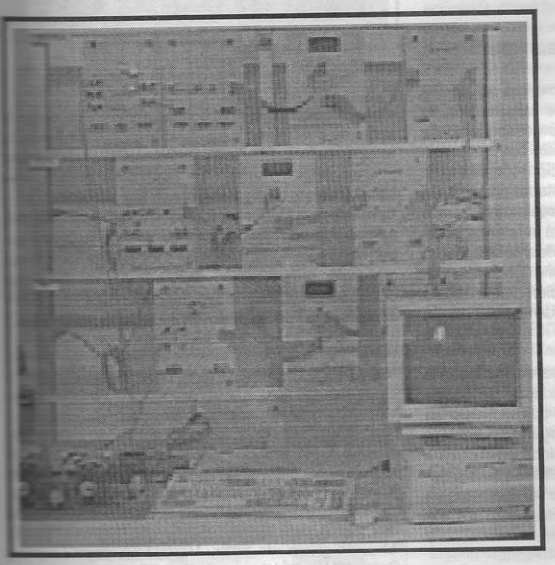
Implementación del FMRLC en REALMATLAB

Control inteligente adaptativo en tiempo real



Marisol Osorio, Esp.
Profesora Asociada,
Universidad Pontificia
Bolivariana Medellín. Colombia.

Diego Marín Arango, Ing.
Ingeniero de Sofware I+D, Celsa
S.A. Medellín. Colombia.

Grupo A+D Automática y Diseño -UPB
Escuela de Ingenierías.
Facultad de Ingeniería Eléctrica y Electrónica.
Universidad Pontificia Bolivariana. Medellín.
Colombia.
a+d@upb.edu.co

Recibido : Mayo 2.002
Aceptado: Junio 2.002

RESUMEN

En este artículo se describe la herramienta REALMATLAB, creada por el grupo A+D de la Escuela de Ingenierías de la Universidad Pontificia Bolivariana así como su utilización en aplicaciones de control inteligente.

PALABRAS CLAVES

Control, difuso, sentido común MATLAB.

ABSTRACT

This article describes REAL-MATLAB, a tool developed by A+D group from the Engineering School of the Pontificia Bolivariana University. Its usafolness for intelligent control applications use full is also presented.

KEYWORDS

Fuzzy, control, common-sense, MATLAB.

1. INTRODUCCIÓN

La necesidad de validar las técnicas de análisis y diseño de sistemas de control mediante la experimentación es evidente. Existe en este momento una tendencia creciente a virtualizar la experimentación, reemplazando por simulaciones la experiencia

directa.Si bien ésta es una solución que reduce costos y extiende el cubrimiento potencial del laboratorio a más personas, no siempre es conveniente alejar al estudiante de la experiencia real. Puede optarse entonces por una solución intermedia, construyendo sistemas que estén destinados específicamente a la experiencia de aprendizaje de los conceptos claves del control y que se usen paralelamente con los simuladores.
Los trabajos llevados a cabo por el Grupo A+D, Automática y Diseño, generaron una herramienta llamado REALMATLAB [9.] para su manejo que permite la comprobación rápida y confiable de nuevos algoritmos de control. Además, facilita la realización de las prácticas de laboratorio y permite la realizacion de prototipos de sistemas de control de manera rápida. En este artículo se explorará la utilización de esta herramienta para control inteligente adaptativo en tiempo real comprobando el algoritmo [3.][4.][5.][7.] [8.].

2. CONFIGURACIÓN DEL SISTEMA

A grandes rasgos, los principales elementos del sistema son: Interfaz de usuario, *toolbox* de construcción de prototipos de sistemas y de post-procesamiento de señal desarrollados bajo Matlab y enlace con un sistema de adquisición de datos y temporización, construido en *ActiveX*. En la Figura 1 se observa la interfaz principal. En el primer *frame*, en la sección inferior izquierda, se puede definir el tipo de señal que constituirá la referencia del sistema. Es posible seleccionar como fuente de señal un potenciómetro (*hardware*) o una señal generada por *software*, que puede ser de tipo función matemática o arbitraria. Esta selección se realiza por medio del menú que aparece en el *frame*.

En el *frame* de la sección inferior central se encuentra la selección de la configuración del control. Se puede controlar la planta en lazo abierto o cerrado, usar un control predeterminado o generar alguno de manera personal. El sistema trae ya predefinidos los controles clásicos P, PI, PD, PID, Adelanto, Atraso y Adelanto-Atraso. El período de muestreo puede ser cambiado en tiempo real y se pueden archivar diferentes secciones de tiempo de las señales de control, referencia y retroalimentación de un ensayo determinado.

3. FUNCIONAMIENTO GENERAL DEL SISTEMA

El REALMATLAB funciona en el entorno MATLAB. Para utilizarlo, debe abrirse el Editor de Comandos del MATLAB y debe escribirse en el *prompt* >> realmatlab esto ocasionará la aparición de la interfaz principal (Figura 1) y la ventana que corresponde a los parámetros del control P.
Las señales de control aparecerán graficadas en tiempo real en la figura central (que es

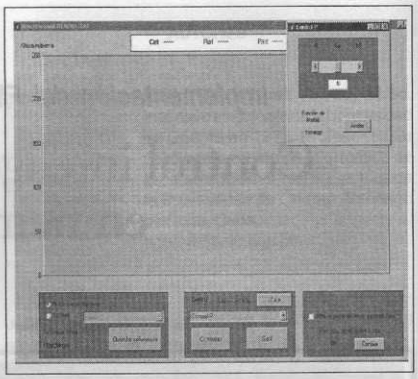


Figura 1. *Interfaz principal de REALMATLAB.*

manifestación visible del control *ActiveX*), en negro la señal de control, en azul la referencia y en verde, la retroalimentación. El usuario puede almacenar cualquier intervalo de señales que considere conveniente seleccionando el cuadro "Archivar Secuencias".
Es bueno resaltar que el protocolo ActiveX tuvo una gran aplicabilidad en la programación de este sistema pues permitió implementar todas las opciones de tiempo real: graficación, temporización y lectura de puertos. El enfoque del presente artículo no permite profundizar sobre este aspecto pero puede verse más información en [1] [5] [9.].

4. APLICACIONES AL CONTROL INTELIGENTE

Usuarios avanzados del sistema pueden crear sus propias señales y sistemas de control siguiendo los pasos sencillamente ilustrados por los creadores del programa en el texto Ambiente Interactivo para Trabajo Experimental en Tiempo Real usando Matlab[1]. Esta característica ha sido usada para apoyar el proyecto de investigación "Mejoras al método *Fuzzy Model Reference Learning Control* FMRLC" del grupo A+D, Automática y Diseño, en las fases Interpolador y Extrapolador para verificar en tiempo real las simulaciones realizadas. Para ello se generó un controlador difuso adaptativo con el método FMRLC [2] mejorado con características de Intrapolación y Extrapolación [3],[4] con estructura de interfaz gráfica y función de *callback*. Con la misma metodología se trabaja actualmente para verificar la influencia del modelo de referencia y otros métodos de adaptación de base de reglas en controladores *fuzzy*. A continuación se realizará una breve explicación introductoria del FMRLC, con el fin de exponer las mejoras que fueron implementadas sobre

directa. Si bien ésta es una solución que reduce costos y extiende el cubrimiento potencial del laboratorio a más personas, no siempre es conveniente alejar al estudiante de la experiencia real. Puede optarse entonces por una solución intermedia, construyendo sistemas que estén destinados específicamente a la experiencia de aprendizaje de los conceptos claves del control y que se usen paralelamente con los simuladores.
Los trabajos llevados a cabo por el Grupo A+D, Automática y Diseño, generaron una herramienta llamado REALMATLAB [9.] para su manejo que permite la comprobación rápida y confiable de nuevos algoritmos de control. Además, facilita la realización de las prácticas de laboratorio y permite la realizacion de prototipos de sistemas de control de manera rápida. En este artículo se explorará la utilización de esta herramienta para control inteligente adaptativo en tiempo real comprobando el algoritmo [3.][4.][5.][7.] [8.].

2. CONFIGURACIÓN DEL SISTEMA

A grandes rasgos, los principales elementos del sistema son: Interfaz de usuario, *toolbox* de construcción de prototipos de sistemas y de post-procesamiento de señal desarrollados bajo Matlab y enlace con un sistema de adquisición de datos y temporización, construido en *ActiveX*. En la Figura 1 se observa la interfaz principal. En el primer *frame*, en la sección inferior izquierda, se puede definir el tipo de señal que constituirá la referencia del sistema. Es posible seleccionar como fuente de señal un potenciómetro (*hardware*) o una señal generada por *software*, que puede ser de tipo función matemática o arbitraria. Esta selección se realiza por medio del menú que aparece en el *frame*.

En el *frame* de la sección inferior central se encuentra la selección de la configuración del control. Se puede controlar la planta en lazo abierto o cerrado, usar un control predeterminado o generar alguno de manera personal. El sistema trae ya predefinidos los controles clásicos P, PI, PD, PID, Adelanto, Atraso y Adelanto-Atraso. El período de muestreo puede ser cambiado en tiempo real y se pueden archivar diferentes secciones de tiempo de las señales de control, referencia y retroalimentación de un ensayo determinado.

3. FUNCIONAMIENTO GENERAL DEL SISTEMA

El REALMATLAB funciona en el entorno MATLAB. Para utilizarlo, debe abrirse el Editor de Comandos del MATLAB y debe escribirse en el *prompt* >> realmatlab esto ocasionará la aparición de la interfaz principal (Figura 1) y la ventana que corresponde a los parámetros del control P.
Las señales de control aparecerán graficadas en tiempo real en la figura central (que es

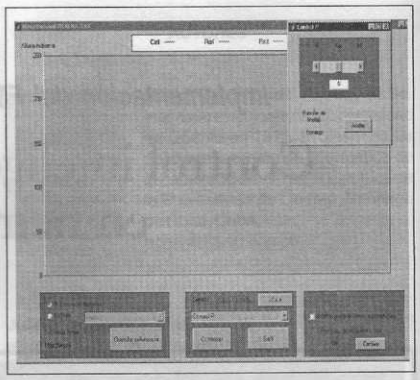


Figura 1. *Interfaz principal de REALMATLAB.*

manifestación visible del control *ActiveX*), en negro la señal de control, en azul la referencia y en verde, la retroalimentación. El usuario puede almacenar cualquier intervalo de señales que considere conveniente seleccionando el cuadro "Archivar Secuencias".
Es bueno resaltar que el protocolo ActiveX tuvo una gran aplicabilidad en la programación de este sistema pues permitió implementar todas las opciones de tiempo real: graficación, temporización y lectura de puertos. El enfoque del presente artículo no permite profundizar sobre este aspecto pero puede verse más información en [1] [5] [9.].

4. APLICACIONES AL CONTROL INTELIGENTE

Usuarios avanzados del sistema pueden crear sus propias señales y sistemas de control siguiendo los pasos sencillamente ilustrados por los creadores del programa en el texto Ambiente Interactivo para Trabajo Experimental en Tiempo Real usando Matlab[1]. Esta característica ha sido usada para apoyar el proyecto de investigación "Mejoras al método *Fuzzy Model Reference Learning Control*/FMRLC" del grupo A+D, Automática y Diseño, en las fases Interpolador y Extrapolador para verificar en tiempo real las simulaciones realizadas. Para ello se generó un controlador difuso adaptativo con el método FMRLC [2] mejorado con características de Intrapolación y Extrapolación [3],[4] con estructura de interfaz gráfica y función de *callback*. Con la misma metodología se trabaja actualmente para verificar la influencia del modelo de referencia y otros métodos de adaptación de base de reglas en controladores *fuzzy*. A continuación se realizará una breve explicación introductoria del FMRLC, con el fin de exponer las mejoras que fueron implementadas sobre

el REALMATLAB y los experimentos realizados, así como las conclusiones obtenidas.

4.1 FMRLC convencional y mejora implementada sobre REALMATLAB

Para una somera explicación sobre el FMRLC, considérese que el control es ejecutado sobre la planta SISO (*Single input single output*) de segundo orden definida en la primera parte de este artículo. Se usan dos variables: error y cambio en error (e y de, respectivamente) y se asigna a cada una 11 funciones de membresía. En la Figura 2 puede observarse esta matriz de reglas.

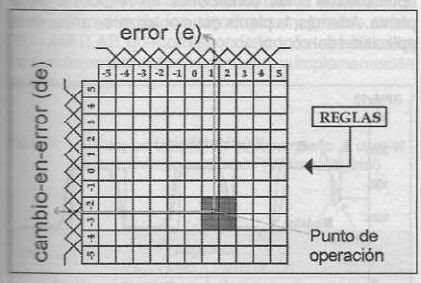


Figura 2. *Representación de las reglas en un control difuso de dos entradas y una salida.*

El Mecanismo de Aprendizaje, cuyo esquema se muestra en la Figura 3, consiste en otro sistema difuso que tiene como entradas el error de referencia Y_e y el cambio en error de referencia Y_c, y como salida, la magnitud del ajuste que debe realizarse a las reglas de las zonas en las cuales se está ejecutando la acción de control.

Las reglas del mecanismo de aprendizaje se fundamentan en el criterio de que si la salida de la planta se ajusta a lo previsto en el modelo de referencia, se deben dejar las reglas del controlador tal como están pero si no se deben modificar de tal manera que el accionar del controlador tienda a modificar la salida de la planta para que se parezca al modelo de referencia.

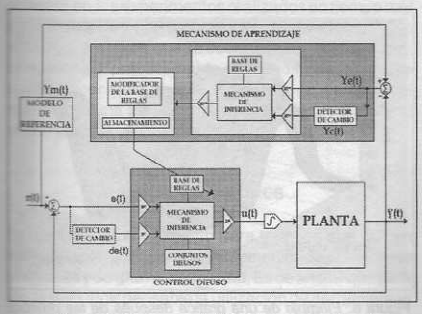


Figura 3. *Esquema del FMRLC convencional.*

La primera mejora consistió en aprovechar el conocimiento adquirido en cada instante que normalmente se usaría únicamente para adaptar las reglas de la zona de operación activa para modificar también las reglas del controlador en zonas vecinas. Esto se realizó mediante la extrapolación aunque parcial porque no modifica todas las reglas a partir de una extrapolación de casos particulares sino que modifica una vecindad.

Así se minimiza la posibilidad de sobregeneralizar; ésto es lo que llamamos ExFMRLC, "*Fuzzy controller extrapolating adaptation methodology*"[3]. La segunda consistió en interpolar el valor de cada una de las reglas activadas considerando el valor de las reglas vecinas según su validez o certeza. La interpolación se realiza después del mecanismo de inferencia y antes de la interfaz de concreción. Esto se hace para cada una de las reglas activadas de forma tal que el valor final después de la interfaz de concreción es mejor que el obtenido con el FMRLC original, procedimiento denominado IFMRLC "Interpolador para Control Difuso Adaptativo" [4], [7] y [8].

4.2 Implementación del FMRLC en Realmatlab

El código básico del algoritmo del FMRLC se encuentra disponible en la referencia [4]. El programa se Implementó inicialmente como simulación y luego se usó el REALMATLAB [1] y se ejecutó en un computador personal tipo PENTIUM de 350Mhz con sistema operativo Windows 95. Dado que este control fue uno de los primeros implementados sobre el REALMATLAB fue necesario hacer experimentos iniciales con controles difusos sin adaptación con el fin de calibrar las constantes de escalamiento para sintonizar las funciones de membresía.

En las simulaciones previas la planta considerada era de tercer orden con un polo en cero (tipo1), aunque era no lineal. Por ello, bastaba con utilizar un control PD. En la planta real, dado que era de segundo orden, fue necesario implementar además de la acción proporcional y la derivativa, una acción integral con el fin de asegurar error de estado estable cero. Esto incrementó la complejidad del control difuso en tiempo real. Después de implementado el control difuso sin adaptación pero con estrategia PID y estimadas las constantes de escalamiento se procedió a implementar la parte destinada específicamente a la adaptación. En el cumplimiento de este objetivo se pudo apreciar la ventaja de la herramienta pues las simulaciones estaban programadas en Matlab, lo que permitió utilizar directamente y sin cambios el algoritmo ya escrito. Lo único que fue necesario hacer fue identificar dentro del

programa las variables que debían permanecer sin cambios entre ejecuciones y aquellas que debían responder al mecanismo de adaptación ajustándose al aprendizaje obtenido de la planta.

Fue necesario implementar modificaciones al programa ejecutor principal de REALMATLAB (enable.m) con el fin de acoger los tipos de variables específicas que requiere un control difuso adaptativo, entre ellas el error, su cambio, las matrices que contienen los pesos de las reglas y aquellas que almacenan los valores de certeza en el caso del interpolador. Para ello, se definió dentro del programa un arreglo que contuviera todo tipo de variables: escalares, matrices y vectores. Originalmente, las únicas variables almacenadas durante los ciclos de control eran las que contenían la referencia, el valor de la retroalimentación, el de la señal de control y el error acumulado.

Esto le da mucha más flexibilidad al programa y permite implementar no sólo el esquema en cuestión sino otros tipos de algoritmos de control.

Dentro de las mejoras que este trabajo introdujo a la versión inicial del REALMATLAB se encuentra también la generación de un archivo que contiene las condiciones iniciales del control entre ellas, el caso supuesto inicial de la matriz de reglas.

La descripción exhaustiva de las modificaciones realizadas al programa principal y la manera en que éstas pueden ser utilizadas para implementar nuevos controladores puede ser vista en [4].El desarrollo del control difuso adaptativo también requirió de una interfaz gráfica que se muestra en la Figura 4. En ella se puede apreciar que es posible seleccionar el tipo de condición inicial para la superficie de control, el tipo de control difuso a implementar: FMRLC original, con interpolador o con extrapolador así como la señal de referencia entre las que proporciona el REALMATLAB original y otras definidas específicamente para las simulaciones iniciales, con el fin de estudiar comparativamente con ellas el comportamiento del control en tiempo real.

Para hacer más fáciles las comparaciones se estructuraron los datos en la misma forma en que los arrojaban las simulaciones iniciales con lo que se pudo contrastar uno a uno, como se puede observar en la Figura 5. Se compararon las salidas del sistema, los criterios de error y el estado final de adaptación estable del sistema entre lo obtenido en las simulaciones previas y lo apreciado sobre el control en tiempo real.

Pudo apreciarse que el desempeño del control sigue siendo bueno cuando se implementa en tiempo real siempre que se tenga en cuenta preservar un período de muestreo mayor o igual a 50 ms. con el fin de asegurar que el algoritmo de control tenga tiempo de ejecutarse. Es de anotar que este período de muestreo se ajusta óptimamente a las condiciones de respuesta de la planta. Además, la planta debe polarizarse antes de la aplicación del control.

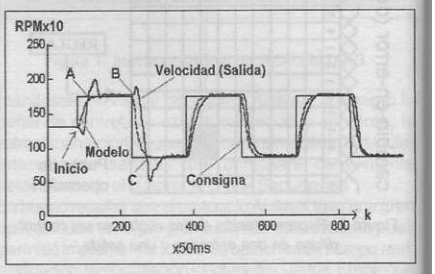


Figura 5. *Gráfica obtenida de las pruebas en Tiempo Real, Usando el REALMATLAB*

Es importante anotar que llevar la planta a su punto de trabajo requiere la aplicación de un control PID clásico durante 5 segundos, tiempo estimado con base en el t conocido de estudios anteriores sobre la planta [9], que está alrededor de 1 segundo. Después de la polarización de la planta es posible dejar todo el control en manos del control difuso adaptativo aún sin aprendizaje previo.

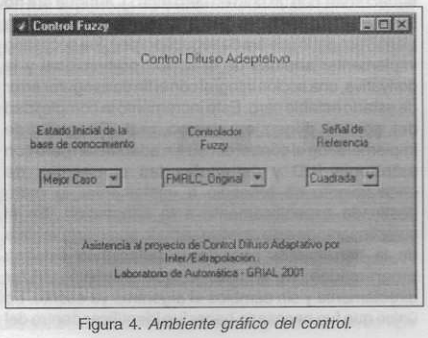


Figura 4. *Ambiente gráfico del control.*

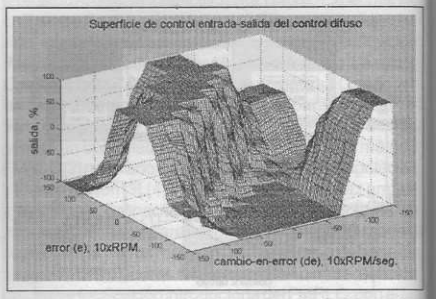


Figura 6. *Ejemplo de una gráfica después de las pruebas en Tiempo Real*

En la Figura 5 se muestran las gráficas temporales de aprendizaje obtenidas. Para su análisis detallado, puede verse [8]. En las gráficas puede apreciarse un comportamiento que se ajusta a las predicciones que se habían obtenido previamente en la simulación.

La Figura 6 representa la superficie de control generada por la base de conocimiento inicializada con el PeorSupuesto (El peor de los casos en el que se le dan órdenes contradictorias con la realidad al sistema). Se presenta aquí para que sirva como base de comparación con las superficies de control obtenidas después de efectuado el aprendizaje al realizar los experimentos.
Como referencia se presentan en la Tabla 1 los resultados de la utilización de los algoritmos mejorados sobre REALMATLAB, que corresponden con los obtenidos en las simulaciones realizadas antes de la implementación en tiempo real.

Tabla 1. *Mejora porcentual en el desempeño al usar el nuevo método propuesto (Versus Original)*

Inicialización	Tipo de Algoritmo	Mejora Promedio	
		Simulación	Tiempo Real
1. Bien Supuesto	FMRLC	0%	0%
	IFMRLC	9%	9%
	ExFMRLC	60%	35%
2. Sin Supuesto	FMRLC	0%	0%
	IFMRLC	13%	5%
	ExFMRLC	65%	27%
3. Peor Supuesto	FMRLC	0%	0%
	IFMRLC	88%	28%
	ExFMRLC	99%	83%

En la columna de "Mejora Promedio" puede observarse que en general existe coherencia entre la mejora promedio obtenida en los experimentos realizados en tiempo real y los resultados obtenidos de las simulaciones. Sin embargo, la mejora que se obtiene en las simulaciones es mayor en prácticamente todos los casos, lo que puede hacer pensar que el estudio en tiempo real muestra de una manera más realista lo que sucede al utilizar el algoritmo mejorado.

5. CONCLUSIONES

Es posible utilizar el Matlab para trabajos que no son exclusivamente de simulación, obteniéndose sistemas amigables y de gran aplicabilidad. Es de anotar que para ello resulta indispensable la utilización de herramientas complementarias como el protocolo *ActiveX* en este caso. En este trabajo se aprecia la posibilidad que abre el REALMATLAB de hacer ensayos en tiempo real para cualquier control que haya sido simulado previamente en Matlab pues basta con tomar el algoritmo sin mayores cambios y hacer adaptaciones menores como eliminar las características cíclicas de la simulación y dejar las sucesivas aplicaciones de la acción controladora en manos del control *ActiveX* que regula la ejecución del período de muestreo.
Una ventaja importante observada cuando se ejecuta el control en tiempo real usando REALMATLAB es que los datos obtenidos de las sucesivas ejecuciones son almacenados en memoria donde quedan disponibles para usar luego las grandes capacidades de graficación, análisis estadístico e identificación de sistemas de Matlab sin siquiera tener que hacer cambio de formato alguno. Por medio de esta metodología de utilización de estrategias de tiempo real es posible encontrar una predicción más ajustada del comportamiento de los sistemas de control prototipo en plantas reales.

Agradecimientos

Durante la realización del trabajo presentado se pusieron de manifiesto las grandes ventajas del trabajo conjunto entre pregrado y posgrado pues, gracias a la colaboración del ingeniero (en ese momento estudiante en ejecución de su trabajo de grado de pregrado) Carlos González y del estudiante Luis Eduardo Tobón, fue posible darle al sistema las características de tiempo real de que goza en el momento presente y comprobar su capacidad de extensión a todo tipo de estrategias de control que puedan ser implementadas en Matlab.

6. REFERENCIAS BIBLIOGRAFÍCAS

[1] OSORIO Marisol. Ambiente Interactivo para Trabajo Experimental en Tiempo Real usando Matlab. Tesis. Medellín 2001.
[2] PASSINO Kevin. Fuzzy Control Systems. Ponencias del III Congreso de la Asociación Colombiana de Automática. 1998
[3] BETANCUR Manuel – Ortiz, Alex. *Fuzzy Controller Extrapolation Adaptive Method.* Ponencias del IX Congreso Latinoamericano de Automática. 2000.
[4] PEDRAZA Marta – Marín Diego – Betancur Manuel. Control Difuso Adaptativo por Inter/Extrapolación. Tesis. Medellín 2001
[5] OSORIO Marisol – González Carlos. Ambiente interactivo para trabajo experimental en tiempo real usando Matlab. Editorial UPB. Medellín. 2001.
[6] EDDON Guy – Eddon Henry. ActiveX visual basic 50. Microsoft Press – Mc Graw-Hill/Interamericana de España. 1997

[7] PEDRAZA Marta – Marín Diego – Betancur Manuel. Interpolador para Control Difuso Adaptativo IFMRLC. Congreso Internacional de Inteligencia Computacional (CIIC). Colombia. 2001.

[8] PEDRAZA Marta – Marín Diego – Betancur Manuel. Sentido Común para Controladores Difusos Extrapolando el FMRLC, Revista Electrónica y Computación Edición N° 17. Colombia 2002. 68p-77p.

[9] OSORIO Marisol–Tobón Luis Andrés REALMATLAB. En Memorias TAEE 2002. V Congreso de Tecnologías aplicadas a la Enseñanza de la Electrónica. Las Palmas de la Gran Canaria. España. 13-15 de febrero de 2002

AUTORES

Marisol Osorio Cárdenas. Ingeniera Electrónica, 1992, UPB. Especialista en Automática UPB 2001, y estudios de Maestría en Ingeniería, 2001, UPB. Profesora Asociada y Directora de Ingeniería Eléctrica y Electrónica UPB 2002. Coordinadora del Grupo de Investigación en Automática hasta julio de 2002. Universidad Pontificia Bolivariana. Teléfono 415 9020, ext. 9560, Medellín, Colombia.
mosorio@logos.upb.edu.co

Diego Ignacio Marín Arango. Ingeniero Electrónico, 2001, UPB. Ingeniero de Software en I+D de CELSA S.A.

«Los años enseñan muchas cosas que los días desconocen».
RALF W. EMERSON