SEP
SECRETARÍA DE EDUCACIÓN PÚBLICA

TECNOLÓGICO NACIONAL DE MÉXICO
Instituto Tecnológico de la Laguna

DIVISIÓN DE ESTUDIOS DE POSGRADO E INVESTIGACIÓN

"Laboratorio virtual de sistemas no lineales usando *Easy Java Simulations*"

POR

Ing. Ana Isabel García Carrillo

TESIS

PRESENTADA COMO REQUISITO PARCIAL PARA OBTENER EL GRADO DE MAESTRO EN CIENCIAS EN INGENIERÍA ELÉCTRICA

DIRECTOR DE TESIS
Dr. Alejandro Enrique Dzul López

CODIRECTOR DE TESIS
Dr. Víctor Adrián Santibáñez Dávila

ISSN: 0188-9060

RIITEC: (21)-TMCIE-2014
Torreón, Coahuila. México,
Diciembre 2014

"2014, Año de Octavio Paz"

Torreón, Coah., 17/Diciembre/2014
Dependencia: DEPI/CPCIE
Oficio: DEPI/CPCIE/162/2014
Asunto: Autorización de impresión de tesis

C. ANA ISABEL GARCÍA CARRILLO
CANDIDATA AL GRADO DE MAESTRA EN CIENCIAS EN INGENIERÍA ELÉCTRICA.
PRESENTE

Después de haber sometido a revisión su trabajo de tesis titulado:

"LABORATORIO VIRTUAL DE SISTEMAS NO LINEALES USANDO EASY JAVA SIMULATIONS"

Habiendo cumplido con todas las indicaciones que el jurado revisor de tesis hizo, se le comunica que se le concede la autorización con número de registro **RIITEC: (21)-TMCIE-2014**, para que proceda a la impresión del mismo.

ATENTAMENTE
EDUCACIÓN TECNOLÓGICA FUENTE DE INNOVACIÓN

SECRETARÍA DE EDUCACIÓN PÚBLICA
INSTITUTO TECNOLÓGICO de la Laguna
División de Estudios de Posgrado e Investigación

DR. JOSÉ LUIS MEZA MEDINA
Jefe de la División de Estudios de Posgrado e Investigación
del Instituto Tecnológico de la Laguna

JLMM/RDSR

TECNOLÓGICO NACIONAL DE MÉXICO
Instituto Tecnológico de La Laguna

"2014, Año de Octavio Paz"

Torreón, Coah., 12/Diciembre/2014

DR. JOSE LUIS MEZA MEDINA
JEFE DE LA DIVISIÓN DE ESTUDIOS DE POSGRADO E INVESTIGACIÓN

Por medio de la presente, hacemos de su conocimiento que después de haber sometido a revisión el trabajo de tesis titulado:

"Laboratorio virtual de sistemas no lineales usando Easy Java Simulations"

Desarrollado por la **C. Ana Isabel García Carrillo**, con número de control **M1213010** y habiendo cumplido con todas las correcciones que se le indicaron, estamos de acuerdo que se le conceda la autorización de la fecha de examen de grado para que proceda a la impresión de la misma.

ATENTAMENTE
EDUCACIÓN TECNOLÓGICA FUENTE DE INNOVACIÓN

Dr. Alejandro E. Dzul López
Asesor/Director de Tesis

Dr. Víctor A. Santibáñez Dávila
Coasesor de Tesis

Dr. Francisco Jurado Zamarripa
Comité Tutorial

Dr. Miguel A. Llama Leal
Comité Tutorial

Agradecimientos

Agradezco a:

- Los doctores Alejandro Dzul y Adrián Santibáñez, mis asesores, por todos sus consejos y apoyo durante el proceso de realización de esta tesis. También a todos los demás profesores del posgrado de Mecatrónica y Contro, por los conocimientos impartidos durante los cursos.

- A mi padre, por su ayuda y atención; a mi madre, cuyo recuerdo me inspira a seguir adelante cada día; a mi hermano Rodolfo, cuyos logros me han servido como ejemplo y a mi hermana Alejandra con quien siempre he compartido los tiempos tanto gratos como difíciles.

- A los maestros Alejandro Flores, Daniel García, Claudio Olivas y Manuel Nieto; mis compañeros con quienes compartí el curso de posgrado y lo hicieron más ameno.

- A mis amigos Albertina Blanco, Itzelle Oronoz, Juan Carrillo, Dinora Sáenz, Fo-lyu Yong, Jorge Balderas y otros más a quiene me acompañaron (o extrañé) durante estos dos años.

- Por último, quisiera agradecer a CONACYT por el apoyo recibido en forma de la beca de posgrado.

Dedicado a la memoria de mi madre, María Luisa

Laboratorio virtual de sistemas no lineales usando *Easy Java Simulations*

Ana Isabel García Carrillo

Resumen

Este trabajo describe la creación de un laboratorio virtual de sistemas no lineales creado utilizando la plataforma *Easy Java Simulations* (EJS), el cual es un paquete de *software* diseñado especialmente para evaluar simulaciones interactivas en el lenguaje de programación Java.

En el presente documento se aborda el uso general de EJS, desde su instalación hasta la descripción de los elementos que lo conforman, así como los pasos para implementar una simulación; como lo es la creación del modelo general del sistema y el diseño de una interfaz gráfica. A lo largo del documento se explica a fondo como introducir en el ambiente de simulación las ecuaciones que describen el sistema a estudiar, la manera de crear métodos y subrutinas para ser empleadas en una simulación, y los elementos gráficos necesarios para recrear la visualización de la simulación.

También se detalla, a manera de un manual del usuario, como utilizar las simulaciones creadas para realizar experimentos. Finalmente, se definen las características y propiedades de los sistemas utilizados para las simulaciones del laboratorio virtual.

Palabras Clave: EJS, interfaz gráfica, modelo, variables, elementos gráficos, control, sistemas no lineales.

Virtual Laboratory of Non-linear Systems using Easy Java Simulations

Ana Isabel García Carrillo

Abstract

In the present paper the creation of a virtual laboratory of nonlinear systems using the Easy Java Simulations platform, which is a software application especially designed to evaluate interactive simulations in the Java programming language, is described.

Over the course of this paper the general use of EJS will be addressed, from its installation to the description of the diferent elements which conform it and the steps to create a simulation: like how to create a system's model and the process to design a graphic interface. In the course of the document it'll be explained how to introduce the equation that describe a system in the simulation enviroment, the way to create methods and subrutines to use inside of a simulation and the graphic elemnts needed to create the view of our simulation.

Also, the use of the designed simulations will be described in the manner of an user's manual. Lastly, the characteristics and properties of the systems used in the virtual laboratory will be defined.

Key words: EJS, graphic interface, model, variable, graphic elements, control, non linear system.

Índice general

Índice de figuras

Índice de cuadros

Capítulo 1

Introducción

En la actualidad, la enseñanza en las áreas de ingeniería puede aprovecharse de las diferentes herramientas de simulación en computadora para reproducir los fenómenos físicos y los efectos de la interacción del alumno con el modelo de una planta, mediante la creación de laboratorios virtuales interactivos[4] y [5]. Con las herramientas y recursos disponibles para la simulación de sistemas, el alumno puede reafirmar lo aprendido en teoría a través de una serie de experimentos virtuales.

La interactividad es uno de los aspectos más importantes a la hora de diseñar simulaciones con fines pedagógicos, en especial en el campo de la ingeniería de control. Gracias a las simulaciones interactivas, los alumnos pueden explorar diferentes configuraciones y probar la respuesta de los sistemas sin la necesidad de tener acceso al sistema físico en cuestión, lo que da más libertad de experimentación al evaluar diferentes leyes de control.

Es para este aspecto de interactividad que *Easy Java Simulations* resulta extremadamente útil. EJS está diseñado específicamente con el propósito de crear simulaciones científicas con un alto nivel de interactividad; dispone internamente de un mecanismo propio para la descripción de modelos y sistemas de ingeniería de control. También cuenta con una amplia librería de elementos visuales interactivos y fácilmente parametrizables que permiten la creación de interfaces gráficas.

A pesar de que EJS utiliza el lenguaje de programación Java para generar las simulaciones, no es necesario que el usuario tenga un alto conocimiento del mismo, lo cual simplifica ampliamente la creación de aplicaciones en simulación, lo que representa una importante ventaja para EJS sobre otros paquetes de programación orientada a objetos.

1.1. Objetivo de la tesis

1.1.1. Objetivo general

Creación de un laboratorio virtual de sistemas no lineales usando EJS.

1.1.2. Objetivos específicos

- Implementar simulaciones para los siguientes sistemas de una entrada y una salida: péndulo simple, circuito RLC, motor de corriente directa y sistema masa-resorte-amortiguador.
- Diseño de un simulador para el robot prototipo de dos grados de libertad.
- Documentar y describir el proceso de diseño del laboratorio virtual.

1.1.3. Actividades a realizar

- Descargar y leer la documentación para los usuarios de EJS.
- Descargar e instalar EJS, tomando nota del proceso necesario para su instalación.
- Recolectar y explorar diferentes simulaciones creadas por otros usuarios en EJS para familiarizarse con la implementación de simulaciones e interfaces gráficas.
- Crear una simulación de práctica: Generador de retratos de fase.

1.2. Estructura del documento

La estructura del presente trabajo de tesis se divide de la siguiente forma:

En el capítulo 2 se hace una pequeña introducción a EJS, se presenta paso a paso la instalación y la preparación previa necesaria para utilizar EJS en cualquier computadora. Después se hace una descripción superficial del uso de EJS y de las partes necesarias para implementar una simulación.

A continuación, en el capítulo 3 se describe más extensamente el uso de EJS para la creación de simulaciones. Todos los elementos y partes de una simulación, junto con su

interfaz gráfica, son explicados a detalle, tanto sus características como su función y la manera en que sus propiedades deben ser modificadas para que se comporten de la manera deseada.

El capítulo 4 consiste de una especie de manual del usuario con el fin de informar lo referente al uso y propósito de cada uno de los elementos que conforman la interfaz gráfica de cada simulación.

Para terminar, en el capítulo 5 se describen los sistemas utilizados en las simulaciones. Este último capítulo se enfoca especialmente en definir las ecuaciones en espacio de estados de los sistemas, que es la forma utilizada en EJS para introducir las ecuaciones que describen la evolución de las dinámicas el sistema.

Capítulo 2

Easy Java Simulation (EJS)

2.1. Introducción a *Easy Java Simulations*

En el estudio del tema de control automático resulta extremadamente útil para los estudiantes tener el acceso a simulaciones interactivas con apoyos visuales que permitan observar el comportamiento de los sistemas y valorar el desempeño de los diferentes controladores que son implementados.

Al poder visualizar el comportamiento del sistema, ya mediante gráficas y modelos en 3D que representen la respuesta física del sistema, el alumno tiene una mejor comprensión de los conceptos estudiados en clase. *Easy Java Simulations*, es un software que funciona como herramienta para crear simulaciones interactivas en el lenguaje de programación Java. El objetivo principal de EJS consiste en simplificar la tarea de creación del ambiente gráfico de la simulación de manera que el autor pueda enfocarse principalmente en la realización del modelo matemático, definición las variables, y diseño de los algoritmos necesarios para describir el sistema o fenómeno que a estudiar.

Los sistemas utilizados en las simulaciones creadas en EJS pueden ser tanto sencillos como extremadamente complejos, sin embargo, EJS tiene integradas varias características que también facilitan la creación de los algoritmos de programación, como por ejemplo, la resolución automática de ecuaciones diferenciales. El autor de EJS es el Dr. Francisco Esquembre, profesor de la Universidad de Murcia, quien lo distribuye gratuitamente bajo la licencia GNU GPL, en http://fem.um.es/Ejs/, donde pueden encontrarse las versiones más recientes de EJS así como manuales de usuario y simulaciones creadas por miembros de la comunidad; documentos que son extremadamente útiles para familiarizar al programador

con el proceso de diseño y desarrollo de simulaciones.

2.2. Instalación

2.2.1. Preparación previa

EJS puede utilizarse en cualquier sistema operativo que sea compatible con la máquina virtual de Java, sin embargo en este caso nos enfocaremos en la instalación en el sistema operativo Windows; en este trabajo se utiliza la versión 7 de Windows, pero el proceso es similar para las versiones XP, Vista y 8. Antes de poder utilizar EJS en una computadora, debemos asegurarnos de que esta cuente con el software necesario para ejecutar EJS, es decir, debe tener instalado *Java Runtime Environment* (JRE) y *Java Standard Edition Development Kit* (JDK). Se debe tener cuidado en no confundir estos dos programas. JRE está generalmente presente en todas las computadoras puesto que es necesaria para utilizar páginas web con elementos Java, y aunque permitirá visualizar las simulaciones no será posible generar o crear nuevas simulaciones únicamente con JRE.

Los vínculos para descargar JRE y JDK pueden encontrarse en la sección de descargas de la página oficial de EJS, o también podemos encontrarlos en la página de desarrolladores de Java http://www.oracle.com/. La instalación de ambos es un proceso sencillo, solo requiere correr un archivo ejecutable de instalación.

Un detalle importante después de la instalación de JDK es que cada vez que sea utilizado es necesario especificarle al ejecutable la dirección completa de las variables del entorno del sistema o PATH. Para especificarla permanentemente, se deben seguir los siguientes pasos (suponiendo que el usuario esté utilizando el sistema operativo Windows 7):

1. Abrir el Panel de Control de Windows; se selecciona Sistema.

2. En la pestaña de Opciones Avanzadas se selecciona Variables del entorno.

3. Se añade la dirección de la carpeta de instalación de JDK a la variable PATH en Variables del Sistema. La dirección de JDK es generalmente C:\Archivos de programa\Java\jdk1.8.0\bin, pero puede cambiar, dependiendo de la versión de JDK

Figura 2.1: Ventana de variables del sistema

que se tenga instalada o del idioma del sistema operativo. En la figura 2.1 se pueden observar las ventanas para modificar las variables del sistema, solo es necesario escribir la dirección anteriormente mencionada, separada con punto y coma de las otras variables.

Una vez se que hayan realizado los pasos anteriores la computadora está lista para utilizar completamente EJS, pero para utilizar algunos elementos de figuras en 3D en las simulaciones de EJS también es necesario instalar el software de Java 3D, que también puede ser descargado de la página oficial de Java.

2.2.2. Instalación de EJS

La instalación de EJS consiste básicamente en descomprimir un archivo ZIP, que se puede encontrar en la página web anteriormente mencionada. Es posible descomprimir la carpeta en cualquier ubicación de la computadora, pero se recomienda evitar los espacios en el nombre de la carpeta así como en las carpetas madre.

Una vez descomprimida la carpeta solo es necesario ejecutar el archivo Ejsconsole.jar para comenzar a utilizar EJS. Si la configuración previa se realizó correctamente, EJS detectará la ubicación de JDK, de lo contrario será necesario especificar la ruta del directorio donde JDK está instalado. Ahora es posible comenzar a crear simulaciones en EJS.

Figura 2.2: Ventana de opciones de EJS

2.3. Uso general de EJS

Al ejecutar EJS se abrirán dos ventanas, la primera ventana tiene tres pestañas que permiten configurar EJS para el uso en la computadora, ver la figura 2.2. En la pestaña de opciones básicas (*basic options*) se puede seleccionar la máquina virtual de Java que se utilizará y el espacio de trabajo donde la simulación será guardada; también tiene opciones de apariencia e idioma. Todo esto ya tiene una configuración inicial y por lo general no es necesario cambiar los valores predeterminados. Las opciones avanzadas (*advanced options*) son para usuarios más experimentados de Java, permitiendo usar máquinas virtuales alternas o utilizar argumentos adicionales para EJS. La tercer pestaña es la pestaña del área salida, es de utilidad para que el usuario sepa si EJS se inició correctamente, o para identificar los errores que puedan estar afectando al programa y no le permitan trabajar correctamente.

La segunda ventana es donde se empiezan a crear las simulaciones, ver la figura 2.3. A continuación, se inicia con una explicación breve de cada una de las partes de una simulación que se permiten crear:

2.3.1. Descripción

Como su nombre lo dice, esta opción permite crear una descripción para la simulación en el lenguaje HTML, que se mostrará al usuario en forma similar a una página web, sin embargo, no es necesario conocer a fondo este lenguaje de programación para realizar una página de descripción. EJS presenta un editor similar a los procesadores de texto, en este es posible elegir el tamaño y tipo de letra, insertar símbolos matemáticos, tablas, imágenes, hacer listas; en fin, todo lo necesario para informar al usuario del contenido de

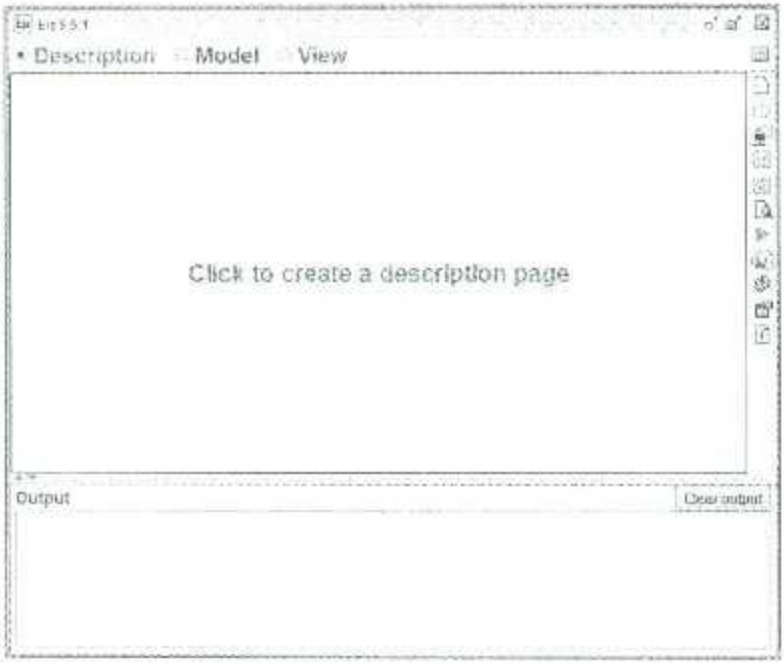

Figura 2.3: Ventana para la creación de simulaciones

la simulación. Si ya se conoce el lenguaje HTML, también cuenta con la opción de editar el código de la página directamente y personalizarla al gusto del programador, en caso de que las opciones ofrecidas por EJS no cumplan con los requerimientos solicitados por el programador.

2.3.2. Modelo

Es la parte más importante de la simulación, dado que aquí es donde se describe el modelo del sistema a simular. Esta parte se divide en variables, inicialización, evolución, relaciones fijas, personalizados y elementos. A continuación se describirán brevemente, más adelante se explicarán a fondo:

- Variables: Se crean tablas de las variables a utilizar en la simulación; se define su nombre, valor inicial, tipo y dimensiones (en caso que de la variable sea un arreglo).
- Inicialización: Aquí se establecen valores iniciales que necesiten algún tipo de cálculo inicial.

- Evolución: Aquí se describe la evolución del sistema, generalmente por medio de un sistema de ecuaciones diferenciales, definiendo la variable independiente y su incremento, el método numérico para resolver las ecuaciones y su tolerancia.
- Relaciones fijas: Aquí se definen las relaciones entre variables que se mantienen fijas durante toda la evolución de nuestro sistema, dichas relaciones pueden servir como restricciones para evitar errores de cálculo.
- Personalizado (*Custom*): Para hacer uso de esta opción, el usuario debe tener conocimiento de programación en Java, puesto que permite escribir métodos personalizados para ayudar ya sea con los cálculos del sistema o para cambios en los gráficos de la simulación.
- Elementos: Aquí se agregan elementos externos a la simulación: servidores web, hardware, bases de datos, etc.

2.3.3. Vista

Aquí se desarrolla la parte gráfica de nuestra simulación, mostrándose una interfaz orientada a objetos con diferentes elementos, como etiquetas de texto, gráficas, figuras en 2D y 3D y elementos interactivos para controlar la simulación.

2.3.4. Otros elementos

En la parte derecha de la pantalla que se muestra en la figura 2.3 se puede encontrar una columna con varios botones para realizar diferentes acciones, las cuales se listan a continuación:

- Información de la simulación: Abre una ventana que permite modificar la información interna de la simulación, similar al "Acerca de" que aparece comúnmente en diferentes programas. Aquí es posible introducir datos como el nombre del autor, título, agregar un logo de la institución, etc.
- Crear nueva simulación: Crea un nuevo archivo EJS para trabajar.
- Abrir desde el área de trabajo: Abrir una simulación creada previamente.

- Leer de una biblioteca digital de EJS: Se conecta por medio del internet a una base de datos donde se pueden descargar simulaciones creadas por otros usuarios.
- Guardar: Guarda el trabajo en la simulación actualmente activa.
- Guardar como nuevo archivo: Guarda la simulación en un nuevo archivo.
- Buscar: Permite buscar una cadena de texto, ya sea en el código del modelo o en la vista de la simulación.
- Correr simulación: Ejecuta la simulación.
- Traducir la interfaz de simulación: Permite cambiar el lenguaje en el que EJS se presenta. El programa viene en idioma inglés y para cambiar el idioma es necesaria una conexión a internet para descargar los archivos de idioma.
- Empaquetar la simulación: Convierte la simulación de un archivo .cjs a un ejecutable .jar, que puede correrse en cualquier máquina con Java.
- Opciones de EJS: Permite cambiar diversas opciones como la apariencia de la interfaz de EJS (color de la ventana, tipo de letra, tamaño, etc.), opciones de la exportación a HTML, las bases de datos a las que podemos conectarnos para descargar simulaciones de otros usuarios, seleccionar la versión de Java con la que se está trabajando y datos del usuario.
- Ayuda e información: Este botón es básicamente un vínculo a la página web oficial en inglés de EJS.

Capítulo 3

Diseño del simulador

En esta capítulo se explicará a detalle cómo crear una simulación en EJS, con descripciones de cada uno de los elementos, así como con ejemplos de cómo realizar diferentes acciones para crear la simulación y también la manera de introducir información. Para comenzar con la creación de simulaciones en EJS, primero se supondrá que el usuario ya dispone del modelo (de preferencia en espacio de estados) del sistema a simular y que cuenta también con las leyes de control a evaluar. Para crear estas simulaciones se utilizaron como apoyo aquellas consideradas en las referencias [7]-[10]. Estas simulaciones están disponibles para ser descargadas desde la página de OPS (*Open Source Physics*); se utilizaron principalmente para conocer más a fondo el uso de los elementos visuales de la simulación y para aprender un poco más de la sintaxis de programación propio del lenguaje Java.

3.1. Modelo

En EJS un modelo es definido como un fenómeno físico del cual es posible describir su comportamiento y obtener el valor de las variables que determinan su estado en cualquier instante de tiempo. En la simulación el estado de las variables cambia debido a dos causas:

1. La dinámica interna de la simulación, que en EJS se le denomina evolución.
2. La influencia de factores externos. Como el modelo no existe en el mundo real, depende del programador crear los medios para que el usuario pueda interactuar con la simulación y modificar los valores del modelo para causar efectos similares.

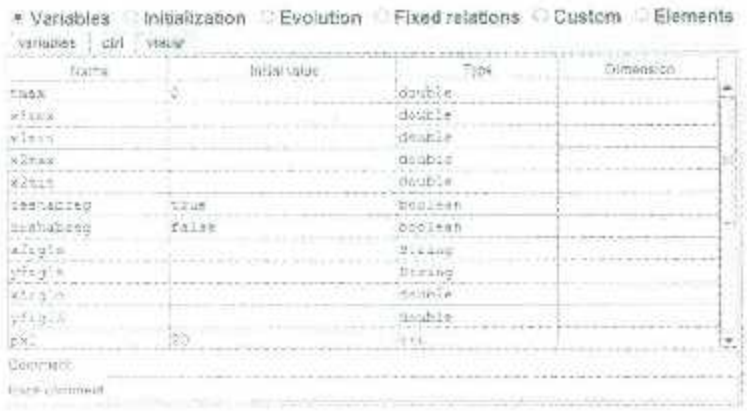

Figura 3.1: Tabla de variablesen EJS.

Se debe tomar en cuenta que los cambios causados directamente por lo anteriormente mencionado también llevarán a cambios indirectos, puesto que hay variables cuyos valores son dependientes de otras variables. Por eso, para poder definir nuestro modelo, se debe establecer lo siguiente:

- Las variables del modelo.
- Las ecuaciones de evolución.
- Las ecuaciones de las relaciones fijas.

3.1.1. Variables

Como se había explicado con anterioridad, en la sección de variables se crean tablas de variables necesarias para realizar la simulación. En la figura 3.1 podemos ver la manera en que las tablas son presentadas en el programa. Las tablas contienen la siguiente información:

- Nombre: Aquí se define el nombre asociado a la variable a identificar en la simulación. También hay que recordar que el lenguaje Java hace distinción entre minúsculas y mayúsculas, por lo que se podría tener dos diferentes variables con el mismo nombre escrito con diferentes caracteres, lo cual no se recomienda para evitar confusiones. El nombre debe ser único en el modelo, puede contener cualquier caracter alfanumérico (a-z, A-Z y 0-9), no hay límite en el número de caracteres que puede ser utilizado para el nombre, pero el primer caracter siempre debe ser alfabético.

- Valor inicial: Como su nombre lo indica, aquí se introduce el valor inicial de las variables. Puede ser un número, cadena o valor lógico, dependiendo del tipo de variable. En caso de que sea una variable con valor fijo, aquí también se establecerá este valor. En caso de no escribir nada en este campo, por lo general, EJS asocia a las variables numéricas un valor inicial de cero, mientras que para los demás tipos de variables se les asocia un valor inicial nulo.

- Tipo: Es el tipo al que pertenece la variable; los tipos disponibles son:

  - *Boolean* (Booleano): Un valor lógico de falso (false) o verdadero (true).
  - *Double* (Doble): Número que contiene valores decimales.
  - *Int* (Entero): Número entero.
  - *String* (Cadena): Una cadena de caracteres en texto.
  - *Object* (Objeto): En este tipo de variables se puede contener diferentes tipos de información relacionada a la programación orientada a objetos; para dar un ejemplo, se podría crear una variable que contenga una referencia al valor de un color. El uso de este tipo de variable es generalmente reservado para personas con experiencia en programación en Java.
  - Dimensión: La dimensión de la variable, en caso de que esta sea un arreglo; si se deja este campo en blanco la variable tendrá una dimensión de 1x1, es decir, se considera un escalar. Las dimensiones deben escribirse entre corchetes, por ejemplo, si escribimos [50], nuestra variable será un vector con 50 elementos, pero si escribimos [2][50], tendremos luego una matriz de 2 renglones con 50 columnas. También podríamos escribir, por ejemplo, [n][n] y tendríamos una matriz de dimensiones variables que podría ser determinada por el usuario al ejecutar la simulación.

A continuación se mostrará en la tabla 3.1 un ejemplo de cómo se introducirían tales variables:

Tabla 3.1: Ejemplo de variables

Variable	Nombre	Valor inicial	Tipo	Dimensión
Masa	m	0	double	
Ley de control	ctrl	"PD"	string	
Tiempo máximo de simulación	tmax	10	int	
Deshabilitar opciones	deshab	false	boolean	
Vector de velocidades	v		double	[1][2]

3.1.2. Inicialización

Como se mencionó en la sección anterior, podemos iniciar nuestras variables con valores constantes en la columna de "Valor inicial", en la sección de variables, pero hay ocasiones en que se necesitará definir valores más complejos para las variables que requieran de cálculos realizados previamente.

En el panel de inicialización, se tiene un editor al que se puede agregar cuantas páginas sean necesarias y el código puede ser tan complejo o extenso como sea necesario. Se deben utilizar expresiones de Java para crear los algoritmos, también viene con una opción llamada "*CodeWizard* que nos ayuda a introducir métodos simples de Java, como declaraciones condicionales (If-Then-Else), o ciclos (For). El editor contiene al final un pequeño espacio para hacer comentarios sobre el código escrito.

Se debe tener presente que el código escrito aquí sólo se ejecutará una vez, al inicio de nuestra simulación. Por citar un ejemplo, supónga que es necesario calcular la pendiente de una recta al principio de la simulación, el código sería el siguiente:

```
x=x1-x2;
y=y1-y2;
m=y/x;
```

3.1.3. Ecuaciones de evolución

Como ya se mencionó, un sistema puede evolucionar desde su estado actual $x_1, x_2, ...x_n$ a un estado $x_1', x_2', ...x_n'$ cuando se simula el paso del tiempo. Las ecuaciones que dictan

Figura 3.2: Tabla de ecuaciones diferenciales

este cambio se conocen como ecuaciones de evolución en EJS, que se pueden escribir como una ecuación general:

$$x_i' = f_i(x_1, x_2, ...x_n). \tag{3.1}$$

Estas ecuaciones son generalmente una serie de ecuaciones diferenciales, las cuales se pueden obtener al escribir el modelo en espacio de estados. En EJS, las ecuaciones de evolución se introducen en el panel como se puede ver en la figura 3.2.

Primero se selecciona de qué manera queremos introducir las ecuaciones, como código o como una serie de ecuaciones diferenciales. La sección de código está reservada para los usuarios más avanzados, ya que no solo se deben escribir las ecuaciones que describen el sistema, también se debe escribir el algoritmo para resolver ecuaciones. Esto es útil si nuestro fin educativo es enseñar los métodos numéricos para resolver las ecuaciones, pero como el objetivo es únicamente el de simular sistemas dinámicos nos enfocaremos en la opción de ecuaciones diferenciales. Después de seleccionar la opción de ecuaciones diferenciales se le debe dar un nombre a la pestaña creada (en la figura 3.2 la pestaña es llamada "evolución"). Ahora, se describirá cada uno de los elementos que contiene esta página:

- FPS (*Frames per second*, cuadros por segundo): Este parámetro le indicará a EJS cuantas veces por segundo deberá ejecutar las ecuaciones de evolución; puede ser modificado de dos maneras; la primera mediante una barra deslizante a la izquierda del panel, y la segunda se ubica directamente debajo de la barra, como un cuadro

de texto donde es posible introducir directamente el valor que se desea darle. Al momento de ejecutar la simulación, un alto número de FPS causará una animación más fluida, pero debido a los recursos necesarios para realizar la ejecución continúa de las ecuaciones también se puede hacer que la simulación se vuelva excesivamente lenta.

- *Autoplay*: Si está seleccionada, la simulación comenzará automáticamente.
- Indep. Var. (Variable independiente): Como su nombre lo indica, es la variable independiente del sistema, generalmente el tiempo. Es posible escribir el nombre de la variable directamente o apretar el botón a la derecha para elegir una de las variables existentes del tipo *double* como la variable independiente.
- *Increment* (Incremento): Incremento de la variable independiente para cada paso de la simulación. Este valor puede ser tanto un valor del tipo *double*, así como un valor entero. Si se utiliza el valor 0.1, por ejemplo, en cada paso de la evolución de la simulación EJS evaluará la ecuación avanzando del estado actual t al estado siguiente $t + 0.1$. Como parte del sistema de solución, la variable tiempo aumentará por el incremento que se ha especificado, por eso, si se quiere utilizar una variable independiente diferente se deberá crear una nueva página de ecuaciones.

En la parte inferior se ubica una lista de los diferentes tipos de algoritmos numéricos para resolver las ecuaciones; existen los siguientes métodos:

- Euler. Es un método de primer orden, por lo cual tiene un nivel bajo de precisión, solo se recomienda usarlo con fines de estudio.
- Euler-Richardson. Es un método de segundo orden, con un nivel de precisión aceptable y buena velocidad de cómpute. Los autores de EJS recomiendan empezar con este método y solo usar uno más poderoso en caso de que los resultados de simulación no muestren la precisión deseada.
- Runge-Kutta. Un método clásico, de orden 4. Con un nivel de precisión razonable. Este fue el método que se utilizo en las simulaciones del presente trabajo de tesis pues mostraba un buen nivel de precisión sin exigir demasiados recursos de procesamiento.

- Runge - Kutta - Fehlberg (aparecen en la lista solo como "Fehlberg"), Dormand-Prince, Bogacki Shampine. Son métodos con un paso adaptable, utilizando un valor predefinido, puede ajustar automáticamente el paso de integración para que el valor obtenido sea menor al valor anteriormente mencionado, llamado tolerancia. Si se selecciona uno de estos métodos se debe especificar el valor de la tolerancia.

Al lado del cuadro de texto, donde se introduce la tolerancia, se encuentra un botón para añadir eventos a las ecuaciones, que será discutido más adelante.

Ahora se explicará el editor de ecuaciones diferenciales ordinarias. Este editor permite introducir a la simulación un sistema de ecuaciones diferenciales ordinarias, explícitas y de primer orden, es decir, en la que la primera derivada de una variable está expresada en una función de sí misma, o de las demás variables del modelo. Al seleccionar las variables que se utilizarán para expresar las ecuaciones se debe tener cuidado en seleccionar solo variables del tipo *double* (doble).

Como incluso sistemas de mayor orden pueden ser reescritos mediante ecuaciones de primer orden, esta aparente restricción no presenta un problema a la hora de crear la simulación, en cambio, permite resolver gran cantidad de problemas de diferenciación.

Declaración las ecuaciones

Para ilustrar el proceso de declarar variables en el editor se utiliza como ejemplo el modelo del péndulo sin fricción[2]. Primero, considérese la ecuación diferencial que rige el movimiento:

$$ml^2\ddot{q} - mgl\sin(q) = 0 \tag{3.2}$$

donde m es la masa, l la distancia desde el eje de giro al centro de masa, g es la acción de la gravedad, q la posición angular del péndulo respecto a la vertical y $\dot{q}$ es la velocidad angular. Reescribiendo esta ecuación en espacio de estados resulta:

$$\begin{aligned} \dot{x}_1 &= x_2 \\ \dot{x}_2 &= -\frac{g}{l}\sin x_1 \end{aligned} \tag{3.3}$$

donde $x_1 = q$ y $x_2 = \dot{q}$. Ahora que las ecuaciones se encuentran en el espacio de estados, se introducen en la tabla de ODEs de EJS, como se muestra en la tabla 3.2.

Tabla 3.2: Ejemplo de ecuaciones diferenciales

	Estado (*State*)	**Tasa** (*Rate*)
Ecuación 1	$\frac{dx1}{dt}$	x2
Ecuación 2	$\frac{dx2}{dt}$	-(g/l)*(Math.sin(x1))

El proceso es bastante simple, primero se da un doble clic en la columna de la izquierda para introducir el estado que se va a diferenciar, y luego otro doble clic en la columna de la derecha para escribir la expresión correspondiente de la ecuación. Para facilitar el proceso de edición de las ecuaciones puede darse clic derecho en cualquiera de las celdas y aparecerá un menú para seleccionar variables de una lista, así como también una serie de opciones para personalizar la tabla.

A pesar de que en la columna de estados se muestra la variable escrita como $\frac{dx1}{dt}$, por ejemplo, el programador solo debe introducir el nombre de la variable, que en caso anterior sería "x1". Un detalle importante que debe mencionarse es que las ecuaciones deben escribirse en la sintaxis de Java. Para las operaciones aritméticas se utiliza la misma construcción que en la gran mayoría de los lenguajes de programación, pero para expresiones un poco más complejas, como operaciones trigonométricas, logarítmicas o potencias, se debe escribir "Math." antes del operador, por lo que se recomienda consultar una lista de todos los elementos de la clase Math para conocer las operaciones que se pueden realizar y la sintaxis correcta para utilizarlos. En [6] se encuentra la lista completa de los elementos que forman la clase Math. EJS tiene una función de autocompletar que sirve como ayuda para las operaciones de la clase Math, desafortunadamente, esta función no esta disponible para la tabla de ecuaciones diferenciales, solo se activa cuando se llama a la clase Math en los ambientes para escribir código.

El proceso descrito anteriormente es para introducir ecuaciones con variables escalares, si se utilizan variables vectoriales, se seguirán los mismos pasos, pero con algunas diferencias. Por razones de la construcción de EJS, el evaluador de ecuaciones solo puede trabajar con variables simples, así que para los valores vectoriales se debe evaluar cada uno de sus elementos por separado, indicando el índice de los elementos utilizados. También se puede utilizar un índice variable, como por ejemplo, el contador i, siempre y cuando no se tenga otra variable global con el mismo nombre. Cuando se utiliza este índice variable se debe utilizar para todas las variables vectoriales escritas en la columna derecha. Así,

si se tiene una variable vectorial llamada x1, se podrían escribir las variables vectoriales a evaluar como "x1[1]" si se desea especificar el elemento 1, o como "x1[i]" si se utiliza el índice variable.

Otro detalle importante a mencionar es que no solo se pueden evaluar las operaciones predeterminadas escritas por el creador de la simulación en la tabla de variables, también es posible evaluar ecuaciones escritas por el usuario en un cuadro de texto especial llamado "Function", el cual será tratado más adelante en la sección donde se describen los aspectos gráficos de las simulaciones. Para realizar esto se utilizan dos comandos importantes, el primero es el comando "_view.", el cual es extremadamente importante para crear simulaciones en EJS, pues nos permite inspeccionar atributos en los objetos gráficos de las simulaciones; y el segundo comando es "Evaluate", que es exclusivo para los cuadros de texto de funciones y nos permite, como su nombre lo dice, evaluar las funciones que están escritas en el cuadro de texto. La sintaxis para realizar esta evaluación desde la tabla de ecuaciones es la siguiente:

$$\underbrace{\frac{dx1}{dt}}_{\substack{\text{Nombre}\\ \text{del estado}}} = \text{\_view.}\underbrace{\text{function}}_{\substack{\text{Nombre del}\\ \text{cuadro de}\\ \text{texto}}}\text{.evaluate}\underbrace{(x1,x2...xn)}_{\substack{\text{Nombre de}\\ \text{las variables}}};$$

Como se observa, la construcción de la instrucción es similar a las que se han visto anteriormente, pero utilizando una función de Java en lugar de una ecuación. Primero, con la instrucción "_view.", se indica que se está referenciando uno de los elementos gráficos de la simulación, luego se escribe el nombre del cuadro de funciones al cual se está referenciando. Con el comando "Evaluate" se hace la evaluación de la función y al final, entre paréntesis, se escriben las variables a evaluar. Para ejemplificar, digamos que el usuario escribió en el cuadro de funciones la ecuación "x1−x2", por lo cual se deben mandar dos variables a evaluar, si en la simulación las variables tienen el mismo nombre, entonces se escribiría el segmento final como "(x1,x2)".

Eventos de una ecuación diferencial

En algunas ocasiones, al correr la simulación puede que sea que la computadora detecte ciertos cambios o condiciones en las variables que requieren que hagamos algún ajuste en la simulación.

Los eventos que EJS detecta son: el cruce por cero, el cruce positivo o un evento de un

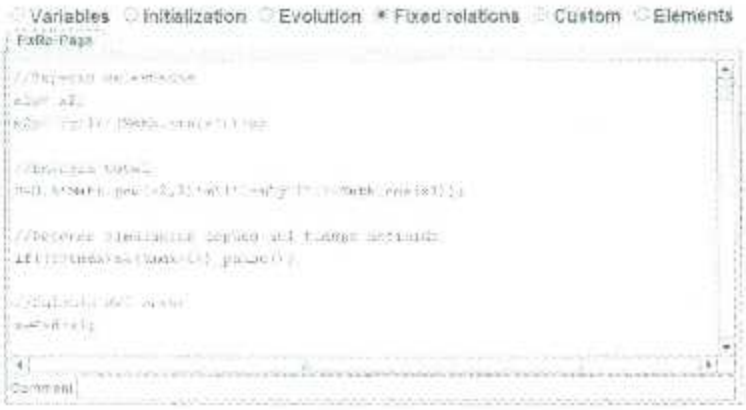

Figura 3.3: Relaciones fijas en la simulación del péndulo simple

estado que sea definido por el usuario.

3.1.4. Relaciones fijas

El panel para la creación de las relaciones fijas, que aparece en inglés en el menú de EJS como *Fixed Relations*, funciona de la misma manera que el panel para crear las ecuaciones de inicialización, como puede verse en la figura 3.3, pero debemos tomar en cuenta que el rol de estas ecuaciones en la simulación es completamente diferente.

Una manera de identificar cuáles de las ecuaciones son relaciones fijas, es pensar en todas la relaciones que deben mantenerse, incluso cuando la evolución del sistema esté detenida, como puede ser durante momentos en que la simulación es pausada o momentos en que el usuario esta interactuando con la simulación.

Las relaciones fijas son ejecutadas siempre al inicio de la simulación (pero después de aquellas especificadas en la sección de inicialización) y también se seguirán ejecutando en cada paso de la simulación y cada vez que el usuario realice un cambio sobre esta. Por lo tanto, cualquier relación entre variables que sea escrita aquí será validada durante el transcurso de toda la simulación. A continuación se muestran algunos ejemplos de las relaciones fijas utilizadas en algunas de las simulaciones del presente trabajo de tesis; para empezar con un ejemplo sencillo de una relación fija, tenemos el cálculo del error de posición

```
xe = xd - x1;
```

donde `xe` es el error de posición, `xd` la posición deseada, y `x1` la posición actual y que

como es bien sabido, la ecuación para calcularlo es una simple resta que se mantiene constante durante toda la simulación. Además de operaciones matemáticas también se pueden realizar operaciones lógicas. Para citar otro ejemplo sencillo:

```
if(t>tmax)_pause();
```

Con esta operación condicional se detiene la simulación con la función interna *_pause* cuando `t`, que es el tiempo de la simulación, llegue hasta un valor *tmax* que es el tiempo total de simulación establecido por el usuario. Para un ejemplo un poco más complejo de otros valores cuyos cálculos se mantienen constantes, considérese el robot de dos grados de libertad[2] cuyo modelo dinámico tiene los siguientes elementos: la matriz de Coriolis, la matriz de inercias y el vector de pares gravitacionales; que deben ser introducidos en la sección de relaciones fijas. La matriz de Coriolis es introducida como se muestra enseguida:

```
C11=-m2*l1*lc2*Math.sin(q2)*qp2;
C12=-m2*l1*lc2*Math.sin(q2)*(qp1+qp2);
C21=m2*l1*lc2*Math.sin(q2)*qp1;
C22=0;
```

Como se puede apreciar, la manera de escribir ecuaciones y operaciones es similar a la de otros lenguajes de programación, la diferencia radica en el uso de la librería "Math." de Java para utilizar la operación trigonométrica seno.

3.1.5. Métodos personalizados

La última opción en el modelo del sistema es la sección de métodos personalizados (*Custom*). El propósito principal de este panel es el de permitir el uso de métodos o subrutinas propias que puedan ser necesarias para diferentes partes de nuestra simulación. Esta sección también puede utilizarse para que usuarios más avanzados en programación en Java puedan importar sus propias librerías de clases y métodos.

La interfaz de esta sección está dividida en dos partes: la parte superior, donde se pueden escribir los métodos personalizados mientras que la parte inferior es utilizada para importar las librerías de Java.

Figura 3.4: Método personalizado para la selección de leyes de control.

Descripción

La función general de los métodos personalizados es la de simplificar partes de la simulación por medio de código de programación. Los métodos personalizados pueden ser utilizados tanto en el modelo de la simulación como en la interfaz gráfica. Para dar un ejemplo, se parte de la idea de que cierto código será utilizado en varias partes de la simulación, así que podemos escribir dicho código aquí e invocarlo desde cualquier parte de la simulación. Otro uso importante de estos métodos es el ser ejecutados como respuesta a la interacción del usuario con la simulación.

La apariencia del editor de métodos personalizados es similar a la de los dos editores de código anteriormente mencionados en las secciones de inicialización y relaciones fijas, pero existen ciertas diferencias. La más importante es que el código aquí escrito solo será ejecutado si es invocado directamente, a diferencia de las relaciones fijas o las relaciones de inicialización que son ejecutadas automáticamente por EJS. Otra diferencia que se tiene es el grado de libertad que podemos usar al momento de escribir los métodos, estos pueden ser tan complejos como sea necesario siempre y cuando estén escritos con una sintaxis correcta de Java.

Creación de métodos personalizados

Lo primero que se debe hacer para crear una serie de métodos personalizados es crear una página en el editor, esto servirá para poder organizarlos en caso de que una gran cantidad de métodos sean utilizados.

La sintaxis que debe llevar un método en Java es la siguiente: es necesario especificar la accesibilidad del método, el tipo de valor que regresará (en caso de que retorne valores, pues no todos los métodos lo hacen) y el nombre del método. También se debe indicar si el método acepta parámetros de entrada así como el tipo de dato de los parámetros.

- Accesibilidad: Esta parte del método se refiere a qué partes de la simulación podrán acceder a este. Si es declarado público (*public*) entonces el método podrá accederse desde cualquier parte de nuestra simulación, ya sea cualquier parte del modelo o de la vista gráfica, incluso podría visualizarse fuera de la simulación. La otra opción de accesibilidad es declarar el método como privado (*private* o dejar en blanco el inicio del método), entonces el método solo podrá ser utilizado dentro del modelo de la simulación. En EJS no hay mucha razón para declarar métodos privados, con excepción de querer restringir el acceso del usuario a algún método durante el tiempo de simulación, pero por lo general realmente no hay problema en declarar todos los métodos públicos.

- Valor de regreso: Aquí se hace referencia a si el método regresa un valor (por lo cual, el método sería una función) o si no lo hace (entonces es una subrutina). En caso de que no retorne ningún valor se escribirá en esta parte *void*. En caso contrario, debe escribirse aquí el tipo de valor que la función regresará, que puede ser cualquier tipo de las variables disponibles en Java. También es necesario especificar la dimensión del valor en caso de tenerla. Otro detalle de los métodos del tipo función es que siempre deben terminar con la instrucción *return* para indicar el contenido de la variable a retornar.

- Nombre del método: No hay muchas restricciones en lo que concierne a los posibles nombres para los métodos, pero se recomienda que el usuario tenga en cuenta las mismas recomendaciones que se hicieron en la sección sobre como nombrar a las variables del sistema.

- Parámetros de entrada: Finalmente, los parámetros de entrada son definidos por una lista con el nombre de la variable y su tipo, con cada par de nombres y tipos separados por comas. Esta lista debe escribirse entre paréntesis después del nombre del método. Al invocar el método desde otra parte de la simulación se debe tener cuidado en siempre enviar el tipo y número correcto de parámetros de entrada. En caso de no tener parámetros de entrada, se puede dejar esta lista en blanco.

EJS provee la forma básica de un método a la hora de crear una página en el editor, la cual puede ser cambiada al gusto del programador. Se puede crear cuantos métodos deseerros en cada página. También es posible dar a la página un nombre que ayude a identificar los métodos que contiene. A continuación se muestra un ejemplo de un método utilizado una simulación de esta tesis:

```
public void seleccionar() {
if(((tipo).equals("PD"))&&(_view.tabbedPanel.getSelectedIndex()==2)) {
func="kp*xe-kv*x2";
}
if(((tipo).equals("PID"))&&(_view.tabbedPanel.getSelectedIndex()==2)) {
func="kp*xe-kv*x2+ki*dxe"; }
}
```

Esta subrutina es llamada "seleccionar" y es usada en todas las simulaciones donde es posible para el usuario elegir entre los controladores PD y PID. Es pública y no regresa ningún valor. Su propósito es examinar una variable del tipo *string*, llamada *tipo* (que está ligada a una lista de tipos de control en la vista de la simulación) y la evalúa con el operador *.equals*[1] que permite que la condición se cumpla solo si el valor de la variable *tipo* es exactamente igual al valor contra el cual la estamos evaluando. La segunda condición se cumple cuando el usuario ha seleccionado la segunda página del panel, que es donde se simulan los controladores del tipo PD. De acuerdo a las condiciones que se cumplan, el valor de la variable *func*, que representa la ley de control, cambiará para reflejar el tipo de control que el usuario ha elegido.

A continuación se presenta otro ejemplo de un método personalizado ahora más elaborado:

```
public void calcular () {
if (_view.plottingPanel.getMouseButton()!=_EjsConstants.LEFT_MOUSE_BUTTON)
return;
_pause();
```

[1] *.equals* es utilizado para evaluar cadenas de caracteres; en caso de comparar valores numéricos o lógicos se utiliza el doble signo de igual, "==".

```
t = 0;
x1 = _view.plottingPanel.getMouseX();
x2 = _view.plottingPanel.getMouseY();
_view.trace.moveToPoint(x1,x2);
_play(); }
```

Esta subrutina es llamada "calcular" y es utilizada en la simulación para obtener retratos de fase. Su función principal es la de obtener un punto de inicio indicado por el usuario para las trayectorias del retrato de fase. Primero tiene una condición donde examinará el panel del retrato de fase para saber si un evento ha ocurrido, en este caso el evento es que el usuario dé un clic en el panel. También se puede observar que esta subrutina tiene una función interna, que es una función predeterminada de EJS que regresa un valor booleano verdadero en caso de que el usuario presione el botón izquierdo del mouse. A continuación, se tiene otra función predeterminada de EJS, que pausa la simulación y regresa el tiempo a cero. Esto es necesario para crear una nueva trayectoria en caso de que ya exista una previa. Después, se examina el panel del retrato de fase, obteniendo la posición en la que el usuario dio clic, con los atributos $getMouseX$ y $getMouseY$, que se almacenarán en las variables $x1$ y $x2$ y que servirán como los valores iniciales. El siguiente paso es mover la línea que dibuja la trayectoria, que es un objeto llamado *trace*, al punto inicial de la trayectoria. El paso final es simplemente volver a iniciar la simulación.

3.1.6. Elementos

En esta sección se pueden agregar una serie de elementos externos a nuestra simulación, ya sea tanto de software como de hardware. En la figura 3.5 se muestra la pantalla para agregar estos elementos. Para el presente trabajo no se utilizó ninguna de las opciones disponibles en esta sección, pero se considera interesante mostrar que EJS es aparentemente compatible con las tarjetas de adquisición de datos de Arduino, lo que deja una opción interesante para crear la maqueta de un sistema para controlar con EJS.

3.2. Vista

En esta sección se comenzará a describir el proceso de la creación de la parte visual de la simulación desarrollada en esta tesis. Para el estudio del control, oportunidad de tener

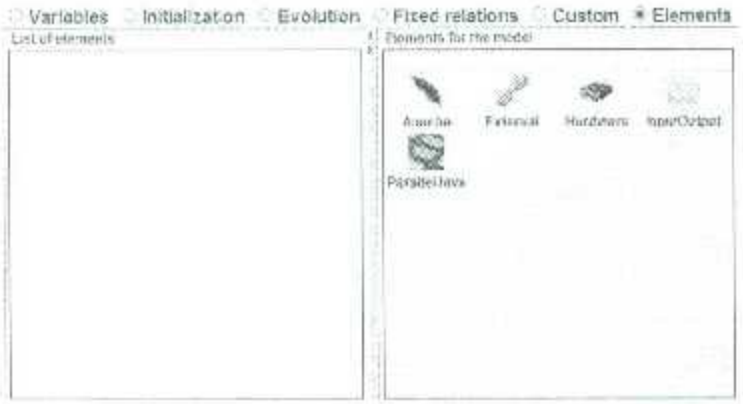

Figura 3.5: Panel para agregar elementos a EJS

un apoyo visual resulta de gran ayuda para que el alumno pueda observar los efectos y reacciones físicas que las leyes de control pueden tener sobre un sistema y así poder tener un mejor entendimiento de los conceptos que pueden resultar un poco abstractos. EJS provee una amplia cantidad de opciones para elaborar interfaces gráficas para nuestra simulación, desde gráficas en 2D y 3D, áreas de dibujo y los objetos básicos de programación visual como son los cuadros de texto, etiquetas, menús, etc. La ventaja que EJS ofrece es que la librería de objetos visuales están especializados en la visualización de datos y con fines experimentales, además su uso ha sido simplificado tanto como sea posible, lo cual facilita especialmente el proceso de hacer la simulación interactiva, que para propósitos pedagógicos es uno de nuestros principales objetivos.

3.2.1. Interfaz gráfica

Las interfaces gráficas de las simulaciones creadas en EJS se construyen a través de una estructura en árbol que contiene los elementos que se han seleccionado para formar la interfaz, presentados como una serie de bloques que se pueden observar en la figura 3.6.

Todos estos elementos gráficos son los bloques de construcción de la interfaz y cada uno de ellos ocupa un área de la pantalla creada y cumple con una función específica. Estos elementos están clasificados por tipo y cada uno tiene una serie de características internas llamadas propiedades que se pueden modificar con el fin de hacer que dicho elemento se vea y se comporte como es necesario. Las propiedades son por lo general valores estáticos, pero también pueden cambiar dinámicamente en el caso de elementos animados. Una de las propiedades más importantes de los elementos de simulación son las llamadas acciones,

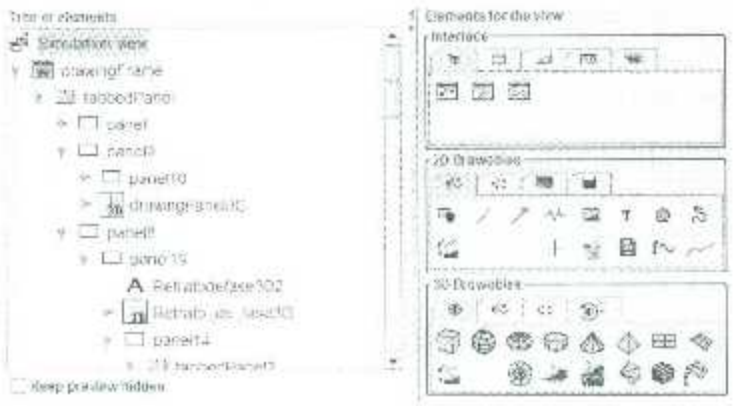

Figura 3.6: Panel para la creación de interfaces gráficas

que permiten definir la manera en que la interfaz se comportará cada vez que el usuario interactúe o modifique con la simulación, mediante acciones como un clic del mouse, o cambiar el contenido dentro de un cuadro de texto.

La librería gráfica de EJS es extensa, por lo que en este capítulo se enfocará en definir los tipos generales en que se clasifican los elementos, y para aquellos elementos individuales se abordará únicamente los utilizados en el presente trabajo. Los tipos en que se dividen los elementos gráficos son los siguientes: contenedores, elementos básicos y elementos de dibujo.

3.2.2. Contenedores

Los contenedores son aquellos elementos que, como su nombre indica, contienen a los demás elementos en un área en la pantalla de la computadora. En la jerarquía de árbol, el elemento contenedor es conocido como *padre*, mientras que todos los elementos contenidos son llamados *hijos*. En la figura 3.6, en la parte izquierda, se puede observar una lista de elementos, donde el segundo en la lista llamado *drawingFrame* es el contenedor padre, mientras que todos los que le siguen son los elementos hijos. El primer elemento padre, llamado *SimulationView*, no es propiamente un elemento y sirve únicamente como la raíz en la jerarquía de árbol de la simulación, el primer contenedor hijo de esta raíz se convertirá en la ventana principal de la simulación a menos de que otro elemento sea designado como la ventana principal. Esto se hace dando clic derecho en el elemento que se quiera marcar como la ventana principal, lo que hará que se abra un menú donde podemos seleccionar la opción *MainWindow*. Existen tres tipos de contenedores: Marco de dibujo, marco de

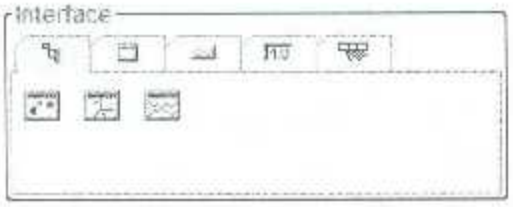

Figura 3.7: Lista de contenedores

dibujo 3D y marco de graficación. Se puede ver en la figura 3.7 el orden según fueron mencionados.

Marco de dibujo

El marco de dibujo, llamado en inglés *drawingFrame*, es el tipo básico de contenedores. En este contenedor podemos utilizar todos lo elementos gráficos básicos y todos los elementos de dibujo en 2D. Al agregarlo al árbol de la simulación, EJS nos provee una forma básica para ejemplificar la manera en que se estructuran estas ventanas. El marco de ejemplo contiene una figura en 2D de un círculo contenido en un entorno de dibujo, los botones de *play* y *reset* (que sirven para ejecutar y regresar la simulación a los valores iniciales respectivamente), y un cuadro de texto para modificar el valor de una variable. Esta ventana puede modificarse como se desee, pero se recomienda dejar intactos los botones de control, ya que sus propiedades están previamente especificadas con las acciones internas de control. En la figura 3.8 se observa la vista de una simulación creada en un marco de dibujo en 2D.

Marco de dibujo 3D

Este contenedor es básicamente igual al contenedor anterior, la diferencia principal es que en este tipo de contenedores es posible utilizar los elementos de dibujo en 3D. La forma básica que este contenedor presenta es casi exactamente igual a la del contenedor 2D, con la excepción de que en lugar de la forma del círculo se tiene una partícula 3D contenida en un entorno de dibujo en tercera dimensión. También contiene los botones de control anteriormente mencionados.

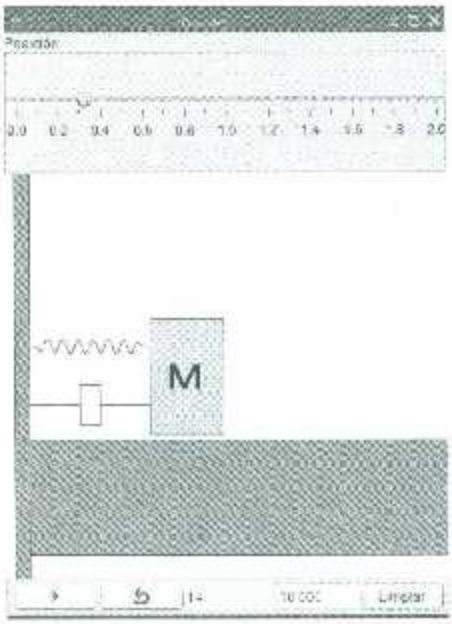

Figura 3.8: Marco de dibujo en 2D, sistema masa-resorte-amortiguador

Marco de graficación

El contenedor de graficación es más sencillo que los anteriores, conteniendo una gráfica que contiene una traza que puede ser asociada con dos variables para sus ejes x y y y los botones de control de la simulación.

Propiedades de los contenedores

Al dar doble clic sobre un elemento gráfico en el árbol que representa la vista de la simulación, aparecerá una ventana donde pueden modificarse los valores de sus propiedades. Para los contenedores, la gran mayoría de sus propiedades tiene propósitos explícitos, indicados por su nombre, como son: nombre, tamaño, color de fondo, visibilidad, tipo de letra, etc. La propiedad más importante, y que se explicará más a fondo, es la propiedad *Layout* o distribución, que aparece en la figura 3.9 en el tercer renglón de la primera columna.

Distribución en un contenedor

Entre la gran mayoría de programas que nos permiten crear aplicaciones con interfaces gráficas para el usuario, EJS es un poco más rudimentario a la hora de agregar elementos

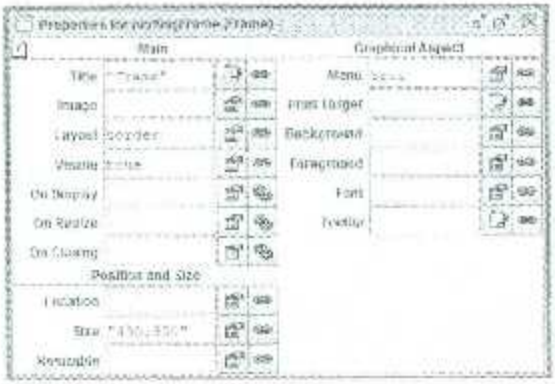

Figura 3.9: Propiedades de los contenedores

a las ventanas que se han creado. En general, se tiene una ventana y se eligen elementos de una lista para arrastrarlos a la posición deseada; también es posible cambiar su tamaño con un clic del ratón y moverlo libremente por la ventana. En EJS, en cambio, la posición y tamaño de los elementos gráficos están regidos por la distribución del espacio (*layout*) de su elemento padre. Algunas distribuciones determinan automáticamente la posición y el tamaño de los hijos, pero por lo general la posición es dictada por el orden que los hijos tienen en la jerarquía de árbol de la simulación.

Para un usuario principiante, esta manera de construir las interfaces gráficas puede resultar un poco confusa y algo molesta, por lo que es necesario familiarizarse con las distribuciones de los contenedores antes de comenzar a crear vistas de simulaciones. Con un poco de práctica y conocimiento previo, las primeras creaciones de vistas se volverán procesos menos frustrantes, y con el uso continuo, el entendimiento de cómo funciona la distribución de elementos se simplificará y en poco tiempo se sentirá como una manera más natural de diseño. A continuación se describen los principales tipos de distribución:

- *Flow* (Flujo): Distribuye a los hijos en una línea horizontal, justificados de manera similar a los párrafos creados en un procesador de texto. Los elementos hijos pueden ser justificados a la derecha, izquierda o centrados.

- *Border* (Borde): Esta distribución puede ordenar a los hijos en el borde del contenedor, ya sea arriba, abajo, izquierda o derecha. También puede colocar un hijo al centro del contenedor. Cuando se elige está opción, al agregar los hijos al contenedor padre aparecerá una ventana para que le sea asignado un lugar.

- *Grid* (Cuadrícula): Con esta distribución puede crearse una cuadrícula de $n \times n$ elementos en nuestro contenedor, donde todos los elementos tendrán la misma altura y anchura. La posición de los hijos es determinada por la posición del hijo en la jerarquía de árbol, el hijo en la posición más alta estará en el primer lugar de la cuadrícula y los demás hijos se irán colocando con el orden en que sean agregados. Sin embargo, no hay que olvidar que es posible mover a los hijos de lugar en la jerarquía para colocarlos en el recuadro deseado.

- *Horizontalbox* (Caja horizontal): La caja horizontal funciona de una manera similar a la cuadrícula, pero con una única fila y sin forzar a los hijos a tener el mismo tamaño.

- *Verticalbox* (Caja vertical): Similar al anterior, pero ahora con una sola columna y con hijos que pueden tener una altura diferente.

Se menciona como una recomendación, utilizar principalmente las distribuciones de cuadrícula y las cajas horizontales y verticales, puesto que se observó que logran crear una presentación un poco más uniforme y es más fácil determinar qué posición tendrán los elementos agregados al contenedor. Aunque la distribución de cuadrícula tiene la desventaja de forzar a todos los elementos en ella a tener el mismo tamaño, también puede resultar útil para una mejor presentación visual. Para finalizar, también existe una opción para determinar el espacio de separación tanto vertical como horizontal en los hijos, que se encuentra en la parte inferior de la figura 3.10.

3.2.3. Elementos básicos

El grupo de elementos básicos está formado por una serie de elementos que pueden ser utilizados para decorar la vista, editar o visualizar variables o invocar acciones de control. Los elementos básicos están agrupados en un recuadro con varias pestañas que podemos observar en la figura 3.11, la pestaña seleccionada contiene los elementos de entornos y marcos, que son los que se comenzarán a explicar.

Entornos y marcos

Estos elementos funcionan de una manera similar a la de los contenedores, con la diferencia de que estos entornos generalmente van dentro de un contenedor. De los elementos

Figura 3.10: Opciones de distribución

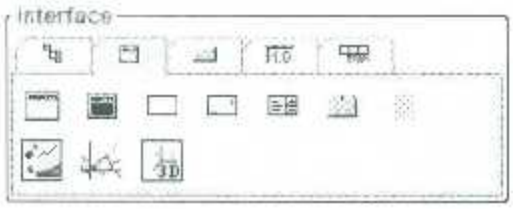

Figura 3.11: Elementos básicos, entornos y marcos

que observamos en la figura 3.11, los primeros dos son de hecho contenedores, el primero siendo una ventana vacía que funciona como los marcos mencionados en la sección de contenedores, y el segundo una ventana de dialogo. Los elementos detallados a continuación son los paneles, y los entornos de dibujo y gralicación.

Panel

El elemento de panel, en la figura 3.11 el tercer elemento de la primera fila, funciona de una manera similar a los contenedores y ayuda a agrupar y ordenar los demás objetos en el contenedor. Al igual que los contenedores, tiene la propiedad de distribución para posicionar diferentes elementos en su interior. Los paneles son extremadamente útiles para la agrupación de elementos, ya que en EJS no es posible seleccionar y mover varios elementos al mismo tiempo, pero si están dentro de un panel solo es necesario mover su panel padre para cambiar su posición. Otro uso que se puede dar a los paneles, es utilizarlos como espacio vacío en las cuadrículas, para una mejor presentación visual de los elementos. En la figura 3.11, después del panel común, está el panel desplazable que a diferencia del panel anterior tendrá barras de desplazamiento horizontal y vertical para desplazarse a través del mismo.

Entorno de dibujo en 2D

El entorno de dibujo 2D, como su nombre indica, es un área donde podemos colocar diferentes figuras y gráficos en dos dimensiones. A diferencia de los contenedores, los entornos de dibujo no tienen la propiedad de distribución y los elementos en su interior se posicionan de acuerdo con coordenadas (x, y) especificadas por el usuario, ya sean constantes para elementos estáticos, o en forma de variables para elementos que tendrán efectos de animación. En las propiedades del entorno de 2D el usuario puede establecer el valor mínimo y máximo de x y y y en base a esto definir las posiciones de los elementos dentro del entorno. En la sección de Elementos de dibujo en 2D se explicarán más a fondo los tipos de dibujos que pueden ser situados en este entorno.

Entorno de dibujo en 3D

El entorno de dibujo en 3D es casi completamente igual que el entorno en 2D, con la diferencia de que aquí se tendrán tres coordenadas de localización (x, y, z). Otra diferencia

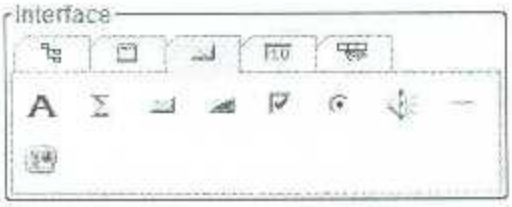

Figura 3.12: Elementos básicos, botones y decoración

importante, es que en el entorno de 3D no solo debe indicarse el tamaño total de entorno por medio de coordenadas, también debemos especificar el *zoom* y posición inicial de la cámara del entorno. Se recomienda dejar los valores predeterminados, pues cuando la simulación esté siendo ejecutada, es muy probable que el usuario rote la cámara para observar el contenido del entorno desde diferentes perspectivas. Un detalle importante a mencionar es que para crear gráficas de valores en 3D es necesario utilizar un entorno de dibujo, mientras que para crear gráficas en 2D se utiliza un entorno de graficación.

Entorno de graficación

El entorno de graficación crea una gráfica de dos ejes que puede ser cartesiana o polar con el fin de graficar diferentes variables. Es bastante simple de configurar, seleccionar sus valores máximos y mínimos, nombrar los ejes para identificar las variables que representan, mostrar la cuadrícula o cambiar los tipos de ejes. La parte más importante, a la hora de crear las gráficas, son los elementos que agregamos a su interior que serán descritos más adelante.

Botones y decoración

La siguiente pestaña de elementos básicos es la de los objetos de decoración y botones. Estos elementos pueden resultar extremadamente familiares para quienes hayan creado interfaces gráficas con anterioridad. El primer elemento que observamos en la figura 3.12 es una etiqueta, que como es sabido, es un texto estático. Le sigue una ecuación, que es un cuadro de texto que permite introducir ecuaciones matemáticas usando la sintaxis de LaTeX (a pesar de que la opción está disponible, cuando se utilizó este elemento durante el presente trabajo, las ecuaciones no se hacían visibles a la hora de ejecutar la simulación). Los dos elementos siguientes son botones que se asocian con métodos personalizados para llamar diferentes acciones en la simulación. Le siguen los *checkBox* y los *radioButton*, que

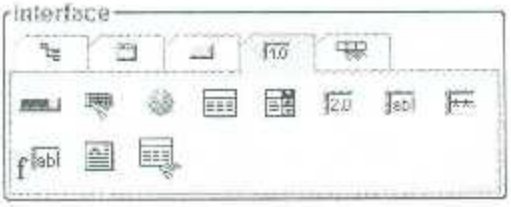

Figura 3.13: Elementos básicos, entrada y salida

se asocian con variables booleanas para modificar sus valores; el elemento representado con una bocina es también un *checkBox*, pero con sonido. Los últimos dos elementos son una línea de separación y una imagen que puede ser asociada a una cámara web para mostrar vídeo.

Entrada y salida

La siguiente pestaña nos muestra los llamados elementos de entrada y salida, cuyo principal objetivo es mostrar y modificar los valores de las variables del modelo. Podemos ver los elementos que conforman este grupo en la figura 3.13.

Para asociar una variable con uno de estos elementos, el proceso es el siguiente, primero se agrega un elemento de entrada/salida a nuestra simulación y damos doble clic sobre este, lo que llevará a ver sus propiedades. En la ventana de propiedades, que es como la que se observa en la figura 3.14, la primera propiedad es la llamada variable, se puede escribir directamente el nombre de la variable a la que queremos asociar el elemento o presionar el botón con la imagen de una cadena al lado derecho del cuadro de texto, lo cual abrirá una lista con las variables que pueden asociarse con el elemento seleccionado. Es recomendable usar la lista de variables para evitar asociar variables que no son compatibles con el elemento con el que se está trabajando.

En otros programas tradicionales para crear aplicaciones con interfaces visuales, el proceso de asociar variables es generalmente logrado usando métodos de programación de bajo nivel que necesitan que el usuario tenga un conocimiento más extenso del lenguaje de programación, mientras que EJS simplifica este proceso puesto que su librería de elementos gráficos está adaptada para el propósito de la creación de simulaciones. En EJS todo el código necesario para la conexión entre el modelo y los elementos visuales de la simulación son generados automáticamente, sin necesidad de que el usuario conozca el proceso necesario.

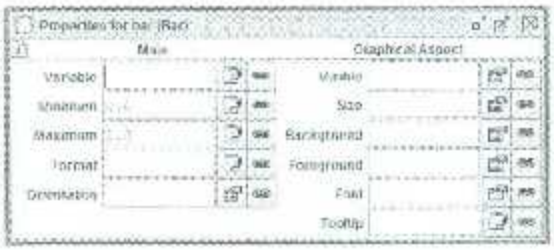

Figura 3.14: Propiedades de una barra

- Barra (*Bar*): El primer elemento es una barra que se llenará o vaciará dependiendo del valor de una variable numérica. Este elemento es únicamente para mostrar gráficamente el valor de una variable.

- Barra de desplazamiento (*Slider*): El elemento que le sigue es una barra de desplazamiento que contiene un indicador que se desplaza a través de la barra dependiendo del valor de su variable asociada. A diferencia de la barra anterior, en esta barra es posible que el usuario arrastre el indicador para modificar el valor de la variable durante la ejecución de la simulación.

- Perilla (*Knob*): Similar a la barra de desplazamiento anterior, pero ahora se presenta como una perilla. Al igual que con la barra de desplazamiento, el indicador de la perilla se moverá de acuerdo con el valor de su variable asociada y el usuario podrá girar la perilla para modificar ese valor.

- Tabla de datos (*Datatable*): Este elemento se asocia con una variable de dimensiones múltiples y muestra los valores de todos sus elementos.

- Lista (*ComboBox*): Es una lista que muestra una serie de elementos. Se asocia con una variable de texto que toma el valor de la cadena que esté escrita en el elemento seleccionado en la lista.

- Campo (*Field*): En apariencia, es un cuadro de texto, pero solo es posible escribir caracteres numéricos sobre este. Se puede utilizar tanto para mostrar como para modificar el valor de una variable numérica, ya sea un entero o un valor doble.

- Campo de texto (*TextField*): Similar al campo anterior, pero este elemento es compatible con variables *string*, por lo cual podemos escribir cadenas de caracteres alfanuméricos.

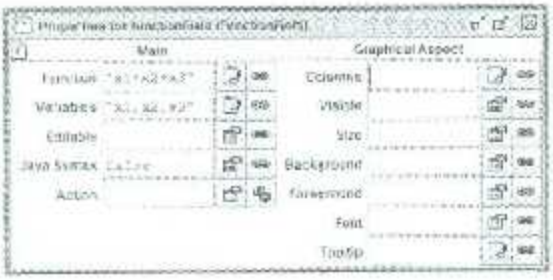

Figura 3.15: Propiedades de un campo de función

- Campo de contraseña (*PasswordField*): Idéntico al campo anterior, con la única diferencia que los caracteres no son mostrados, sino que son ocultados por un símbolo.

- Campo de función (*FunctionField*): Este es un campo extremadamente importante, puesto que permite escribir funciones que EJS puede evaluar. Las funciones son variables de texto, pero al evaluarlas se obtendrá un valor numérico. Debemos tener cuidado con las siguientes propiedades que se muestran en la figura 3.15. La primer propiedad es la de función, que es el equivalente a la variable asociada en los elementos anteriores. Es en ella donde se escribe la función a evaluar. Le sigue la propiedad *variables*, que es donde se especifica el nombre de las variables utilizadas en la función. Se debe tener cuidado a la hora de escribir la función en el campo para no utilizar variables no especificadas. Por ejemplo, si en la propiedad de variables se escribe "x1,x2", entonces tenemos dos variables llamadas x1 y x2, pero si el usuario escribe la función como "x1+x2−x3", entonces la simulación se encontrará con un error de cálculo puesto que la variable x3 no está especificada. Otra propiedad importante es la de la sintaxis de Java (*Java syntax*), que es una propiedad booleana. Si su valor es verdadero entonces cuando se define una función en el campo se tendrá que utilizar la sintaxis de Java, es decir, invocar la librería Math para operaciones más complejas. Si su valor fuera falso, entonces se puede escribir la variable como es generalmente escrita en otros lenguajes de programación. Para ejemplificar esto, con la sintaxis de Java se escribiría una función como "Math.sin(x1)−x2", mientras que sin la sintaxis la función sería "sin(x1) − x2". En la sección de cómo declarar las ecuaciones de evolución se dio un ejemplo de cómo evaluar las funciones mostradas en un campo de función.

- Área de texto (textArea): Es un cuadro de texto donde puede escribirse una serie de

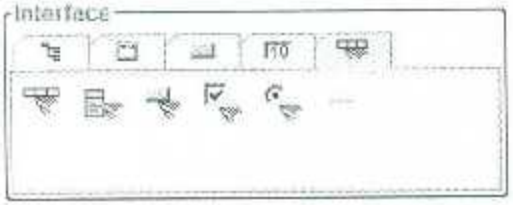

Figura 3.16: Elementos básicos, menús

líneas, a diferencia de las anteriores, solo sirve para mostrar un texto, que aunque puede modificarse no se asocia con ninguna variable.

- Panel de un arreglo (arrayPanel): Es casi idéntico a la tabla de datos anteriormente mencionada, con la diferencia de que este panel es editable para modificar los valores del los elementos del arreglo o variable dimensionada a la que este asociado.

Menús

En la última pestaña se visualizan los elementos que permiten crear menús horizontales en la parte superior del contenedor. Para crear un menú primero se agrega una barra de menús, que es el primer elemento que observamos en la figura 3.16. A continuación, con el siguiente elemento se crea un menú. Los elementos siguientes son cada uno de los objetos que conformaran el menú, que son elementos que ya se han explicado: botones, *checkboxes* y *radiobuttons*.

3.2.4. Elementos de dibujo 2D

Los elementos de dibujo son el principal aporte de EJS a la creación de simulaciones visuales. Consisten en una serie de elementos que pueden ser incluidos en los entornos de dibujo anteriormente mencionados para crear animaciones que ayudan a visualizar los diferentes estados y cambios en el modelo de la simulación.

Estas animaciones pueden ser tan simples o complejas como sea necesario, en dos o tres dimensiones, y contener una gran cantidad de objetos en su interior. A continuación se mencionará con detalle los elementos que pueden usarse en un entorno de dibujo en 2D.

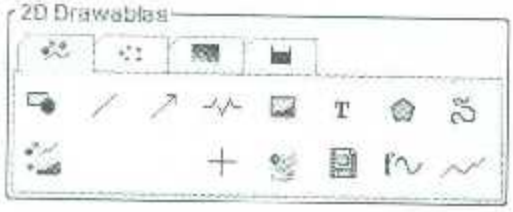

Figura 3.17. Elementos de dibujo en 2D, básicos

Elementos básicos

En la figura 3.17 se muestra la primera pestaña en la sección de los elementos de dibujo en 2D, que muestran los elementos básicos.

- Forma (*Shape*): Es el elemento más básico, que permite crear una forma ya sea elíptica o rectangular. Se puede animar este objeto de diferentes maneras, la primera es la de cambiar su posición. Al dar doble clic a la forma en 2D se abrirán sus propiedades y en ella se encuentran las propiedades de posición en x ($PosX$) y posición en y ($PosY$), que al asociarlas con una variable de valor dinámico harán que dicho objeto se mueva a través del entorno de dibujo. Otra manera de animar un es cambiar su tamaño, en la ventana de propiedades mencionada anteriormente, se encuentran las propiedades de tamaño en x ($SizeX$) y tamaño en y ($SizeY$) que permiten determinar el tamaño del objeto como un valor fijo, o si las asociamos a una variable entonces el tamaño de dicho objeto cambiará de acuerdo al valor de esa variable. En la ventana de propiedades también existen otras opciones para modificar el objeto como son el color y anchura de línea, el color de relleno, el estilo de la figura (elíptica o rectangular), etc.

- Segmento (*Segment*): Es otro elemento extremadamente básico; es simplemente una línea o segmento. Tiene básicamente las mismas propiedades que la forma y también es posible animarlo de manera similar. Un detalle que debe tenerse en cuenta con este elemento es el uso de las propiedades de tamaño en x y y cuya función es en realidad dictar la dirección de la línea. Por ejemplo, si se tiene una línea posicionada en la coordenada (0,0) del entorno de dibujo y en la propiedad de tamaño en x se tiene un 1 y en la propiedad de tamaño en y se tiene un 2, entonces se creará una diagonal que va desde la coordenada (0,0) a la coordenada (1,2).

- Flecha (*Arrow*): Casi exactamente igual al segmento, con la diferencia de que en lugar de ser una línea simple es una flecha.
- Resorte (*Spring*): Este elemento dibuja un resorte, cuya longitud puede ser asociada a una variable para simular el efecto de extensión o contracción de un resorte.
- Imagen (*Image*): Este elemento lee una imagen desde un archivo externo y la despliega dentro del entorno de dibujo. Se puede escalar dicha imagen para hacerla del tamaño deseado.
- Texto (*Text*): Es una etiqueta que puede ser utilizada en un entorno en 2D.
- Polígono (*Polygon*): Similar a la primera forma mencionada, con la diferencia básica que este elemento forma un polígono de tantos lados como el usuario desee. Esta figura puede ser animada en una manera igual que la forma básica.
- Rastro (*Trail*): A diferencia de los elementos anteriores, a pesar de que el rastro puede colocarse también en un entorno de dibujo en 2D, su principal uso es en el entorno de graficación. El rastro es quien nos permite graficar los valores de diferentes variables escalares. Para hacerlo, primero debe agregarse un rastro al entorno de graficación, después se da doble clic en el rastro para abrir sus propiedades. En la parte superior de la ventana de propiedades están las propiedades de entrada (input) donde se pueden elegir las dos variables a graficar. Por ejemplo, si se desea graficar el valor de la variable $x1$ con respecto al tiempo t, entonces en la propiedad de *inputX* se selecciona la variable t, del tiempo, y en la propiedad de *inputY* se selecciona la variable $x1$.
- Grupo (group): Este elemento sirve para agrupar elementos dentro del entorno de dibujo en 2D.
- Curva analítica (analytic curve): Este elemento permite dibujar una curva a través de una función. Para crear una curva analítica, en las propiedades primero se debe especificar el nombre de una variable para las funciones analíticas que se usarán para las coordenadas; después escribir las dos funciones que describen la forma de la curva en función de la variable que fue especificada. Por ejemplo, si en la propiedad de *variable* se escribe "x", ahora las expresiones en las propiedades de las coordenadas en X y Y deben escribirse en función de "x". Si en la propiedad X la expresión fuera

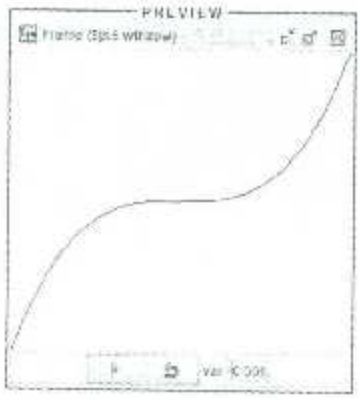

Figura 3.18: Ejemplo de curva analítica

simplemente "x" y en la propiedad de Y sea "x^3" se obtendrá la curva mostrada en la figura 3.18

- Traza (trace): La traza funciona de una manera similar al rastro, pero en lugar de ser asociada a dos variables escalares la traza es asociada a una única variable de dimensiones $n \times 2$ o $2 \times n$ y grafica la acumulación de los pares de puntos.

Los elementos que se omitieron fueron el cursor, que como su nombre lo indica, crea un cursor en el entorno de dibujo; el elemento de imagen de cámara (CamImage) que despliega una imagen dinámica al asociarse con una cámara web y el elemento de vídeo que lee un archivo de vídeo externo y lo reproduce en la simulación.

Otros elementos

En la siguiente pestaña de dibujo en 2D se encuentran básicamente los mismos elementos que en la primera, pero ahora para formar conjuntos de objetos en un solo elemento. La tercera pestaña muestra elementos para gráficas y campos, estos elementos no fueron usados en el presente trabajo, con excepción de los campos de vectores (los elementos tres y cuatro en la segunda fila), que pueden ser utilizados para crear retratos de fase.

En la figura 3.20 se muestra un ejemplo de un retrato de fase creado con el campo analítico de vectores. Primero, se nombran las dos variables, en este caso se uso x y y. Después solo se tienen que escribir las funciones que describen el retrato de fase del sistema en las propiedades de Componente X y Componente Y. Otro detalle importante a considerar son los números de puntos a graficar, también se pueden cambiar aspectos

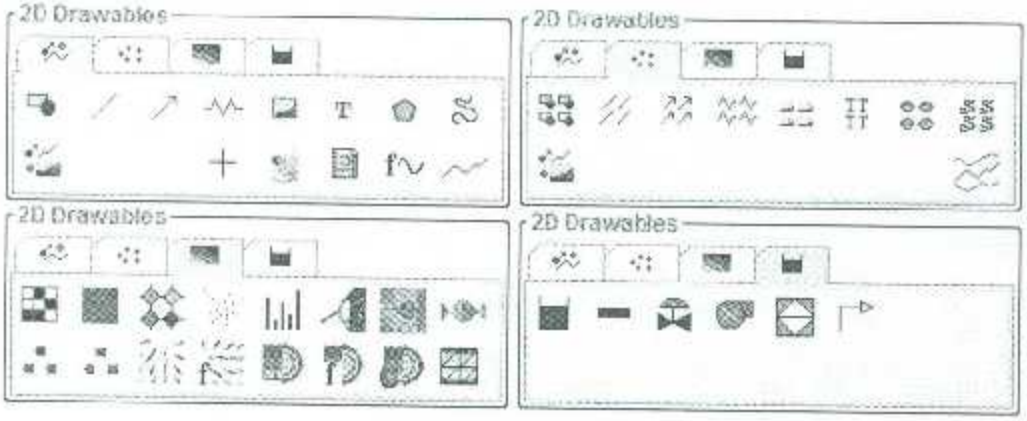

Figura 3.19: Elementos de dibujo en 2D

Figura 3.20: Ejemplo de un retrato de fase

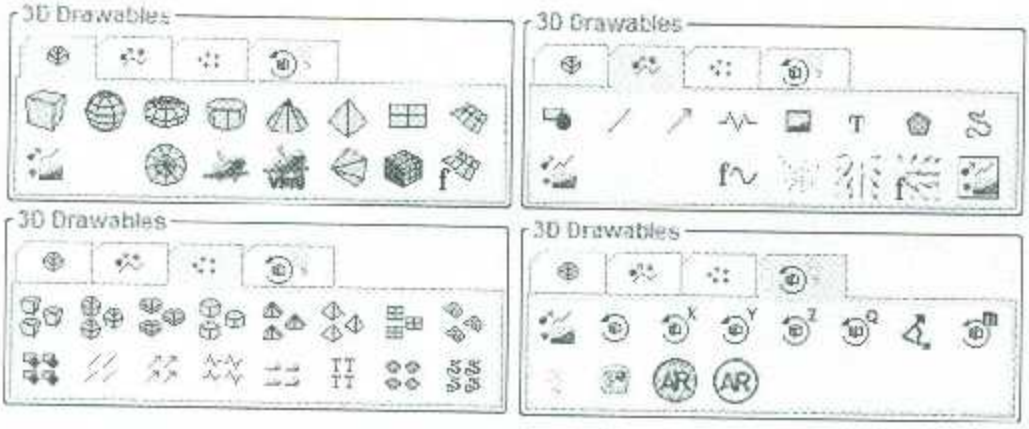

Figura 3.21: Elementos de dibujo en 3D

como el tamaño de las flechas o el color, para darle una mejor presentación al retrato de fase.

En la última pestaña de la figura 3.19 se tienen diferentes elementos de control, como son tanques, bombas, válvulas y tuberías. Estos elementos son muy útiles a la hora de crear simulaciones de sistemas que contienen estos elementos, sin embargo, para el presente trabajo no fueron utilizados.

3.2.5. Elementos de dibujo 3D

Como se mencionó con anterioridad también es posible utilizar gráficos tridimensionales en nuestras simulaciones, a continuación se detallarán los elementos de dibujo que podemos utilizar para este fin.

Objetos en 3D

En la primera pestaña de los elementos de dibujo en 3D encontramos los objetos en 3D, que son principalmente una serie de cuerpos tridimensionales que pueden modificarse como sea necesario: cubos, esferas, discos, cilindros, prámides e incluso planos. En la segunda fila pueden encontrarse dos elementos que son representados con la imagen del transbordador espacial, que son los objetos 3D y los objetos de realidad virtual. Los objetos 3D son modelos en 3D creados con el lenguaje de programación Java, con el paquete de Java 3D. Los objetos de realidad virtual son modelos importados desde archivos .vrml, que pueden ser creados con diferentes paquetes de dibujo asistido por computadora y que

Figura 3.22: Modelo VRML de un robot de dos grados de libertad en EJS

ayudan a crear mejores modelos para utilizarse en la simulación.

En la figura 3.22 se observa un modelo VRML de un robot de dos grados de libertad que fue importado a una simulación de EJS. Hay que notar dos desventajas en el uso de estos modelos: la primera es que el modelo se importa sin la información del color, así que en EJS aparecerá como un cuerpo completamente blanco, lo que hace un poco difícil reconocer detalles en el modelo (por lo que si se planea crear un modelo VRML para utilizar en una simulación de EJS se recomienda que este no sea muy elaborado, puesto que los detalles más finos se perderán). La otra desventaja es que las simulaciones que contienen este tipo de figuras pueden volverse lentas a la hora de crear animaciones, dependiendo del poder de procesamiento de la computadora en la que se esté trabajando, y en el tiempo total de simulación. Aún así, ya que para crear modelos en Java 3D por medio de código se necesita un alto grado de conocimiento en programación en Java, mientras que el uso de paquetes de dibujo asistido por computadora es más conocido por los estudiantes, los objetos VRML son la opción más viable cuando se requiera cuerpos tridimensionales un poco más sofisticados que los ofrecido por EJS.

Elementos básicos en 3D y conjuntos

En la segunda pestaña de la figura 3.21 se ubican los elementos de dibujo básico, donde encontramos los mismos elementos que los elementos básicos para el entorno en 3D, siendo la única diferencia que ahora también será necesario considerar el eje z a la hora de definir la posición y tamaño del objeto.

En la tercera pestaña están los conjuntos de objetos 3D, que al igual que los elementos básicos, funcionan de manera similar a sus equivalentes en 2D, pero ahora para los cuerpos

Figura 3.23: Rotación de las articulaciones del robot de dos grados de libertad

tridimensionales.

Herramientas y utilidades

En la cuarta pestaña de la figura 3.21 se encuentra esta serie de elementos, los más importantes a notar son los elementos de rotación, identificados por un cubo y una flecha. Estos elementos se agregan como hijos a un cuerpo en 3D para crear el efecto de rotación. En la figura 3.23 puede verse un ejemplo de cómo son utilizados en el robot de dos grados de libertad. Los elementos de rotación pueden especificar el eje con respecto al cual rotarán a su objeto padre. En el caso del elemento de rotación que no especifica el eje el usuario tendrá que especificarlo en las propiedades del elemento. También existe un elemento para rotar por medio de cuaterniones (Q) y otro para matrices de transformación (m).

En la figura 3.23 se observan las propiedades de una rotación con respecto al eje z. En la propiedad de ángulo (*angle*) se introduce el ángulo de rotación, que en este caso es la variable que representa la posición angular de la primera articulación del robot. En la propiedad de origen (*origin*) se especifica el punto de origen de la rotación, en coordenadas x,y,z. Si se escribiera "0,0,0" se toma como origen el centro del cuerpo. La última propiedad simplemente habilita o deshabilita la rotación.

3.3. Acciones

Otra particularidad de las simulaciones corresponde al uso de acciones cuando el usuario interactúe con la vista de la simulación. En EJS, se puede asociar a los elementos que forman la simulación por medio de las propiedades de cada elemento. En la figura 3.24 se muestran las propiedades de un *ComboBox*. En la parte inferior, con letras rojas se encuentra la propiedad de acción (*Action*). Es aquí donde se pueden asociar métodos

Figura 3.21: Propiedades de un ComboBox con la ventana de acciones disponibles

personalizados o predeterminados a nuestro objeto.

Al presionar el botón a la derecha del campo de acciones, el botón con dos engranes, se abrirá una lista con todos los métodos disponibles asociables al objeto. Los métodos creados por el usuario aparecen escritos en rojo y en la parte superior de la lista, mientras que los métodos predeterminados de EJS aparecen escritos en azul y están acompañados por una descripción de qué acción realizarán. Una vez agregada una acción, esta se ejecutará cada vez que el usuario interactúe con el objeto durante la simulación, que por ejemplo, en el caso de *ComboBox* sería seleccionar uno de los elementos en su lista.

3.4. Crear ejecutables

Ya que las aplicaciones de simulación serán creadas para distribuirse y no todas los usuarios tienen a la mano los recursos necesarios para utilizar EJS, también es posible crear ejecutables de nuestras simulaciones que pueden ser utilizados en cualquier computadora que tenga una máquina virtual de Java.

El proceso es extremadamente sencillo, solo debe presionarse el botón de empaquetar, que en la figura 2.3 es el antepenúltimo en la columna de la derecha. Después de presionarlo EJS creará automáticamente un archivo ejecutable .jar que se guardará en la carpeta llamada "*output*", en el directorio de EJS. Un detalle a cuidar a la hora de crear los ejecutables de las simulaciones, si tenemos archivos externos, que pueden ser imágenes, archivos de audio y vídeo o modelos .vrml, estos deben estar guardados en la carpeta "*source*" de EJS. Aunque a la hora de crear simulaciones se puede llamar estos archivos desde cualquier locación en nuestra computadora, cuando se crea un ejecutable estos no estarán disponibles para lectura a menos de que, como se mencionó anteriormente, se

encuentren en la carpeta "*source*" en el momento de la creación del archivo ejecutable.

3.5. *Launcher*

Launcher es una herramienta distribuida junto con EJS para resolver el problema de la distribución y presentación de las simulaciones creadas con EJS. *Launcher* es una aplicación de Java que sirve para presentar las aplicaciones creadas con EJS como programa ejecutable en un solo paquete, siendo lo único necesario para ejecutarlas una máquina virtual de Java presente en la computadora del usuario.

Las aplicaciones creadas con *Launcher* se ejecutan al dar doble clic en el nombre del archivo y después *Launcher* automáticamente despliega el programa especificado. *Launcher* presenta la colección de simulaciones en una estructura de árbol que es posible organizar en carpetas de acuerdo con el tema o nivel de estudios. También se puede acompañar las simulaciones con una descripción en un archivo de texto o html. Para ejecutar una simulación se da doble clic al nodo que representa la aplicación, representado con una flecha verde.

Para crear una aplicación con *Launcher* se utiliza el mismo botón que para crear los ejecutables .jar. Se da clic derecho en el botón de empaquetar (mencionado anteriormente para crear los ejecutables), que abrirá un menú donde se selecciona la opción "*Package simulations with Launcher*", que abre un menú con todas las simulaciones disponibles en la carpeta de "*source*", solo deben seleccionarse las simulaciones deseadas y EJS creará automáticamente la aplicación. Una vez creada la aplicación, está puede encontrarse en la carpeta de "*export*" de EJS.

Si es necesario modificar aspectos de la aplicación *Launcher*, solo debe abrirse, y en el menú superior de "File->Edit", se abrirá la ventana donde se pueden hacer una serie de modificaciones a la aplicación creada. La figura 3.25 muestra el editor de *Launcher*. Para un manual completo y más detallado del uso de la herramienta de *Launcher* ver referencia [3].

Figura 3.25: Aplicación creada con *Launcher*

Capítulo 4

Uso de las simulaciones creadas

En este capítulo se detallará cómo utilizar las simulaciones que fueron creadas en EJS, con el fin de ayudar al usuario a no encontrarse con dificultades durante el uso de las simulaciones. Se describirán los diferentes procesos para utilizar las aplicaciones, así como lo que es posible realizar con ellas y las diferentes opciones que ofrecen.

4.1. Iniciar la aplicación

Iniciar las aplicaciones es un proceso sencillo, existen dos maneras de hacerlo dependiendo de los archivos que tengamos a nuestra disposición y el propósito con el que estemos usando la simulación.

4.1.1. Iniciar con *Launcher*

La primera opción es iniciar utilizando la aplicación creada con *Launcher*. Esta opción es utilizada cuando solo tenemos disponibles los ejecutables .jar de las simulaciones y el lanzador de aplicaciones, o si la computadora con la que estamos trabajando no cuenta con el programa de JDK. También es recomendable utilizarla si nuestro único propósito es realizar algunos experimentos con la simulación desarrollada.

Como se mencionó con anterioridad, lo único que debemos hacer es abrir el ejecutable de *Launcher*, que nos mostrará la lista de simulaciones que corresponden a la colección de ese lanzador, y solo tenemos que dar doble clic en el nodo de la simulación deseada para que esta arranque.

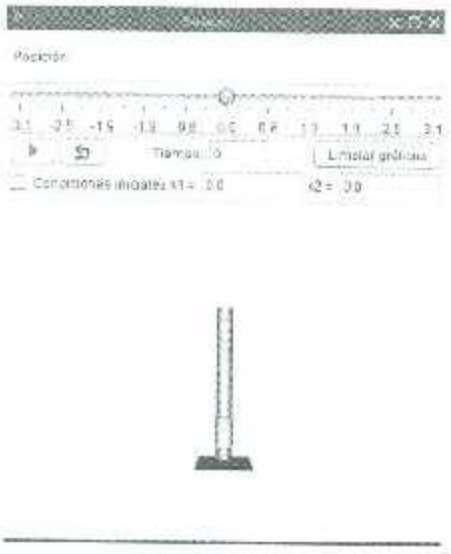

Figura 4.1: Ventana que muestra el modelo 3D del sistema

4.1.2. Iniciar con EJS

La segunda opción es iniciar la simulación con EJS. Está opción es recomendable principalmente cuando no solo queremos utilizar la simulación, sino cuando se requieran hacer modificaciones a esta. Como se mencionó en el capítulo de instalación, para utilizar esta opción es necesario tener instalado JDK y Java 3D para poder utilizar EJS en su totalidad.

Para abrir una simulación con EJS primero se ejecuta el archivo *Ejsconsole.jar* que se encuentra en la carpeta de EJS, lo que abrirá las dos ventanas mostradas en las figuras 2.2-2.3. En la ventana de 2.3, en la columna derecha se presiona el botón de "Abrir archivo" que nos permite seleccionar el archivo de simulación que deseamos abrir. Seleccionando el archivo .ejs de la simulación para abrirla, ahora podrá ser modificada si así se desea. Si se desea ejecutar la simulación, se presiona el botón en forma de flecha verde que puede verse en la columna derecha de la ventana en la figura 2.3.

4.2. Uso general

4.2.1. Sistemas de una entrada y una salida

Primero se dará una explicación de cómo utilizar las simulaciones de los sistemas que se crearon de una entrada y una salida como son: péndulo simple, el sistema masa-resorte-amortiguador, el circuito RLC y el motor de corriente directa. Una vez iniciada la simulación se despliegan dos ventanas ante el usuario que se muestran en las figuras 4.1 y 4.2. Comenzando con la ventana de la figura 4.1, en la parte superior se encuentra una barra de desplazamiento que indica y permite al usuario cambiar la variable de salida del sistema (en el ejemplo mostrado, la posición angular del péndulo). En la segunda fila de elementos, está el botón de *Play* que inicia la simulación. No debemos confundir este botón con aquel que ejecuta la aplicación. El botón de *Play* es para iniciar el experimento después de ejecutar la aplicación de la simulación. Cuando se inicia la simulación este botón cambia al botón de *Pause* y sirve para detener la simulación, pero las variables del sistema mantienen los valores obtenidos durante el experimento.

El siguiente botón es el de *Reset*, que regresa la simulación a sus valores iniciales, que los valores especificados para cada variable en la tabla de variables. A continuación está el campo de texto del tiempo, donde el usuario indica cual será el tiempo total de simulación (en caso de dejarlo en cero, la simulación continuará por un tiempo indefinido hasta que el usuario decida detenerla). La función del siguiente botón está explícita en su nombre, limpia las gráficas obtenidas en la simulación. La tercera fila es la de condiciones iniciales, si el *CheckBox* está seleccionado es posible modificar las condiciones iniciales de los estados de nuestro sistema, en caso de no utilizarlas estas siempre iniciarán en cero.

En la parte inferior se muestra un modelo en 3D (o en el caso del sistema masa-resorte-amortiguador, un dibujo en 2D) del sistema. Esta representación del sistema está animada y cambia de acuerdo con la evolución de los estados durante la simulación.

En la siguientes ventana encontramos las opciones para configurar la simulación. La primera pestaña es llamada "Sistema" y es mostrada en la figura 4.2. En la sección de "Datos del sistema", en la parte derecha de la ventana, podemos encontrar una serie de campos de texto para definir los valores de los parámetros de nuestro sistema, el valor predeterminado es 1, pero el usuario puede reemplazar estos valores por los que desee. En la parte inferior encontramos las ecuaciones en espacio de estados que describen el comportamiento del sistema. En la parte derecha está un diagrama en 2D del sistema.

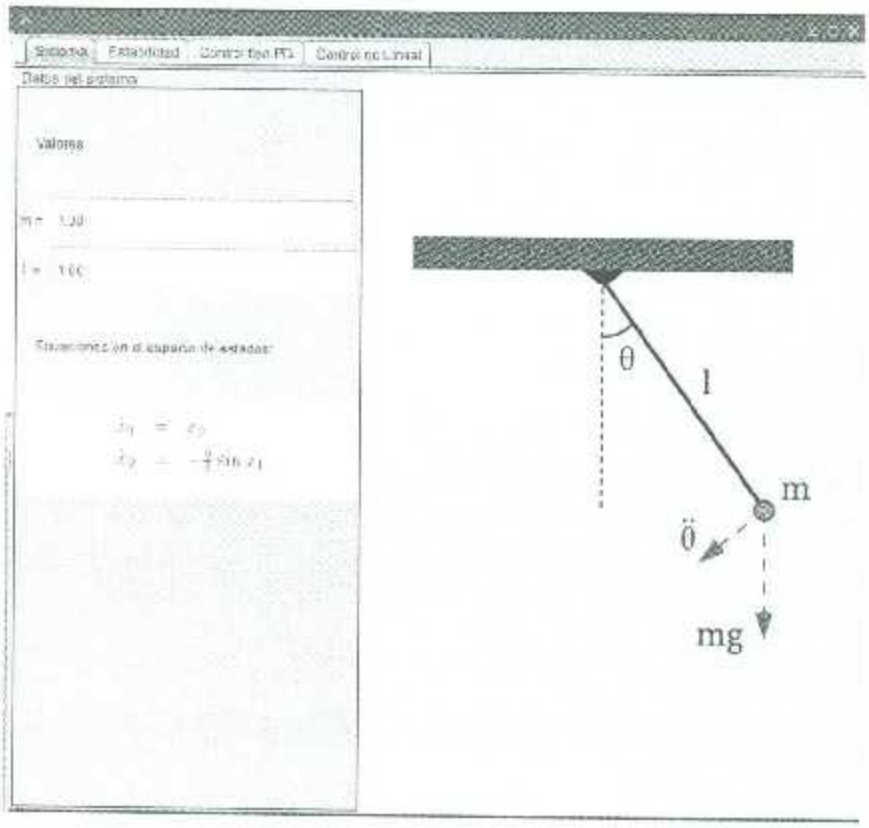

Figura 4.2: Ventana de opciones y graficación, mostrando la pestaña de Sistema

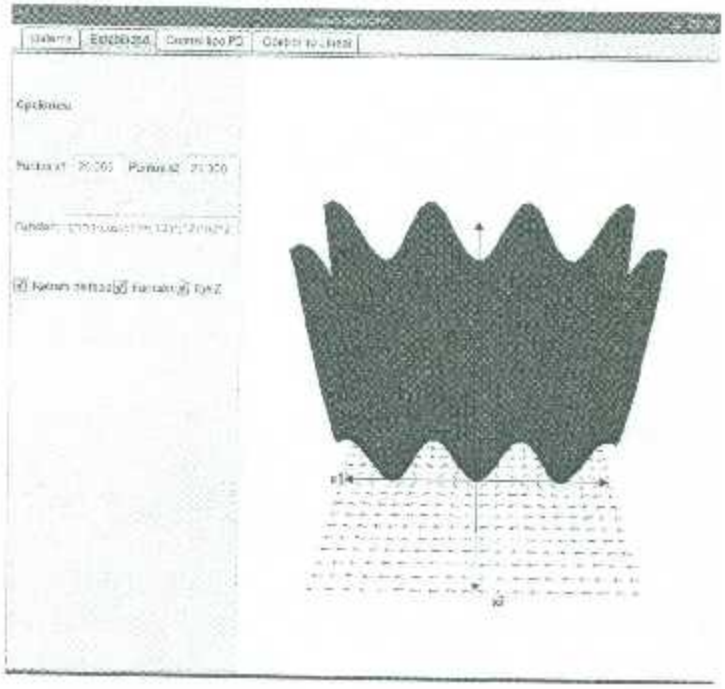

Figura 4.3: Pestaña de Estabilidad

A continuación encontramos la pestaña de "Estabilidad" en la figura 4.3. En esta pestaña, en la parte derecha de la pantalla, se encuentra una gráfica en 3D que nos muestra el retrato de fase de nuestro sistema en lazo abierto. En la parte derecha encontramos una serie de opciones para personalizar el retrato de fase. En el campo de texto de *funcin* se puede escribir la ecuación para una función tridimensional a mostrar junto con el retrato de fase para mostrar estabilidad.

En la siguiente pestaña podemos seleccionar los controladores PD, mostrada en la figura 4.4. En la parte izquierda superior de la ventana se encuentran las opciones. Hay dos *RadioButton* para elegir si se utilizará un controlador de regulación o uno de seguimiento. Debajo de cada uno de los *RadioButton* se encuentra un *ComboBox* con los controladores disponibles. Para regulación se tiene el controlador PD, PD con compensación de gravedad, PD con compensación precalculada y PID. Para seguimiento se tienen los controladores PD+, PD con compensación y Par Calculado.

Cuando se haya seleccionado un controlador, en el campo de texto de "u=" se escribirá automáticamente la ley de control elegida. Las leyes de control escritas en el campo funcionan como un ejemplo o como una base, pueden ser dejadas tal y como están o ser

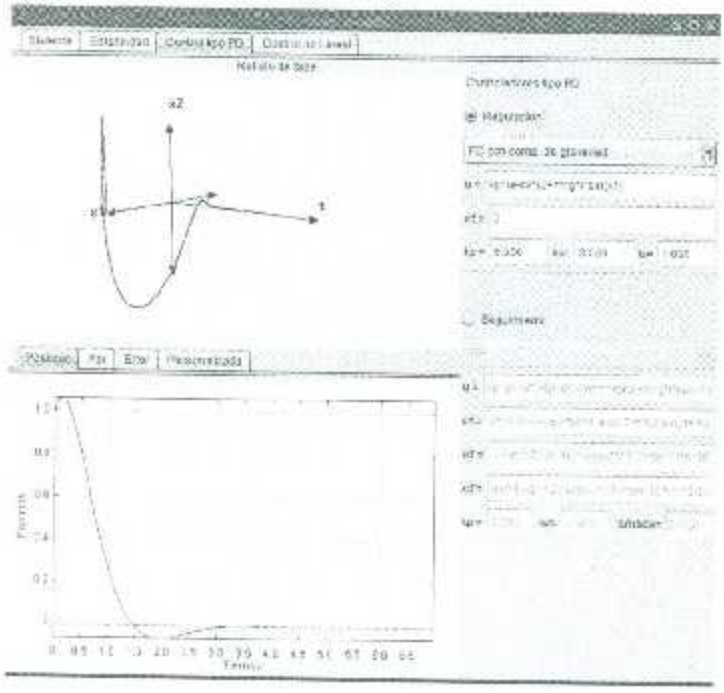

Figura 4.4: Pestaña de controladores PD

modificadas por el usuario. En el campo de "xd..." se escribe la salida deseada para el sistema, en el caso de los controladores en regulación, debemos recordar que la posición deseada debe ser expresada en radianes; mientras que para los controladores de seguimiento también se debe proporcionar al sistema la primera y segunda derivada de la función de salida deseada. Para los controladores en seguimiento, la simulación también brinda una función de ejemplo para la salida, pero al igual que con las leyes de control esta puede ser modificada o reemplazada por la que el usuario desee. Por debajo de los campos de texto donde se escribe la salida deseada se encuentran los campos para modificar las ganancias las cuales el usuario puede definir libremente.

En la parte superior derecha de la pantalla se encuentra el retrato de fase pero ahora asociado al sistema en lazo cerrado; en esta gráfica tridimensional podemos observar como evoluciona la trayectoria de la salida a través del tiempo de simulación mediante una traza tridimensional. En la parte inferior se encuentran cuatro pestañas donde los resultados del experimento serán graficados. En la primera pestaña se tiene una gráfica de la salida con respecto al tiempo, la segunda pestaña muestra una gráfica de la entrada con respecto al tiempo, en la tercera pestaña se muestra la gráfica del error con respecto al tiempo y en la

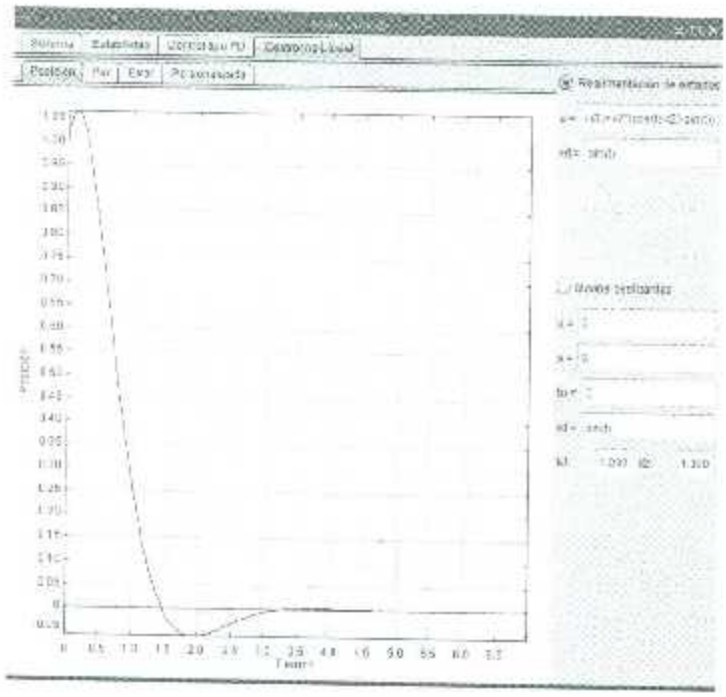

Figura 4.5: Ventana de controladores

gráfica de personalizada es posible elegir de una lista dos variables a graficar. Las variables disponibles son: los dos estados del sistema y sus derivadas, el tiempo, el error y la energía total.

En la última pestaña se tienen los controladores no lineales, donde tenemos dos ejemplos: el control por realimentación de estados y el control por modos deslizantes. Al igual que en la pestaña anterior, primero elegimos con los *RadioButton* el tipo de controlador deseado, si elegimos el control por realimentación debemos proporcionar al sistema la ley de control y la función de entrada. Para el control por medio de modos deslizantes también se deben proporcionar las funciones s y $\dot{s}x$, así como la posición deseada. En la parte derecha de la pantalla se encuentran las gráficas de las mismas variables que para los controladores PD, solo que esta vez se graficarán los resultados de los controladores no lineales.

Para resumir, se muestran los pasos para realizar un experimento con esta aplicación:

1. Ejecutar el archivo de la simulación, con cualquiera de los dos métodos mencionados.

2. En la pestaña de sistema, cambiar los valores de los parámetros si es necesario.

3. Seleccionar el tipo de control, ya sea PD o no lineal.

4. Seleccionar el controlador.

5. Introducir la salida deseada y los valores de las ganancias.

6. Si es necesario para el experimento, modificar los valores iniciales en la pantalla mostrada en la figura 4.1.

7. Una vez que todas las preparaciones están listas, presionar el botón de *Play* para comenzar la simulación.

8. Durante la simulación se puede mover la barra de desplazamiento de la salida para crear perturbaciones al sistema.

9. Una vez terminado el experimento, presionar *Pause* para detener.

Una vez pausada la simulación se pueden modificar todas la especificaciones anteriores, y si se desea regresar el sistema a su estado inicial se presiona el botón de *Reset*.

Almacenamiento de las gráficas obtenidas en un archivo de imagen

Dado que al realizar experimentos es necesario reportar los resultados, en la simulaciones creadas con EJS tenemos la opción de guardar las gráficas obtenidas en archivos de imagen con posibles diferentes formatos. Se puede tomar simplemente una captura de pantalla utilizando la función de "Imprimir pantalla" de la computadora, pero EJS también incluye una herramienta para realizar la captura de las gráficas que generamos. Para guardar las gráficas lo primero que debemos hacer es dar clic derecho en el entorno de dibujo o en el entorno de graficación el cual se requiera exportar en una imagen, esto abrirá el menú que se observa en la figura 4.6. Se selecciona la opción de elementos "Elements option", que despliega otro menú donde existe una lista de los entornos de dibujo y graficación de nuestra simulación. De nuevo, se selecciona el elemento deseado y esto desplegará un tercer menú, donde se dá clic en la opción de "Snapshot". Cabe hacer notar que a pesar de que al abrir la lista de entornos de simulación es posible seleccionar todos los entornos existentes en la simulación, si se elige abrir la herramienta de "Snapshot" para un entorno diferente al que se hizo clic derecho para abrir el menú se desplegará una ventana en blanco sin la gráfica deseada.

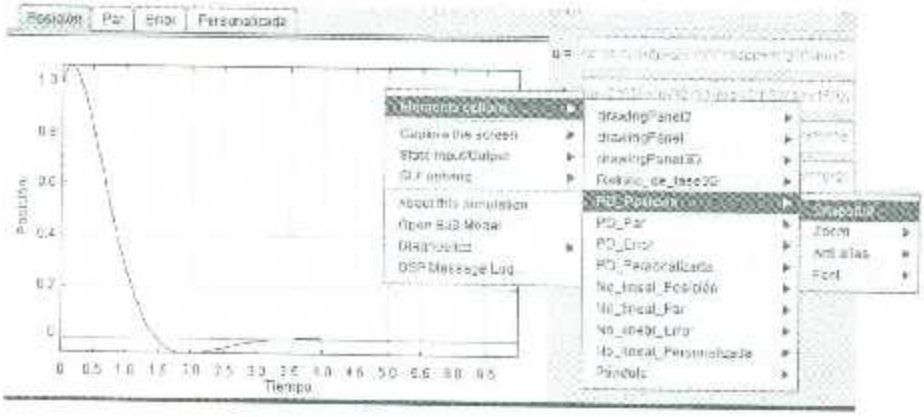

Figura 4.6: Menú de captura de gráficas

Después de dar clic sobre "Snapshot" la herramienta para la captura de gráficos se abrirá. Aquí el usuario puede hacer unas pequeñas modificaciones en la gráfica. Al dar clic derecho sobre el área de graficación, que es el menú mostrado en la figura 4.7. Con *ZoomIn* se pueden hacer acercamientos a un área de la gráfica, dependiendo del lugar donde se haya dado clic, el área donde se hará el acercamiento se muestra como un recuadro color magenta; y con *ZoomOut* se hace un alejamiento. Con la opción de *Autoscale* la gráfica regresará al zoom que tenía por defecto. Si queremos cambiar los valores máximos y mínimos de los ejes x y y de nuestra gráfica seleccionamos la opción de *Scale*, que abrirá una pequeña pantalla donde podemos cambiar estos valores. Ahora, para guardar solo se selecciona "File->Save as..." y se elige el formato de imagen con el que se deseа guardar el archivo y la carpeta donde se quiera guardarla. EJS generalmente utiliza la carpeta *output* pero se puede elegir cualquier ubicación disponible de la PC.

Si en cambio, se desea tomar la captura de un entorno de dibujo en 2D o 3D, lo que se realiza es dar clic derecho en el entorno que se desea capturar, y en el menú desplegado seleccionar "Capture the screen->Snapshot", lo que nos lleva directamente a la ventana para seleccionar la carpeta donde el archivo será guardado. Es necesario mencionar que no es posible utilizar este método de captura cuando el modelo 3D esté utilizando la implementación de Java 3D, si se quiere realizar la captura con este método se debe cambiar la implementación a Simple 3D en la ventana de propiedades del entorno de dibujo 3D.

En la figura 4.8 se observa la diferencia entre un modelo con la implementación simple en 4.8a y la implementación Java 3D en 4.8b. En la implementación simple el modelo

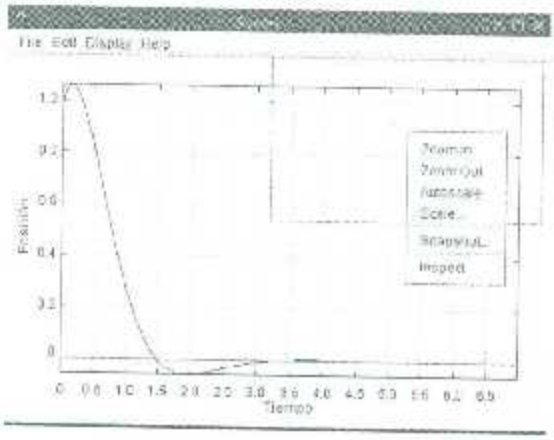

Figura 4.7: Herramienta para captura de gráficas

(a) Modelo con la implementación de 3D simple (b) Modelo con la implementación de Java 3D

Figura 4.8: Diferencias del tipo de implementación de objetos 3D

se ve un poco más burdo y una malla cubre la superficie del cuerpo, mientras que con Java 3D la superficie del modelo es de un color sólido, pero se muestran los efectos de la iluminación. Sin Java 3D los modelos de VRML no pueden ser visualizados, así que para el robot de dos grados de libertad no es posible tomar una captura de pantalla utilizando las herramientas de EJS.

Cómo agregar una nueva ley de control

Uno de los objetivos de la simulación es probar diferentes leyes de control, por lo que si el usuario desea probar leyes diferentes a las ofrecidas en la simulación, ahora se describirá el proceso para agregar una nueva ley de control. Se pueden utilizar dos métodos, uno para una ley que se desee probar en una sola ocasión, o si se desea agregar la ley permanentemente al programa.

Para utilizar una ley de control en una sola ocasión lo único que debe hacerse es escribir la ley de control en el campo de entrada " u = " y presionar el botón "Enter" (si el fondo del campo de texto se vuelve rojo, quiere decir que hay un error de sintaxis en la ley de control o se está utilizando una variable no existente en el modelo del sistema), se recomienda utilizar los campos de los controladores PD y utilizar el campo de entrada dependiendo si la ley de control es para regulación o seguimiento. Si una vez se haya probado la ley proporcionada por el usuario se selecciona otra ley de control de los ejemplos que se proveen en la simulación, la ley que el usuario escribió será reescrita con el ejemplo seleccionado.

Como agregar una ley de control permanentemente

Los pasos para agregar una nueva ley de control permanentemente serían los siguientes, asumiendo que el usuario abrió la simulación desde EJS y tiene instalado correctamente el programa de JDK:

1. Si la ley de control es para regulación, abrimos las propiedades del *ComboBox* para seleccionar los controladores PD en regulación, en el caso que sea una ley de seguimiento se abrirán las propiedades del *ComboBox* apropiado.

2. Agregar el nombre de la simulación al *ComboBox* de opciones: Primero nos dirigimos a la vista de la simulación para poder modificar las propiedades de los *ComboBox*. En la figura 3.24 se observan las propiedades de un *ComboBox*. En la propiedad de "Options" debe escribirse un nombre para identificar la nueva ley de control, separada

por un punto y coma de las leyes anteriores, y se presiona "Enter" para guardar los cambios.

3. Ahora es necesario dirigirse al modelo de la simulación y en la sección de métodos personalizados. Si el controlador es para regulación, nos dirigimos a la sección llamada "Seleccionar controlador PD en regulación" y agregamos una nueva entrada al método *seleccionar*, el nuevo código sería el siguiente:

```
if(((tipo).equals("(nombre de la ley de control)"))&&
(_view.tabbedPanel.getSelectedIndex()==2))
{
func="(expresión de la ley de control)";
}
```

Que se puede copiar de una de las otras entradas y pegar al final del método, para solo hacer los cambios necesarios de nombre y ecuación de la ley de control.

4. Si en cambio, se agregara una ley de control en seguimiento, nos dirigimos a la sección "Seleccionar controlador PD en seguimiento" y creamos una nueva entrada en el método *seleccionar3* con el siguiente código:

```
if((tipos).equals("(nombre de la ley de control)"))
{
_view.function5.setFunction("(expresión de la ley de control)");
ents="(entrada)";
derent="(derivada de la entrada)";
dderent="(segunda derivada de la entrada)";
}
```

De nuevo, lo importante a cambiar son el nombre y la expresión que define la ley de control. En las expresiones para definir la entrada y sus derivadas podemos dejar la misma que en las demás entradas, o cambiarla por si se necesita probar una función de entrada en especial.

5. Guardamos el proyecto de simulación, y si los pasos anteriores fueron realizados correctamente, ahora en la lista de leyes de control disponibles se tendrá una nueva entrada lista para usarse.

4.2.2. Robot de dos grados de libertad

Otra de las simulaciones creadas es la del robot manipulador de dos grados de libertad. El robot utilizado es el robot prototipo tomado de la referencia [2]. La simulación del robot de dos grados de libertad es similar en uso a las simulaciones de los sistemas de una entrada y una salida. A continuación se describirá el uso de de esta simulación.

Al abrir la simulación, con los métodos mencionados con anterioridad, ante el usuario se desplegarán las dos ventanas principales que conforman esta simulación. La primera ventana, mostrada en la figura 4.9 contiene el modelo en 3D del robot así como los controles principales de la simulación.

En la parte superior de la ventana se encuentran dos barras deslizantes, que al igual que con los sistemas de una entrada una salida, permiten modificar las variables de salida del sistema para crear perturbaciones durante la simulación o cambiar los valores iniciales de las variables, en este caso las posiciones angulares de las articulaciones del robot. La posición de ambas articulaciones está expresada en radianes. Debajo de las dos barras deslizantes se encuentra el modelo 3D del robot, que como se explicó anteriormente, solo es visible si la aplicación de Java 3D está instalada en la computadora donde se esté ejecutando la simulación.

En la parte inferior de la ventana está ubicado un conjunto de controles para la simulación, el primero es el botón de *Play* seguido por el botón de *Reset*. Después se tiene el campo para indicar el tiempo de simulación, que al igual que con las simulaciones anteriores, si se deja en cero la simulación continuará hasta que el usuario decida detenerla. En los dos últimos se proporciona la información del sistema, en "Ver parámetros" se abrirá una ventana con los parámetros del sistema y sus valores. En el siguiente botón, llamado "Ver ecuaciones", se abrirá una ventana con el modelo dinámico del robot.

En la segunda ventana se encuentran, al igual que con las simulaciones anteriores, una serie de campos para configurar las leyes de control utilizadas en la simulación. Esta ventana se divide en un panel superior con dos pestañas, donde podemos elegir entre leyes de control de seguimiento y leyes de control en regulación; y el panel inferior tenemos tres pestañas: en la primera se muestran las gráficas de error y posición de la articulación 1, en la segunda se muestran las gráficas de error y posición de la articulación 2, mientras que en la tercera pestaña se tiene la gráfica personalizada.

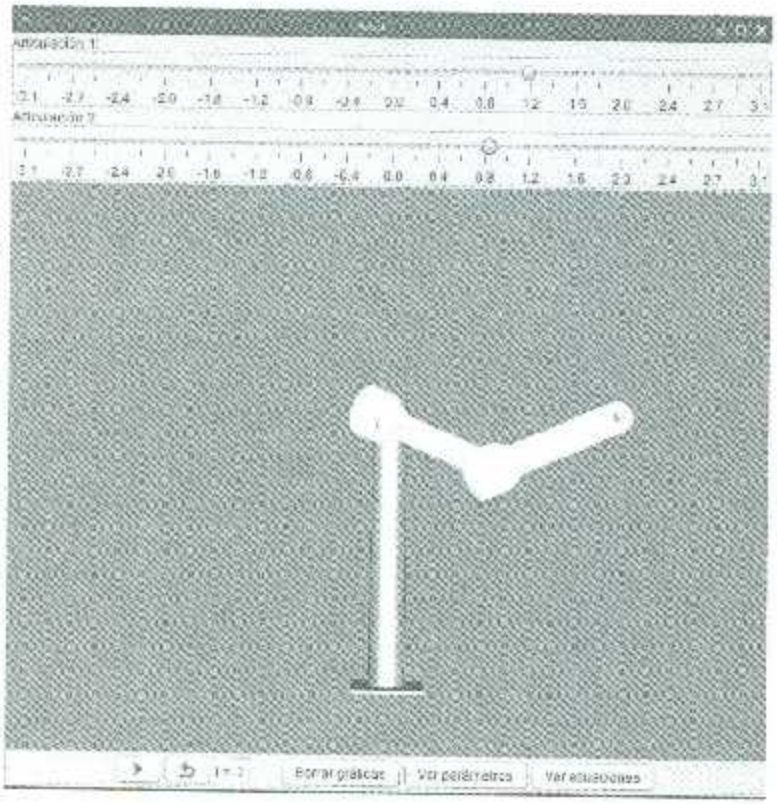

Figura 4.9: Modelo 3D del robot y controles de la simulación

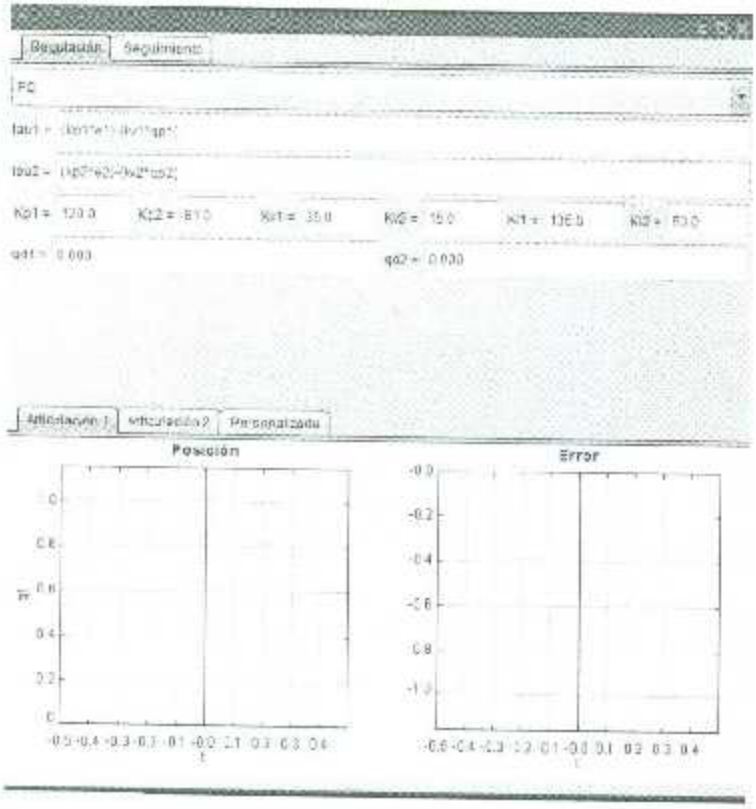

Figura 4.10: Opciones de control en regulación

4.2.3. Control por regulación

En la pestaña de control por regulación, que puede ser observada en la figura 4.10 encontramos primero un *ComboBox*, con una serie de controladores PD en regulación. Los tipos disponibles son PD, PD con compensación de gravedad, PD con compensación precalculada de gravedad y PID. Una vez que se haya seleccionado una ley de control, los campos de "tau1" y "tau2" se completarán automáticamente con las entradas deseadas, y al igual que con los sistemas de una entrada una salida, el usuario puede modificar estas expresiones y utilizar una modificación de la ley de control, siempre y cuando esté escrita con la sintaxis correcta. Debajo de los campos de entrada se encuentra una serie de campos para modificar las ganancias de los controladores, los últimos campos de texto son para introducir la posición deseada.

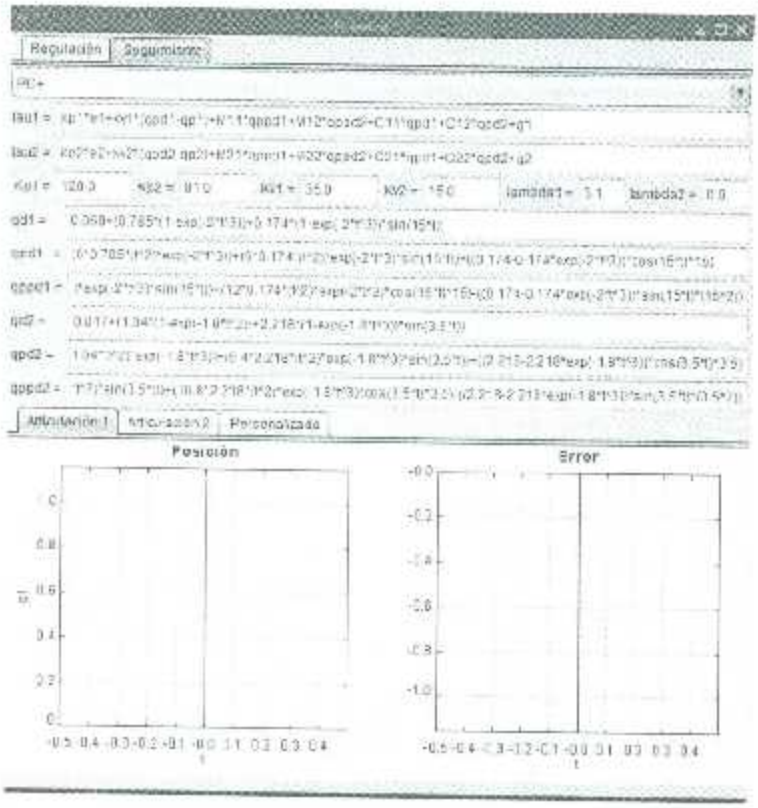

Figura 4.11: Opciones de control en seguimiento

4.2.4. Control por seguimiento

En la segunda pestaña, ahora mostrada en la figura 4.11, se encuentran los controladores por seguimiento. La distribución de los elementos es esencialmente la misma que en la pestaña anterior. En el *ComboBox* creado para elegir el tipo de controlador, ahora se tendrá disponibles los controladores PD+, PD con compensación y Par Calculado. La principal diferencia con la pestaña anterior son los campos de la trayectoria deseada, puesto que ahora existen dos trayectorias, de las cuales también se deben escribir sus primeras dos derivadas para lograr el control por seguimiento.

4.2.5. Panel de graficación

En este último panel se grafican los resultados de nuestros experimentos en tres diferentes pestañas, en la figura 4.12 se muestra una serie de gráficas obtenidas al realizar la simulación con los parámetro mostrados en la figura 4.11, pero utilizando el controlador

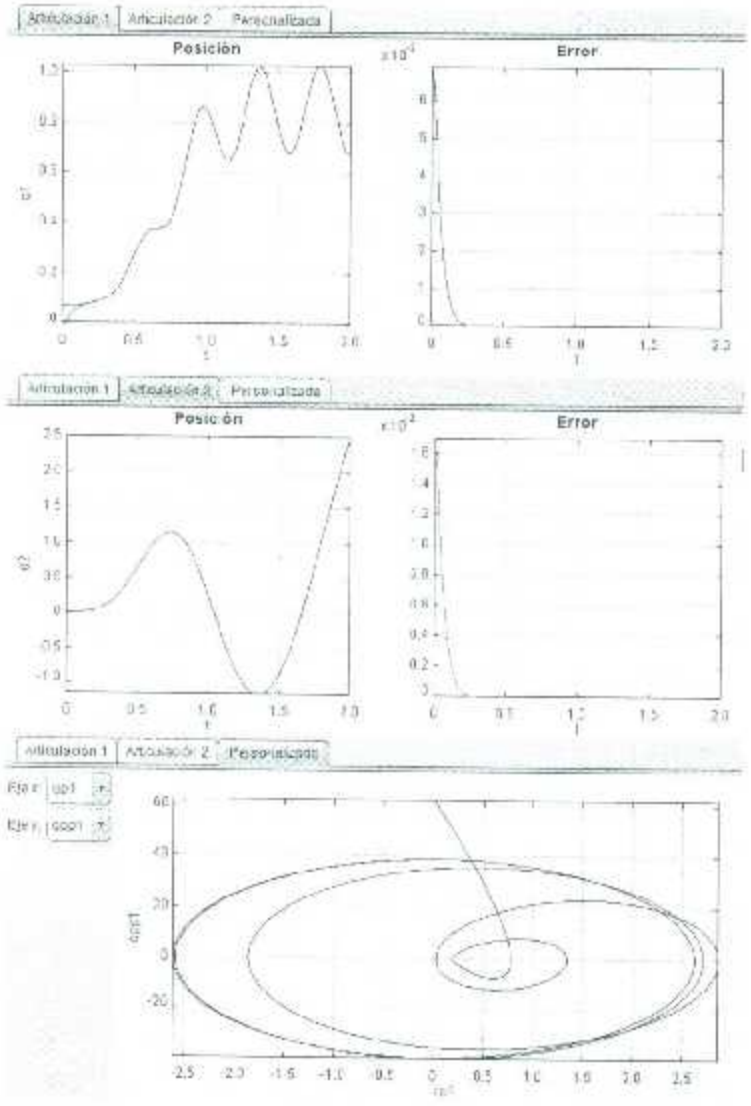

Figura 4.12: Gráficas obtenidas durante la simulación del controlador Par Calculado

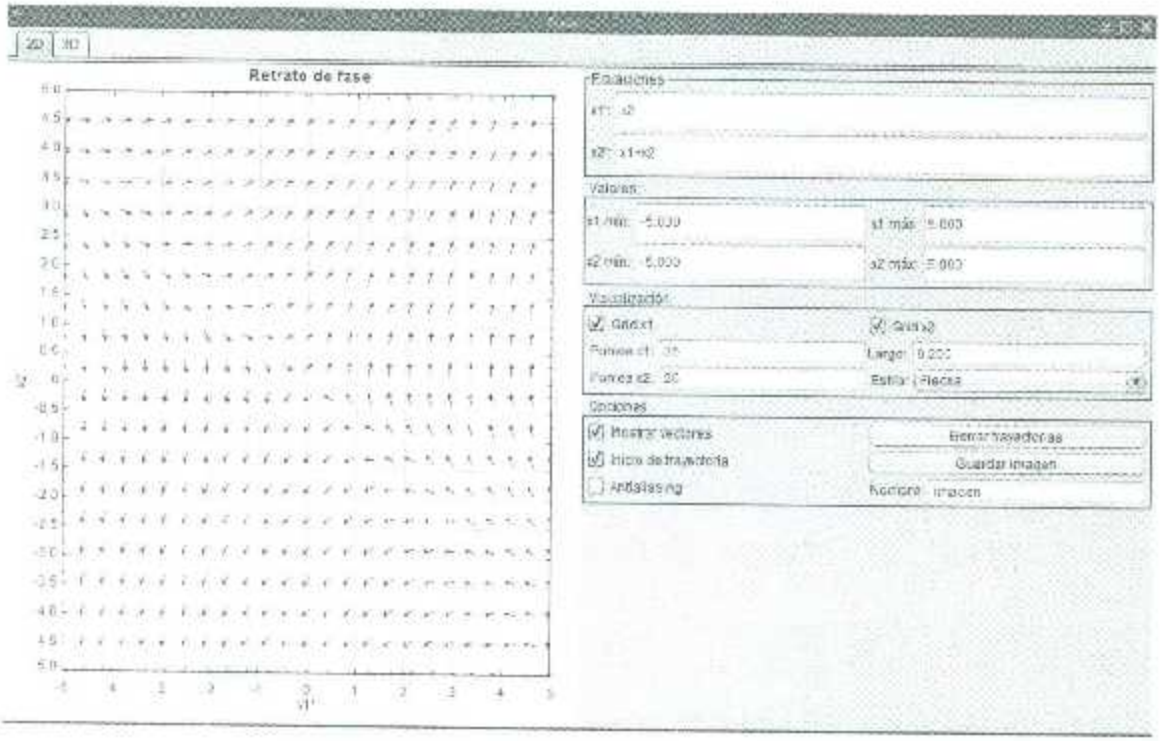

Figura 4.13: Interfaz para la creación de retratos de fase en 2D

Par Calculado. Si se requieren hacer capturas de estas gráficas solo es necesario seguir los pasos que ya fueron descritos en la sección anterior. Si en cambio, se desea tomar una captura del modelo del robot, solo queda recurrir a la función de "Imprimir pantalla" de la computadora.

4.2.6. Retratos de fase

La siguiente simulación de la que se hablará en este apartado tiene como función crear retratos de fase tanto en 2D como en 3D. Esta simulación en particular fue la primera que se elaboró y su objetivo era practicar y familiarizarse con EJS antes comenzar con la simulaciones de los sistemas anteriormente vistos. La simulación tiene una única ventana, dividida en dos pestañas. La primera pestaña es para crear retratos de fase en dos dimensiones, mientras que la segunda ventana se utiliza para crear retratos de fase en tres dimensiones. Cada una de las pestañas está dividida en dos paneles, uno que muestra el retrato de fase y otro panel donde se hacen las configuraciones.

Retratos de fase en 2D

Para crear un retrato de fase, lo único que el usuario debe hacer es escribir las ecuaciones que describen el comportamiento del sistema en la sección de "Ecuaciones", en la parte superior derecha de la ventana. Con estas ecuaciones, el campo de vectores en el área del retrato de fase cambiara para reflejar el sistema descrito. Debajo de la opción de "Ecuaciones" se tiene una serie de opciones para mejorar la visualización. Valores máximos y mínimos de la gráfica, el número de puntos a graficar, el estilo y tamaño del marcador de los vectores, o podemos incluso ocultar la cuadrícula de la gráfica. Lo anteriormente descrito puede ser observado en la figura 4.13.

Una vez que se tiene un retrato de fase, el usuario puede proceder a dar clic en cualquier punto de la gráfica, para indicar el inicio de una trayectoria y ver como esta evoluciona a través del tiempo. El inicio de la trayectoria está marcado por un punto color rojo que podemos ocultar en el *CheckBox* de "Inicio de trayectoria". Se pueden crear tantas trayectorias como el usuario desee, estas también pueden borrarse con el botón de "Borrar trayectorias". En las figuras 4.15 y 4.16 se ven dos ejemplos de retratos de fase creados de esta manera, describiendo al péndulo simple sin fricción y al oscilador de Van der Pol.

Retratos de fase en 3D

La siguiente pestaña, mostrada en la figura 4.14, permite visualizar retratos de fase en tres dimensiones, al igual que en la pestaña anterior, el usuario solo debe escribir las ecuaciones que describen al sistema en la parte superior derecha de la ventana. A continuación se muestran las opciones de visualización para modificar el campo vectorial en 3D para mejorar la visualización del retrato de fase, sin embargo, las opciones de personalización son un poco más limitadas para este elemento de dibujo. Uno de los principales cambios es la inclusión del botón "Ajustar zoom", que ajusta el tamaño del retrato de fase al tamaño de la pantalla.

Si el usuario desea guardar los retratos de fase obtenidos tiene dos opciones: seguir los pasos que fueron descritos en las secciones de las simulaciones de los sistemas, o presionar los botones de "Guardar imagen", presentes en las interfaces.

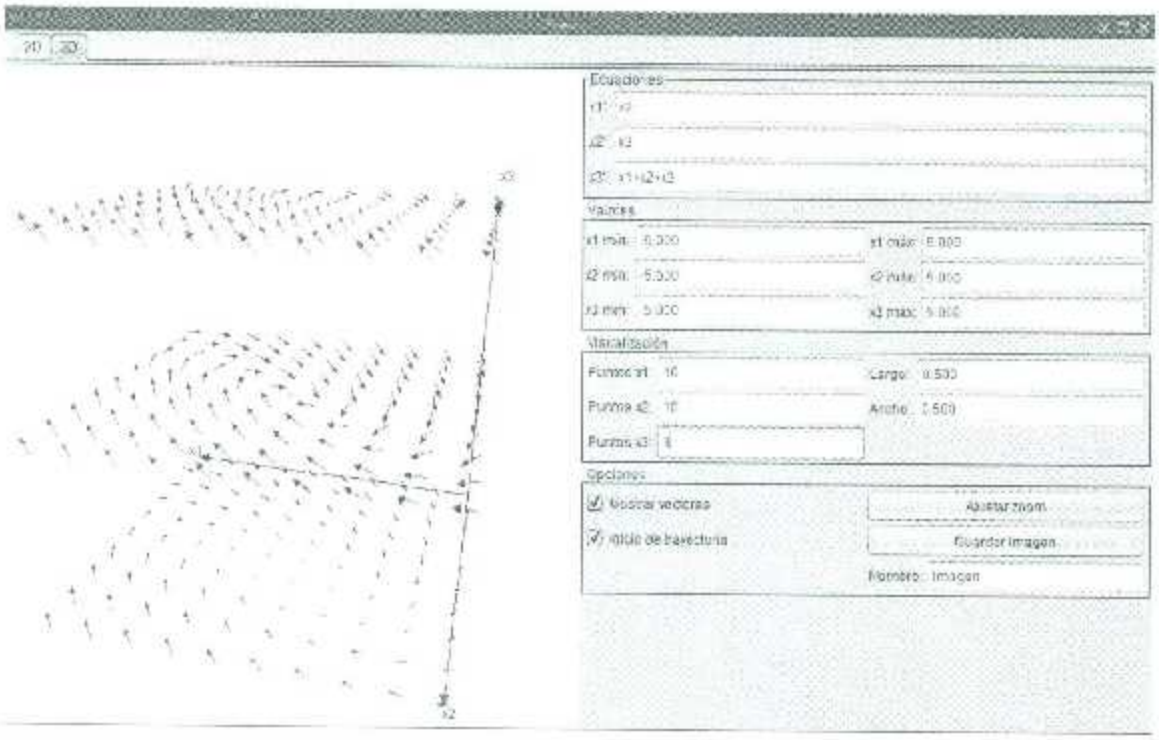

Figura 4.14: Interfaz para la creación de retratos de fase en 3D

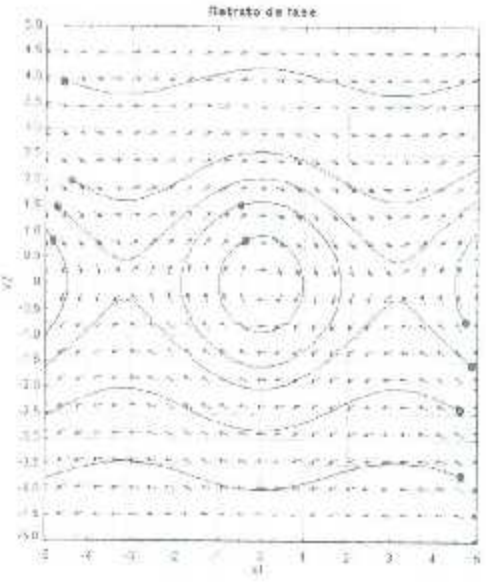

Figura 4.15: Retrato de fase del péndulo simple obtenido con la aplicación

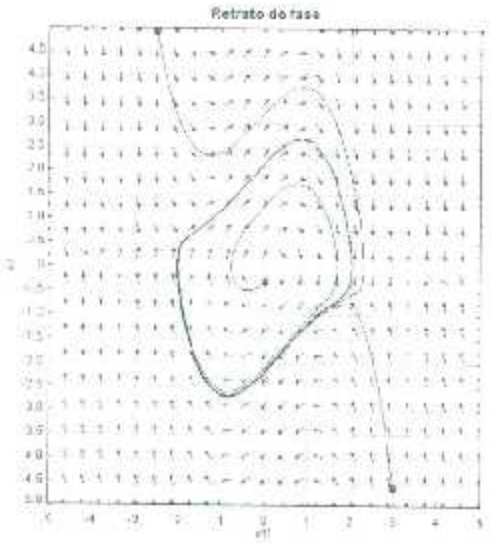

Figura 4.16: Retrato de fase del oscilador de Van der Pol obtenido con la aplicación

Capítulo 5

Sistemas usados en las simulaciones

Dado que para crear las simulaciones en EJS solo necesitamos el sistema representado en espacio de estados, esta sección se enfoca a mostrar estas ecuaciones.

5.1. Sistemas de una entrada-una salida

5.1.1. Péndulo simple

Basándose en el ejemplo 5.1 de la referencia [2] tenemos que:

$$ml^2\ddot{q} + mgl - \sin(q) = \tau \tag{5.1}$$

donde m es la masa, l la distancia desde el eje de giro al centro de masa, g es la acción de la gravedad, q la posición angular del péndulo respecto a la vertical y $\dot{q}$ es la velocidad angular. Reescribiendo esta ecuación en el espacio de estados se tiene:

$$\begin{aligned} \dot{x}_1 &= x_2 \\ \dot{x}_2 &= \frac{1}{ml^2}\left[\tau - mgl\sin x_1\right] \end{aligned} \tag{5.2}$$

5.1.2. Motor de corriente directa

Considérese el siguiente circuito que representa a un motor de corriente directa de imán permanente:

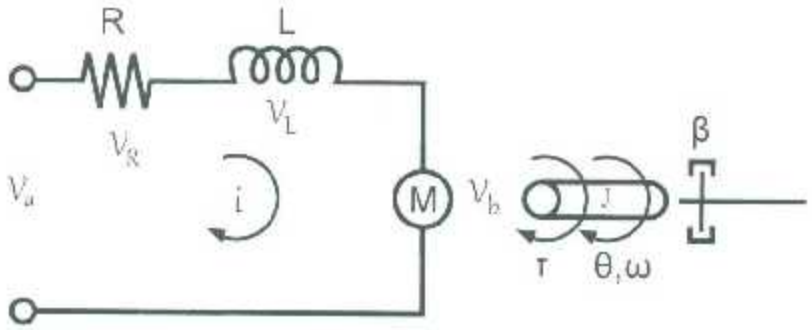

Figura 5.1: Diagrama del motor de corriente directa

donde:

J	=	Inercia
ω	=	Velocidad angular
θ	=	Desplazamiento angular
β	=	Coeficiente fricción
τ	=	Par
i	=	Corriente de armadura
R	=	Resistencia de armadura
L	=	Inductancia
V_a	=	Voltaje de armadura
V_b	=	Voltaje debido a la FEM

del cual obtenemos las siguientes ecuaciones:

$$V_a - iR + L\frac{di}{dt} - V_b \tag{5.3}$$

$$V_b = K_b\omega \tag{5.4}$$

$$\tau_e = Ki \tag{5.5}$$

$$\tau_m = J\frac{d\omega}{dt} + \beta\omega \tag{5.6}$$

$$\tau_m = \tau_e \tag{5.7}$$

donde:

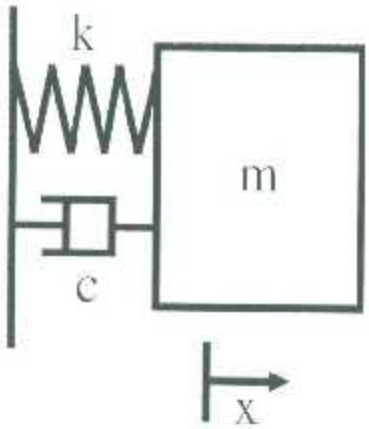

Figura 5.2: Diagrama del sistema masa-resorte-amortiguador

K_b = Constante de proporcionalidad de la FEM
K = Constante de par del motor
τ_m = Par mecánico
τ_e = Par eléctrico

nos proporciona la siguiente ecuación de transferencia:

$$\frac{\omega(s)}{V_a(s)} = \frac{K}{LJs^2 + (RJ + L\beta)s + (R\beta + KK_b)} \tag{5.8}$$

Considerando el desplazamiento angular θ como la salida y estableciendo el valor de $L = 0$ obtenemos la siguiente representación en espacio de estado:

$$\begin{aligned} x_1 &= \theta, \quad \dot{x}_1 = \dot{\theta} = \omega = x_2 \\ x_2 &= \omega, \quad \dot{x}_2 = \dot{\omega} = \frac{K}{JR}u - \frac{1}{J}(\beta - \frac{KK_b}{R})x_2 \\ V_a &= u \end{aligned}$$

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & -\frac{1}{JR}(\beta - \frac{KK_b}{R}) \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{K}{J} \end{bmatrix} u \tag{5.9}$$

5.1.3. Sistema masa-resorte-amortiguador

Se tiene el siguiente sistema masa-resorte-amortiguador mostrado en la figura 5.2, con la siguiente ecuación que describe la dinámica del sistema:

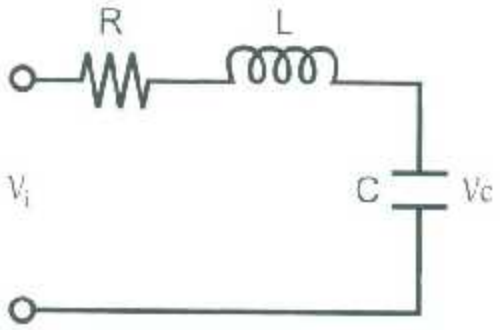

Figura 5.3: Diagrama del circuito RLC en serie

$$m\ddot{x} - c\dot{x} - k = F \tag{5.10}$$

donde m es la masa, k la constante del resorte, c el amortiguamiento del sistema, $\ddot{x}$ la aceleración, F la fuerza, $\dot{x}$ la velocidad, y x es desplazamiento de la masa. Haciendo $x = x_1$, $\dot{x} = x_2$, $F = u$ y reescribiendo en el espacio de estados

$$\begin{aligned} \dot{x}_1 &= x_2 \\ \dot{x}_2 &= -\tfrac{k}{m}x_1 - \tfrac{c}{m}x_2 + u \end{aligned} \tag{5.11}$$

5.1.4. Circuito RLC

Se tiene el siguiente circuito RLC en serie mostrado en la figura 5.3, con la siguiente ecuación que describe la dinámica del sistema:

$$\ddot{V}_c + \frac{R}{L}\dot{V}_c + \frac{1}{LC}V_c - \frac{1}{LC}V_i \tag{5.12}$$

donde L es el valor de la inductancia, R el valor de la resistencia, C el valor del capacitor, V_c el voltaje en el capacitor y V_i el voltaje de entrada. Haciendo $V_c = x1$, $\dot{V}_c = x2$ y $V_i = u$, y reescribiendo en el espacio de estados:

$$\begin{aligned} \dot{x}_1 &= x_2 \\ \dot{x}_2 &= -\tfrac{1}{LC}x_1 - \tfrac{R}{L}x_2 + \tfrac{1}{LC}u \end{aligned} \tag{5.13}$$

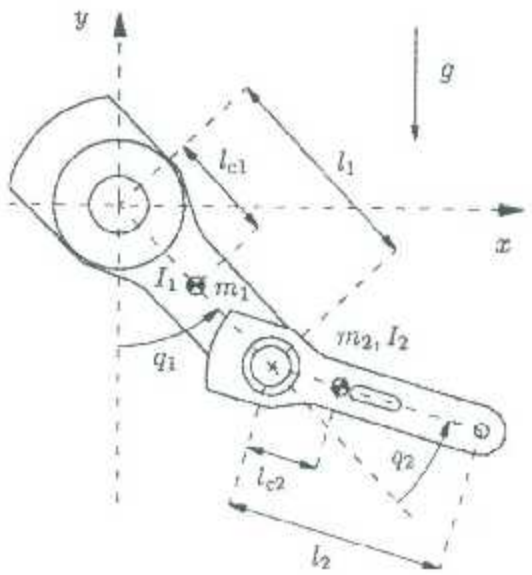

Figura 5.4: Robot prototipo de 2 grados de libertad

5.2. Robot prototipo

Se tiene el siguiente robot prototipo de dos grados de libertad:

Tabla 5.1: Parámetros

Descripción	Notación	Valor	Unidades
Longitud del eslabón 1	l_1	0.450	m
Longitud del eslabón 2	l_2	0.450	m
Distancia al centro de masa (eslabón 1)	l_{c1}	0.091	m
Distancia al centro de masa (eslabón 2)	l_{c2}	0.048	m
Masa eslabón 1	m_1	23.902	kg
Masa eslabón 2	m_2	3.880	kg
Inercia eslabón 1 respecto al centro de masa	I_1	1.266	kg m^2
Inercia eslabón 2 respecto al centro de masa	I_2	0.093	kg m^2
Aceleración de la gravedad	g	9.81	m/ s^2

Modelo dinámico

Su modelo dinámico es el siguiente [2]:

$$\begin{bmatrix} M_{11}(\boldsymbol{q}) & M_{12}(\boldsymbol{q}) \\ \boldsymbol{q}M_{21}(q) & M_{22}(\boldsymbol{q}) \end{bmatrix} \ddot{\boldsymbol{q}} + \begin{bmatrix} C_{11}(\boldsymbol{q},\dot{\boldsymbol{q}}) & M_{12}(\boldsymbol{q},\dot{\boldsymbol{q}}) \\ C_{21}(\boldsymbol{q},\dot{\boldsymbol{q}}) & M_{22}(\boldsymbol{q},\dot{\boldsymbol{q}}) \end{bmatrix} \dot{\boldsymbol{q}} - \begin{bmatrix} g_1(\boldsymbol{q}) \\ g_2(\boldsymbol{q}) \end{bmatrix} = \tau$$

donde q_1 y q_2 son las posiciones articulares, $\dot{q}_1$ y $\dot{q}_2$ son las velocidades articulares. Los elementos de la matriz $M(\boldsymbol{q})$ son los siguientes:

$$\begin{aligned} M_{11}(\boldsymbol{q}) &= m_1 l_{c1}^2 + m_2 \left[l_1^2 + l_{c2}^2 + 2 l_1 l_{c2} \cos(q_2) \right] + I_1 + I_2 \\ M_{12}(\boldsymbol{q}) &= m_2 \left[l_{c2}^2 + l_1 l_{c2} \cos(q_2) \right] + I_2 \\ M_{21}(\boldsymbol{q}) &= m_2 \left[l_{c2}^2 + l_1 l_{c2} \cos(q_2) \right] + I_2 \\ M_{22}(\boldsymbol{q}) &= m_2 l_{c2}^2 + I_2 \end{aligned}$$

Los elementos de la matriz de fuerzas centrífugas y de Coriolis son:

$$\begin{aligned} C_{11}(\boldsymbol{q},\dot{\boldsymbol{q}}) &= -m_2 l_1 l_{c2} \sin(q_2) \dot{q}_2 \\ C_{12}(\boldsymbol{q},\dot{\boldsymbol{q}}) &= -m_2 l_1 l_{c2} \sin(q_2) \left[\dot{q}_2 + \dot{q}_1 \right] \\ C_{21}(\boldsymbol{q},\dot{\boldsymbol{q}}) &= m_2 l_1 l_{c2} \sin(q_2) \dot{q}_2 \\ C_{22}(\boldsymbol{q},\dot{\boldsymbol{q}}) &= 0 \end{aligned}$$

por último, el vector de fuerza gravitacionales tiene los siguientes elementos:

$$\begin{aligned} g_1(\boldsymbol{q}) &= \left[m_1 l_{c1} + m_2 l_1 \right] + m_2 l_{c2} g \sin(q_1 + q_2) \\ g_2(\boldsymbol{q}) &= m_2 l_{c2} g \sin(q_1 + q_2) \end{aligned}$$

5.3. Trayectorias deseadas

Con el propósito de evaluar, mediante simulaciones numéricas, la prestación de los controladores de movimiento, se ha elegido la siguiente trayectoria de movimiento articular [2]:

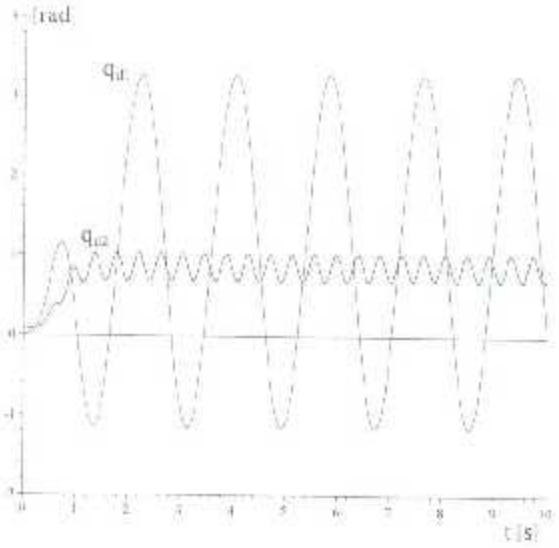

Figura 5.5: Trayectorias deseadas

$$\begin{bmatrix} q_{d1} \\ q_{d2} \end{bmatrix} = \begin{bmatrix} a_1 \\ a_2 \end{bmatrix} - \begin{bmatrix} b_1\left[1-e^{-2.0t^3}\right] + c_1\left[1-e^{-2.0t^3}\right]\sin(\omega_1 t) \\ b_2\left[1-e^{-1.8t^3}\right] + c_2\left[1-e^{-1.8t^3}\right]\sin(\omega_2 t) \end{bmatrix} [rad] \qquad (5.14)$$

donde $a_1 = \pi/45$[rad], $b_1 = \pi/4$[rad], $c_1 = \pi/18$[rad] y $\omega_1 = 15$[rad/s] son parámetros de la referencia de posición deseada para la primera articulación mientras que $a_2 = \pi/180$[rad], $b_2 = \pi/3$[rad], $c_2 = 25\pi/36$[rad] y $\omega_2 = 3.5$[rad/s] son los parámetros de la referencia de posición deseada para la segunda articulación. La figura 5.5 muestra el perfil de la trayectoria deseada para ambas articulaciones.

A partir de las posiciones deseadas (5.14) es posible obtener analíticamente las velocidades deseadas, que vienen dadas en las siguientes expresiones:

$$\begin{aligned} \dot{q}_{d1} &= 6b_1t^2e^{-2.0t^3} - 6b_1t^2e^{-2.0t^3}\sin(\omega_1 t) - \left[c_1 - c_1e^{-2.0t^3}\right]\cos(\omega_1 t)\omega_1 \\ \dot{q}_{d2} &= 5.4b_2t^2e^{-1.8t^3} - 5.4b_2t^2e^{-1.8t^3}\sin(\omega_2 t) + \left[c_2 - c_2e^{-1.8t^3}\right]\cos(\omega_2 t)\omega_2 \end{aligned} \qquad (5.15)$$

De la misma manera, se procede para obtener analíticamente las aceleraciones deseadas:

$$\begin{aligned}\ddot{q}_{d1} =& -12b_1te^{-2.0t^3} - 36b_1t^4e^{-2.0t^3} + 12c_1te^{-2.0t^3}\sin\omega_1 t \\ & - 36c_1t^4e^{-2.0t^3}\sin\omega_1 t + 12c_1te^{-2.0t^3}\cos\omega_1 t\omega_1 \\ & - \left[c_1 - c_1e^{-2.0t^3}\right]\sin(\omega_1 t)\omega_1^2\,[\mathrm{rad/s^2}]\end{aligned}$$

$$\begin{aligned}\ddot{q}_{d2} =& 10.8b_2te^{-1.8t^3} - 29.2b_2t^4e^{-1.8t^3} + 10.8c_2te^{-1.8t^3}\sin\omega_2 t \\ & - 29.2c_2t^4e^{-1.8t^3}\sin\omega_2 t + 10.8c_2te^{-1.8t^3}\cos\omega_2 t\omega_2 \\ & - \left[c_2 - c_2e^{-1.8t^3}\right]\sin(\omega_2 t)\omega_2^2\,[\mathrm{rad/s^2}]\end{aligned} \tag{5.16}$$

5.4. Leyes de control utilizadas

Las leyes de control utilizadas como ejemplo en las simulaciones fueron tomadas de [2]. A continuación en las tablas 5.2 y 5.3 se listan las leyes de control utilizadas y las ecuaciones que las describen.

Tabla 5.2: Controladores PD en regulación

Nombre	Ley de control
PD	$\tau = K_p\tilde{q} - K_v\dot{q}$
PD con compensación de gravedad	$\tau = K_p\tilde{q} - K_v\dot{q} + g(q)$
PD con compensación de precalculada gravedad	$\tau = K_p\tilde{q} - K_v\dot{q} + g(q_d)$
PID	$\tau = K_p\tilde{q} - K_vq + K_i\xi,\ \xi = \tilde{q}$

Tabla 5.3: Controladores PD en seguimiento

Nombre	Ley de control
PD+	$\tau = K_p\tilde{q} - K_v\dot{q} + M(q)\ddot{q}_d + C(q,\dot{q})\dot{q}_d + g(q)$
PD con compensación	$\tau = K_p\tilde{q} + K_v\dot{\tilde{q}} + M(q)\left[\ddot{q}_d + \lambda\dot{\tilde{q}}\right] + C(q,\dot{q})\left[\dot{q}_d - \lambda\tilde{q}\right] + g(q)$
Par calculado	$\tau = M(q)\left[\ddot{q}_d + K_p\tilde{q} + K_v\dot{\tilde{q}}\right] + C(q,\dot{q})\dot{q}_d + g(q)$

donde se tiene los vectores:

$$q = [q_1, q_2]^T \tag{5.17}$$

$$\tilde{q} = [\tilde{q}_1, \tilde{q}_2]^T \tag{5.18}$$

$$\dot{q} = [\dot{q}_1, \dot{q}_2]^T \tag{5.19}$$

$$\ddot{q} = [\ddot{q}_1, \ddot{q}_2]^T \tag{5.20}$$

K_p, K_v y K_i son matrices simétricas definidas positivas por diseño,

$$K_p = \begin{bmatrix} k_p & 0 \\ 0 & k_p \end{bmatrix} \tag{5.21}$$

$$K_v = \begin{bmatrix} k_v & 0 \\ 0 & k_v \end{bmatrix} \tag{5.22}$$

$$K_i = \begin{bmatrix} k_i & 0 \\ 0 & k_i \end{bmatrix} \tag{5.23}$$

y λ se define como:

$$\lambda = K_v^{-1}K_p \tag{5.24}$$

5.5. Resultados en simulación

A continuación se muestran algunos resultados de simulaciones realizadas con el programa:

5.5.1. Péndulo

Utilizando el controlador PD de seguimiento PD con compensación.

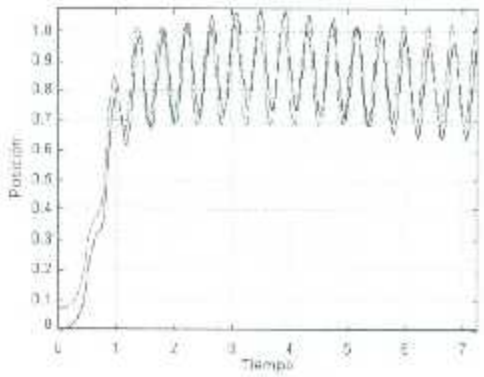

Figura 5.6: Posición

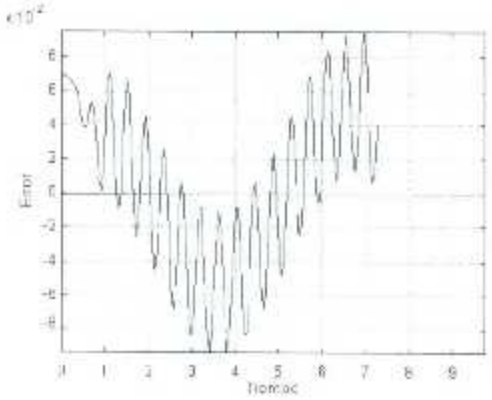

Figura 5.7: Error

5.5.2. Motor de corriente directa

Utilizando el controlador por regulación PID.

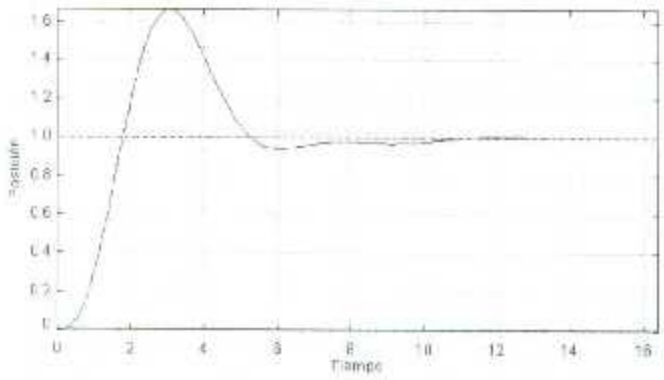

Figura 5.8: Posición

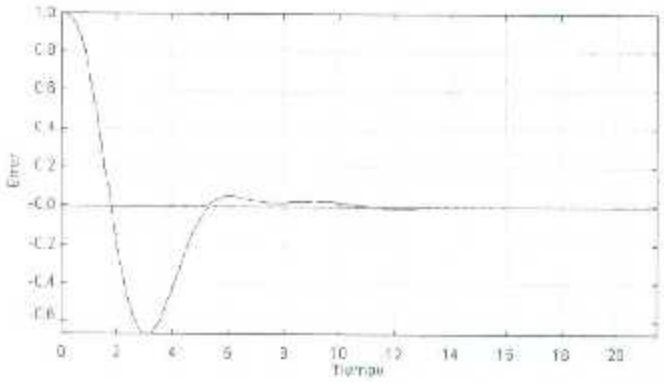

Figura 5.9: Error

5.5.3. Sistema masa-resorte-amortiguador

Utilizando el controlador por regulación PID.

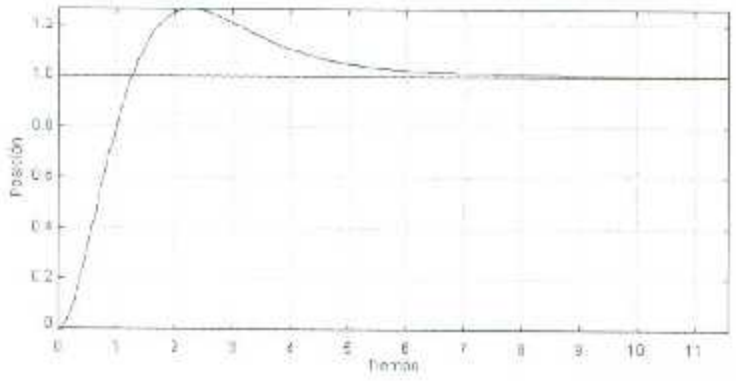

Figura 5.10: Posición

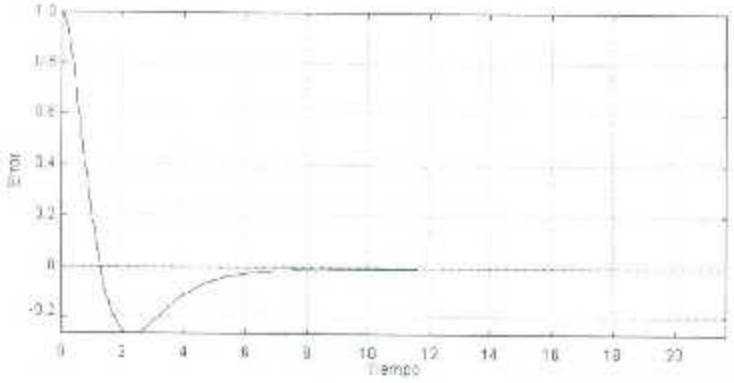

Figura 5.11: Error

5.5.4. Robot prototipo de 2 grados de libertad

Utilizando el controlador PD por seguimiento Par Calculado.

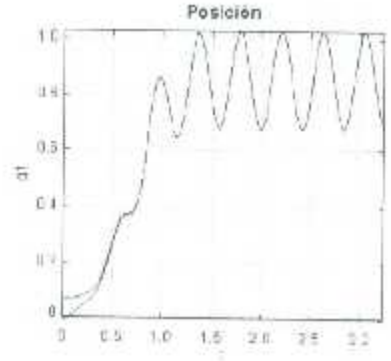

Figura 5.12: Posición de la articulación 1

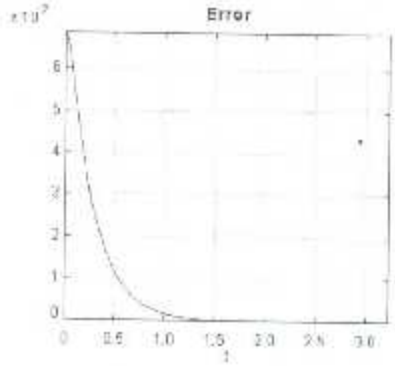

Figura 5.13: Error de la articulación 1

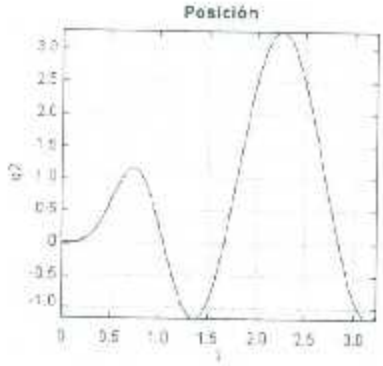

Figura 5.14: Posición de la articulación 2

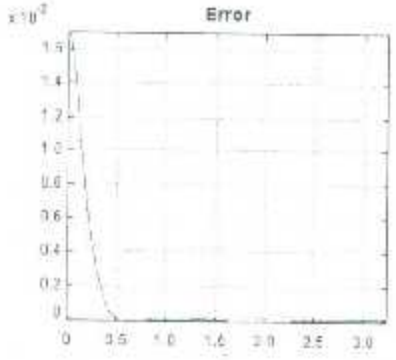

Figura 5.15: Error de la articulación 2

Capítulo 6

Conclusiones

En la actualidad el uso de simulaciones con interfaces gráficas como apoyo pedagógico resulta de gran ayuda para que los alumnos puedan visualizar mejor los conceptos estudiados y así tener un mejor entendimiento de los temas de estudio. En el área de control esto se vuelve importante para que los alumnos puedan observar cómo las leyes diseñadas afectarían a un sistema físico, dado que una maqueta del sistema no siempre está disponible. Una segunda opción son las simulaciones en computadora. Sin embargo, por lo general se necesita de muy alto nivel de conocimientos de programación para crear tales simulaciones.

EJS es, en definitiva, una herramienta extremadamente útil y poderosa para crear fácilmente simulaciones de sistemas físicos. Su principal ventaja es que la persona que crea las simulaciones no necesita tener un gran conocimiento de programación para desarrollar modelos matemáticos e interfaces gráficas. EJS incluso genera automáticamente el código para crear los enlaces entre el modelo y la interfaz automáticamente, mientras que el usuario solo necesita navegar a través de ventanas de diálogo para poder determinar estos enlaces sin necesidad de entrar a fondo en el código fuente del programa.

Algunas desventajas que se encontraron con EJS provienen del hecho de que es aún un proyecto en desarrollo, el programa aún presenta pequeños errores (como por ejemplo, para abrir una simulación diferente después de cerrar otra, es necesario cerrar y volver a abrir la aplicación de EJS). También se observaron algunos problemas con los gráficos en 3D, al modificarlos podían causar que el archivo .ejs de la simulación fuera corrompido y se volviera imposible abrirlo correctamente. Otro problema es que la documentación no entra en mucho detalle en el uso del programa, el manual distribuido no ha sido actuali-

zado en varios años y no está al pendiente de los cambio más recientes al programa. Sin embargo, gracias a la librería de simulaciones disponibles en la página de OPS se puede encontrar gran cantidad de ejemplos realizados por otros usuarios para aprender por medio de práctica y observación el uso de los elementos de EJS.

Con EJS se crearon una serie de simulaciones para conformar un laboratorio de sistemas no lineales. En este laboratorio, que contiene un sistema masa-resorte-amortiguador, un circuito RLC, un motor de corriente directa, un péndulo simple y el robot prototipo de dos grados de libertad, el usuario puede probar leyes de control, ya sea del tipo PD en regulación y seguimiento como un par de leyes de control no lineales; estas leyes de control pueden ser modificadas y también es posible añadir leyes nuevas. En estas simulaciones se incluyeron modelos en 2D y 3D que representan al sistema físico y además gráficas de los estados del sistema, que contienen una herramienta para ser exportadas como archivos de imagen. Con todo esto se le da al alumno una mejor visualización de los problemas de control y de como los controladores afectan el comportamiento de un sistema.

Se menciona cómo posibilidad de trabajos a futuro la creación de una maqueta física controlada desde EJS por medio de las tarjetas adquisitoras de Arduino. Otro trabajo posible es utilizar la recientemente añadida posibilidad de crear simulaciones para el sistema operativo Android, permitiendo su uso en tablets y teléfonos celulares.

Bibliografía

[1] Esquembre Francisco, "Easy Java Simulations", Open Source Physics, 2003.

[2] Kelly Rafael y Santibáñez Víctor, "Control de movimiento de robots manipuladores", Pearson Education, Madrid, 2003.

[3] Belloni Mario and Christian Wolfgang, "Open Source Physics: A User's Guide with Examples", Chapter 15: Authoring Curricular Material, 2005.

[4] G. Farias, F. Esquembre, J. Sanchez, S. Dormido, "Laboratorios virtuales remotos usando Easy Java Simulations y Simulink", Departamento de Informática y Automática, UNED, Madrid, 2006.

[5] G. Farias, F. Esquembre, J. Sanchez, S. Dormido et al. "Desarrollo de laboratorios virtuales, interactivos y remotos utilizando Easy Java Simulations y modelos Simulink", Departamento de Informática y Automática, UNED, Madrid, 2004.

[6] Oracle Open World, "Class Math", https://docs.oracle.com/, 1993-2014.

[7] Wolfgang Christian and Francisco Esquembre, "Function Visualizer Model", 2009.

[8] Wolfgang Christian and Francisco Esquembre, "ODE Direction Field Plotter Model", 2012.

[9] Spencer Wheaton, "Computational Physics Resources: Smooth Particle Interpolation", 2014.

[10] Loo Kang Wee, "Geostationary Earth Orbit Satellite Model", 2012.