CONEXIÓN DE ECOSIMPRO Y SYSQUAKE EN UNA HERRAMIENTA INTERACTIVA DE CONTROL NO LINEAL

Sebastián Dormido Bencomo
Dpto. Informática y Automática, Facultad de Ciencias
UNED, Fac. de Ciencias, Dpto. de Informática y Automática. Avda. Senda del Rey 9, 28040 Madrid, Spain
e-mail: sdormido@dia.uned.es

Ramón Perez Vara y Pedro Cobas
Empresarios Agrupados
Magallane, 3, 28015 Madrid, Spain
e-mail: pce@ecosimpro.com, rpv@empre.es

Resumen

SysQuake es una herramienta orientada a la simulación y a la visualización de datos científicos. Mediante una concepción innovadora del concepto de gráficos interactivos SysQuake, cuyo lenguaje de base es esencialmente un clónico de Matlab, proporciona primitivas de base muy potentes y simples para la resolución de problemas matemáticos complejos donde la interactividad juega un papel clave. Esta es particularmente la situación que se plantea en los problemas de control donde con frecuencia hay que correlacionar representaciones en los dominios frecuencial y temporal. En este trabajo se describe como conectar SysQuake con EcosimPro en un ejemplo concreto que muestra como esta interactividad puede ser enormemente útil en los lenguajes de modelado acausal orientado a objetos de sistemas híbridos.

Palabras Clave : Educación en control, EcosimPro, Interactividad, SysQuake.

1 INTRODUCCIÓN

Las ideas, del control automático son realmente ricas en contenido visual que puede representarse de forma intuitiva y geométrica. Estos contenidos visuales se pueden emplear para presentar tareas y manejar conceptos y métodos y manipularlos en la resolución de los problemas.

Las ideas básicas del control automático a menudo surgen de situaciones visuales muy específicas y todos los expertos en control reconocen la gran utilidad que supone partir de estas concreciones cuando necesitan manejar los correspondientes objetos abstractos.

Esta forma de actuar con atención explícita a las representaciones específicas potenciales para explicar las relaciones abstractas que son de interés es lo que denominamos *visualización en control*.

Nuestra percepción es fundamentalmente visual y por lo tanto no nos debe sorprender que las ayudas mediante visualizaciones estén tan presentes en nuestro trabajo. Con gran frecuencia utilizamos procesos simbólicos, diagramas visuales y otras formas de procesos imaginativos que nos permite adquirir lo que se podía llamar una cierta intuición de lo abstracto.

La visualización aparece así como algo profundamente natural tanto en los orígenes del control automático como en el descubrimiento de nuevas relaciones entre objetos matemáticos y también por supuesto en la transmisión y comunicación de nuestro conocimiento del control.

Estos aspectos intuitivos son probablemente muchos más difíciles de explicitar y asimilar por los estudiantes precisamente porque muy a menudo se encuentran en el sustrato menos consciente de la actividad del especialista (Dormido, 2002).

A partir de estas consideraciones generales el computador se puede considerar como una herramienta que nos permite visualizar y manipular de forma interactiva los objetos que son propios del control automático. El objetivo último es facilitar la comprensión de los conceptos que se tratan de transmitir y enseñar a nuestros estudiantes.

Tradicionalmente los sistemas se diseñan siguiendo un proceso iterativo. Las especificaciones del problema no se suelen utilizar normalmente para calcular el valor de los parámetros del sistema porque no hay fórmula explícita que las relacione de forma directa. Esta es la razón de dividir cada iteración en dos fases. La primera a menudo llamada síntesis, consiste en calcular los parámetros desconocidos del sistema, tomando como base un grupo de variables de diseño (que están relacionadas con las especifica-

ciones). Durante la segunda fase, llamada análisis, el comportamiento del sistema se evalúa y se compara con las especificaciones. Si no concuerdan, se modifican las variables de diseño y se lleva a cabo una nueva iteración. Es posible, sin embargo, fundir ambas fases en una de manera que la modificación de los parámetros produce un efecto inmediato. De esta manera el proceso de diseño se hace realmente dinámico y el estudiante percibe el gradiente del cambio en el criterio de comportamiento para los elementos que se están manipulando. Esta capacidad interactiva permite identificar mucho más fácilmente los compromisos que se pueden alcanzar.

En estos años se han desarrollado muchas herramientas destinadas a la educación en control. Muchas ideas y conceptos interesantes fueron implementados por el grupo del Prof. Åström en Lund. En este contexto merecen destacarse los conceptos de *cuadros dinámicos* y *sistemas interactivos virtuales* introducidos por Wittenmark *et al*., 1998. El objetivo principal de estas herramientas es tratar de obtener una participación mucho más activa de los estudiantes en cursos de control automático.

En esencia, un cuadro dinámico es una colección de ventanas gráficas que se manipulan simplemente utilizando el ratón. Los estudiantes no tienen que escribir ninguna sentencia de código o aprender nada extra al objetivo central de su aprendizaje. Si los estudiantes cambian cualquier elemento activo en las ventanas gráficas se inicia de forma automática un nuevo procesamiento y presentación de los resultados. De esta manera se percibe como sus modificaciones afectan a los resultados obtenidos.

Estas herramientas interactivas, intentan "desmitificar" conceptos matemáticos abstractos a través de la visualización de ejemplos seleccionados ad hoc. En la actualidad una nueva generación de paquetes de software ha creado una alternativa interesante para el aprendizaje interactivo del control automático (Garcia and Heck, 1999). Su principio de funcionamiento se basa en objetos que permiten una manipulación gráfica directa. Durante estas manipulaciones, los objetos se actualizan de forma inmediata, de forma que la relación entre los objetos se mantienen en todo momento. *Ictools and CCSdemo* (Johansson *et al*., 1998; Wittenmark *et al*., 1998), desarrollado en el Departamento de Control Automático de la Universidad de Lund y *SysQuake* en el Instituto de Automática de la Escuela Politécnica Federal de Lausanne, (Pyguet, 1999).

Con esta filosofía se ha desarrollado una herramienta interactiva para explicar conceptos básicos de control no lineal (Dormido *et al*., 2002) utilizando SysQuake como herramienta de trabajo. El problema que plantea SysQuake es que no incorpora de forma automática la detección de eventos discretos planificados en el estado lo que complica extraordinariamente el cálculo de las trayectorias en el espacio de estados. Sin embargo esta detección se hace de forma muy natural en los lenguajes de modelado acausal como EcosimPro. Desde esta perspectiva en el presente trabajo se propone la conexión de SysQuake y EcosimPro para simplificar la solución interactiva de esta clase de problemas.
En la sección 2 se expone brevemente el problema planteado. En la sección 3 se explica la forma de llevar a cabo la conexión de un modelo escrito en EcosimPro con SysQuake. En la sección 4 se describe brevemente la solución adoptada en el caso de la aplicación de control no lineal descrita en la sección 2. Finalmente en la sección 5 se exponen las conclusiones de este trabajo.

2. DESCRIPCIÓN DE LA APLICA-CIÓN DE CONTROL NO LINEAL

Los sistemas que se pueden estudiar tienen la forma que se describe en la Figura 1. El elemento lineal es cualquier función de transferencia propia (está previsto que versiones futuras de la herramienta incorporen retardos puros de tiempo).

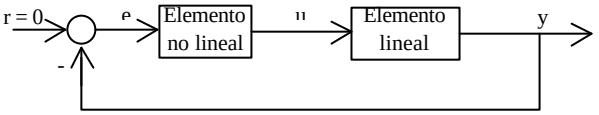

Figura 1: Estructura del sistema

El elemento no lineal tiene la forma general que se muestra en la Figura 2 que cubre una gran variedad de no linealidades que son lineales a tramos con simetría impar y que incluye la mayoría de las no linealidades multivaluadas clásicas (Sridhar, 1960). Entre muchas otras permite describir las siguientes no linealidades: zona muerta, saturación, zona muerta + saturación, relé y relé con histéresis.

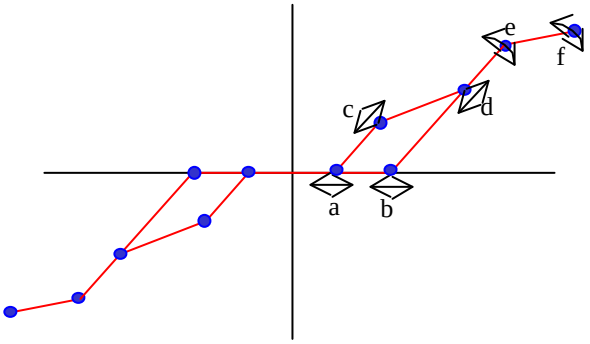

Figura 2: No linealidad genérica

La configuración de la no linealidad se efectúa de forma interactiva: el usuario sólo tiene que mover con el ratón los puntos activos (que se representan por

pequeños círculos) en la ventana "Elemento no lineal" de la herramienta. La dirección de las flechas indica el tipo de desplazamiento que se puede realizar sobre cada uno de los puntos.

En la Figura 3 se muestran el resto de las ventanas que incorpora la herramienta. Estas ventanas son: "Parámetros", "Función de transferencia", "Entrada al elemento no lineal", "Parte real y parte imaginaria de la función descriptiva", "Curva de Nyquist y lugar crítico", "Trayectoria fásica", "Entrada" y "Salida". Para un análisis detallado de la aplicación véase Dormido *et al*., 2002

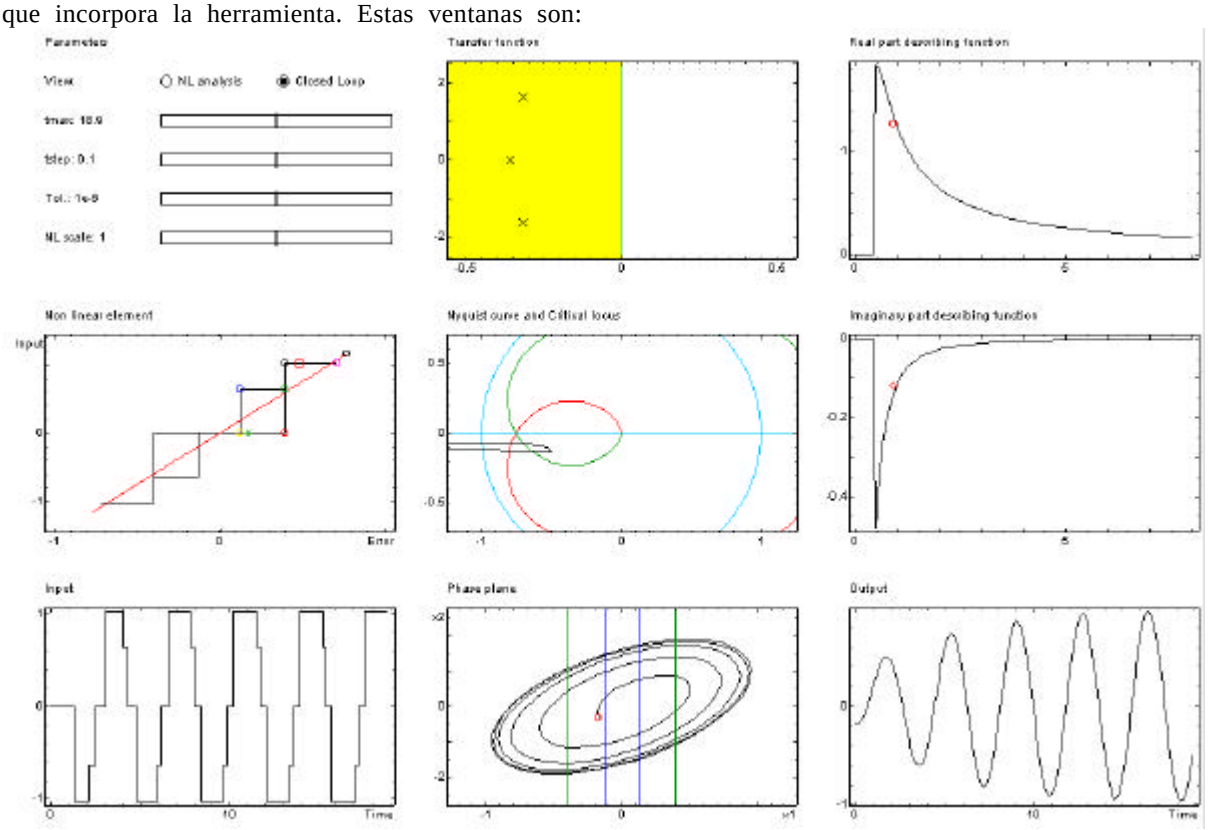

Figura 3 Vista múltiple de las ventanas que incorpora la herramienta

3. CONEXIÓN DE UN MODELO ECOSIMPRO CON SYSQUAKE

Conectar un modelo generado con EcosimPro con SysQuake es fácil, pues hay que crear simplemente una *interfaz dll que llame al modelo de Ecosim.* Como siempre suele ocurrir en estos casos, lo difícil es hacerlo la primera vez, pues para las sucesivas veces el marco de programación será idéntico o muy parecido. En un futuro se podría evaluar la generación automática de la dll, pero por el momento requiere un trabajo manual de programación.

Hay que empezar diciendo que para realizar esta interfaz se requiere lo siguiente:

- Conocer MS Visual C++. Concretamente la creación y manejo de librerías dinámicas y programación en C++.
- Leer el manual de SysQuake acerca de la conexión a rutinas externas.
- Leer el manual de conexión de EcosimPro con SysQuake [Cobas, 2002].

El diagrama de la Figura 4 muestra el proceso de conexión:

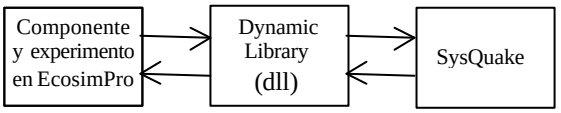

Figura 4: Diagrama de conexión Ecosim-SysQuake

En primer lugar hay que modelar el sistema con EcosimPro y probarlo muy exhaustivamente. Esto nos garantizará que el modelo construido es robusto y que está preparado para trabajar con cualquier dato de entrada. Esta tarea puede ser ardua y difícil de llevar a cabo para modelos no triviales. Hay que tener en cuenta que uno de los puntos fuertes de SysQuake es que podemos hacer que el usuario cambie de forma interactiva cualquier dato de entrada, por eso hay que garantizar que el modelo responderá adecuadamente.

Una vez que el modelo de EcosimPro funciona correctamente para cualquier tipo de entrada podremos pasar a conectarlo con SysQuake. Los pasos que deben seguirse son:

- Determinar cuales son los datos de entrada y salida al modelo
- Crear un experimento que integre el modelo un tiempo que puede venir dado por SysQuake
- Crear unos arrays en el experimento que nos almacenen las variables que queremos devolver a SysQuake
- Probar este experimento exhaustivamente
- Crear una dll con el código generado que enganche este modelo con SysQuake
- Si procede (manejo avanzado de SysQuake) programar en SysQuake un código para conectarse a la función externa de Ecosim (dll) que defina adecuadamente las entradas y salidas a la función. Si no se crea ningún código siempre se podrá llamar directamente a la función creada desde SysQuake (ver ejemplo más adelante)
- Arrancar SysQuake y comprobar que la dll carga la función dll desarrollada
- Ejecutar SysQuake cambiando las variables y comprobando que los resultados que se dibujan en SysQuake son correctos.

A continuación se explicarán a muy grandes rasgos (para detalles ver Cobas, 2002) estos pasos con un componente trivial en EcosimPro que resuelve la ecuación "x= sin(TIME)+d1", donde x será la variable a calcular, TIME el tiempo actual de integración y d1 un dato conocido que el usuario podrá enviar desde SysQuake.

Para ello en primer lugar se modela el componente en Ecosim:

```
COMPONENT pruSysQuake
                DATA
   REAL d1 = 1.9
                DECLS
   REAL x
CONTINUOUS
   x = sin(TIME) + d1
END COMPONENT
```

A continuación se escribe un experimento que integre este modelo un tiempo y almacene los valores de TIME y x en un array:

```
EXPERIMENT exp1 ON
pruSysQuake.default
DECLS
   CONST INTEGER maxDatos= 400
   REAL mtime[maxDatos]
   REAL mx[maxDatos]
   INTEGER i
BODY
    TIME= 0 - CINT
    i = 1
WHILE (INTEG_CINT() !=INTEG_END)
  mtime[i]= TIME --rellena con t
    mx[i] = x      --rellena con x
    i = i + 1
END WHILE
END EXPERIMENT
```

El dato de entrada va a ser "d1" y los arrays de salida serán "mtime" y "mx". Para el ejemplo se integrará un intervalo fijo que se inicializa en la dll. En un caso más sofisticado se puede hacer que TSTOP y CINT sean datos de entrada.

3.1 CREACIÓN DE LA DLL

La creación de la dll se hace generando un proyecto en Visual C++ para lo cual se elige el proyecto de nueva creación como "Win32 Dynamic-link library". Este proyecto incluirá las clases generadas por Ecosim, así como otras que usa SysQuake (ver detalles en Cobas, 2002). La función principal de la dll debe tener una estructura del tipo siguiente:

```
static lme_int32
ecosimSeno(lme_ref lme,
lme_int32 nargin, lme_int32
nargout)
{
…
}
```

Los argumentos importantes son nargin (número de argumentos de entrada) y nargout (número de argumentos de salida). Ambos se deberán inicializar en esta función. En nuestro caso nargin= 1 (dato "d1") y nargout= 2 (arrays "mtime" y "mx").

Para conectarse al modelo de Ecosim no habrá más que instanciar la clase del experimento y llamar al método initEcosim. Por ejemplo:

```
pruSysQuake _default_exp1
modelo;
initEcosim( &modelo );
```

Ahora se dispone ya de un modelo Ecosim en la dll. A partir de aquí se harán tres cosas:

1- Leer las matrices de entrada de SysQuake y pasárselas al modelo de Ecosim
2- Ejecutar el modelo
3- Leer los resultados y transmitirlos a SysQuake en su formato de matrices

3.2 LECTURA DE ARRAYS DE ENTRADA

Para leer d1, hay que pasarlo como matriz de entrada de dimensión 1. Para ello se debe usar la función LMECB_GetMatrix(), por ejemplo:

```
lme_int32 status=
LMECB_GetMatrix(narg, &nf, &nc,
&D, 0);
if (!status)
  return 0;
modelo.setValueReal("d1", D[0]);
```

La última línea pone el valor que contenga D[0] en el dato "d1" del modelo. Existe otra forma de hacerlo, más engorrosa, pero más rápida. Se puede obtener la dirección interna de cualquier variable del modelo en Ecosim de la siguiente manera:

```
double* d1= (double*)
modelo.getVarAddress("d1");
```

A partir de ahora la variable d1 es como si apuntara a la d1 del modelo de Ecosim. Esto permite hacer por ejemplo:

```
*d1 = D[0];
```

Esto último sería totalmente equivalente a lo hecho por el primer método.

3.3 ASIGNACIÓN DE ARRAYS DE SALIDA

Hay que indicar a SysQuake cuales son las direcciones de memoria de los arrays de salida. Serán dos arrays propios de SysQuake que posteriormente se rellenarán con los resultados.

3.4 PROGRAMAR LA EJECUCIÓN DE LA SIMULACIÓN

Para simular se le puede dar el tiempo inicial, final y el intervalo de comunicación fácilmente desde la dll. También se podría pasar desde SysQuake, pero por simplificar en este caso se establecen tiempos fijos de simulación y método de integración:

```
modelo.TIME= 0;
modelo.TSTOP= 1;
modelo.CINT= 0.2
modelo.IMETHOD= RK4;
```

Después se ejecuta el experimento:

```
modelo.runExperiment(); //
ejecuta el experimento de Ecosim
```

3.5 LECTURA DE ARRAYS DE SALIDA

Finalmente habrá que leer los resultados de EcosimPro y pasárselos a SysQuake.

```
modelo.getArray1D("mtime", Time,
nDatos);  // Lee Time
modelo.getArray1D("mx", X,
nDatos);   // Lee X
```

Esta llamada rellena los arrays Time y X con los valores leídos en "mtime" y "mx". A partir de ahora SysQuake ya tiene los resultados.

3.6 INICIALIZACIÓN DE LA FUNCIÓN EXTERNA

Aparte de esta función "ecosimSeno" habrá que crear una función llamada "InstallFn" que cargue la dll en modo automático y una estructura lme_fn estática que defina el nombre y el numero de argumentos. La misión de esto será que SysQuake detecte de forma automática que hay una nueva función a cargar.

3.7 EJECUCIÓN DE SYSQUAKE

Una vez generada la dll se pondrá en el directorio LMEExt de SysQuake y cuando se ejecute el programa SysQuake, se deberá cargar de forma automática (debe aparecer un mensaje que así lo indique). A partir de ahora la función ecosimSeno() estará disponible en SysQuake y podrá ser llamada con distintos valores de d1 y obtendríamos dos matrices: una primera con los valores de TIME y otra con los de X. Por ejemplo si se le pasa la función:

```
ecosimSeno([1.9])
```

devuelve

```
mtime[0, 0.2, 0.4, 0.6. 08, 1.0]
mx   [1.9, 2.0986, 2.2894,
2.4646,  2.6173, 2.7414 ]
```

Esta función podrá ser usada desde el código de SysQuake como si fuera una función más de la librería de SysQuake.

4. CONEXIÓN DE UN MODELO ECOSIMPRO CON SYSQUAKE

Desde el punto de vista del cálculo de las trayectorias de la aplicación de control no lineal descrita en la sección 2. El problema es una concatenación de tramos de trayectorias lineales entre cada dos "puntos de cruce" consecutivo por la característica no lineal que puede modificarse de forma interactiva por el usuario. Como Sysquake no dispone de mecanismos automáticos para la detección de los puntos de cruce esto queda como tarea que hay que desarrollar dentro de un margen de tolerancia definido previamente. El problema es particularmente difícil desde un punto de vista numérico porque para determinadas no linea-lidades la trayectoria puede presentar características de "modo deslizante".

Una alternativa interesante que se ha implementado en este trabajo es conjugar las capacidades interactivas de SysQuake con la posibilidad de

utilizar los "solvers" que incorpora EcosimPro que detectan automáticamente los puntos de cruce por la no linealidad sin necesidad de que el usuario tenga que preocuparse de ello. En esencia SysQuake se encarga de la modificación interactiva de los elementos siguientes:

1- Característica no lineal, que viene descrita por las coordenadas de los puntos que la definen.

```
[xa xb xc yc xd yd xe ye xf yf]
```

2- Función de transferencia lineal descrita por su modelo en el espacio de estados.

```
[A B C D]
```

3- Condición inicial del vector de estado `x0`

4- Parámetros de la simulación dinámica

```
[tmax tstep metodo_integracion]
```

Por su parte EcosimPro devuelve el vector de instantes de tiempo `t`, el vector de estado `x` y el vector de valores de la salida en esos instantes `y`.

5. CONCLUSIONES

El presente trabajo ha mostrado como se pueden conectar SysQuake y EcosimPro en una aplicación interactiva donde la determinación de los eventos en el estado son detectados por el solver de EosimPro. Este planteamiento simplifica enormemente la escritura del programa y facilita su depuración. Las ideas apuntadas son fácilmente trasladables a situaciones análogas. es posible automatizar aún más el procedimiento de conexión de ambas herramientas. Un aspecto que el trabajo ha puesto de manifiesto es la posibilidad de extender la potencialidad de EcopsimPro dotando a su lenguaje de descripción de "experimentos" con primitivas que permitan el desarrollo de aplicaciones interactivas. Esta tarea no plantea problemas conceptuales cuando las variables interactivas que se modifican no cambian el "índice del problema".

Referencias

Cobas, P.: (2002). "Manual de conexión de un modelo EcosimPro a SysQuake", EA International. Nov-2002

Dormido, S. (2002) Control Learning: Present and Future. Plenary Lecture. *15th Triennial World Congress of IFAC, Barcelona, Spain*

Dormido, S., F. Gordillo, S. Dormido-Canto, J. Aracil (2002). An interactive tool for introductory nonlinear control systems education. *15th Triennial World Congress of IFAC, Barcelona, Spain*.

Garcia, R. C.; Heck, B. S.: (1999). "Enhancing classical controls education via interactive GUI design", *IEEE Control Systems Magazine*, **19**, nº 3, pp. 77-82.

Johansson, M.; Gäfvert, M.; Åström, K. J.: (1998). "Interactive tools for education in automatic control", *IEEE Control Systems Magazine*, **18**, nº 3, pp. 33-40.

Piguet, Y.: (1999). "SysQuake: User Manual", Calerga.

Wittenmark, B.; Häglund, H.; Johansson, M.: (1998). "Dynamic pictures and interactive learning", *IEEE Control Systems Magazine*, **18,** nº 3, pp. 26-32.