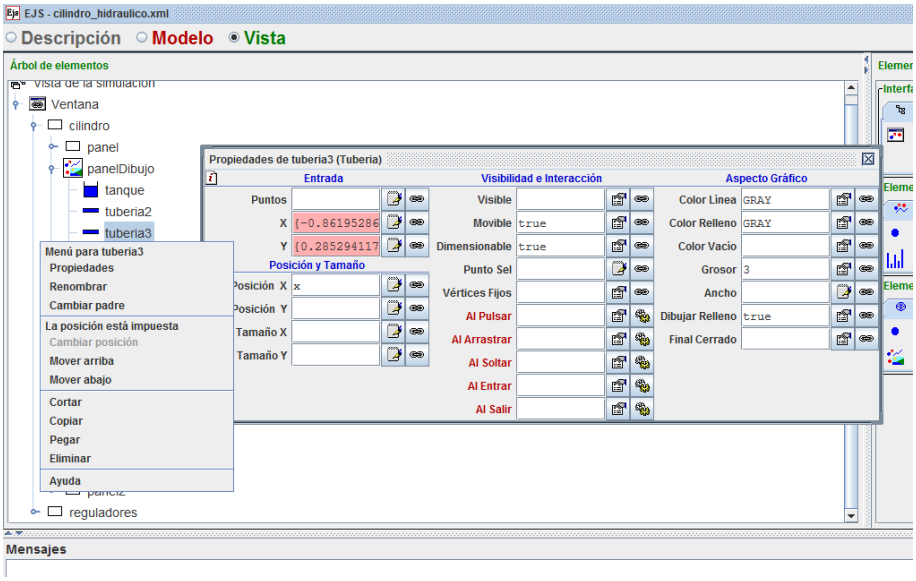

Figura 79: Menú *propiedades* de elemento *tuberia3*

Como ejemplo de texto interactivo se escoge *desplazamiento_1*, que informa de cuando el desplazamiento *x* alcanza un valor de 0.2 m *(ver figura 80).*

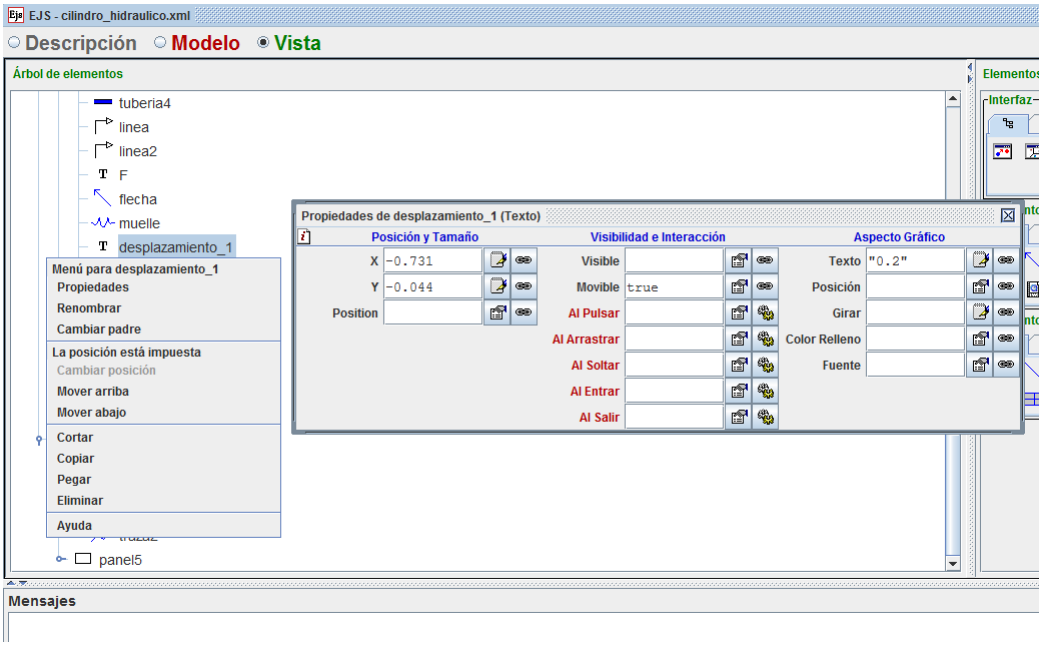

Figura 80: Menú *propiedades* de texto interactivo *desplazamiento_1*

El contenedor *panel* sigue una distribución *border*; este contenedor tiene dos hijos: *panel_deslizadores* en la posición de arriba y *panel_campos* en la posición centro *(ver figura 81).*

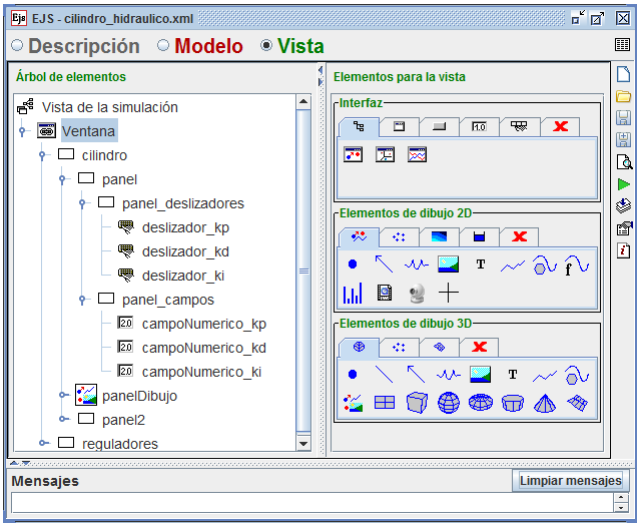

Figura 81: Composición contenedor *panel*

El contenedor *panel_deslizadores* sigue una distribución *rejilla* de 3 filas y una sola columna; los elementos contenidos son deslizadores que tienen como cometido variar el valor de *kp*, *kd* y *ki* y con ello estudiar el comportamiento de los distintos reguladores. Si se analiza el diseño del *deslizador_ ki* se obtiene que el valor mínimo que puede tomar *ki* es 0.01; el valor máximo es 10, el formato de presentación es decimal, la orientación del deslizador es horizontal, el color de la fuente negro, etc. (*ver figura 82*).

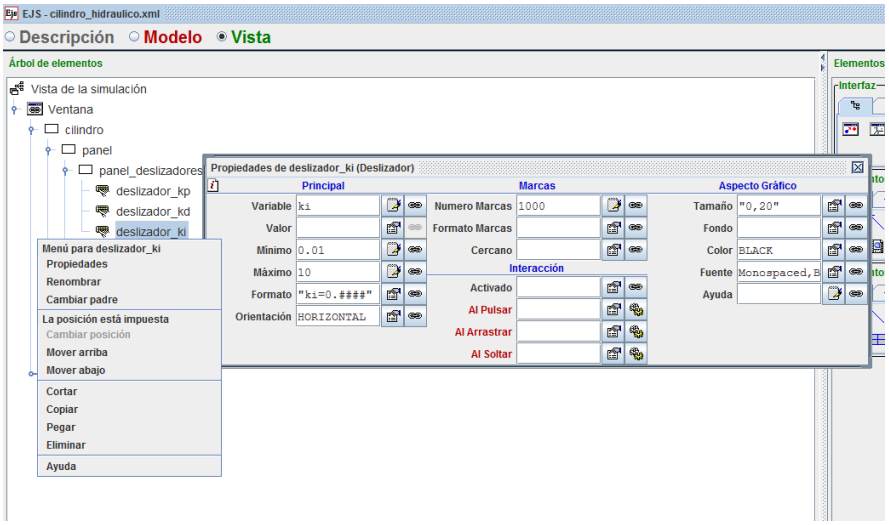

Figura 82: Menú *propiedades* de elemento *deslizador_ki*

El contenedor *panel_campos* alberga una serie de campos para mostrar y modificar el valor numérico de *kp*, *kd* y *ki*. Tomando como ejemplo a *compoNumerico_kp*; el valor inicial de

kp es uno y el formato de este campo es decimal (*ver figura 83*).

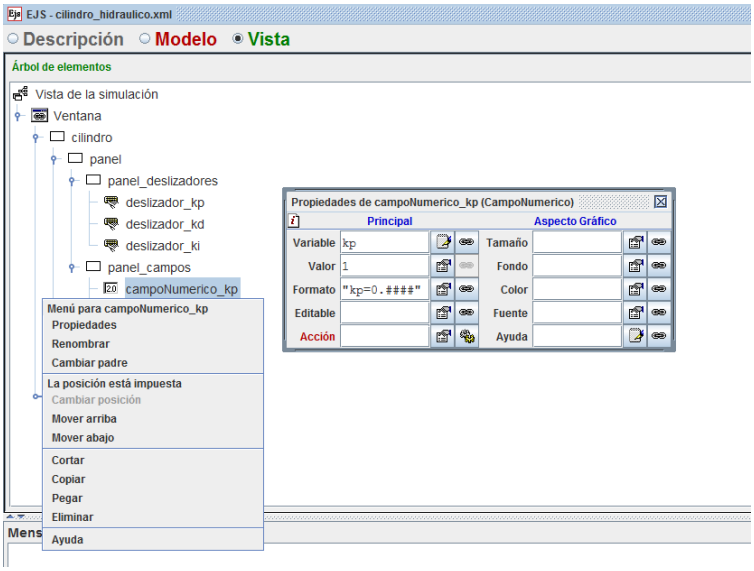

Figura 83: Menú *propiedades* de *campoNumerico_kp*

El contenedor *panel2* sigue una política de *rejilla* de una sola fila y dos columnas; contiene un campo que permite mostrar el valor que toma la variable *x* y una etiqueta que presenta el presente modelo. Dentro del menú *propiedades* de etiqueta podemos elegir las características de la fuente (*ver figuras 84 y 85*).

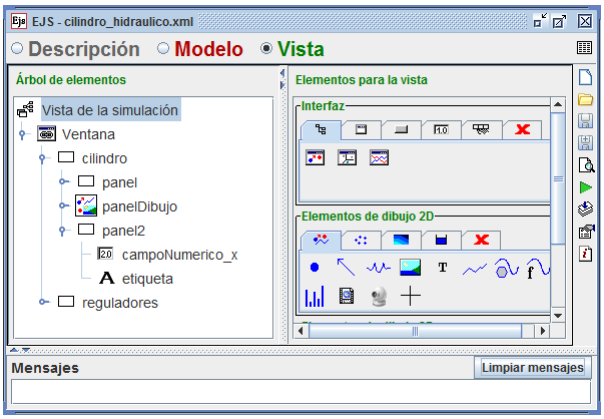

Figura 84: Elementos de contenedor *panel2*

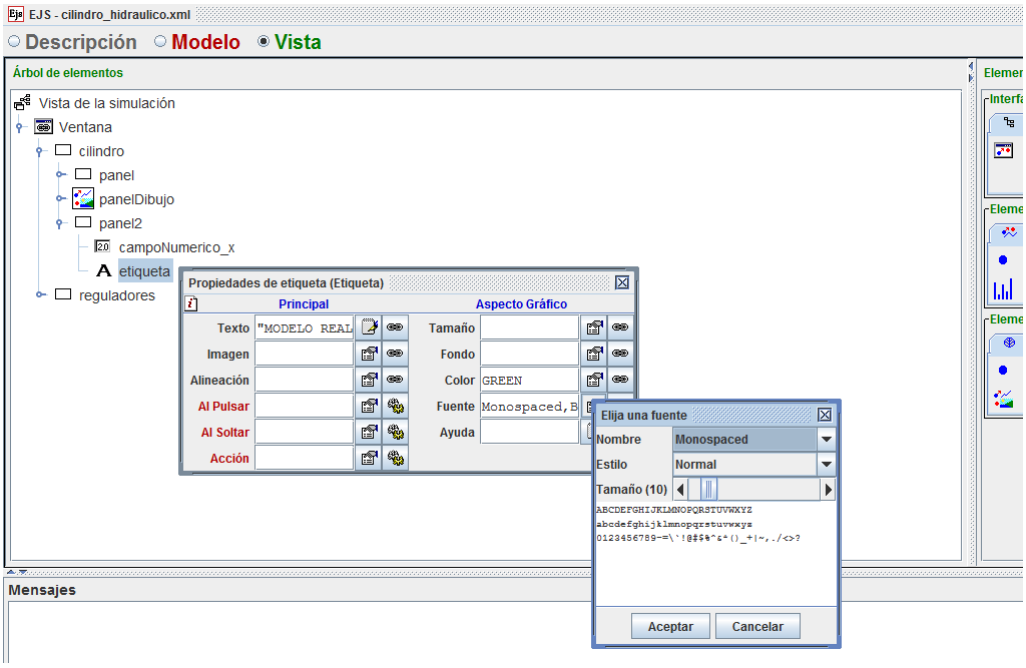

Figura 85: Menú *propiedades* de *etiqueta*

A continuación se estudia el contenedor *reguladores*, con distribución *VBOX* y padre de *panel5*, *panel6*, *panelConEjes* y *panelConEjes2 (ver figura 86).*

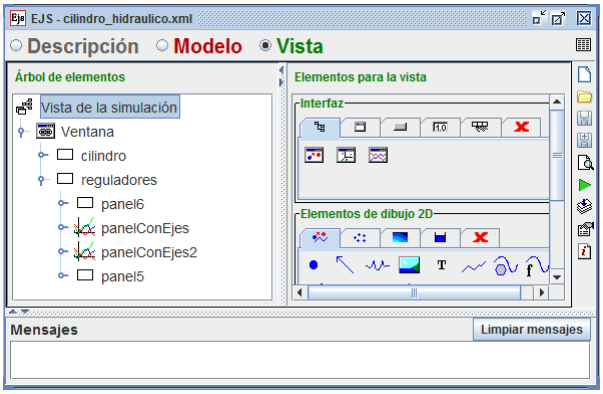

Figura 86: Composición contenedor *reguladores*

En el *panel6*, con una distribución *rejilla* de dos filas y una columna, se obtiene un deslizador que permite modificar el valor de *xref* y un campo para mostrar su valor numérico *(ver figura 87).*

Al variar el valor de *xref* el regulador envía la acción de control al actuador correspondiente y se consigue una posición aproximada a la buscada; esta aproximación es fruto

del tipo de regulador elegido y de las constantes asociadas a este regulador.

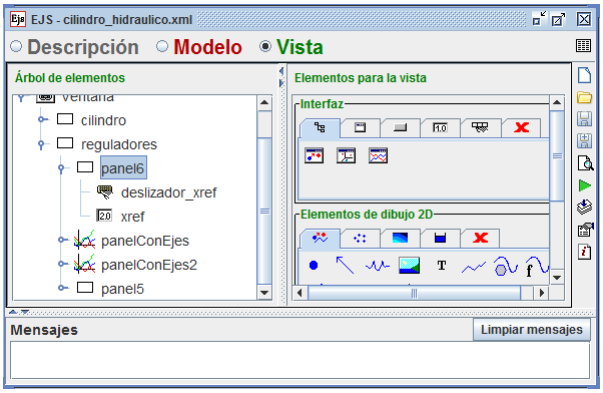

Figura 87: Elementos de *panel6*

El *panelConEjes* es un contenedor para dibujo bidimensional que incorpora un sistema de ejes cartesianos.

El título que recibe este panel es "posición VS tiempo", el eje x representa el tiempo y el eje y representa *x* y *xref* simultáneamente; ya que se van a añadir dos trazas para la misma gráfica. Estas trazas son *traza* y *traza_xref (ver figura 88).* El sentido de añadir la traza con el valor de *xref* es para poder observar cómo está funcionando el regulador.

El eje x alberga valores de tiempo comprendidos entre cero y cien segundos, mientras que los valores que se representan en el eje y van de cero a uno.

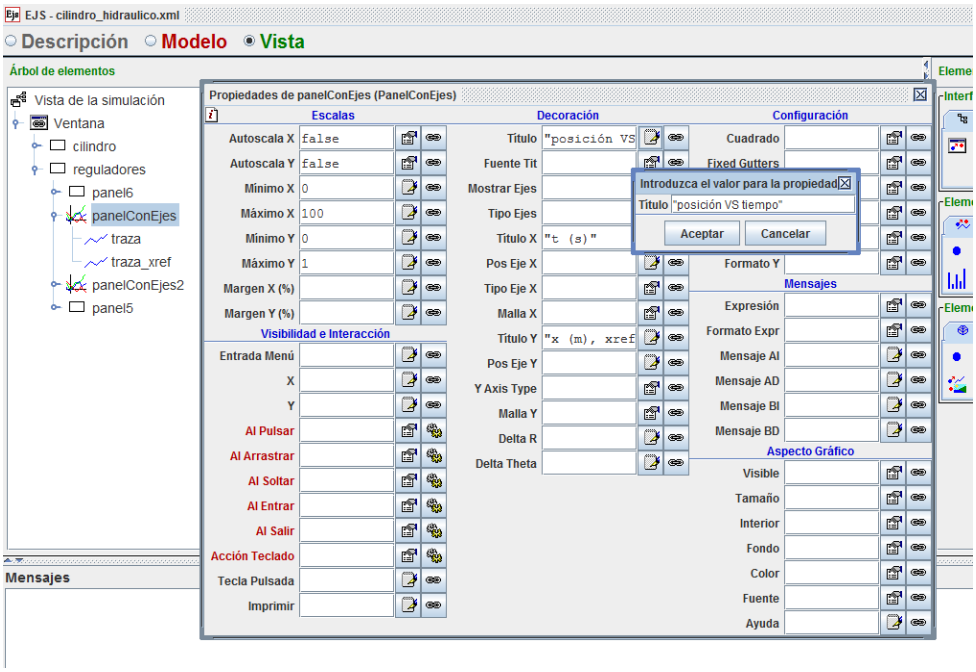

Figura 88: Menú *propiedades* de *panelConEjes*

El contenedor *panelConEjes2* alberga la traza que muestra la representación de la evolución de la acción de control respecto al tiempo (*ver figura 89*).

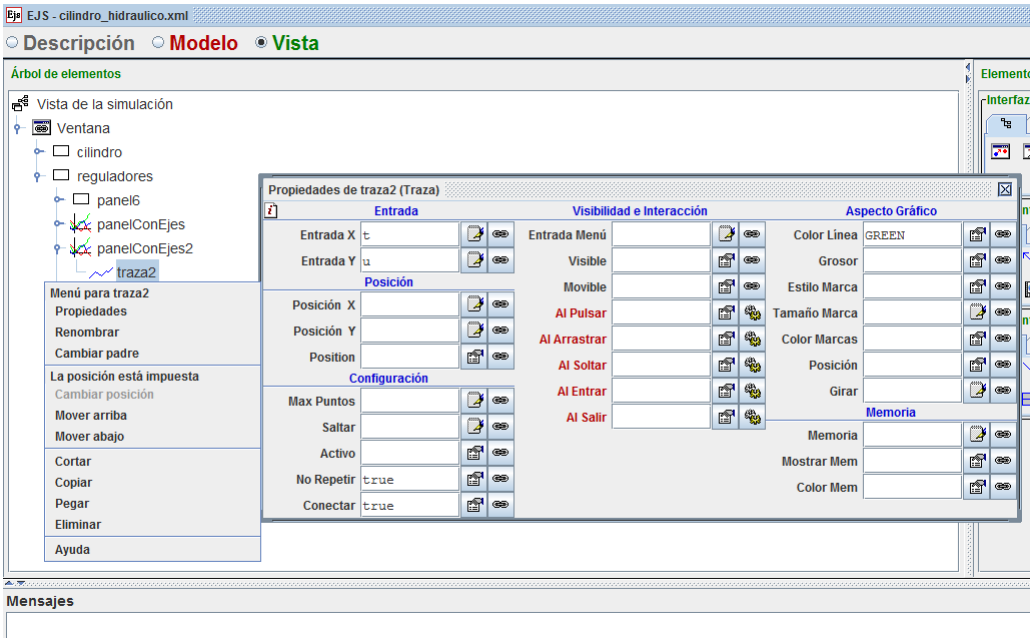

Figura 89: Menú *propiedades* de *traza2*

En el contenedor *panel5*, con distribución *rejilla* de una fila y tres columnas, se obtiene un panel de accionamiento con tres botones para controlar la ejecución de la simulación *(ver figura 90).*

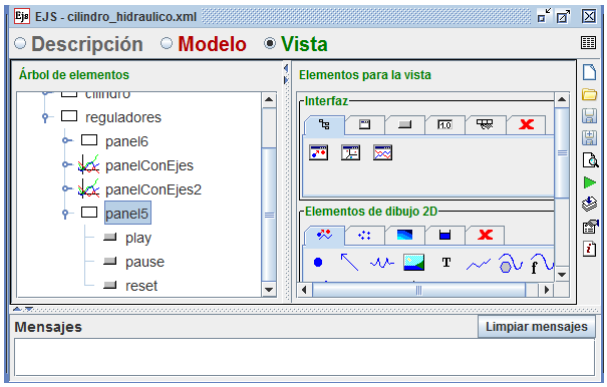

Figura 90: Elementos de contenedor *panel 5*

Al presionar el botón *play* se pone en marcha la evolución de la simulación; cuando la evolución esté parada *(ver figura 91).*

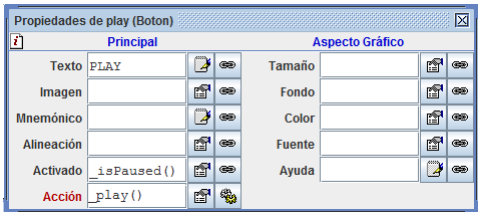

Figura 91: Menú *propiedades* de botón *play*

Cuando se pulsa el botón *pause* se detiene la evolución de la simulación siempre y cuando la evolución está en marcha *(ver figura 92).*

Figura 92: Menú *propiedades* de botón *pause*

Al pulsar *reset* se inicializan las variables al valor adjudicado en las tablas de variables (apartado *variables* dentro de panel *modelo*) y se limpia la vista *(ver figura 93).*

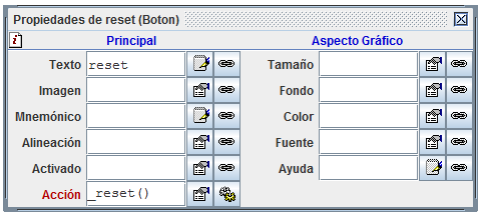

Figura 93: Menú *propiedades* de botón *reset*

Creada la estructura completa del árbol de elementos se muestra cómo queda la interfaz gráfica que aparece cuando se lanza la simulación *(ver figura 94).*

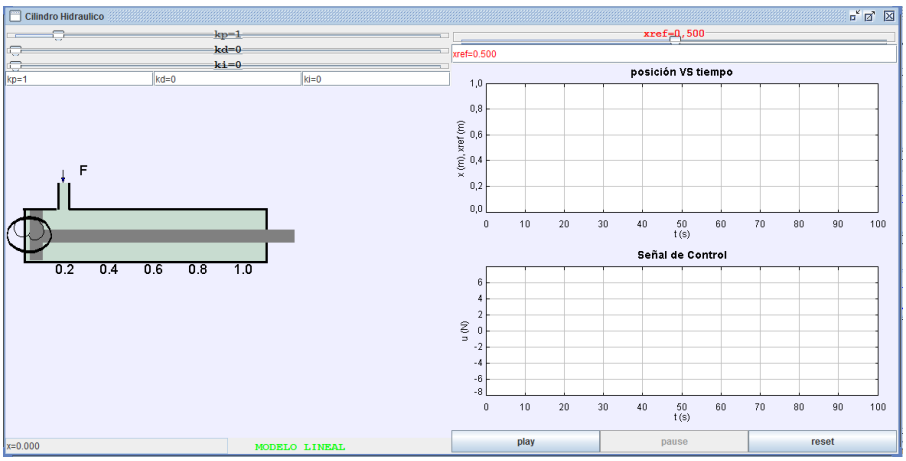

Figura 94: Vista de simulación sistema cilindro

6.2.4 Descripción

En este apartado se crea y edita el código HTML que aparece en el servidor adjunto a la simulación. Se hace una descripción sobre el sistema del cilindro. La *figura 95* muestra el panel *descripción* de *cilindro_hidraulico.xml*.

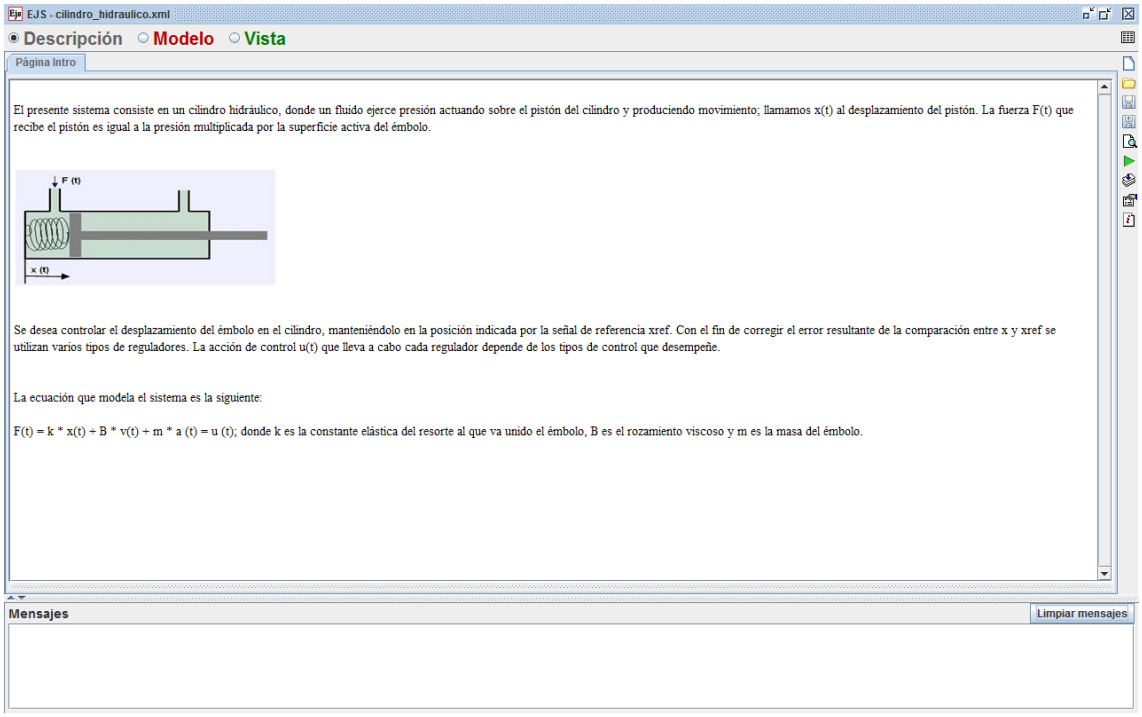

Figura 95: Panel *descripción* sistema cilindro

Capítulo 7

Casos prácticos

7.1 DEPÓSITO

Este apartado engloba un conjunto de pruebas que se llevan a cabo con el fin de averiguar el comportamiento óptimo del depósito ante un cambio en el valor de la señal de referencia. Al variar el valor de *href* el regulador envía la acción de control al actuador correspondiente y se consigue un nivel de líquido aproximado al buscado; esta aproximación depende del tipo de regulador elegido y de las constantes asociadas a este regulador. Se compara también el modelo real con el linealizado.

7.1.1 Reguladores P: Casos 1,2,3

Caso 1. Regulador P

Enunciado: Lanzar la simulación tomando como altura de referencia 0.5 m, como constante proporcional el valor 100, como constante derivativa cero y como constante integral cero. Para el modelo linealizado la altura del líquido en el punto de equilibrio es de 0.3 m. Observar la evolución de salida del sistema para el modelo real y el modelo linealizado.

href = 0.5 m h0 = 0.3 m kp = 100 kd = 0 ki = 0

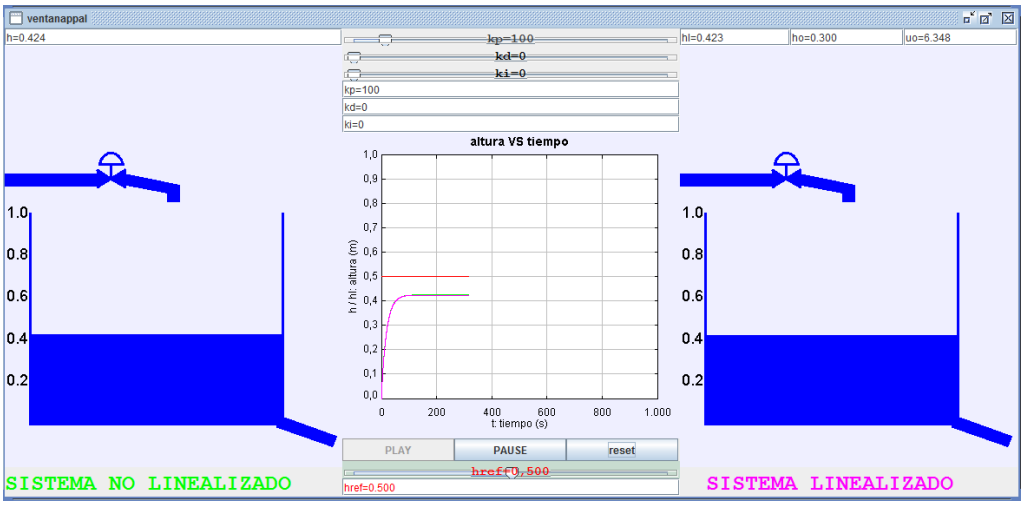

Figura 96: Vista de caso 1 sistema depósito

Con el regulador se busca conseguir una altura de 0.5 m. Con la acción de control propuesta los dos modelos presentan error. Se linealiza en torno a 0.3 m de altura; al no estar

muy alejado del valor de referencia esto implica que la respuesta de los dos sistemas es casi idéntica.

Caso 2. Regulador P

Enunciado: Cambiando el valor de la altura de referencia a 0.8 m y permaneciendo el resto de valores del ejercicio anterior observar la evolución de la salida de los dos modelos.

href = 0.8 m ***h0 = 0.3 m*** ***kp = 100*** ***kd = 0*** ***ki = 0***

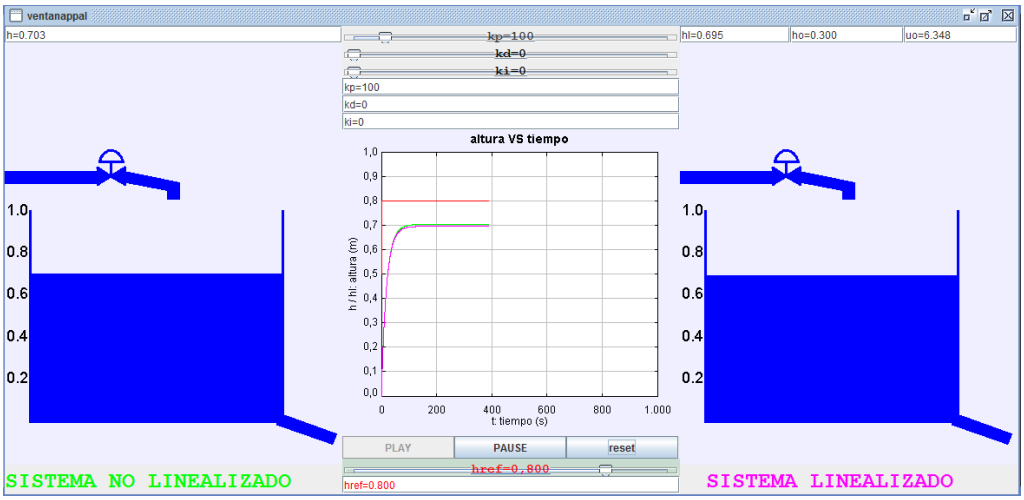

Figura 97: Vista de caso 2 sistema depósito

Teniendo en cuenta que la acción de control del regulador permanece constante las trazas del modelo lineal y el no lineal difieren algo más que en el ejercicio anterior, ya que el punto de equilibrio en torno al cual se linealiza dista más de la altura que se obtiene con el modelo real.

Caso 3. Regulador P

Enunciado: Modificando el valor de la constante proporcional a 1000 para intentar obtener una mejor respuesta y teniendo un valor de 0.2 m de la altura de referencia al inicio de la simulación para luego dar paso a una *href* de 0.5 m; observar la evolución de la salida de los dos modelos.

href = 0.2 - 0.5 m ***h0 = 0.3 m*** ***kp = 1000*** ***kd = 0*** ***ki = 0***

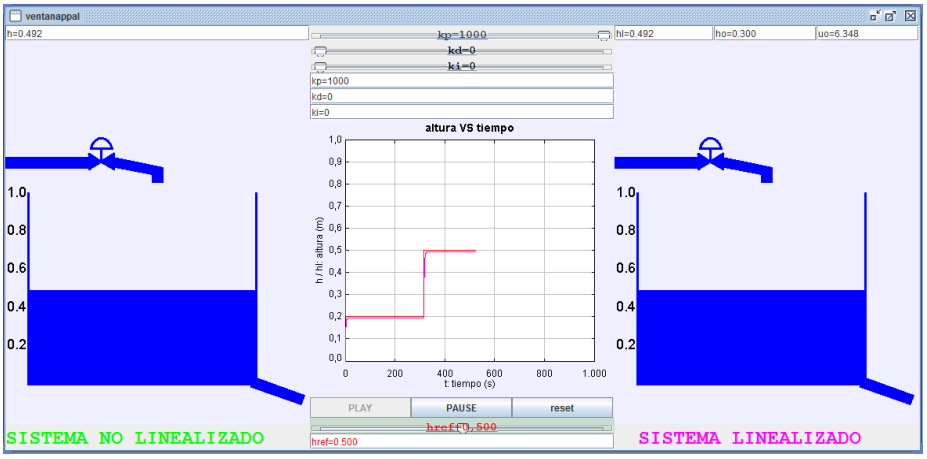

Figura 98: Vista de caso 3 sistema depósito

Con este regulador al aumentar el valor de *kp* se consigue que el error disminuya. Ambos modelos coinciden en su respuesta.

7.1.2 Reguladores PI: Casos 4,5

Caso 4. Regulador PI

Enunciado: Observar la evolución de la salida del sistema para el modelo real y el lineal con un regulador PI cuyas constantes proporcional e integral valen 100 y 10 respectivamente. El valor de *href* es 0.5 m y el de *h0* es 0.3 m.

href = 0.5 m h0 = 0.3 m kp = 100 kd = 0 ki = 10

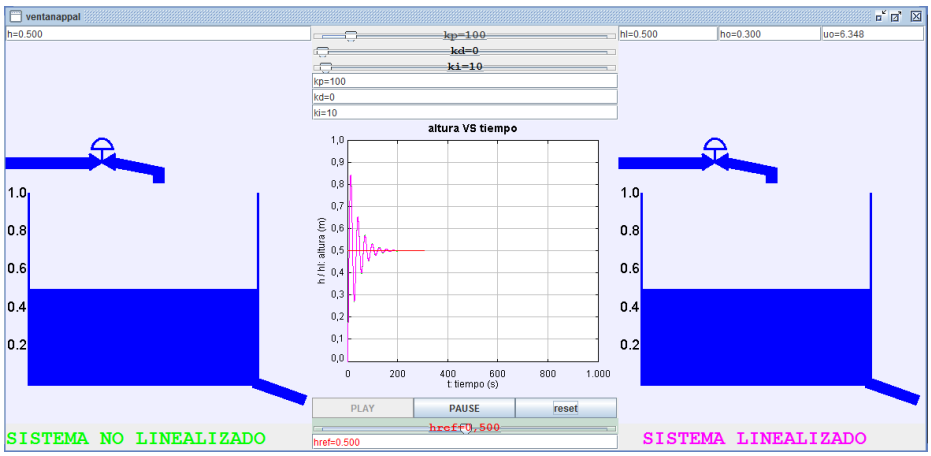

Figura 99: Vista de caso 4 sistema depósito

En la respuesta estática se obtiene un error nulo mientras que la respuesta dinámica de los dos modelos presenta mucha sobreoscilación.

Caso 5. Regulador PI

Enunciado: Para intentar corregir la sobreoscilación en la respuesta dinámica se disminuye el valor de *ki*, pasando a tomar el valor 1. La constante proporcional no se modifica. Observar la evolución de la salida del sistema para el modelo real y el lineal.

href = 0.5 m h0 = 0.3 m kp = 100 kd = 0 ki = 1

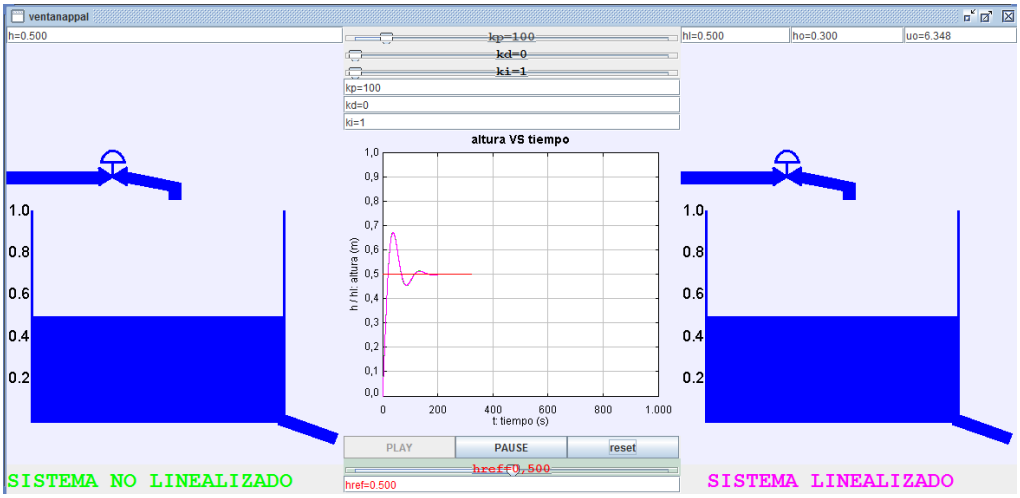

Figura 100: Vista de caso 5 sistema depósito

7.1.3 Reguladores PID: Casos 6,7

Caso 6. Regulador PID

Enunciado: Estudiar el uso de un regulador PID para obtener una mejor respuesta. Los valores de la constante proporcional, integral y derivativa son 100, 1 y 100 respectivamente. La altura de referencia continua con el valor 0.5 m y el valor de la altura en el punto de equilibrio en torno al cual se linealiza es 0.3 m. Observar la evolución de la salida del sistema para el modelo real y el lineal.

href = 0.5 m h0 = 0.3 m kp = 100 kd = 100 ki = 1

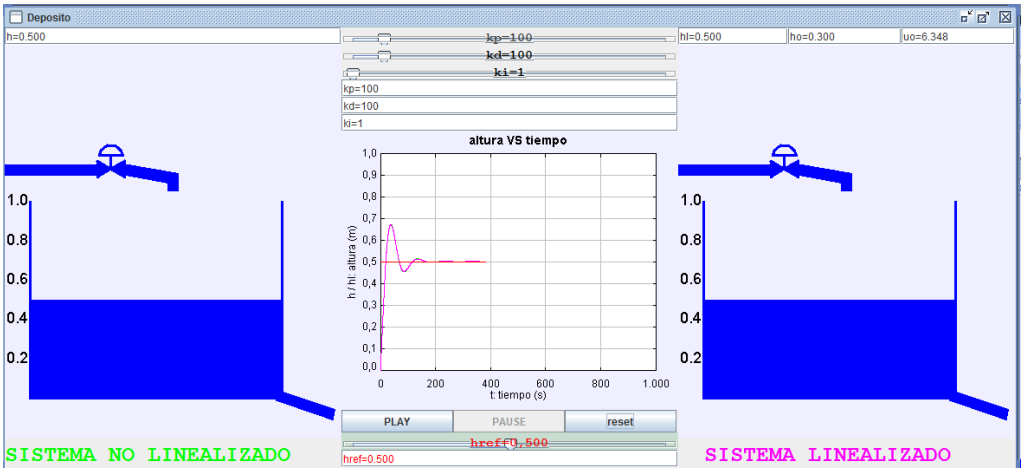

Figura 101: Vista de caso 6 sistema depósito

Se obtiene una respuesta más rápida que en el caso 5, con poco pico de oscilación y sin error.

Caso 7. Regulador PID

Enunciado: Continuando con el mismo regulador que en el caso anterior, teniendo una altura del líquido en el punto de equilibrio de 0.3 m y variando la señal de referencia de 0.2 a 0.8 m; observar la evolución de la salida del sistema para el modelo real y el lineal.

href = 0.2 -0.8 m ***h0 = 0.3 m*** ***kp = 100*** ***kd = 100*** ***ki = 1***

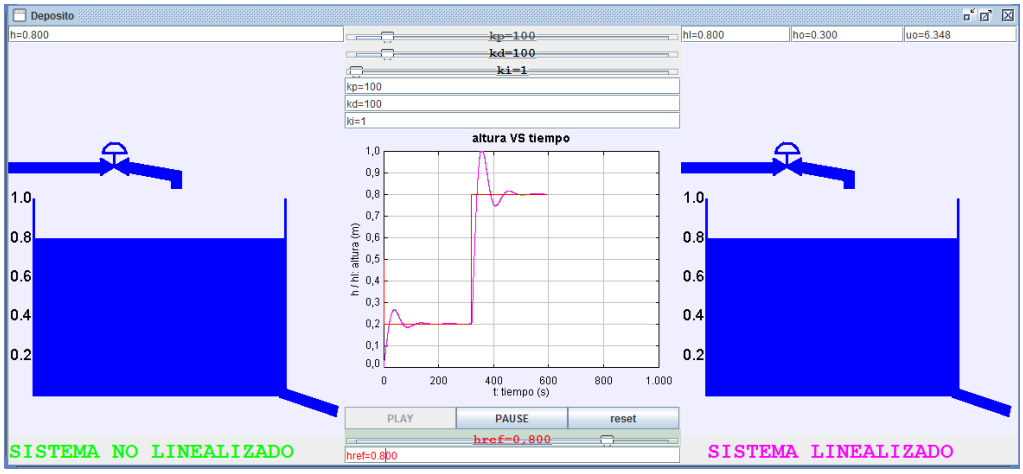

Figura 102: Vista de caso 7 sistema depósito

Se observa un buen comportamiento de parte de los dos modelos. La respuesta es idéntica aún estando lejos del punto de equilibrio. Este regulador es el que mejor ajustado está de todos los casos estudiados.

Conclusión: El regulador hace que sea menor la diferencia entre el modelo lineal y no lineal ya que tiende a reducir el error y hace que se tienda a aproximar la respuesta al valor de *set-point* deseado.

7.2 AVIÓN

En este punto se lleva a cabo el descriptivo de una serie de pruebas que se realizan con el fin de averiguar la mejor respuesta del cilindro hidráulico ante un cambio en el valor de *set_point*. Esta respuesta depende del tipo de regulador y de las constantes asociadas a él.

7.2.1 Reguladores PI (de partida): Caso 1

Caso 1. Regulador PI

Enunciado: Tomando el valor 0.5 u como altura de referencia, como constante proporcional el valor 1, como constante derivativa cero y como constante integral 0.01; lanzar la simulación y observar la evolución de la altura y del ángulo del avión respecto al eje horizontal.

href = 0.5 m ***kp = 1*** ***kd = 0*** ***ki = 0.01***

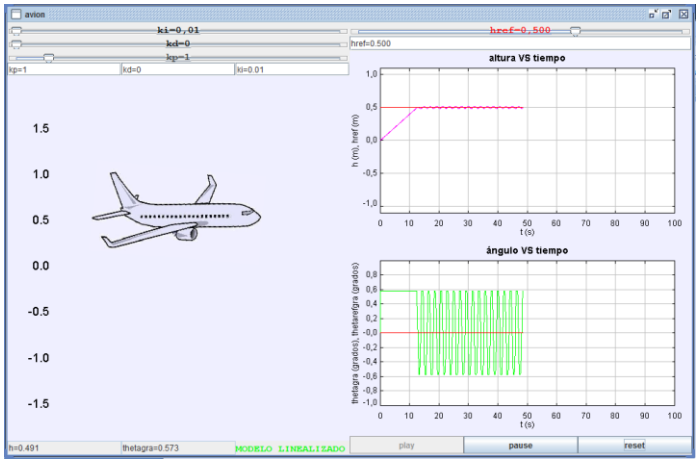

Figura 103: Vista de caso 1 sistema avión

La altura inicial del avión es 0 u, al introducir el valor de *set-point* el avión sube y permanece oscilando en altura y ángulo.

7.2.2 Reguladores PID: Caso 2

Caso 2. Regulador PID

Enunciado: Se modifica la altura de referencia a un valor de -0.5 u, las constantes proporcional e integral continúan igual que en el ejercicio anterior y se añade la acción derivativa, tomando *kd* el valor 10; lanzar la simulación. ¿Es efectiva la acción derivativa?

href = -0.5 u ***kp = 1*** ***kd = 10*** ***ki = 0.01***

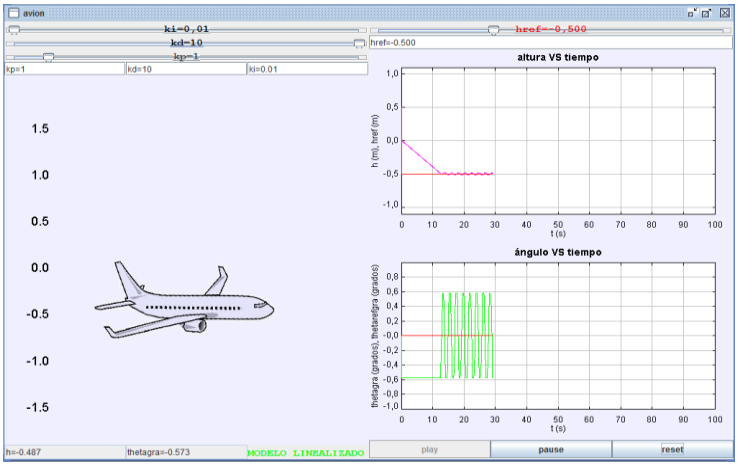

Figura 104: Vista de caso 2 sistema avión

La acción derivativa no es efectiva porque en la respuesta dinámica el ángulo se satura en los cambios, por lo que es constante y su derivada nula. En el permanente no tiene prácticamente influencia.

7.2.3 Reguladores PI: Casos 3,4,5,6,7

Caso 3. Regulador PI

Enunciado: Manteniendo la altura de referencia al valor -0.5 u, la constante derivativa se anula, la integral permanece como en el ejercicio anterior y se modifica el valor de *kp* a 5; lanzar la simulación y observar la evolución de la altura y del ángulo del avión respecto al eje horizontal.

href = -0.5 u ***kp = 5*** ***kd = 0*** ***ki = 0.01***

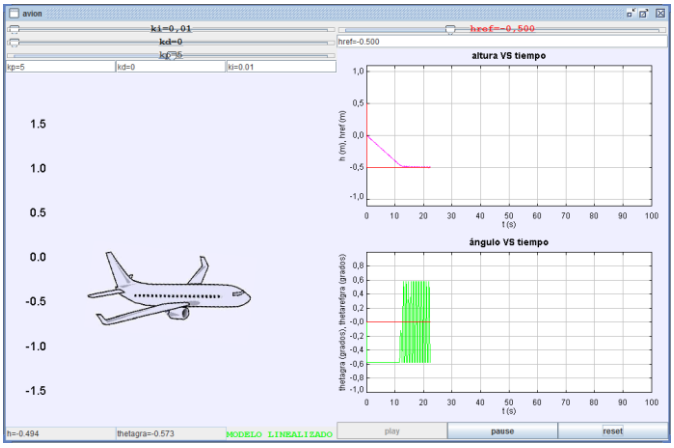

Figura 105: Vista de caso 3 sistema avión

El error en altura disminuye, pero las oscilaciones en ángulo aumentan. Tendremos menos confort.

Caso 4. Regulador PI

Enunciado: Continuando a la altura de referencia de valor -0.5 u, las constante integral permanece como en el ejercicio anterior y se modifica el valor de *kp* a 0.1; lanzar la simulación

y observar la evolución de la altura y del ángulo del avión respecto al eje horizontal.

href = -0.5 u ***kp = 0.1*** ***kd = 0*** ***ki = 0.01***

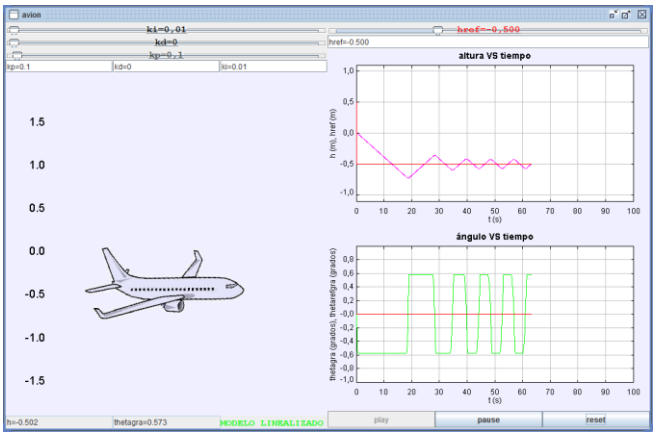

Figura 106: Vista de caso 4 sistema avión

El error en altura aumenta, en cambio las oscilaciones en ángulo disminuyen. Existe un mayor confort.

Caso 5. Regulador PI

Enunciado: Manteniendo la altura de referencia al valor -0.5 u y la constante proporcional al valor 0.1, se modifica el valor de *ki* a 0.5; lanzar la simulación y observar la evolución de la altura y del ángulo del avión respecto al eje horizontal.

href = -0.5 u ***kp = 0.1*** ***kd = 0*** ***ki = 0.5***

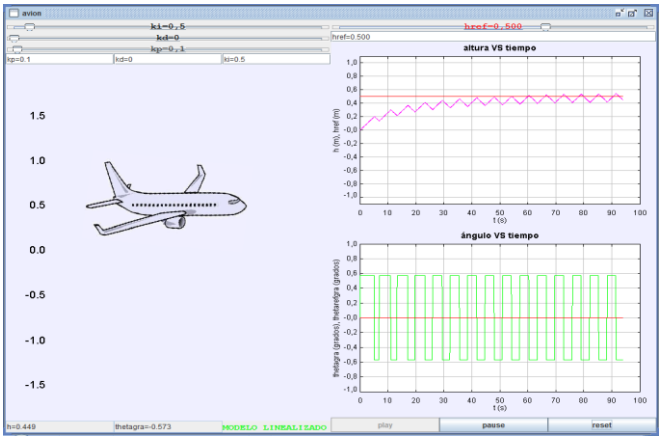

Figura 107: Inicio de la simulación en el caso 5 sistema avión

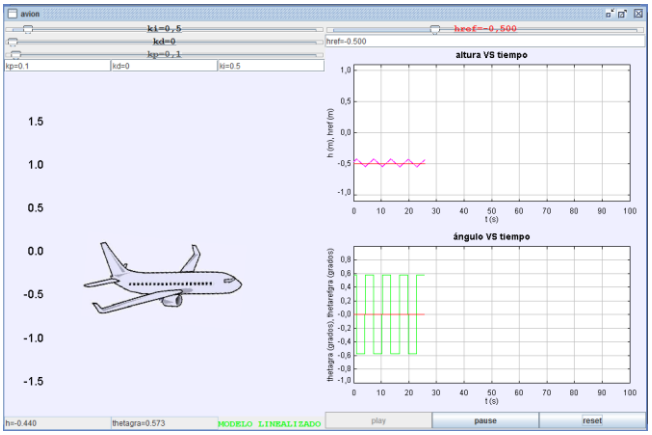

Figura 108: Régimen permanente en la simulación de caso 5 sistema avión

En el transitorio se observa que el avión presenta un tiempo de establecimiento más largo.

Respecto al caso 4: una vez alcanzado el régimen permanente el error en altura se mantiene pero aumentan las oscilaciones del avión.

Caso 6. Regulador PI

Enunciado: Permaneciendo la altura de referencia al valor -0.5 u y la constante de proporcionalidad a 0.1, se modifica el valor de *ki* a 6; lanzar la simulación y observar la evolución de la altura y del ángulo del avión respecto al eje horizontal.

href = -0.5 u ***kp = 0.1*** ***kd = 0*** ***ki = 6***

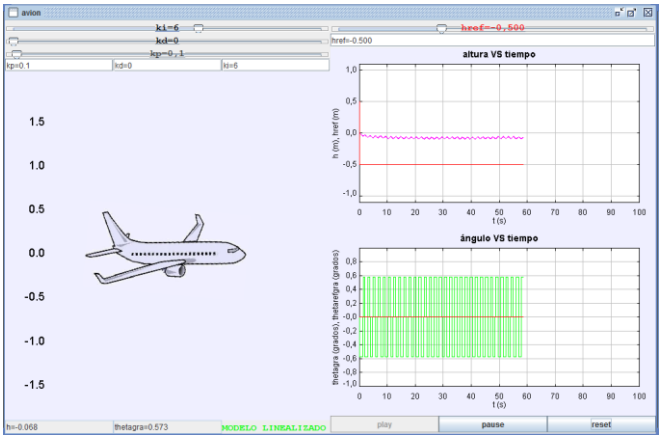

Figura 109: Vista de caso 6 sistema avión

Con este regulador el resultado que se obtiene es pésimo, ya que el valor de *ki* es demasiado grande y el valor de *kp* pequeño.

Caso 7. Regulador PI

Enunciado: Cambiando la altura de referencia al valor 0.5 u y tomando las constantes proporcional e integral los valores 1 y 6 respectivamente; lanzar la simulación y observar la evolución de la altura y del ángulo del avión respecto al eje horizontal.

href = 0.5 u ***kp = 1*** ***kd = 0*** ***ki = 6***

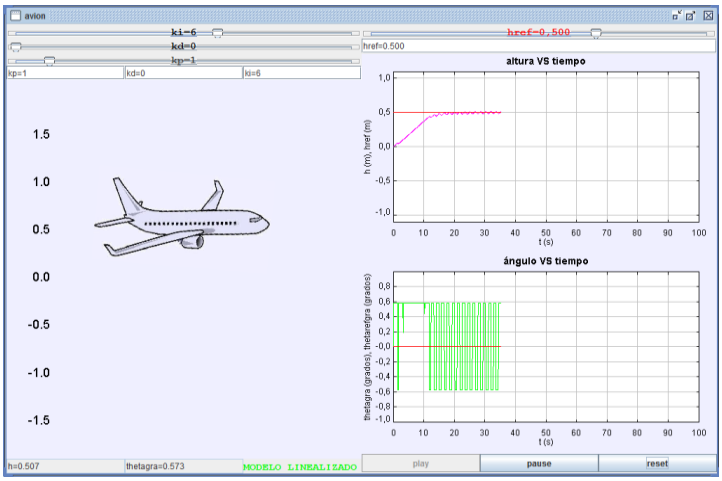

Figura 110: Vista de caso 7 sistema avión

Al aumentar el valor de *kp* la respuesta dada por este regulador es aceptable. El avión inicialmente tarda en reaccionar ya que aparecen oscilaciones en el transitorio debido a la acción integral grande.

Conclusión: Para un mayor confort de los pasajeros la constante proporcional *kp* debe ser grande (del orden de 5) en el transitorio y pequeña (en torno a 0.2) en el régimen permanente para evitar las oscilaciones en ángulo. *ki* debe ser lo más pequeña posible sin llegar a ser cero para evitar que el sistema se haga inestable. *kd* no afecta pues al estar el ángulo saturado en la respuesta dinámica, es constante y la acción derivativa se anula.

Dependiendo del objetivo de control es mejor una opción u otra. Si lo que se busca es la comodidad de los pasajeros, dando prioridad a la respuesta estática, el caso óptimo es el número cuatro.

7.3 CILINDRO HIDRÁULICO

Este apartado describe un conjunto de pruebas que se llevan a cabo con el fin de averiguar el comportamiento óptimo del cilindro hidráulico ante un cambio en el valor de la señal de referencia. El resultado depende del tipo de regulador y de las constantes asociadas a él.

7.3.1 Reguladores P: Casos 1,2,3

Caso 1. Regulador P

Enunciado: Tomar como posición de referencia 0.5 m, como constante proporcional el valor 1, como constante derivativa cero y como constante integral cero. Observar la evolución de la salida y la señal de control.

xref = 0.5 m ***kp = 1*** ***kd = 0*** ***ki = 0***

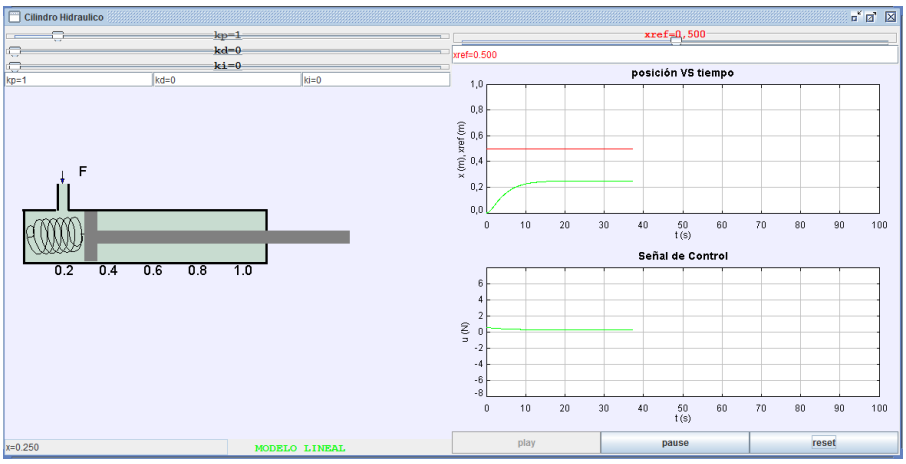

Figura 111: Vista de caso 1 sistema cilindro hidráulico

El sistema es lento y sobreamortiguado, obteniéndose un error de 0.250 m. La acción de control, la fuerza, comienza con un pico y va disminuyendo hasta que permanece constante para mantener la posición del émbolo, evitando que el muelle le haga retroceder.

Caso 2. Regulador P

Enunciado: Se toma como posición de referencia la misma que en el ejercicio anterior y se cambia el valor de la constante proporcional para que tome el valor 10. Observar la evolución de la salida y la señal de control.

xref = 0.5 m ***kp = 10*** ***kd = 0*** ***ki = 0***

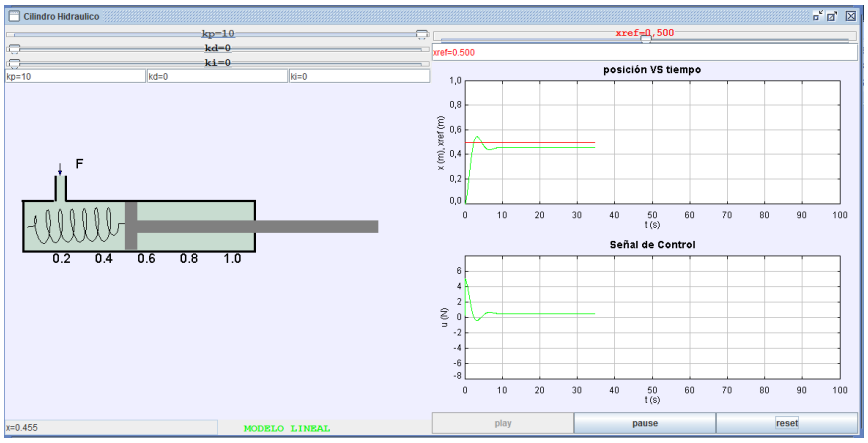

Figura 112: Vista de caso 2 sistema cilindro hidráulico

Para los valores con que hemos lanzado la simulación el sistema responde más rápido que en el caso anterior y en su respuesta presenta sobreoscilación. El error disminuye, ya que se obtiene un valor de *x* de 0.455 m; es decir, un error de 0.045 m.

La acción de control comienza con un pico pronunciado para intentar adaptarse al cambio y luego permanecer constante.

Caso 3. Regulador P

Enunciado: Cambiar la posición de referencia a 0.2 m y posteriormente retomar el valor de 0,5 m; se mantiene el valor 10 de la constante proporcional. Observar la evolución de la salida y la señal de control.

xref = 0.2 -0.5 m ***kp = 10*** ***kd = 0*** ***ki = 0***

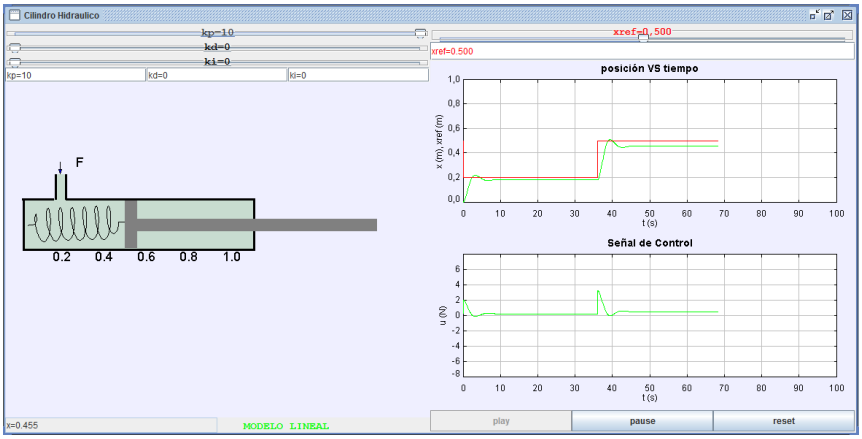

Figura 113: Vista de caso 3 sistema cilindro hidráulico

La salida es rápida, con sobreoscilación y presentando siempre un error en el régimen permanente.

Se observa que la señal de control intenta actuar para compensar los cambios. Al principio el cambio que se produce es brusco, mientras que después intenta contrarrestar el error que se va produciendo.

7.3.2 Reguladores PI: Casos 4,5,6

Caso 4. Regulador PI

Enunciado: Manteniendo constante el valor de la señal de referencia a 0.5 m; dar a *ki* el valor 1; y el valor 10 a *kp*. Observar la evolución de la salida y la señal de control.

xref = 0.5 m ***kp = 10*** ***kd = 0*** ***ki = 1***

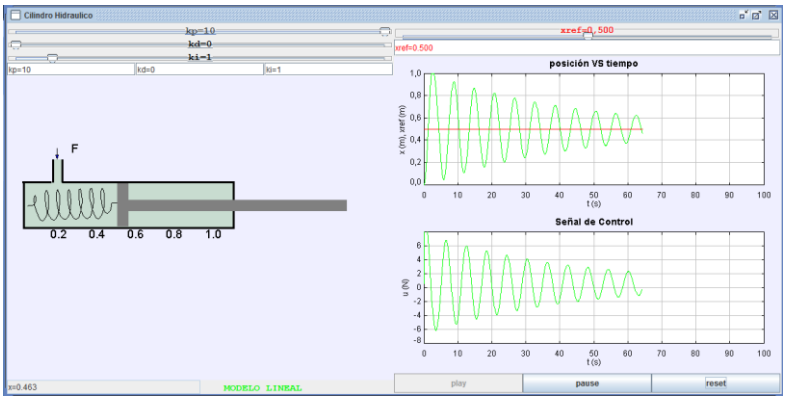

Figura 114: Inicio de la simulación de caso 4 sistema cilindro hidráulico

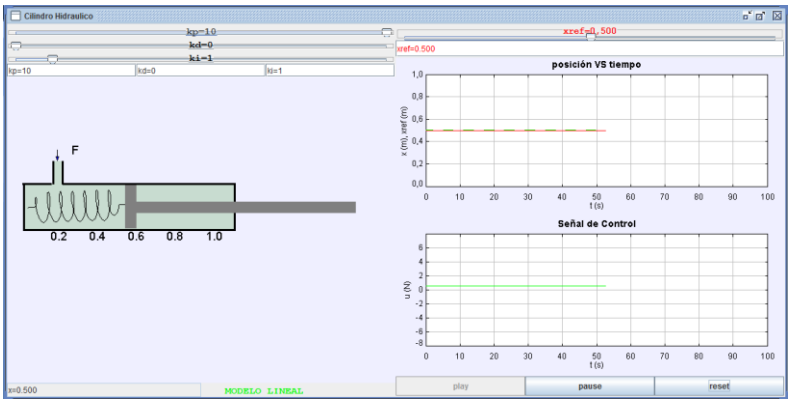

Figura 115: Fin de la simulación de caso 4 sistema cilindro hidráulico

El muelle oscila; sus oscilaciones van disminuyendo lentamente hasta que el error llega a cero.

La señal de la acción de control también oscila intentando contrarrestar el efecto del error.

Caso 5. Regulador PI

Enunciado: Al inicio hacer la simulación con una señal de referencia de 0.2 m; luego cambiar a 0.5 m. Mantener la constante proporcional con el valor 10 y cambiar la constante de integración a 0.1. Observar la evolución de la salida y la señal de control.

xref = 0.2 - 0.5 m ***kp = 10*** ***kd = 0*** ***ki = 0.1***

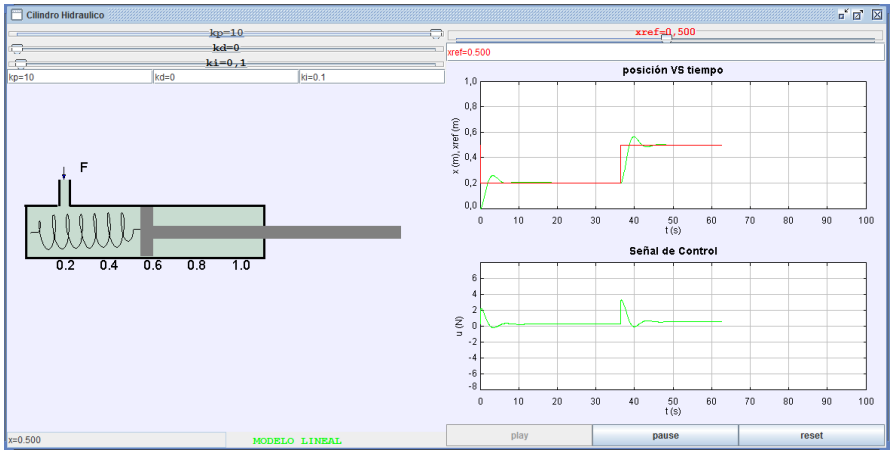

Figura 116: Vista de caso 5 sistema cilindro hidráulico

La respuesta es rápida y con poca sobreoscilación; se consigue anular el error en régimen permanente.

Caso 6. Regulador PI

Enunciado: Permaneciendo la señal de referencia a un valor de 0.5 m, teniendo un valor de *ki* igual a 0.2 y variando la constante proporcional de 1 a 6; observar la evolución de la salida y la señal de control.

xref = 0.5 m ***kp = 1 - 6*** ***kd = 0*** ***ki = 0.2***

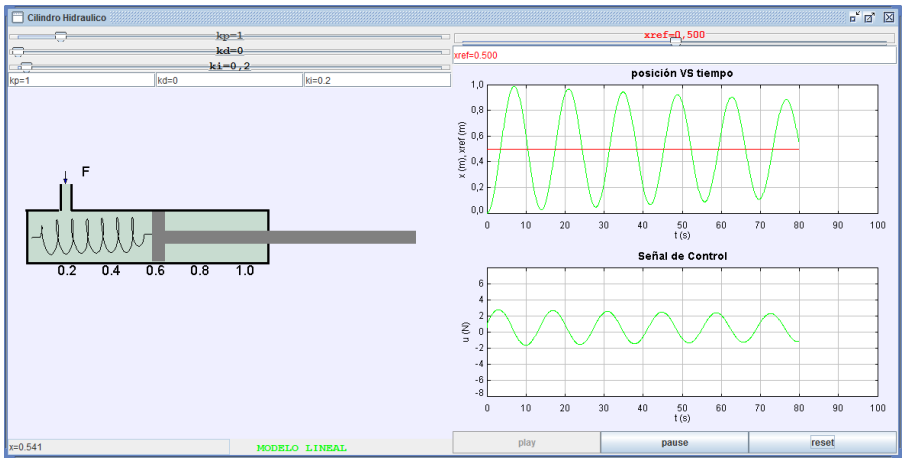

Figura 117: Simulación de caso 6 sistema cilindro hidráulico para kp igual a uno

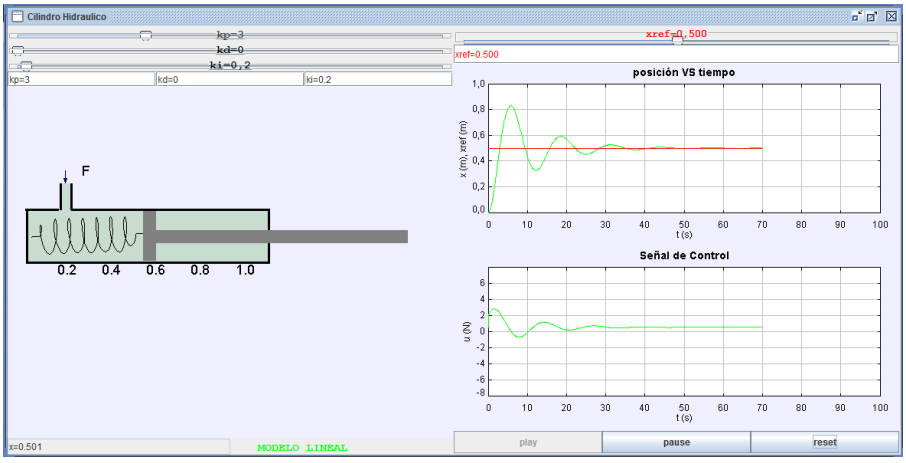

Figura 118: Simulación de caso 6 sistema cilindro hidráulico para *kp* igual a tres

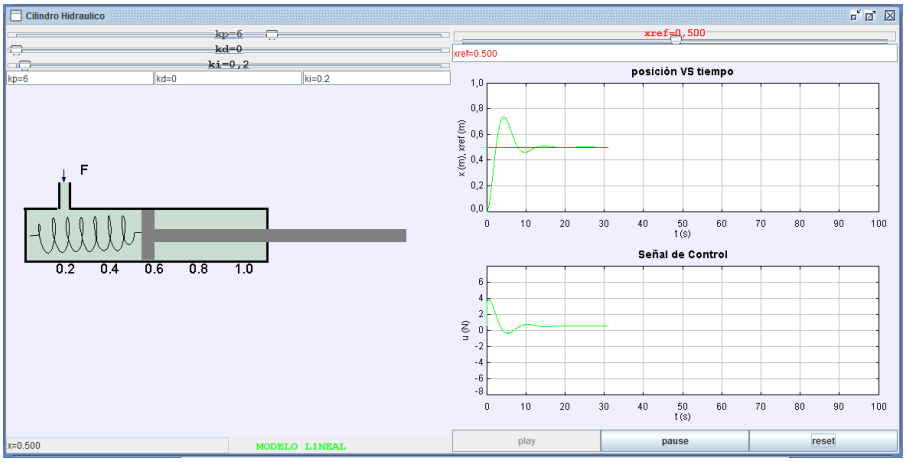

Figura 119: Simulación de caso 6 sistema cilindro hidráulico para kp igual a seis

Se puede observar que a medida que aumenta el valor de *kp* el sistema alcanza de manera más rápida el régimen permanente.

7.3.3 Reguladores PD: Casos 7,8

Caso 7. Regulador PD

Enunciado: Simular un regulador PD, con una constante de proporcionalidad de 1, una constante derivativa de 10 y cambiando la señal de referencia de 0.1 a 0.8 m. Observar la evolución de la salida y la señal de control.

xref = 0.1 – 0.8 m ***kp = 1*** ***kd = 10*** ***ki = 0***

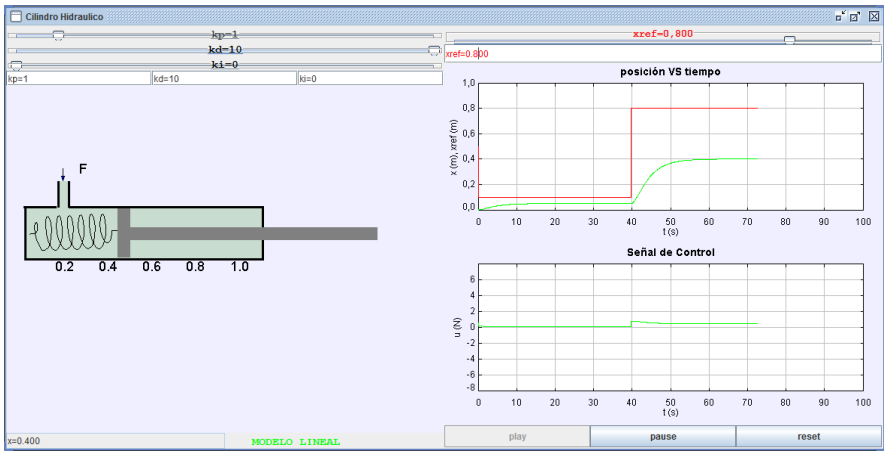

Figura 120: Vista de caso 7 sistema cilindro hidráulico

El sistema responde muy rápido inicialmente pero presenta un gran error.

Caso 8. Regulador PD

Enunciado: Simular un regulador con acciones proporcional y derivativa, con una constante de proporcionalidad de valor 10, manteniendo el valor de la constante derivativa a 10 y cambiando la señal de referencia de 0.1 a 0.8 m. Observar la evolución de la salida y la señal de control.

xref = 0.1 – 0.8 m ***kp = 10*** ***kd = 10*** ***ki = 0***

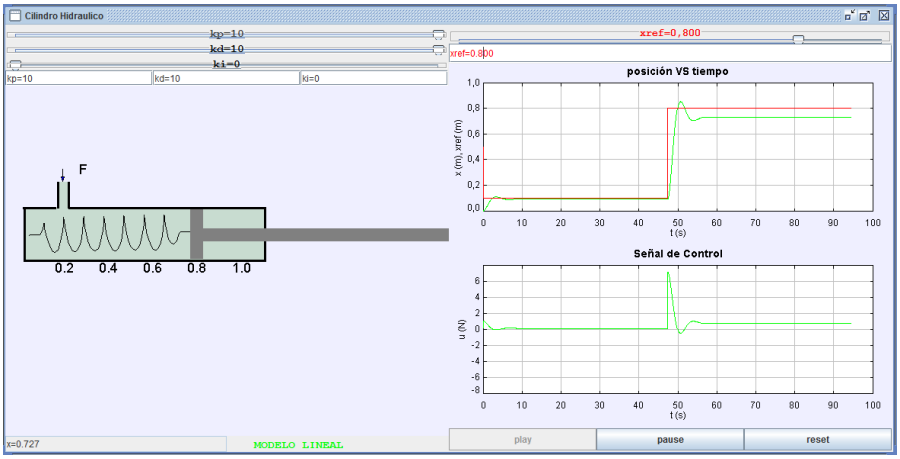

Figura 121: Vista de caso 8 sistema cilindro hidráulico

Al aumentar el valor de *kp* el sistema reacciona de manera más rápida que en el caso anterior y el error que se obtiene es menor aunque grande.

7.3.4 Reguladores PID: Caso 9

Caso 9. Regulador PID

Enunciado: Simular un regulador PID, con una constante de proporcionalidad de valor 10, con un valor de la constante derivativa de 10 y un valor de *ki* de 0.1. Cambiar la señal de referencia de 0.1 a 0.8 m y observar la evolución de la salida y la señal de control.

xref = 0.1 – 0.8 m ***kp = 10*** ***kd = 10*** ***ki = 0.1***

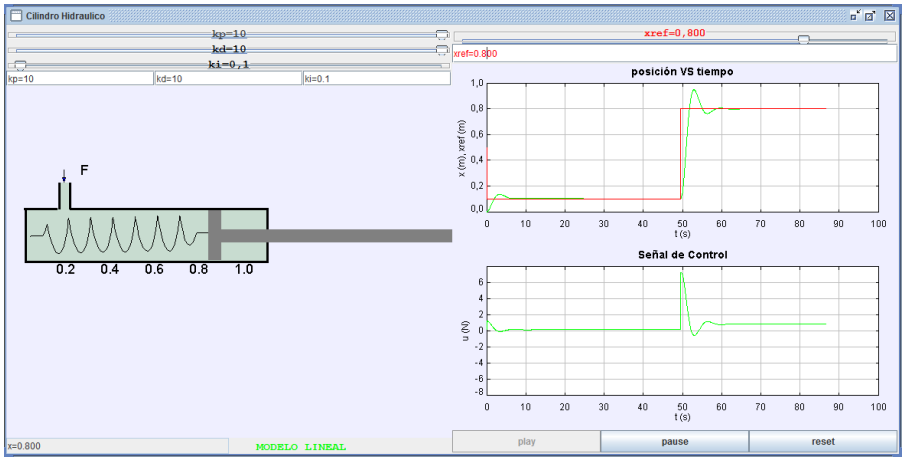

Figura 122: Vista de caso 9 sistema cilindro hidráulico

Con estos valores de las constantes obtenemos una respuesta rápida y un error nulo, aunque a consta de un pico inicial grande en la señal de control.

Por todo lo que se ha comentado en cada caso práctico se concluye que el quinto es la mejor opción de las estudiadas.

Capítulo 8

Conclusiones y trabajos futuros

8.1. CONCLUSIONES

En este proyecto se han implementado tres sistemas con el objetivo de que los reguladores PID sean estudiados de forma práctica por parte de los alumnos, ejecutando simulaciones remotas mediante un navegador Web. También se han añadido mejoras importantes tanto en la representación gráfica, realizada en los sistemas desarrollados previamente, como en la reestructuración del código. A su vez se han desarrollado tres prácticas que pueden servir de soporte en el aprendizaje de la asignatura *Señales y Sistemas* y de futuras asignaturas de Ingeniería de Control, presentándose tres ejercicios a modo de ejemplo.

Se ha comprobado que la implantación de laboratorios remotos y virtuales ofrece nuevas posibilidades en la educación al aumentar las oportunidades de experimentación. Dichos laboratorios tienen como ventaja los horarios de experimentación, ya que los sistemas simulados de los experimentos están disponibles a cualquier hora y para cualquier persona. Además de ampliar la disponibilidad temporal permiten que no sea necesario la presencia física en los laboratorios para realizar las prácticas, lo cual puede ser interesante, por ejemplo, para personas que compaginan estudios con trabajo o para alumnos con discapacidades. Hay que destacar también que su uso no aporta exclusivamente capacidades y conocimientos de la propia asignatura, sino que además proporciona a los alumnos experiencia en el uso de las TIC o en el desarrollo de trabajos en grupo de manera virtual.

Las simulaciones han podido confeccionarse con éxito pero cabe decir que *EJS*, si bien ha resultado ser una herramienta eficiente y cómoda, presenta pequeñas deficiencias. La principal de ellas es la existencia de numerosas limitaciones a la hora de escribir las ecuaciones y expresiones matemáticas, lo cual obliga en numerosas ocasiones a tener que buscar soluciones alternativas para introducir las fórmulas deseadas en los modelos de las simulaciones. Este defecto cobra mayor importancia cuanto más complejo es el modelo matemático a implantar.

Por último debo decir que, aunque el uso de los laboratorios virtuales posee muchas ventajas tal y como se ha detallado previamente, en algunas ocasiones se requiere de los laboratorios reales para un correcto entendimiento del caso práctico (resolver problemáticas específicas de la interacción directa con los equipos como son la puesta a punto de los mismos, el cableado, etc.). Sólo de esta manera se puede alcanzar la meta pedagógica de que los alumnos aprendan o entiendan conceptos acerca de los sistemas de forma virtual antes de llegar al laboratorio y así aprovechar al máximo el tiempo dedicado a la práctica presencial.

8.2. TRABAJOS FUTUROS

Siguiendo con el proceso de implantación de laboratorios remotos y virtuales, con este proyecto se pretende asentar la base de futuras herramientas del mismo tipo o posibles ampliaciones y mejoras de ésta misma.

Entre estas mejoras y herramientas nuevas, podrían realizarse:

- Nuevas prácticas simuladas para la asignatura Señales y Sistemas
- Prácticas virtuales para otras asignaturas del Departamento de Ingeniería de Sistemas y Automática y para otros campos del conocimiento
- Dar el salto a los laboratorios remotos, intentando conectar el hardware a herramientas informáticas que permitan al alumno hacer prácticas reales con los equipos de la Universidad desde su casa.

.

Bibliografía

[1] Creación de simulaciones interactivas en Java. Aplicación a la enseñanza de la Física. Francisco Esquembre. Pearson Prentice Hall.

[2] Desarrollo de prácticas remotas virtuales de Señales y Sistemas mediante EJS. Carlos Villa Carmona. Universidad Carlos III de Madrid.

[3] Examen de Señales y Sistemas. Febrero del 2005.
Universidad Carlos III de Madrid.

[4] http://ittz.blogspot.com/2009/01/tipos-de-control.html

[5] http://www.juntadeandalucia.es/averroes/ies_sierra_magina/d_tecnologia/bajables/2%20bachillerato/SISTEMAS%20AUTOMATICOS%20DE%20CONTROL.pdf

[6] http://www.sapiensman.com/control_automatico/

[7] http://taninos.tripod.com/viscosidad.htm

[8] Introducción a los sistemas de control automático. José María Marcos Elgoibar. Bellisco, Ediciones técnicas y científicas.

[9] Laboratorios remotos y virtuales para la enseñanza. S. Dormido, J. Sánchez, H. Vargas, S. Dormido-Canto, R. Dormido, N. Duro, G. Farías, Ma. A. Canto. Departamento de Informática y Automática de la UNED.

[10] Laboratorios virtuales en la educación. Arcadio de la Cruz Rodríguez, José Antonio Guerra García, Eduardo Lazarín Meyer. División Preparatoria y Dirección de Innovación para la Academia Departamento de Ciencias Instituto Tecnológico y de Estudios superiores de Monterrey. Campus Estado de México.

[11] Teoría de sistemas. Fernando Matía, Agustín Jiménez, Rafael Aracil. Sección de publicaciones de la Universidad Politécnica de Madrid.