Simulación de procesos con controladores lógico-programables (PLC's)

M. García, M. Llorca, J. Oro, D. Solà y E. Barberà*
IQS School of Engineering, Universitat Ramon Llull
Via Augusta 390, 08017-Barcelona, Spain

Process simulation with Programmable Logic Controllers (PLC' s)

Simulació de processos amb controladors lògics programables (PLC)

Recibido: 2 de enero de 2014; revisado: 12 de mayo de 2014; aceptado: 13 de mayo de 2014

RESUMEN

La realización de prácticas en el laboratorio de control de procesos presenta como mayor dificultad el coste asociado al funcionamiento del proceso, debido al consumo de reactivos, de energía o a la generación de residuos. Por tanto se propone trabajar con un proceso simulado en un controlador lógico- programable o PLC, con lo que se realiza una práctica de bajo coste y más cercana a la realidad que la pura simulación numérica.

Palabras clave: Control de procesos, simulación, controladores lógico programables

SUMMARY

The laboratory of process control can be very expensive, the high cost is associated to the process operation, reagents, energy or waste generation. Therefore, hybrid simulation can be a very atractive option. In this work, process simulation with PLC is proposed as a low cost alternative closer to reality than pure numerical simulation.

Key words: Process control, simulation, PLC

RESUM

Les pràctiques de control de processos presenten un elevat cost associat amb l'operació del procés, a causa del consum de reactius, energia o per la generació de residus. Per tant, la simulació híbrida pot ser una opció molt atractiva. En aquest treball es proposa la simulació de processos amb controladors lògics programables o PLC com pràctica de baix cost i mès propera a la realitat que la pura simulació numèrica.

Paraules clau: Control de processos, simulació, controladors lògics programables

1.- INTRODUCCIÓN

El dominio de las técnicas de control de procesos requiere la realización de prácticas que permitan al operador adquirir la habilidad suficiente. Dado el elevado coste de las plantas experimentales se tiende a la ejecución de prácticas por simulación por ordenador, siendo Simulink® de Matlab® uno de los programas más populares.
Aunque las prácticas de control de procesos realizadas por simulación son muy útiles, adolecen del defecto de la no manipulación de los elementos reales de control ni de las diferentes señales que intervienen. Tampoco permiten probar el comportamiento de nuevos controladores.
Estos inconvenientes se pueden resolver mediante la simulación del proceso mediante sistemas electrónicos (1 y 2) construido con amplificadores operacionales, lo que permite disponer de procesos con la función de transferencia del orden deseado que deben ser conectados al controlador mediante las señales eléctricas correspondientes a la variable controlada (salida del proceso) y a la señal reguladora (entrada al proceso).
Únicamente existe la imposibilidad de construir un proceso de tiempo muerto mediante elementos electrónicos analógicos, lo que impide la modelización de muchos procesos de control reales.
La simulación de un proceso de tiempo muerto mediante técnicas digitales no tiene una gran dificultad, basta generar una cola de espera en la que la entrada actual se almacena durante un tiempo igual al tiempo muerto.
En el presente trabajo se propone la utilización de un controlador lógico-programable o PLC para implementar la simulación del proceso. Ésta se realizará mediante bloques de primer o de segundo orden subamortiguado y uno o más bloques de tiempo muerto. La simulación se deberá ejecutar periódicamente con un tiempo de ejecución suficientemente pequeño para que el comportamiento sea correcto, aproximándose suficientemente al funcionamiento del proceso continuo simulado.
La señal reguladora, que viene del controlador, se introducirá al PLC simulador mediante una entrada analógica y la variable controlada se sacará del PLC a través de una salida analógica (Figura 1).

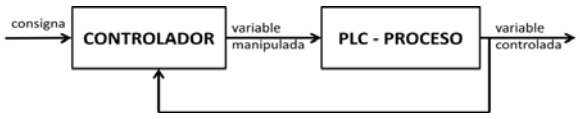

Figura. 1.- *Diagrama de bloques*

El controlador será exterior al PLC simulador y podrá ser otro PLC de igual o diferente modelo y fabricante, un ordenador actuando como controlador y la interfase adecuada o bien un equipo controlador, ya sea digital o analógico.
En consecuencia, el objetivo de este trabajo es la programación de un PLC S-300 de Siemens® o de un PLC Premium de Schneider Electric® para que se comporten como procesos con una función de transferencia compuesta por unidades de primer o segundo orden subamortiguado y de tiempo muerto.

2.- MATERIALES Y MÉTODOS

El hardware y software utilizado es el siguiente:
Siemens®: CPU compacta 314C-2DP con entradas y salidas analógicas de 12 bits. Step 7® Professional 2006 (3).
Schneider Electric®: PLC Modicon Premium TSXPSY2600 (fuente de alimentación), TSXP572634 (CPU), TSXAEY414 (entradas analógicas de 16 bits) y TSXASY410 (salidas analógicas de 11 bits + signo). UnityPro® v2.2 (3).
Tanto la entrada (variable manipulada) como la salida (variable controlada) se han considerado diferencias de potencial (voltios).

3.- PLC S-300 SIEMENS®

3.1.- Primer orden

Un proceso de primer orden viene caracterizado por la siguiente función de transferencia en s normalizada a ganancia unidad:

$$\frac{C(s)}{VM(s)} = \frac{1}{\tau s + 1}$$

Ecuación 1

donde *C(s)* y *VM(s)* son las variables controlada (salida) y manipulada (entrada) respectivamente y τ, la constante de tiempo.
Esta ecuación puede transformarse en la correspondiente función discreta (en z) mediante el conocido procedimiento de Tustin, obteniendo:

$$\frac{C(z)}{VM(z)} = \frac{T + Tz^{-1}}{T + 2\tau + (T - 2\tau)z^{-1}}$$

Ecuación 2

donde T es el periodo de simulación.

Su implementación recursiva es:

$$c(k) = \frac{T \cdot [vm(k) + vm(k-1)] - (T - 2\tau) \cdot c(k-1)}{T + 2\tau}$$

Ecuación 3

donde k indica el instante actual

El programa, realizado en lenguaje de lista de instrucciones (AWL) de Step7® (3), se estructura en un bloque de función para poder disponer de un bloque de datos asociado que permita su utilización repetida, para poder simular procesos de orden superior.
La tabla de variables asociada a este bloque de función es:

Tipo	identificador		Comentario	
IN	r	real	entrada,	vm(k)
	periodo	real	periodo de simulación,	T en s
	tau	real	constante de tiempo,	τ en s
OUT	c	real	salida,	c(k)
STAT	cp	real	salida anterior,	c(k-1)
	rp	real	entrada anterior,	vm(k-1)

FB1

- segmento 1

```
L     #r     // entrada del proceso, vm(k)
L     #rp    // entrada anterior, vm(k-1)
+R
L     #periodo  // T
*R            // T[vm(k)+vm(k-1)]
T     #c
L     #tau
L     #tau
+R            // 2τ
L     #periodo
TAK           // intercambio de ACU1 y ACU2
-R            // (T-2τ)
L     #cp   // salida anterior, c(k-1)
*R                          // (T-2τ)c(k-1)
L     #c
TAK
-R            // T[vm(k)+vm(k-1)]-(T-2τ)c(k-1)
T     #c
L     #tau
L     #tau
+R
L     #periodo
+R            // (T+2τ)
L     #c
TAK
/R            // [T[vm(k)+vm(k-1)]-(T-2τ)c(k-1)]/(T+2τ)
T     #c
```

- segmento 2

```
L     #r    // entrada actual, vm(k)
T     #rp   // guardar entr. actual en entr. previa, vm(k-1)
L     #c    // salida actual, c(k)
T     #cp   // guardar sal. actual en sal. previa, c(k-1)
```

Se ha utilizado la propia variable de salida, c, para ir almacenando los resultados intermedios.
Como la ejecución ha de ser de forma periódica con un control ajustado del tiempo, es preferible utilizar un bloque de organización de ejecución periódica como el OB35, fijando su periodo de ejecución con un valor igual al periodo de simulación.
Como ejemplo de aplicación se ha considerado la simulación de un proceso de segundo orden con amortiguamiento crítico asociando dos bloques de primer orden a dos bloques de datos, DB1 y DB4, en éstos se almacenan los parámetros y datos de entrada y de salida actuales y previos.
El bloque de organización **OB35** queda:

```
L     #OB35_EXC_FREQ
ITD               // paso a entero doble
DTR               // paso a real
L     1000.0
/R                // paso a segundos
T     MD  10
CALL FB  1,DB1
      r     :=MD14
      periodo:=MD10
      tau   :=5.000000e+000
      c     :=MD22
CALL FB  1,DB4
      r     :=MD22
      periodo:=MD10
      tau   :=5.000000e+000
      c     :=MD18
```

El periodo de ejecución del bloque OB35 se ha fijado en 100 ms, este valor está almacenado como entero sencillo en la variable OB35_EXC_FREQ, las primeras instrucciones permiten pasar este valor a real en segundos (MD10).
En el programa principal, OB1, se sitúan las instrucciones necesarias para adquirir la entrada del proceso a partir de una entrada analógica, que, después de convertida a número real, se guarda en la marca MD14 y emitir la salida, marca MD18, a través de una salida analógica, después de ser convertida adecuadamente.

3.2.- Segundo orden subamortiguado

La función de transferencia de un proceso de segundo orden subamortiguado con ganancia unidad (Ecuación 4) conduce, mediante la aplicación del método de Tustin, a su forma discreta (Ecuación 5).

$$\frac{C(s)}{VM(s)} = \frac{\omega_o^2}{s^2 + 2\xi\omega_o s + \omega_o^2}$$

Ecuación 4

donde ω_o es la frecuencia de resonancia y ξ, el coeficiente de amortiguamiento.

$$\frac{C(z)}{VM(z)} = \frac{\omega_o^2 T^2(1 + 2z^{-1} + z^{-2})}{(\omega_o^2 T^2 + 4\xi\omega_o T + 4) + (2\omega_o^2 T^2 - 8)z^{-1} + (\omega_o^2 T^2 - 4\xi\omega_o T + 4)z^{-2}}$$

Ecuación 5

donde T es el periodo de simulación.
Su implementación en forma recursiva es:

$$c(k) = \frac{B^2 \cdot [vm(k) + 2 \cdot vm(k-1) + vm(k-2)] - A_1 \cdot c(k-1) - A_2 \cdot c(k-2)}{A_0}$$

Ecuación 6

donde:

$$k = instante\ actual$$
$$B = \omega_0 T$$
$$A_0 = (\omega_o^2 T^2 + 4\xi\omega_o T + 4)$$
$$A_1 = (2\omega_o^2 T^2 - 8)$$
$$A_2 = (\omega_o^2 T^2 - 4\xi\omega_o T + 4)$$

La tabla de variables del bloque de función desarrollado para su implementación es:

Tipo	identificador		Comentario
IN	r	real	entrada, vm(k)
	periodo	real	periodo de simulación, T en s
	frec_res	real	frecuencia de resonancia, ω_0 en rad/s
	coef_am	real	coeficiente de amortiguamiento, ξ
OUT	c	real	salida, c(k)
STAT	cp1	real	salida anterior, c(k-1)
	cp2	real	salida anterior, c(k-2)
	rp1	real	entrada anterior, vm(k-1)
	rp2	real	entrada anterior, vm(k-2)
TEMP	B	real	$\omega_0 T$
	A0	real	$\omega_0^2 T^2 + 4\xi\omega_0 T + 4$
	A1	real	$2\omega_0^2 T^2 - 8$
	A2	real	$\omega_0^2 T^2 - 4\xi\omega_0 T + 4$

El bloque simulador del proceso es el siguiente:

FB2

- cálculo de los coeficientes

```
L     #frec_res        //ω0
L     #periodo         //T
*R
```

```
T     #B          //ω0T
L     4.000000e+000
L     #coef_am    //ξ
*R
L     #B
+R
L     #B
*R
L     4.000000e+000
+R
T     #A0         //A0 = (4*ξ+B)*B+4
L     #B
L     #B
*R
L     2.000000e+000
*R
L     -8.000000e+000
+R
T     A1          //A1 = B*B*2-8
L     -4.000000e+000
L     #coef_am
*R
L     #B
+R
L     #B
*R
L     #4.000000e+000
+R
T     #A2         //A2 = (-4*ξ+B)*B+4
```

• cálculo de la respuesta

```
L     #rp1
L     2.000000e+000
*R
L     #rp2
+R
L     #r
+R
L     #B
*R
L     #B
*R
T     #c          //[r(k-1)*2+r(k-2)+r(k)]*B*B
L     #cp1
L     #A1
*R
L     #c
TAK
-R
T     #c          //[r(k-1)*2+r(k-2)+r(k)]*B*B-c(k-1)*A1
L     #cp2
L     #A2
*R
L     #c
TAK
-R
L     #A0
/R
T     #c                  //salida, c(k)
```

• actualización de las variables

```
L     #rp1
T     #rp2        //vm(k-2)
L     #r
T     #rp1        //vm(k-1)
L     #cp1
T     #cp2        //c(k-2)
L  #c
T  #cp1                   //c(k-1)
```

Este bloque, igual que en el caso anterior, se llama desde el bloque de organización **OB35**.

```
L     #OB35_EXC_FREQ
ITD
DTR
L     1000.0
/R
T     MD    10
CALL FB     2,DB2
      r     :=MD14
      periodo  :=MD10
      frec_res :=1.000000e-001
      coef_amort:=3.000000e-001
      c     :=MD18
```

El periodo de ejecución del bloque OB35 se ha fijado en 100 ms, en coherencia con el dato de periodo.
Igual que en el caso anterior, la señal de entrada se debe guardar en la marca MD14 y la señal de salida queda almacenada en la marca MD18.

3.3.- Tiempo muerto

Para simular un tiempo muerto basta almacenar la entrada durante un tiempo igual al tiempo muerto. La solución más simple es establecer un vector en que la primera posición acoge la entrada más reciente y la última posición contiene el valor más antiguo que constituirá la salida, después de cada salida se desplazan todos los valores. Este desplazamiento puede consumir una gran cantidad de tiempo de ejecución si la dimensión del vector, que es igual al cociente entre el tiempo muerto, θ, y el periodo de simulación, T, es grande.
Para incrementar la eficiencia de la ejecución se puede implementar un vector circular en el que en lugar de desplazar los valores se utiliza un puntero. Éste indica el lugar donde se encuentra el valor a extraer y donde se introducirá la nueva entrada, luego el puntero se incrementa en una unidad. Sólo hay que tener en cuenta que al sobrepasar la dimensión del vector se debe reiniciar.

El equivalente discreto del tiempo muerto θ, es:

$$\frac{C(z)}{VM(z)} = z^{-N}$$

Ecuación 7

donde N = θ/T.

El bloque de datos correspondiente a la función desarrollada es:

Tipo	identificador		Comentario
IN	r	real	entrada
	dead_time	int	tiempo muerto (N)
OUT	c	real	salida
STAT	pos	int	posición puntero

En este caso no hay cálculo de la salida sino que sólo se ha de gestionar el almacén de datos. Primero se calcula la

posición del valor que debe salir y se reinicializa si se supera la dimensión del almacén. El valor contenido en dicha posición es el que devuelve la función y será sustituido por la entrada actual.

FB3

```
• nueva posición
   L     #dead_time
   L     #pos
   ==I
   SPBN     _000
   L     0
_000:  INC 1
   T     #pos
• puntero
   L     4
   *I
   SLD  3
   L     P#8.0
   +D
   LAR1
• salida y almacenamiento de la entrada
   L     DID [AR1,P#0.0]
   T     #c
   L     #r
   T     DID [AR1,P#0.0]
```

Se ha fijado el periodo de ejecución del bloque **OB35** a 100 ms, igual que en los casos anteriores, en este bloque de organización se debe incluir:

```
CALL FB  3,DB3
       r     :=MD14
       dead_time:=20
       c     :=MD18
```

Igual que en los casos anteriores, la variable de entrada debe situarse en la marca MD14 y la de salida queda en la marca MD18.
En este ejemplo se ha considerado un tiempo muerto de 2 segundos, por lo que, siendo el periodo de simulación de 100 ms, el número de posiciones correspondiente es N = 20.
En el caso de necesitar simular un tiempo muerto mayor bastará aumentar el valor de N, incrementando las posiciones de memoria disponibles en el correspondiente bloque de datos o concatenar las llamadas al bloque de función que sean necesarias asociando cada llamada a un bloque de datos (DB) diferente.

4.- PLC SCHNEIDER ELECTRIC MODICON PREMIUM®

En este caso la tarea es más sencilla dado que el programa UnityPro® ya contiene algunos bloques de función en la librería, concretamente se pueden utilizar los bloques: LAG_FILTER y QDTIME, que se comportan como un proceso de primer orden y de tiempo muerto, respectivamente. Únicamente será necesario programar una función que se comporte como un proceso de segundo orden subamortiguado.

4.1.- Primer orden

La función LAG_FILTER se encuentra en la librería CONT_CTL/Conditioning de UnityPro, su representación en lenguaje ST (texto estructurado) es:

```
LAG_FILTER_Instance(IN:=InputValue, GAIN:=GainFactor,
   LAG:=LagTimeConstant, TR_I:=InicializationInput,
   TR_S:=InitializationType, OUT:=>Output)
```

Descripción de los parámetros de entrada:

IN	REAL	Valor de entrada, vm(k)
GAIN	REAL	Ganancia
LAG	TIME	Constante de tiempo, τ
TR_I	REAL	Entrada de inicialización
TR_S	BOOL	Tipo de inicialización
		“1” = Tracking
		“0” = Automático

Descripción del parámetro de salida:

OUT	REAL	Salida

Las variables y constantes hacen referencia a la ecuación 1.

En Automático, TR_S = 0, el módulo de función se procesará produciendo la respuesta esperada. En Tracking, TR_S = 1, la salida mantiene el valor de inicialización, TR_I. Si se desea simular un proceso de ganancia unidad y constante de tiempo de 5 segundos la llamada a introducir en la tarea MAST es:

```
LAG_FILTER_Instance(IN:=entrada,
      GAIN:=1.0,
      LAG:=t#5s,
      TR_I:=0.0,
      TR_S:=0,
      OUT:=>salida)
```

Así como en Step 7 se ha considerado la ganancia del proceso igual a la unidad, en este caso, la rutina de UnityPro contempla la introducción del valor.
Tal como se ha mencionado en el apartado Materiales y métodos, se han considerado las señales en voltios, por lo que la variable de salida se deberá multiplicar por 1000 y pasar a entero antes de enviarla a la salida analógica seleccionada, dado que la salida se debe hacer como entero en mV. De la misma forma la señal de entrada que el convertidor analógico/digital da como entero en mV, se deberá pasar a real en V.

4.2.- Segundo orden subamortiguado

Se ha programado una función FB derivada denominada SECOND_ORDER utilizando lenguaje ST y partiendo de la ecuación 6.

Descripción de las entradas y salida:

IN	REAL	entrada, vm(k)
RES_FR	REAL	frecuencia de resonancia en rad/s, ω_0
COEF_AM	REAL	coeficiente de amortiguamiento, ω
SIM_TIME	REAL	periodo de simulación en segundos, T
OUT	REAL	salida, c(k)

Las variables internas son:

OUT1	REAL	salida, c(k-1)
OUT2	REAL	salida, c(k-2)
IN1	REAL	entrada, vm(k-1)
IN2	REAL	entrada, vm(k-2)

Listado:

```
B:=RES_FR*SIM_TIME;
A0:=B*(B+4.0*COEF_AM)+4.0;
A1:=2.0*B*B-8.0;
A2:=B*(B-4.0*COEF_AM)+4.0;
OUT:=(B*B*(IN+2.0*IN1+IN2)-A1*OUT1-A2*OUT2)/A0;
OUT2:=OUT1;
OUT1:=OUT;
IN1=IN;
```

Para su ejecución se programa la tarea MAST como periódica con un periodo de ejecución igual al periodo de simulación, por ejemplo:

```
SEC_OR_IN1 (IN:=entrada,
   RES_FR:=0.1,
   COEF_AM:=0.3,
   SIM_TIME:=0.1,
   OUT=>salida);
```

en donde SEC_OR_IN1 es una instancia de la función derivada desarrollada.

4.3.- Tiempo muerto

En la misma librería mencionada se encuentra la función QDTIME, su representación en ST es:

```
QDTIME_Instance (IN:=InputValue, T_DELAY:=DeadTime,
        TR_I:InitializationInput, TR_S:=InitializationType,
        OUT:=>Output, READY:=>InternalBufferFlag)
```

Descripción de los parámetros de entrada:

IN	REAL	Valor de entrada
T_DELAY	TIME	Tiempo muerto
TR_I	REAL	Entrada de inicialización
TR_S	BOOL	Tipo de inicialización "1" = Tracking "0" = Automático

Parámetros de salida:

OUT	REAL	Salida
READY	BOOL	Estado del búfer "1" = lleno "0" = no lleno

Este bloque dispone de un búfer de 128 elementos, por lo que el tiempo muerto debe ser como máximo de 128 veces el periodo de simulación.

5.- CONCLUSIONES

Se han programado diversos bloques de función para simular procesos de primer y segundo orden subamortiguado y de tiempo muerto para PLC's Simatic S-300 de Siemens®.

Para los PLC's Premium de Schneider® no ha sido necesario programar los procesos de primer orden y de tiempo muerto porque ya existen funciones disponibles en la librería, por lo que únicamente se ha tenido que programar el proceso de segundo orden subamortiguado.

Se ha comprobado el correcto funcionamiento de las funciones programadas y de las existentes en la librería, recogiendo las respuestas sobre registro de papel, lo que ha permitido comprobar que las respuestas corresponden a los procesos simulados.

REFERENCIAS

1 Barberà, E.; Nevado, I.; Molins, J.J.: "El microordenador personal y sus aplicaciones al control de procesos: III Simulación híbrida de sistemas de control", *Afinidad*, **42**, 587-92, Noviembre-Diciembre 1985

2 Barberà, E.: "Hybrid simulation: a cheap alternative in process control laboratory" (oral communication), 11th Mediterranean Congress of Chemical Engineering, Barcelona, 21-24 october 2008

3 Molins, J.J.; Barberà, E.: "Autómatas programables: Step7® y UnityPro®", IQS, 2012

AGRADECIMIENTO

Se agradecen especialmente las sugerencias del revisor que han contribuido de forma importante a la mejora de la redacción del presente artículo.