Simulación en tiempo real de un MPC utilizando herramientas de software libre

Erik Marichal Arbona[1], Ronald Ugas Lago[2], Alberto Aguado Behar[3], Rafael F. Tanda Martínez[4]

[1] Universidad de las Ciencias Informáticas (UCI), Dirección de Energía., emarichal@uci.cu

[2] Universidad de las Ciencias Informáticas (UCI), Facultad 7 rugas@uci.cu

[3 y 4] Instituto de Cibernética, Matemática y Física (ICIMAF), Departamento de Control Automático aguado,tanda@icimaf.inf.cu

RESUMEN

En este trabajo se implementa un controlador predictivo basado en modelos ARX en forma de bloque para el CACSD *Scicos*. El controlador es simulado sobre una plataforma RCP basada en herramientas de código abierto y libre distribución, como *Scicoslab/Scicos*, RTAI-LAB y el sistema operativo de tiempo real *Linux/RTAI*. Para la validación de esta estrategia de control se utiliza un proceso de segundo orden con un retardo considerablemente grande. La visualización y recolección de los datos, así como el ajuste de los parámetros de la tarea de tiempo real en ejecución se hacen mediante un osciloscopio virtual llamado *Qrtailab*.

Palabras claves: Controlador predictivo basado en modelos, Linux/RTAI, Sistemas de tiempo real.

Real-Time simulation of a model-based predictive controller using free software tools

ABSTRACT

In this paper we implemented a ARX model-based predictive controller in block form for CACSD Scicos. The controller is simulated on a RPC (Rapid Prototyping Controller) environment based on open source and free tools such as Scicoslab/Scicos, RTAI-LAB and real-time operating system Linux/RTAI, an extension of the standard Linux kernel. The controller was programmed in C language as a Scicos computational function. To validate this control strategy using a second order process with a considerably larger delay. Visualization and data collection as well as adjusting the parameters of the real-time target execution are performed by a virtual oscilloscope called Qrtailab.

Key words: Model-based predictive controller, Real time systems, RTAI.

INTRODUCCIÓN

Los prototipos rápidos de control o RCP (*Rapid Controller Prototyping*) fueron introducidos en la década del 90 por la industria automotriz, con el objetivo de diseñar, analizar y probar de forma iterativa las estrategias de control. Los entornos RCP están compuesto por dos elementos: una aplicación CACSD (Computer Aided Control System Design) y un hardware provisto de un sistema operativo de tiempo real. Uno de los entornos RCP más populares está basado en el *Matlab/Simulink/Realtime-Workshop* que a pesar de ser muy potente y de contar con una excelente interfaz gráfica, tiene como principal inconveniente su costo[1]. Una sugerente alternativa de código abierto y libre distribución es la integración entre *Scicoslab/Scicos* y *Linux/RTAI*.

Una herramienta llamada RTAI-LAB es quién garantiza la compatibilidad entre *Scicoslab/Scicos* y *Linux/RTAI*, esta provee a *Scicos* de un editor gráfico de diagramas de bloques de *Scicoslab*; de un nuevo generador de código, una versión modificada del original, para que los programas compilados puedan ejecutarse sobre *Linux/RTAI*. También suministra una librería con un conjunto de bloques entre ellos: señales de entradas y otros que permiten a los sistemas diseñados, interactuar con tarjetas de adquisición de datos. Por último,

proporciona un osciloscopio virtual llamado *Xrtailab* para el monitoreo y la recolección de datos del proceso, así como para el ajuste de los parámetros de las tareas de tiempo real en ejecución.

Scicos brinda la posibilidad de adicionar nuevos toolboxes a partir de los requerimientos del usuario, que indudablemente contribuyen a aumentar las prestaciones de esta herramienta. Cuenta, además, con una comunidad de desarrollo encargada de incrementar sus potencialidades. A pesar de ello, hay pocas contribuciones que implementan estrategias de control avanzado y las librerías que trae incluidas por defecto, sólo ofrecen técnicas de control clásico. Son precisamente estas limitaciones las que motivan en este trabajo a la creación de un nuevo bloque *Scicos* con un controlador predictivo basado en modelo, conocido como PREDARX.

El controlador PREDARX es una variante del GPC (*General Predictive Controller*) que en lugar de utilizar un modelo CARIMA utiliza un modelo clásico ARX, el cual resulta más fácil y sencillo de identificar. Al igual que el GPC este regulador se caracteriza por su robustez y flexibilidad.

MATERIALES Y MÉTODOS

En este epígrafe se describen las principales características de las herramientas de software libre utilizadas para el desarrollo y puesta a punto del *Controlador Predictivo basado en modelos ARX* que se propone. También quedan reflejados los métodos utilizados en la elaboración de los bloques *Scicos* referentes a dicho regulador.

SCICOSLAB/SCICOS

Es un software de código abierto destinado al cálculo científico creado por un grupo de investigadores que inicialmente fueron los desarrolladores de *Scilab* en el INRIA (*Institut National de Recherche on Informatique et on Automatique*) y puede ser descargado libremente desde su sitio[1] oficial en Internet. Este incluye un centenar de funciones específicas y de propósito general para el cálculo numérico, organizado en librerías llamadas *toolboxes*, que cubren diferentes aéreas como simulación, optimización, sistemas de control y procesamiento de señales.

Scicos es un importante toolbox para *Scicoslab* que proporciona un editor gráfico de diagramas de bloques para la construcción y simulación de sistemas dinámicos. Estos bloques representan funciones fundamentales predefinidas en *Sicoslab* o definidas por el usuario[2].

RTAI (*Real Time Application Interface*)

Esta interfaz es una extensión del kernel estándar de *Linux* desarrollado por el *Departamento de Ingeniería Aeroespacial del Politécnico de Milán (DIAPM)*, Italia. *RTAI* le atribuye al sistema operativo *GNU/Linux* propiedades de tiempo real. Asume al kernel como si fuera una tarea más, asignándole una baja prioridad. Como se puede apreciar en la figura 1, las interrupciones originadas en el procesador, principalmente señales de error como división por cero, las atiende directamente el kernel estándar, pero las interrupciones de los periféricos, como es el caso de los temporizadores, son manejadas por el *Despachador de Interrupciones* de *RTAI*, y si éstas no requieren servicios de tiempo real, entonces pasan a ser atendidas por el kernel estándar.

RTAI implementa su propio mecanismo IPC (*Inter-Process Communication*) para que las tareas de tiempo real no tengan que usar el sistema de llamadas estándar de *GNU*/*Linux*. Los diferentes mecanismos de IPC están incluidos como módulos del kernel, de esta manera pueden ser cargados sólo cuando sean necesarios.

EL planificador también está desarrollado como un módulo del kernel, lo que facilita la implementación de planificadores alternativos si es necesario. Actualmente hay tres tipos de planificadores que dependen de la arquitectura del procesador:

- *Uniprocesor (UP)*: Es utilizado en plataformas con un solo procesador.
- *Multiprocesador Simétrico (SMP)*: Está diseñado para máquinas SMP y proporciona una interfaz para las aplicaciones de forma que es posible seleccionar el procesador ó procesadores que deben ejecutar una tarea. Si el usuario no especifica un procesador para la tarea, SMP selecciona el procesador en función de la carga de trabajo.
- *Multi-Uniprocesador (MUP)*: Puede ser usado en máquinas con uno o varios procesadores, pero a diferencia de SMP, a las tareas se les debe especificar el procesador que deben usar. Desde un punto de vista positivo, el planificador MUP permite un mecanismo de temporización más flexible para las tareas que los planificadores SMP ó UP.

Otro componente importante de *RTAI* es la interfaz *LXRT*, que permite desarrollar aplicaciones de tiempo real en el espacio de usuario sin tener que crear módulos del kernel[3].

RTAI-LAB

Es una herramienta que permite compilar lenguajes de bloques desarrollados en *Scilab/Scicos*, *Scicoslab/Scicos* o bien en *Matlab/Simulink* para que puedan ser ejecutados como tareas de tiempo real sobre *Linux/RTAI*. Proporciona una nueva librería con un conjunto de bloques que implementan señales de salida, recursos para la comunicación entre procesos como FIFOS, semáforos y mailbox, e interfaces para interactuar con sensores, actuadores y tarjetas de adquisición de datos. Cuenta además con un osciloscopio virtual, que facilita la visualización y recolección de los datos, así como el ajuste de los parámetros del proceso desde cualquier computadora en una red[4].

QRTAILAB

Es un osciloscopio virtual que permite al usuario interactuar con las tareas de tiempo real. Tiene una interfaz muy parecida al que ofrece *RTAI-LAB*, además de brindar las mismas funcionalidades, solo que es mucho más sencillo de compilar e instalar. Esta herramienta es de código abierto y puede ser descargada libremente desde internet.

CONTROLADOR PREDICTIVO

El método que se propone a continuación utiliza como predictor de las salidas un modelo clásico del tipo ARX, descrito por*:*

$$y(t) = A(z^{-1})y(t) + z^{-1}B(z^{-1})u(t) + n(t) \qquad 1)$$

donde:

$$A(z^{-1}) = a_1 z^{-1} + a_2 z^{-2} + \cdots + a_n z^{-n} \qquad 2)$$

$$B(z^{-1}) = b_0 + b_1 z^{-1} + b_2 z^{-2} + \cdots + b_n z^{-n} \qquad 3)$$

son polinomios del mismo grado n. La componente aleatoria o ruido del modelo $n(t)$ es considerado del tipo "drift" o deriva, un proceso estocástico de incrementos independientes, o sea:

$$n(t) = n(t-1) + \xi(t) \qquad 4)$$

siendo $\xi(t)$ una función que tiene propiedades de ruido blanco discreto, estacionario y con esperanza matemática igual a cero.

El método consiste primeramente en calcular los polinomios:

- $E^i(z)$: Polinomio de grado i, sin término independiente.
- $F^i(z^{-1})$: Polinomio de grado $n-1$, con término independiente.
- $P^i(z)$: Polinomio de grado i en z, con término independiente igual a cero.
- $Q^i(z^{-1})$: Polinomio de grado $n-1$ en z^{-1} con término independiente distinto de cero.

A continuación se muestra un algoritmo en seudocódigo que permite hallar los coeficientes de estos polinomios expresados en forma de matrices.

Condiciones iniciales:

$$Q(k\times n) = \begin{bmatrix} b_1 & b_2 & \cdots & b_n \\ 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & 0 \end{bmatrix}$$

$$F(k\times n) = \begin{bmatrix} a_1 & a_2 & \cdots & a_n \\ 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & 0 \end{bmatrix}$$

$$P(k\times k) = \begin{bmatrix} b_0 & 0 & \cdots & 0 \\ 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & 0 \end{bmatrix}$$

$$E(k\times k) = \begin{bmatrix} 1 & 0 & \cdots & 0 \\ 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & 0 \end{bmatrix} \qquad 5)$$

Algoritmo I:

$i = 2 \cdots k$

  $j = 1 \cdots k$

    $E(i, j) = E(i-1, j)$

    $P(i, j) = P(i-1, j)$

  Fin

  $aux = F(i-1, 1)$

  $E(i, i) = aux$

  $P(i, i) = Q(i-1, 1) + aux * B(1)$

  $j = 1 \cdots n-1$

    $F(i, j) = F(i-1, j+1)$

    $Q(i, j) = Q(i-1, j+1)$

  Fin

  $F(i, n) = 0$

  $Q(i, n) = 0$

  $j = 1 \cdots n$

    $F(i, j) = F(i, j) + aux * A(j)$

$$Q(i,\ j) = Q(i,\ j) + aux * B(j+1)$$

Fin

Fin

Se define ahora la matriz auxiliar T_{aux}, de $N2$ filas y Nu columnas, como:

$$T_{aux} = \begin{bmatrix} p_1^1 & 0 & 0 & 0 & \cdots & 0 \\ p_1^1 + p_1^2 & p_2^2 & 0 & 0 & \cdots & 0 \\ p_1^1 + p_1^2 + p_1^3 & p_2^2 + p_2^3 & p_3^3 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & & \ddots & \vdots \\ \sum_{i=1}^{N2} p_1^i & \sum_{i=2}^{N2} p_2^i & \sum_{i=3}^{N2} p_3^i & \cdots & \cdots & \sum_{i=Nu}^{N2} p_{Nu}^i \end{bmatrix} \quad 6)$$

La matriz T_{aux} puede generarse fácilmente a partir de la matriz de coeficientes de los polinomios P^k, para $k = N2$, calculada mediante el Algoritmo 1. En efecto, si se define la matriz de coeficientes P, en la forma:

$$P = \begin{bmatrix} p_1^1 & 0 & 0 & \cdots & 0 \\ p_1^2 & p_2^2 & 0 & \cdots & 0 \\ p_1^3 & p_2^3 & p_3^3 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ p_1^{N2} & p_2^{N2} & p_3^{N2} & \cdots & p_{Nu}^{N2} \end{bmatrix} \quad 7)$$

entonces T_{aux} se puede calcular mediante el siguiente algoritmo:

Condiciones iniciales:

$$T_{aux}(N2 \times Nu) = 0 \quad 8)$$

Algoritmo II:

$j = 1 \cdots Nu$

$T_{aux}(1, j) = P(1, j)$

Fin

$i = 2 \cdots N2$

$j = 1 \cdots i$

$T_{aux}(i, j) = T_{aux}(i-1, j) + P(i, j)$

Fin

Fin

Se extrae ahora de T_{aux} las $N2 - N1 + 1$ últimas filas para formar la matriz T:

$$T = T_{aux}(N1 : N2, 1 : Nu) \quad 9)$$

Con el objetivo de hacer más cómodo el proceso de optimización, resulta conveniente expresar en forma vector-matricial el modelo predictivo:

$$\hat{Y}(t) = T\Delta U(t) + S \quad 10)$$

donde

$$\hat{y}(t) = [\hat{y}(t+N1)\ \hat{y}(t+N1+1) \cdots \hat{y}(t+N2)]^T \quad 11)$$

$$s = \left[y(t) + s^{N1} y(t) + s^{N1} + s^{N1+1} \cdots y(t) + \sum_{i=N1}^{N2} s^i \right]^T \quad 12)$$

$$\Delta U(t) = [\Delta u(t)\ \Delta u(t+1) \cdots \Delta u(t+Nu-1)] \quad 13)$$

En ¡Error! No se encuentra el origen de la referencia.) s^k representa, para $k = N1, N1+1, \cdots, N2$, los términos que dependen de los datos disponibles en el tiempo t y pueden calcularse mediante la ecuación:

$$s^k = z^{-1} Q^k(z^{-1}) \Delta u(t) + z^{-1} F^k(z^{-1}) \Delta y(t) \quad 14)$$

Si se desea que las predicciones de la salida estén próximas a los valores de referencia deseados $r(t+j)$, puede plantearse el problema del control como el de obtener la secuencia de valores presentes y futuros de la variable manipulada $u(t)$, que minimizan a la siguiente función objetivo:

$$J = \sum_{j=N1}^{N2} [\hat{y}(t+j) - r(t+j)]^2 + \sum_{j=0}^{Nu-1} [\beta \Delta u(t+j)]^2 \quad 15)$$

En la que se penalizan, por una parte, las desviaciones de las predicciones de la salida $y(t+j)$ de sus valores deseados $r(t+j)$ y también los cambios de la variable manipulada $u(t+j)$ con un cierto peso β, con el que se puede conseguir un control suficientemente suave.

Con frecuencia los valores de $r(t+j)$ se definen como trayectorias que unen el valor actual de la variable $y(t)$ con un valor futuro deseado $w(t+N2)$, por ejemplo mediante un filtro exponencial de primer orden del tipo:

$$r(t+j) = \alpha r(t+j-1) + (1-\alpha) w(t+N2) \quad 16)$$

$$r(t) = y(t) \quad j = 1, \ldots N2$$

Si $w(t + N2)$ puede conocerse de antemano, es posible seguir una trayectoria planificada; en el caso más frecuente w es constante durante períodos de tiempo relativamente largos y $w(t + N2)$ se hace igual a $w(t)$. La selección del coeficiente α entonces se hace con el criterio de lograr cambios suaves en la salida hasta alcanzar el valor de la referencia; esto es particularmente útil cuando la referencia puede estar sometida a cambios súbitos.

En la solución del problema del control predictivo generalmente se plantea la condición de que la variable manipulada debe permanecer constante después de un cierto número Nu de períodos de control futuros. A Nu se le conoce como el "horizonte de control" y es menor o igual que $N2$. Esta condición permite en muchos casos reducir la dimensionalidad del problema de optimización de $N2-1$ a Nu y además, favorece que se obtengan soluciones estables. Así que:

$$\Delta u(t + j) = 0 \text{ para } j \geq Nu \qquad 17)$$

$N1$ y $N2$ son comúnmente conocidos como los horizontes mínimo y máximo de predicción. El valor de $N1$ por lo general se selecciona igual o ligeramente mayor que el número de períodos de retardo puro presente en el proceso a controlar, temiendo en cuenta que el control seleccionado para el tiempo t, sólo tendrá efecto después de transcurrido el tiempo de retardo puro y no tiene sentido entonces penalizar las desviaciones que no pueden corregirse por el control en el instante t.

Partiendo de la función objetivo cuadrática **¡Error! No se encuentra el origen de la referencia.** y de la expresión **¡Error! No se encuentra el origen de la referencia.**) se obtiene:

$$J_1 = \Delta U(t)^T \left(T^T T + \beta^2 I_{Nu}\right)\Delta U(t)$$

$$J_2 = 2e^T T\Delta U(t) - e^T e$$

$$J = J_1 - J_2 \qquad 18)$$

donde $e = R - S$ es el vector de errores y R los valores futuros de la referencia:

$$R^T = \left[r(t + N1) \;\; r(t + N1 + 1) \cdots r(t + N2)\right]^T \qquad 19)$$

Derivando e igualando a cero **¡Error! No se encuentra el origen de la referencia.**), con el fin de encontrar la secuencia óptima de control, se llega al resultado:

$$\Delta U(t) = \left(T^T T + \beta^2 I_{Nu}\right)^{-1} T^T e \qquad 20)$$

La ecuación matricial 20) ofrece Nu valores de los cambios a aplicar en la variable manipulada. De ellos solamente se aplica el primero, el correspondiente al instante t. En el siguiente período de control $t + 1$ se realizan de nuevo todos los cálculos, de acuerdo con la estrategia de horizonte móvil[5].

BLOQUES SCICOS

Un bloque *Scicos* puede tener dos tipos de entradas y dos tipos de salidas: entradas de activación, entradas normales, salidas de activación y salidas normales. Las entradas y salidas de activación se encuentran en la parte superior e inferior del bloque respectivamente (color rojo), mientras que, las entradas y salidas normales se ubican a los extremos (color negro) tal y como se ilustra en la figura 2.

Las entradas de activación controlan los bloques a través de eventos donde, la ocurrencia de un evento genera una salida calculada a partir de la entrada y el estado interno, para luego proceder a la actualización de dicho estado. Si el bloque se encuentra activo puede generar eventos que controlen otros bloques. Los bloques que no tengan entradas de activación se encuentran permanentemente activos[6].

Construcción de Nuevos Bloques Scicos

Un nuevo bloque puede ser construido como un Súper Bloque, mediante la interconexión de bloques simples, y compilándose posteriormente. La construcción de un nuevo Bloque Simple se basa esencialmente en dos funciones[6]:

- *Función Interfaz*: Determina la interface con el usuario. Es programada en código Scilab, pudiéndose utilizar funciones *Scilab* o desarrolladas por el usuario en ese lenguaje específico.
- *Función Computacional*: Utilizada para especificar el comportamiento dinámico del bloque. Puede ser programada en lenguajes como C, Fortran o el lenguaje propio de Scilab.

Nuevo Bloque *Scicos* (MPC)

El bloque MPC, como se muestra en la figura 3, cuenta con una entrada de activación, una salida normal por donde devuelve la variable de control y 3 entradas normales. Por el puerto de entrada 1 se suministra la señal utilizada en el proceso de identificación, por el segundo puerto se aplica la señal de referencia y por el puerto 3 se le pasa la variable controlada.

El bloque, primero se comporta como un identificador para encontrar mediante el Método de los Mínimos Cuadrados Recursivos[7] un modelo ARX aproximado, requerido para generar las matrices P, F, Q, E definidas en 5). Terminado el período de identificación comienza inmediatamente la acción del controlador.

Los parámetros de configuración para el método de identificación y el controlador se establecen a través de las ventanas de dialogo de las figura 4 y 5. A continuación se describen cada uno de ellos:

- **Períodos**: Número de períodos de muestreo que dura el proceso de identificación.
- **Orden**: Orden del modelo ARX.
- **Olvido**: Factor de olvido exponencial.
- **N1**: Horizonte mínimo de predicción.
- **N2**: Horizonte máximo de predicción.
- **Nu**: Horizonte de control
- **Peso**: Su función es penalizar la variable de control. Valores grandes del coeficiente de peso conllevan a respuestas más estables pero más lentas.
- **Filtro**: Coeficiente del filtro exponencial de primer orden, que admite valores entre 0 y 1.

RESULTADOS OBTENIDOS

Para evaluar en ambiente de simulación la efectividad del controlador PREDARX frente a procesos con grandes retardos de tiempo, se toma como objeto de estudio, el sistema discreto de segundo orden:

$$G_{(z)} = \frac{0{,}011(1+z)z^{-40}}{z^2 - 1{,}757z + 0{,}861}$$

El diagrama de bloque del sistema de control se muestra en la figura 6. El período de muestreo es de 1 segundo y la señal de referencia oscila en forma de una onda cuadrada entre los valores 0 y 10.

El comportamiento del sistema con el controlador PREDARX se muestra en la figura 7.

La gráfica revela el efecto en la salida de una secuencia binaria seudo-aleatorio[7], que se aplica durante los primeros 500 períodos de muestreo con el propósito de identificar un modelo ARX de segundo orden. Una vez culminada la identificación, con: $N1 = 42, N2 = 43, Nu = 3$ y $\beta = 4{,}5$, el controlador comienza a actuar sobre el proceso y rápidamente la salida alcanza la señal de referencia.

CONCLUSIONES

El desarrollo de nuevos bloques para *Scicos* resulta sencillo y permite crear nuevas contribuciones en función de las necesidades que tengan los usuarios.

Implementar estrategias de control en forma de bloques brinda a otros usuarios la posibilidad de utilizarlos sin necesidad de conocer detalles de la programación, centrándose solamente en alcanzar el objetivo fundamental que es lograr un control óptimo a partir de un adecuado ajuste de los parámetros del regulador.

El RCP basado en *Scicoslab*/*Scicos* y *Linux/RTAI* es una excelente alternativa para aquellos que no puedan adquirir en el mercado herramientas equivalentes de software propietario debido a sus elevados costos. Estas herramientas facilitan la implementación y simulación de técnicas de control con requerimientos de tiempo real de un modo simple y rápido.

Como se pudo comprobar en la simulación, el controlador predictivo PREDARX presenta un buen desempeño frente a procesos con retardos considerables y bajo condiciones de tiempo real duro.

Las experiencias expuestas en este trabajo pueden emplearse con fines docentes en cursos de pregrado o de posgrado y abren la posibilidad, en nuestro país, de desarrollar aplicaciones industriales de control avanzado utilizando herramientas de software libre.

REFERENCIAS

1. **BUCHER, R., DOZIO L., y MANTEGAZZA, P.**: "Rapid control prototyping with Scilab/Scicos and Linux/RTAI," 2002.
2. **CAMPBELL, S. L. V., CHANCELIER, J-P., y NIKOUKHAH, R.**: *"Modeling and Simulation in Scilab/Scicos with Scicoslab 4.4" Ed. Springer Sciences and Business Media Inc.*, 2009.
3. **BUCHER, R. y DOZIO, L.**: "Cacsd under rtai Linux with rtai-lab.", en *Fifth Real-Time Linux Workshop*, 2003.
4. **BUCHER, R. y BALEMI, S.**: *"Scilab/Scicos and Linux/RTAI - a unified approach."*, en *Proceeding of the 2005 IEEE Conference on Control Applications*, Canada, 2005.
5. **AGUADO, A.**: "Controladores predictivos basdos en modelos." en *Control Predictivo, IX Maestría de Informatización Industrial y Automatización*, Ciudad Habana, 2008
6. **DELICADO, B.**: "Introducción al modelado y simulación de sistemas físicos con la toolbox de Scicos.", *Tesis de Doctorado*, Universidad Nacional de Educación a Distancia, 2007.
7. **AGUADO, A. y MARTÍNEZ, A.**: *"Identificación y control adaptable", Ed. Pearson Education SA*, Madrid, España, 2002.

AUTORES

Alberto Aguado Behar, graduado de Ingeniero Electricista desde 1966. Obtuvo el grado de Doctor en Ciencias Técnicas en 1976. Es Investigador Titular y Profesor Titular. Trabaja en el Departamento de Control Automático del ICIMAF e investiga en temas de Identificación de Sistemas, Control Predictivo y Control Inteligente. Imparte cursos sobre estos temas para la Maestría en Automática de la CUJAE. Dirección centro de trabajo: Calle 15 No. 551 entre C y D. Teléfono centro de trabajo y Fax: 832-0319. Dirección e-mail: aguado@icmf.inf.cu.

Erik Marichal Arbona, graduado de ingeniero en Automática en la Universidad Central de Las Villas desde el 2004. Actualmente es profesor Instructor del departamento de Sistemas Digitales de la Facultad 5 de la Universidad de las Ciencias Informáticas. Teléfono centro de trabajo: 835-8942 Dirección e-mail: emarichal@uci.cu.

Ronald Ugas Lago, graduado de ingeniero en Automática en la sede: "Julio A. Mella" de la Universidad de Oriente. Actualmente es profesor Asistente del departamento de Sistemas Digitales de la Facultad 7 de la Universidad de las Ciencias Informáticas. Dirección particular: c/ Hermanos Marín # 109 e/ Riuz Rivera y Prolongación de Corona, Rpto. Veguita de Galo, Santiago de Cuba. Teléfono centro de trabajo: 835-8159 o 837-2727. Dirección e-mail: rugas@uci.cu.

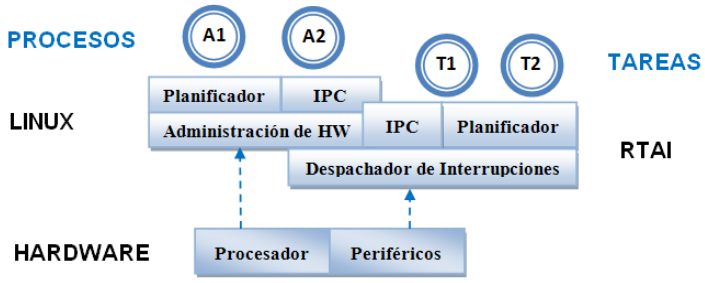

Figura 1. Arquitectura básica de RTAI.

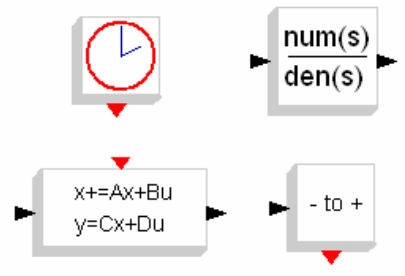

Figura 2. Ejemplos de Bloques Scicos.

Figura 3. Nuevo bloque MPC.

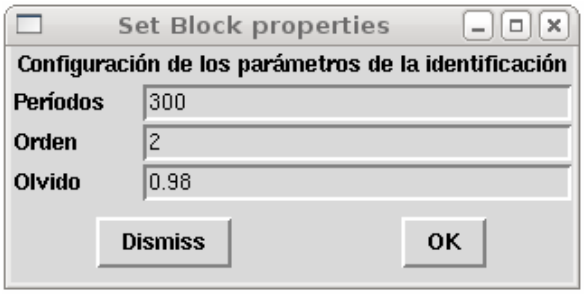

Figura 4. Parámetros para la identificación.

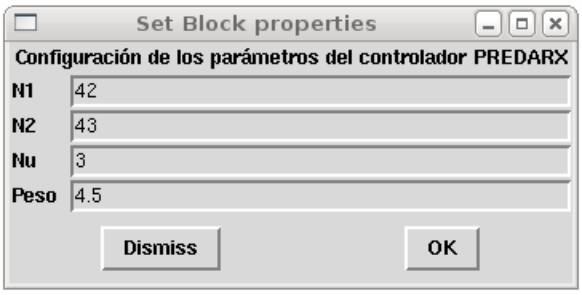

Figura 5. Parámetros del controlador PREDARX.

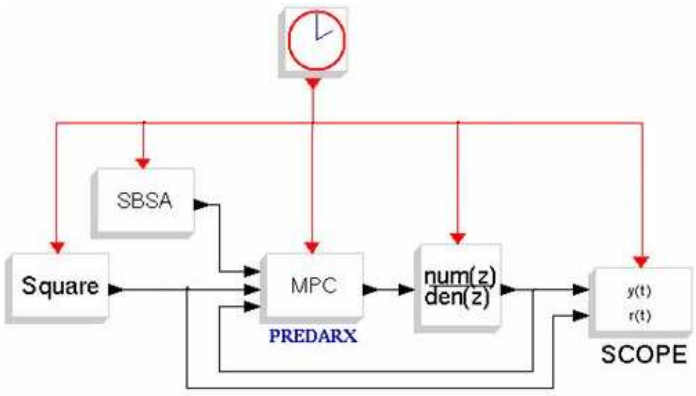

Figura 6. Lazo de control con un regulador PREDARX.

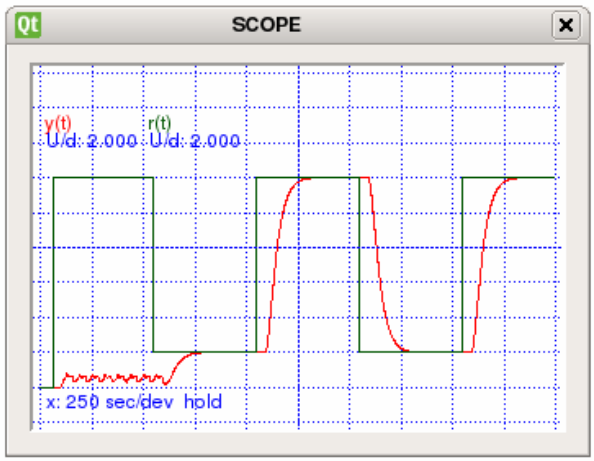

Figura 7. Comportamiento del sistema de control en tiempo real.