LABORATORIOS VIRTUALES REMOTOS USANDO EASY JAVA SIMULATIONS Y SIMULINK

G. Farias[1], F. Esquembre[2], J. Sanchez[1], S. Dormido[1]

[1]Departamento de Informática y Automática, UNED.
Ciudad Universitaria, C.P. 28040, Madrid, España

[2]Departamento de Matemáticas, Universidad de Murcia.
Campus de Espinardo, C.P. 30071, Murcia, España

gfarias@bec.uned.es, fem@um.es,
{jsanchez,sdormido}@dia.uned.es

Resumen

Actualmente Easy Java Simulations (Ejs) permite re-utilizar modelos Simulink para la creación de laboratorios virtuales interactivos. Sin embargo hasta ahora, si el usuario final del laboratorio virtual no dispone de Matlab/Simulink en su ordenador no podrá ejecutar la simulación. Como solución al problema anterior está en proceso de desarrollo y prueba la utilización del servidor Java Internet Matlab (Jim), el cual junto con Ejs permite la creación y ejecución de laboratorios virtuales remotos, con un alto grado de interactividad, sin necesidad de que el diseñador o usuario final requiera en su computador de Matlab/Simulink. Es necesario considerar que la extensión propuesta utiliza una licencia normal de Matlab ya que el requisito de una licencia de Matlab Web Server puede resultar en algunos casos prohibitivo debido a su coste. En el presente artículo, por tanto, se exponen las principales ideas existentes tras el desarrollo mencionado.

Palabras Clave: Educación en Automática, Laboratorio virtual, Laboratorio basado en Web, *Ejs*, Simulink/Matlab.

1 INTRODUCCIÓN

Un informe de la Asociación de Industrias de Tecnologías de Información de los EE.UU. señala lo siguiente: "El cambio de paradigma respecto de nuestros objetivos y modelos educativos está recién comenzando, al confluir las soluciones del siglo XXI con la infraestructura del siglo XX y las tradiciones educativas del siglo XIX". Tal diagnóstico no sólo asume la imparable transformación de las tecnologías de la información en cualquier aspecto de la sociedad, si no que además revela el cambio que está ocurriendo en la educación, pasando de ser una etapa de preparación a un proceso de formación constante.

Un proceso continuo de enseñanza será más exitoso mientras más flexible sea respecto de la hora y el lugar en que se realice la instrucción. La enseñanza de la Ingeniería de Control debe por tanto aprovechar las capacidades facilitadas por las herramientas de simulación e Internet, para reproducir y soportar las principales acciones causa-efecto entre el estudiante y un modelo de la planta, mediante un laboratorio virtual interactivo, ya sea en un formato local o remoto. La literatura y la investigación han obtenido grandes avances en este propósito [1,6].

La interactividad es un aspecto fundamental cuando se diseñan laboratorios virtuales para ser usados con propósitos pedagógicos en el campo de la Ingeniería de Control. Es la interactividad la que permite al estudiante explorar diferentes configuraciones y la calidad de la respuesta del sistema, cuando se intenta adquirir no sólo la teoría, si no además los aspectos estratégicos e intuitivos de los modelos analizados [2,8].

Matlab/Simulink es un paquete de software clásico que proporciona facilidades para la construcción de modelos de forma gráfica mediante diagramas de bloques, por lo que se ha convertido en una herramienta ampliamente utilizado en la industria y la enseñanza de gran parte de las ingenierías. Sin embargo, la interactividad proporcionada por los modelos Simulink, dista bastante del tipo de interactividad que se ha mencionado anteriormente.

Es en este aspecto donde *Ejs* [4] puede resultar muy útil. *Ejs* es una herramienta software gratuita diseñada para el desarrollo de laboratorios virtuales interactivos. Este paquete proporciona un mecanismo propio para la descripción de modelos científicos y sistemas de la Ingeniería de Control. Además, *Ejs* dispone de una enorme cantidad de elementos

Almería 2006 - ISBN: 84-689-9417-0

visuales parametrizables e interactivos, que permiten la rápida construcción de la vista de un laboratorio virtual.

La integración de modelos Simulink en los laboratorios virtuales desarrollados en *Ejs*, ha sido parte de un trabajo constante y fructífero [3,9]. Los resultados obtenidos permiten, de forma muy sencilla al usuario, agregar a un modelo Simulink un alto nivel de interactividad.

Jim (Java Internet Matlab) es un paquete escrito en Java orientado a extender las capacidades de *Ejs* para manipular modelos Simulink. La extensión permite al usuario de *Ejs* ejecutar la simulación sin la necesidad de contar con Matlab/Simulink en su equipo, ya que *Ejs* establece un enlace de red con un equipo remoto, soportado por Jim, que sí posee Matlab/Simulink.

La estructura del presente documento es la siguiente: La sección 2 describe de forma breve el entorno *Ejs*. Posteriormente, en la sección 3, se resumen los pasos requeridos para utilizar los modelos Simulink en *Ejs*. En el apartado 4, se describe el servidor Jim. A continuación en la sección 5 se describe la conexión entre *Ejs*, Jim y Simulink. En la sección 6 se presentan dos ejemplos de conexión remota. Finalmente se analizarán las conclusiones principales.

2 EASY JAVA SIMULATIONS

Ejs es una herramienta de software gratuita diseñada para la creación de simulaciones interactivas en Java [4,5].

El usuario al que está dirigida *Ejs* son estudiantes, profesores e investigadores de ciencias, que poseen un conocimiento básico de programación de computadores, pero que no disponen de una gran cantidad de tiempo para crear una simulación gráfica con un elevado grado de interactividad.

Las simulaciones en *Ejs* son estructuradas en dos partes, el *Modelo* y la *Vista*. En el modelo se describe el comportamiento del sistema mediante variables y código Java o ecuaciones diferenciales ordinarias. Mientras que la vista provee el aspecto gráfico o visual de la simulación. La interfaz de usuario de *Ejs* se presenta en la Figura 1.

Ejs se puede utilizar de forma independiente para crear simulaciones interactivas, ya sea como aplicación o como applet, sin embargo, también es posible utilizar *Ejs* para agregar interactividad a modelos Simulink [3].

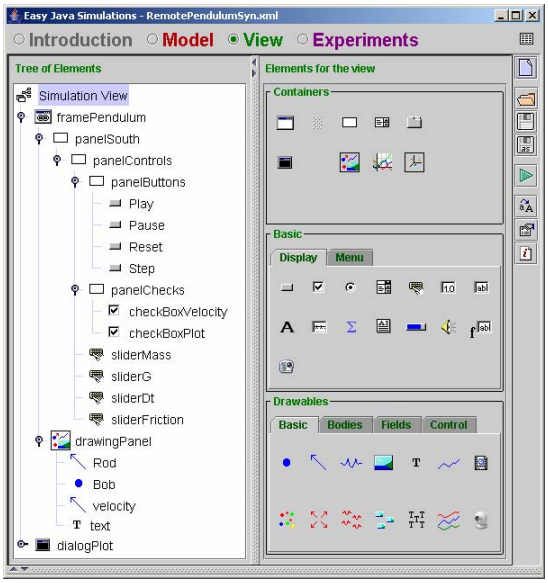

Figura 1. Interfaz de usuario de *Ejs* para la creación de la vista de una simulación. La estructura de árbol a la izquierda corresponde a la vista de la Figura 7.

3 CONECTANDO EJS Y SIMULINK

La comunicación entre *Ejs* y Matlab/Simulink se realiza mediante la librería JMatlink [7], que está basada en JNI (Java Native Interface) y la librería Engine en lenguaje C proporcionada por Matlab. El esquema anterior se observa en la Figura 2.

El procedimiento para la conexión de un modelo Simulink y *Ejs* es simple y se resume en las siguientes cuatro etapas:

- Seleccionar (o modificar) el modelo Simulink.
- Crear y conectar variables *Ejs* con el modelo.
- Controlar la ejecución del modelo.
- Definir la vista y la interactividad con el usuario.

Al seleccionar el modelo Simulink, se debe considerar la potencialidad interactiva que éste posee. En general se recomienda que las variables que serán controladas u observadas por el usuario sean señales explícitas de entrada o salida de algún bloque en el modelo. Ello permitirá realizar la conexión con *Ejs* de forma más sencilla y directa.

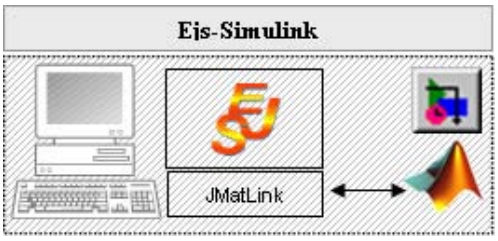

Figura 2. Conexión *Ejs* y Matlab/Simulink.

Respecto a lo anterior, si se requiere aumentar la interactividad de un modelo Simulink para utilizarlo con *Ejs*, es posible que se modifique ligeramente la estructura del mismo de acuerdo a las recomendaciones mencionadas.

En la Figura 3 se presenta el modelo seleccionado que corresponde al que proporciona Simulink para simular un péndulo simple. Nótese que en este caso se requiere una modificación del modelo ya que algunas variables de interés (gravedad, fricción, masa y longitud) están de forma implícita. El modelo modificado se presenta en la Figura 4. Obsérvese que las variables de interés se presentan como entradas o salidas de los bloques del modelo.

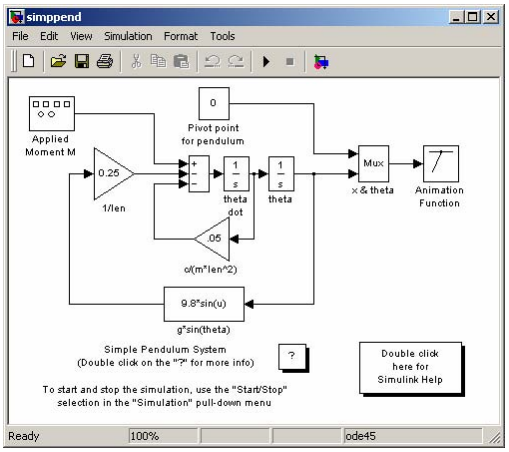

Figura 3. Modelo Simulink seleccionado. Obsérvese que las variables de interés están de forma implícita en el modelo.

En la parte *Modelo* de *Ejs*, el usuario debe especificar en la *Tabla de Variable*s los nombres de las variables, sus dimensiones, el tipo, y si esta conectada o no a una señal de un bloque del modelo Simulink. Para realizar la conexión, *Ejs* proporciona al usuario un cuadro de dialogo en donde se listan los bloques y parámetros disponibles del modelo. *Ejs* también permite realizar la conexión utilizando el modelo Simulink. En la Figura 5 es posible observar cómo se indica a *Ejs* el modelo modificado (*simppendModified.mdl*) y las variables conectadas al mismo.

Para que *Ejs* controle la ejecución sincronizada del modelo Simulink, basta especificar en la subsección *Evolución*, dentro de la sección *Modelo*, el siguiente método proporcionado por *Ejs*:

_external.step (int pasos);

La variable entera *pasos* indica a *Ejs* cada cuantos pasos de integración del modelo Simulink se actualizarán los valores de las variables de *Ejs* conectadas. En general el valor de pasos será uno, aunque puede ser mayor, si se desea que la evolución de la simulación sea más rápida.

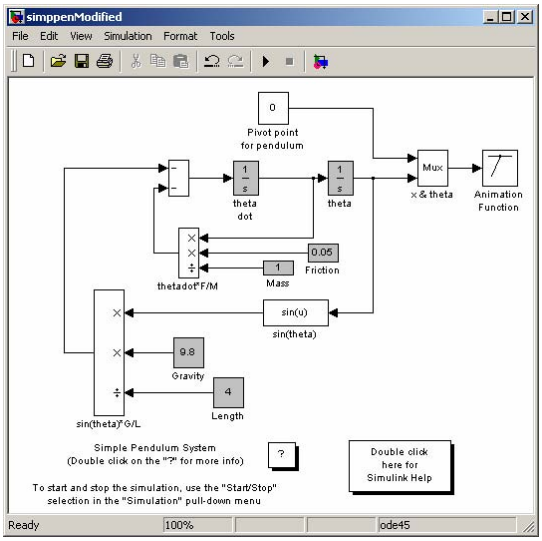

Figura 4. Modelo Simulink modificado. Las variables de interés son entradas o salidas de bloques del modelo.

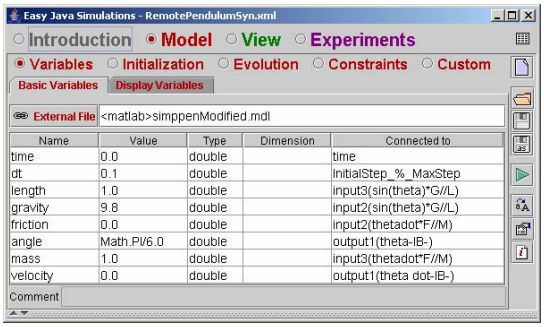

Figura 5. Interfaz de *Ejs* en donde se observa el archivo Simulink y las variables de interés conectadas.

El control de la simulación está sincronizado, debido a que en cada paso de evolución se ejecutan repetitivamente las siguientes acciones:

1) Se actualizan las variables de entrada en el modelo de acuerdo a sus valores en *Ejs*.
2) Se ejecuta un paso de integración en Simulink.
3) Para finalmente obtener los valores de las variables de salida del modelo y actualizar los de la vista.

Finalmente, la cuarta etapa consiste en la definición de la vista, y por tanto la interactividad, ya que los elementos visuales están íntimamente relacionados con la capacidad interactiva de la simulación. Para crear la vista, el diseñador debe utilizar los elementos gráficos disponibles proporcionados por *Ejs* y que se observan en el lado derecho de la Figura 1. Cada elemento de la vista proporciona al diseñador un

Almería 2006 - ISBN: 84-689-9417-0

conjunto de parámetros de configuración, como los que se observan en la Figura 6, para el elemento que representa la bola del péndulo. En este caso las variables de interés como ángulo y velocidad están asociadas por el estado de la bola, mientras que la fricción, masa y gravedad pueden manipularse mediante deslizadores (ver Figura 7).

Además, obsérvese en la misma imagen la existencia de un conjunto de botones que permiten controlar la ejecución de la simulación. También puede distinguirse una ventana más pequeña al lado derecho, que presenta la evolución de la posición angular de la bola respecto al tiempo.

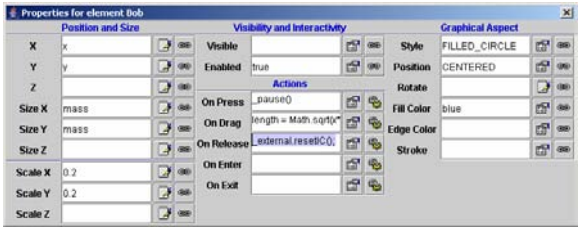

Figura 6. Propiedades del elemento de la vista que representa la bola del péndulo. El texto sombreado corresponde al método *_external.resetIC()*.

Un aspecto importante para la interactividad es la posibilidad de reiniciar la simulación en un estado diferente, en este caso, que la bola comience desde una posición definida por el usuario, para ello *Ejs* proporciona al diseñador un método que modifica y reinicia a un valor arbitrario la salida de un bloque integrador del modelo Simulink. El método suministrado por *Ejs* es el siguiente:

_external.resetIC();

Con lo anterior, el usuario final podrá arrastrar la bola a una posición diferente, y al soltarla, la simulación comenzará a ejecutarse nuevamente pero, en un nuevo estado. En la Figura 6 se observa, en texto sombreado, la incorporación del método anterior en las propiedades de la bola. Una descripción más detallada del procedimiento de conexión puede encontrarse en [3].

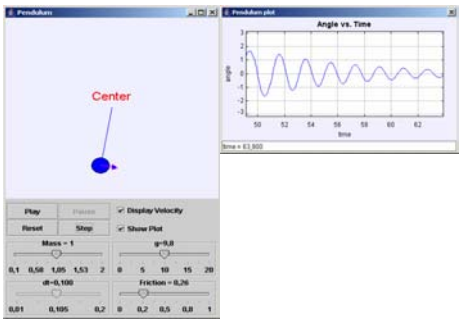

Figura 7. Interfaz gráfica de la simulación.

4 JIM: JAVA INTERNET MATLAB

Para extender las capacidades de conexión de *Ejs* y los modelos Simulink, se decidió desarrollar Jim, un paquete de software que permite conectar un modelo Simulink remoto con una aplicación *Ejs*.

Es importante destacar que Jim es una aplicación escrita en Java que se conecta con una versión de Matlab local, y no con Matlab Web-Server. Al igual que *Ejs*, la librería JMatlink es utilizada para la conexión entre Matlab y Jim.

La comunicación entre Jim y *Ejs* es del tipo cliente/servidor y se resuelve mediante la utilización de sockets TCP/IP, el lado del servidor es soportado por Jim mientras que *Ejs* corresponde al lado del cliente. A continuación se describe brevemente la interfaz de usuario y las opciones proporcionadas por el servidor Jim.

4.1 LA INTERFAZ DE USUARIO DE JIM

La interfaz de usuario de Jim se presenta en la Figura 8. En la aplicación se proporcionan las siguientes opciones para la comunicación entre Jim, *Ejs* y Simulink (los valores por defecto están entre paréntesis):

- Puerto de servicio (*2005*): Puerto en el que se establecerá la comunicación entre *Ejs* y Jim.
- Tamaño del buffer de entrada (*1024)*: Límite máximo del buffer de entrada disponible para el enlace TCP/IP.
- Tamaño del buffer de salida (*1024*): Límite máximo del buffer de salida disponible para el enlace TCP/IP.
- Directorio de Trabajo (*\JimWD*): Carpeta donde se alojarán los modelos Simulink propios del usuario remoto.
- Máximo número de sesiones (*5*): Límite superior de sesiones de Matlab a ejecutar en el servidor.
- Permitir funciones S.O (*no chequeado*): Se habilita al usuario remoto a usar funciones de Matlab (como DOS o el operador !) que afectan al sistema (disponible sólo en Matlab 7.0).
- Permitir archivos externos (*chequeado*): Permite al usuario remoto utilizar un modelo de Simulink propio.
- Archivo de bitácora (*chequeado con opción simple*): Si esta habilitado se guardará en un archivo información de las acciones realizadas por los usuarios externos.

Notar que algunas opciones pueden afectar el rendimiento del servidor. Por ejemplo, si se incrementa el número de sesiones de Matlab el tiempo de respuesta del servidor podría empeorar.

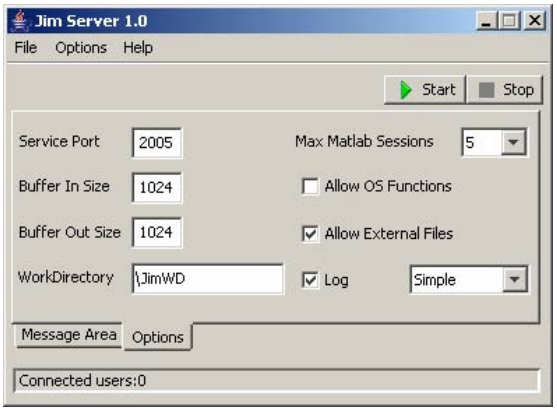

Figura 8. Interfaz de usuario de Jim. Sección de opciones.

Jim puede ser detenido en cualquier momento por el administrador del servidor remoto para realizar una re-configuración, sin embargo esto obligará al usuario remoto a reiniciar la conexión con Jim.

4.2 INICIANDO EL SERVIDOR

Antes de establecer la conexión entre Matlab y *Ejs*, es necesario iniciar el servidor remoto. En el lado del servidor se requieren un ordenador conectado a una red con los siguientes elementos:

- Windows 98 o superior.
- Matlab 5.3 (R11) o superior.
- Máquina Virtual de Java 1.5.
- Java Internet Matlab (Jim) 1.0.

Los tres primeros elementos también son requeridos cuando se establece una conexión local entre Matlab y *Ejs*, por lo que el único nuevo elemento corresponde a Jim.

La conexión utiliza el protocolo TCP/IP, así que se debe asegurar que los ordenadores que se conectarán soportan este tipo de enlace. En este sentido podría ser incluso necesario deshabilitar algunas opciones de seguridad (como por ejemplo el cortafuego de Windows XP) para permitir el uso del protocolo.

Cuando se tenga por seguro que los ordenadores cumplen con las condiciones anteriormente comentadas, entonces el usuario del servidor podrá iniciar el servicio remoto simplemente haciendo doble clic en el archivo *JimW.exe*, y posteriormente pinchando el botón *Start*.

4.3 TIPOS DE ENLACES ENTRE EJS Y JIM

La comunicación entre *Ejs* y un Matlab remoto es bastante similar a una conexión local, en este caso el enlace es llamado **síncrono** porque cada paso de integración de un modelo Simulink es directamente controlado por *Ejs*.

El enlace síncrono presenta un buen rendimiento en modo local, pero cuando la conexión se realiza en forma remota la simulación se ralentiza debido principalmente, a los retardos inherentes de la red. En una red de área local (LAN), probablemente, la ejecución de una simulación no sea tan lenta, pero en una red más extensa como Internet, el rendimiento de la simulación podría llegar a ser demasiado lento. Este inconveniente fue el principal factor que influyó en la creación de un segundo tipo de enlace, denominado **asíncrono**, que evita o por lo menos atenúa los efectos producidos por el retardo de la red.

La ejecución de una simulación del modelo remoto se divide en la realización de todas las etapas o fases de la Tabla 1.

Tabla 1: Fases realizadas para la ejecución de un modelo Simulink remoto.

Fase	*Acción*
1	La simulación envía a Jim los valores de las variables de entrada del modelo.
2	Jim actualiza los valores de las variables de entrada y solicita a Matlab la ejecución de un paso de integración.
3	Matlab ejecuta un paso de integración del modelo Simulink.
4	Jim obtiene del espacio de trabajo de Matlab los valores actualizados de las variables de salida del modelo Simulink.
5	Jim envía los valores actualizados a la simulación.
6	Finalmente, se actualiza la vista de la simulación.

La Figura 9 presenta las fases de la Tabla 1 cuando se realiza la ejecución de una simulación remota. Obsérvese que estas fases no dependen del tipo de enlace seleccionado.

Debido a que el enlace asíncrono necesita reiniciar el modelo Simulink cada vez que ocurre una interacción de usuario, es posible que la simulación no se ejecute de la forma deseada, por lo que se recomienda utilizar un enlace síncrono en una red LAN.

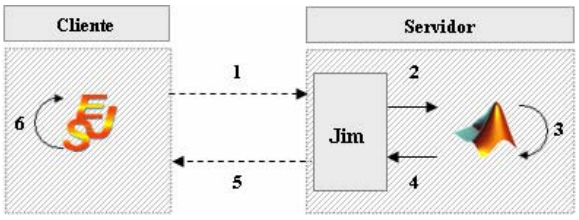

Figura 9: Fases en la ejecución de una simulación remota.

Almería 2006 - ISBN: 84-689-9417-0

4.3.1 Enlace Síncrono

El enlace síncrono entre la simulación de *Ejs* y Matlab es bidireccional, ya que en cada paso de integración se ejecutan las seis fases de la Tabla 1, por lo que la información fluye en ambos sentidos.

Se debe considerar que en la versión remota del enlace síncrono, el retardo no sólo es debido a la red, ya que tanto la simulación de Ejs como Matlab se bloquean mutuamente mientras se realiza algún cómputo. De esta forma el retardo total en cada paso de evolución es la suma del retardo de la red y el del tiempo de ejecución.

Pese a las desventajas mencionadas anteriormente, el enlace síncrono permite a los usuarios de *Ejs* convertir una simulación local en una simulación remota en forma prácticamente instantánea. El comportamiento de la simulación será en ambos casos el mismo, exceptuando los retardos en el enlace remoto anteriormente descrito.

4.3.2 Enlace Asíncrono

Debido a los retardos inherentes en las comunicaciones por red, se decidió desarrollar una forma de realizar el enlace entre Jim y Matlab diferente a la utilizada entre *Ejs* y Matlab.

El enlace síncrono requiere una enorme cantidad de transacciones entre la simulación de *Ejs* y Jim. Sin embargo, mientras más comunicación tiene la simulación más lenta es la evolución de ésta.

Para atenuar los retardos introducidos en versión remota, el enlace asíncrono no mantiene una coordinación constante entre el modelo Simulink y la vista de la simulación. Este "descuido" se basa en la *idea clave* de que el usuario no interactúa con la simulación en todo momento, es decir, el usuario no introduce cambios en los parámetros de la simulación en cada instante, porque si se realiza uno, requerirá tiempo para observar la respuesta del sistema antes de volver a hacer otro.

Debido a lo anterior, la comunicación entre la simulación y Simulink será unidireccional en la mayoría de los pasos de evolución, presentando una bidireccionalidad sólo en los instantes en que el usuario interactúa con la simulación.

En el caso unidireccional, Jim enviará a la simulación los valores de las variables de salida del modelo, mientras que la simulación enviará a Jim los valores de las variables de entrada del modelo sólo en el caso en que el usuario interactúe con la interfaz.

La Figura 10 presenta los cronogramas de fases vistas en la Tabla 1 tanto para el enlace síncrono como asíncrono. Al comienzo un usuario realiza una interacción con la simulación, por lo que todas las fases son ejecutadas, sin embargo cuando la sexta fase es finalizada, en el enlace síncrono la primera fase se realiza de nuevo, mientras que en el enlace asíncrono la primera fase sólo será ejecutada cuando el usuario interactúe nuevamente.

Es fácil observar en los diagramas temporales los tiempos ociosos en el caso síncrono, y los cálculos simultáneos del enlace asíncrono.

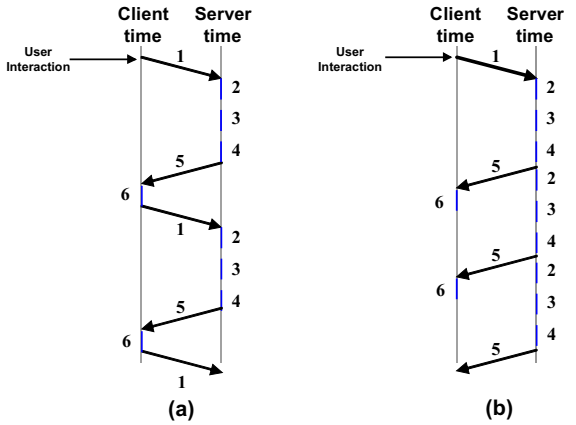

Figura 10: Cronogramas del cliente y servidor para los enlaces síncrono y asíncrono. Los números representan las fases de la Tabla 1.

Como se describió en la sección 3, en la comunicación síncrona, cada paso de evolución de la simulación, tanto en la vista como en el modelo Simulink, es controlado directamente por *Ejs*. Es decir, el enlace es bidireccional en todos los pasos de la evolución. La elección del enlace síncrono hará que la ejecución de la simulación en forma remota sea similar a la local, pero con los retardos propios de la red. Por otra parte, el enlace asíncrono disminuye el efecto del retardo, pero debido a que requiere reiniciar la simulación del modelo Simulink cada vez que el usuario interactúa, es posible que el comportamiento de la simulación no sea similar a su versión local, por ello se recomienda realizar las pruebas necesarias antes de publicar la simulación.

5 EJS, JIM Y SIMULINK

El esquema de conexión entre la simulación de *Ejs*, Jim y Simulink se presenta en la Figura 11. Obsérvese que el cálculo se realiza en el lado del servidor, mientras que en el lado del cliente se tiene la interfaz de la simulación.

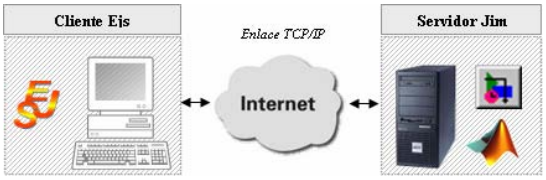

Figura 11. Conexión entre *Ejs*, Jim y Matlab

Uno de los aspectos considerados para el uso de un modelo Simulink remoto, es minimizar los cambios requeridos para convertir una simulación local existente a una versión remota. Esto se ha logrado en forma prácticamente absoluta en el caso del enlace síncrono, mientras que en el asíncrono sólo se requiere realizar algunas pequeñas modificaciones.

5.1 ESTABLECIENDO UN ENLACE SÍNCRONO

En el enlace síncrono el único cambio requerido, para convertir una simulación local existente, es que el usuario especifique la dirección IP y el puerto del servidor remoto. La Figura 12 presenta un ejemplo de lo anterior, en donde se convierte la simulación de la de la sección 3 a una versión remota introduciendo la IP y el puerto en el cuadro de texto *External File*.

Una vez especificado el enlace remoto, no es necesario realizar ninguna modificación más a la simulación. Incluso, si el modelo Simulink no se encuentra en el servidor remoto, *Ejs* se encarga de enviárselo cuando sea necesario.

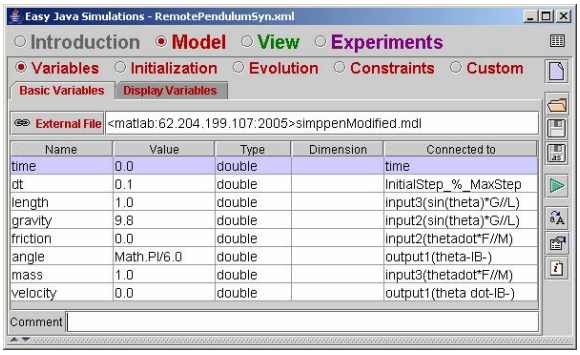

Figura 12. Estableciendo un enlace remoto síncrono.

5.2 ESTABLECIENDO UN ENLACE ASÍNCRONO

Para establecer un enlace asíncrono entre una simulación local existente, y un servidor remoto se debe cambiar la palabra clave *matlab* por *matlabAS*, seguida por la dirección IP y el puerto. Por esta razón en la Figura 12 se debería observar en el cuadro de texto *External File* lo siguiente: "*<matlabAS:62.204….*" en vez de "*<matlab:62.204…..*".

Otra modificación necesaria para establecer el enlace asíncrono, es utilizar el siguiente método para informar a Jim que se han realizado cambios en las variables de entrada del modelo Simulink:

_external.synchronize();

En el caso del modelo del péndulo simple presentado en la sección 3, el método anterior debe utilizarse para detectar la interacción del usuario con los deslizadores tal como se observa en la Figura 13. En este caso el texto sombreado corresponde al método anterior, y es definido en una de las propiedades del elemento de la vista que controla la masa.

Respecto de las propiedades del elemento de la vista que representa la bola del péndulo no se requiere ninguna modificación, porque *Ejs* ejecuta implícitamente un *_external.synchronize()* cuando se ejecuta el método *_external.resetIC()* al cambiar la posición de la bola.

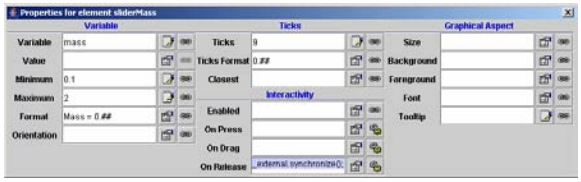

Figura 13. Propiedades del elemento de la vista que controla la masa de la bola. El texto sombreado corresponde al método _external.synchronize();

6 EJEMPLOS

A continuación se presentan dos prototipos desarrollados utilizando las herramientas presentadas anteriormente. Como primer ejemplo se expone la utilización del modelo del péndulo de la Figura 4. Posteriormente se describe una simulación que utiliza funciones de la ToolBox Control System de Matlab. Los ejemplos pueden ser ejecutados desde cualquier ordenador que soporte una conexión TCP/IP.

6.1 MODELO DEL PÉNDULO SIMPLE DE SIMULINK

Esta simulación corresponde al modelo del péndulo simple descrito en la sección 3, utilizando un enlace de tipo asíncrono. En la Figura 14 se observa la aplicación en la que el usuario puede interactuar con diferentes parámetros del modelo del péndulo simple. La aplicación puede ser accedida a través de un navegador con la siguiente dirección:

http://lab.dia.uned.es:8080/rmatlab/pendulum/RemoteSimulinkAsynchronous.html

Almería 2006 - ISBN: 84-689-9417-0

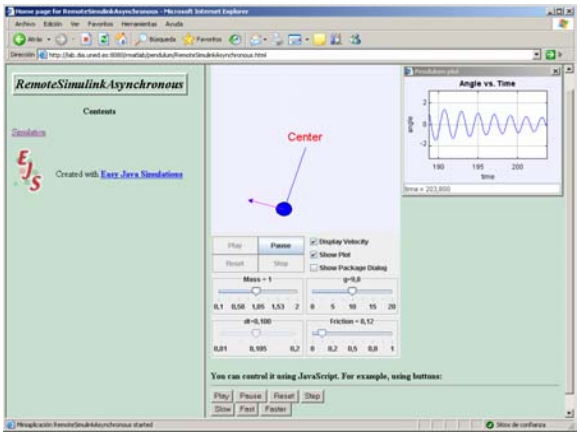

Figura 14. Vista de la aplicación que simula el péndulo simple.

6.2 HERRAMIENTAS BÁSICAS DE CONTROL

A pesar de que en este trabajo no se ha descrito cómo utilizar los métodos propios de *Ejs* para ejecutar o modificar variables en el espacio de trabajo del Matlab remoto, quisiéramos adelantar al usuario de *Ejs* las posibilidades que brinda la conexión mencionada para ejecutar comandos o funciones de Matlab.

En la aplicación de la Figura 15, un usuario puede modificar los polos y ceros de un modelo y observar las gráficas proporcionadas por diferentes funciones de la ToolBox Control System de Matlab. La simulación puede ser accedida a través de la siguiente dirección:

http://lab.dia.uned.es:8080/rmatlab/cbtool/cbTool.html

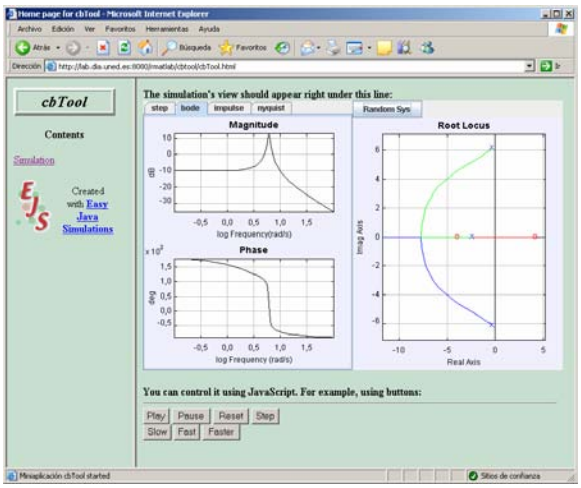

Figura 15. Vista de la aplicación que utiliza funciones de la ToolBox Control System de Matlab.

7 CONCLUSIONES

El desarrollo de laboratorios virtuales y remotos es un hecho de especial relevancia en la Ingeniería de Control, en donde la experimentación con una planta real proporciona un aprendizaje difícil de obtener sin la utilización de la misma.

El aspecto central de este trabajo consistió en describir la extensión incorporada a *Ejs* que permite a un usuario ejecutar una simulación que utiliza un modelo Simulink, sin el requisito de tener instalado Matlab/Simulink en su equipo. Para lograrlo *Ejs* realiza una conexión TCP/IP con un servidor remoto, denominado Jim, que sí posee Matlab/Simulink.

Agradecimientos

Los autores agradecen al Ministerio Español de Investigación a través de la concesión del proyecto DPI2004-01804.

Referencias

[1] Aktan B., Bohus C., Crowl L., Shor M., (1996) "Distance Learning Applied to Control Engineering Laboratories", IEEE Transactions on Education, Vol. 39, n. 3, Agosto, pág. 320-326.

[2] Dormido, S. (2004) "Control learning: Present and future" IFAC Annual Control Reviews, vol. 28, pp 115-136.

[3] Dormido S., Esquembre F., Farias G., Sánchez J. (2005). Adding interactivity to existing Simulink models using Easy Java Simulations, *44th IEEE* CDC/ECC, Sevilla (Spain).

[4] Esquembre F. " Easy Java Simulations", http://fem.um.es/Ejs

[5] Esquembre F., (2005) "Creación de Simulaciones Interactivas en Java", Pearson Educación S.A., Madrid.

[6] Guillet D., Nguyen A., Rekik Y. (2005). Collaborative Web-Based Experimentation in Flexible Enginnering Education, IEEE Transactions on Education, Vol. 48, Nº 4.

[7] Müller S. (2005), www.held-mueller.de/JMatLink/

[8] Sánchez J., Dormido S., Esquembre F. (2005) "The learning of control concepts using interactive tools". Computer Applications in Engineering Education , vol. 13, Nº1, pp 84-98.

[9] Sánchez J., Dormido S., Pastor R., Esquembre F., (2004)"Interactive learning of control concepts using Easy Java Simulations", Plenary Lecture, IFAC Workshop Internet Based Control Education IBCE'04, Grenoble (France), september.