Control con realimentación del estado con MATLAB 6.5 y puerto paralelo

Julio César Gómez Ruiz

||| POLITÉCNICA No. 1 | Medellín, junio - octubre de 2005, p.p. 105-118

Autor

JULIO CÉSAR GÓMEZ
Estudiante de Ingeniería en Instrumentación y Control - décimo semestre, Politécnico Colombiano Jaime Isaza Cadavid. Integrante del Semillero de Investigación en Gases, seis años de experiencia en programación con MATLAB, técnico en computación. Correo electrónico: juliogomez@hispavista.com

Resumen

En este artículo se aborda el control de un sistema de presión, desde la adquisición de datos, pasando por el desarrollo del algoritmo de control y la ejecución del mismo. Se utiliza un software de alto nivel llamado MATLAB®, conocido por sus fuertes herramientas de cálculo numérico y de simulación, que a la vez realiza un control por realimentación del estado, conectado a una tarjeta de adquisición de datos de ocho bits utilizando el puerto paralelo. Se explican las configuraciones del software para el manejo del puerto y las ventajas que se tiene en el campo industrial, tanto en el control clásico como en el control moderno, utilizando este lenguaje.

Palabras claves

Control - Controlador PID - Matlab - observador de estado - puerto paralelo - realimentación.

Abstract

In this article the pressure control system is approached, from the data acquisition, through the development of the algorithm of control and the execution of itself. It uses a high level software called MATLAB®, known by its strong tools for numeric calculus and simulation, and simultaneously makes a state feedback control, connected to a card of data acquisition of eight bits using the parallel port. The configurations of the software for the handling of the port and the advantages in the industrial field are explained, as much as in the classic control as in the modern controls, using this language.

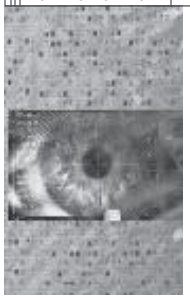
Key words

Control - PID Controller - Matlab - State Observer - Parallel Port - Feedback.

Control con realimentación del estado con MATLAB 6.5 y puerto paralelo

Julio César Gómez Ruiz

||| POLITÉCNICA No. 1 | Medellín, junio - octubre de 2005, p.p. 105-118



Introducción

"Como consecuencia del gran avance experimentado por la computación digital, prácticamente todos los sistemas de control construidos hoy en día se basan en microprocesadores y sofisticados microcontroladores. La utilización de los sistemas de control basados en computador permiten satisfacer especificaciones más exigentes que las que se pueden lograr con los sistemas analógicos así como posibilitar nuevas funcionalidades." [1]

Normalmente el diseño de un algoritmo de control tiene sus dificultades a la hora de elegir la estructura de control que más se ajuste a la dinámica de una planta. En el control clásico es más sencillo el diseño del algoritmo, mientras que empleando las técnicas de control digital se requiere de un poco más de análisis y procedimiento matemático. El lenguaje de programación MATLAB 6.5 [2] es una herramienta indispensable, porque tiene la posibilidad no sólo de simular, sino también de adquirir datos y controlar en tiempo de ejecución un sistema, con algoritmos de control a nivel casi conceptual, gracias a su avanzado cálculo matricial y a sus herramientas: Control System, System Identification e Instrument Control. El avance de estos lenguajes permite al ingeniero incluso ir más allá del control convencional y probar otras alternativas llevan a un avance significativo en la industria.

[1] GARCÍA JAIMES, Luis Eduardo. *Control Digital, Teoría y Práctica.* Politécnico Colombiano Jaime Isaza Cadavid. Medellín, 2004.

[2] El software MATLAB 6.5 utilizado para el ejercicio de este proyecto fue facilitado por la Escuela de Ingenierías de la Universidad Pontificia Bolivariana.

Desarrollo del proyecto

Lo primero que se debe tener en cuenta para diseñar un sistema de control es cuál será la variable medida y controlada y cuál será la variable manipulada. Con esto claro, se deben conocer los límites del sistema, para seguridad tanto del proceso como de los individuos que lo operan.

En este caso se implementó un sistema de control sobre un proceso de presión de un tanque cerrado, es un sistema simple pero suficiente para apreciar la utilidad de los algoritmos de control digital.

En el diagrama mostrado en la figura 1 se puede observar el esquema del sistema utilizado para realizar el control del proceso.

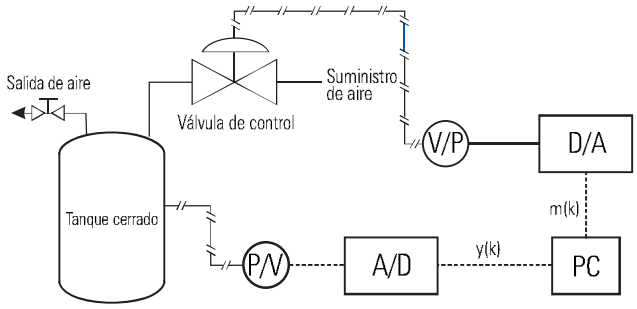


Fig. 1 Proceso para el control de presión

Como se dijo anteriormente, es necesario conocer los límites del sistema. Para este tanque se calibra el transductor de presión de 0 a 25 psi, rango dentro del cual se garantiza la seguridad del sistema y la integridad de los usuarios.

El transductor de presión a voltaje (P/V) entrega una señal de voltaje normalizada de 0 a 5 v dc, correspondiente a la variable que se mide y(k), esta señal análoga es recibida por el convertidor análogo/digital (A/D), el cual la entrega al computador en una medida proporcional a su resolución. Para el caso de ocho bits será de 0 a 255 en binario.

Con los datos en el computador, el algoritmo de control procesa y(k) y entrega un valor de salida m(k) por el puerto al convertidor digital/análogo (D/A). Esta señal que es de 0 a 5v dc, la recibe el convertidor de voltaje a presión (V/P), que traduce la señal eléctrica a presión de aire de 3 a 15 psi, señal que necesita la válvula para abrir o cerrar según el valor de la señal que recibe.

Configuración del software

MATLAB® es un lenguaje de alto nivel utilizado en el campo científico y de ingeniería, por la gran capacidad de procesamiento numérico y simulación. Hasta aquí no hay nada nuevo, pero en las versiones 6.X permite manejar los puertos del computador y, en este caso se utiliza el puerto paralelo para la adquisición y control de un proceso con una tarjeta de ocho bits.

Configuración física del Puerto

El Puerto paralelo del computador, es utilizado para realizar comunicación con dispositivos de salida como impresoras, o con otros equipos de procesamiento o computadores. Este puerto consta de 25 pines, los cuales están distribuidos en puertos lógicos de la siguiente forma:

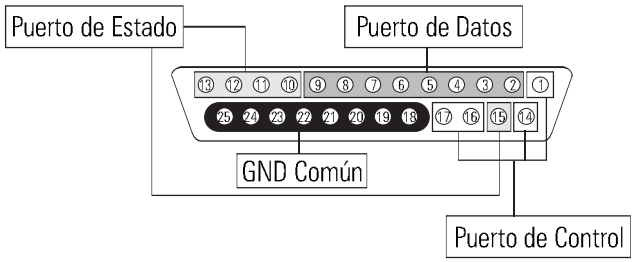


Fig. 2 Estructura física del puerto paralelo

Puerto	Orden físico del byte / Número del pin							
	7	6	5	4	3	2	1	0
Datos	9	8	7	6	5	4	3	2
Control					*17	16	*14	*1
Estado	*11	10	12	13	*15			

Tabla 1. Estructura lógica del puerto paralelo

Nota: Los pines marcados (*) son de hardware invertido.

- El puerto de Datos se puede configurar como entrada o salida.
- El puerto de Control se puede configurar como entrada o salida.
- El puerto de Estado sólo se configura como entrada.

- Para realizar la configuración del puerto, se debe conocer el funcionamiento de la tarjeta de adquisición de datos, la cual funciona de la siguiente manera:
- La salida de control se entrega a la tarjeta por el puerto de datos, en pocas palabras hay que configurar este puerto como salida.
- Los datos del proceso, la tarjeta los entrega en 2 nibbles, (el orden alto y bajo se configuran desde software), este dato se recibe por el puerto de estado, se debe hacer la corrección del dato por el hardware invertido.
- Por el puerto de control se le ordena a la tarjeta que seleccione el canal, haga la conversión análoga/digital, envíe el primer nibble (alto o bajo) y luego el segundo nibble, hay que tener en cuenta el hardware invertido para este puerto.

Para empezar, la configuración en MATLAB 6.5, se hace de la siguiente forma:

```
>> puertos=daqhwinfo('parallel')

puertos =

        AdaptorDllName: 'C:\MATLAB6p5\toolbox\daq\daq\private\mwparallel.dll'
    AdaptorDllVersion: 'Version 2.2    (R13) 28-Jun-2002'
          AdaptorName: 'parallel'
           BoardNames: {'PC Parallel Port Hardware'}
     InstalledBoardIds: {'LPT1'}
  ObjectConstructorName: {''  ''  'digitalio('parallel','LPT1')'}
```

- El comando daqhwinfo, entrega información del hardware de adquisición en forma de estructura, en este caso se pide información del puerto paralelo.

```
>> existencia=puertos.InstalledBoardIds

existencia =

    'LPT1'
```

- Se asigna a una variable los identificadores de los puertos paralelos instalados en el PC.

```
>> lpt = existencia {1}

lpt =

LPT1
```

- Se le asigna a la variable lpt, el primer puerto encontrado (o el único encontrado).

```
>> port_io=digitalio('parallel',lpt)

Display Summary of DigitalIO (DIO) Object Using 'PC Parallel Port Hardware'.

        Port Parameters:  Port 0 is port configurable for reading and writing.
                          Port 1 is port configurable for reading.
                          Port 2 is port configurable for reading and writing.

          Engine status:  Engine not required.

DIO object contains no lines.
```

■ El comando digitalio construye un objeto de entrada o salida digital del puerto asociado al adaptador parallel, con el identificador lpt.

```
>> Data=addline(port_io,[0:7],0,'out')

  Index:  LineName:  HwLine:  Port:  Direction:
  1       'Pin2'     0        0      'Out'
  2       'Pin3'     1        0      'Out'
  3       'Pin4'     2        0      'Out'
  4       'Pin5'     3        0      'Out'
  5       'Pin6'     4        0      'Out'
  6       'Pin7'     5        0      'Out'
  7       'Pin8'     6        0      'Out'
  8       'Pin9'     7        0      'Out'
```

■ El comando addline añade líneas al objeto de entrada o salida digital (port_io). Aquí se configuran todos los pines del puerto de datos como salidas digitales.

```
>> Status.Dato=addline(port_io,[1:4],1,'in')

Status =

    Dato: [4x1 dioline]
```

■ Aquí se configura el puerto de estado como entradas digitales, de 1 a 4 se recibe el nibble.

```
>> Status.Error=addline(port_io,0,1,'in')

Status =

    Dato: [4x1 dioline]
   Error: [1x1 dioline]
```

■ Aquí se configura el bit 0 del puerto de estado. Nota: Aunque la configuración del puerto de estado es en realidad del bit 3 al 7, Matlab permite una facilidad de interpretarlo y configurarlo como bits 0 a 4.

```
>> Control.Strobe=addline(port_io,0,2,'out')

Control =

    Strobe: [1x1 dioline]
```

■ Aquí se configura el bit 0 del Puerto de Control como salida digital.

```
>> Control.Autofeed=addline(port_io,1,2,'out')

Control =

     Strobe: [1x1 dioline]
   Autofeed: [1x1 dioline]
```

■ Se configura el bit 1 como salida digital.

```
>> Control.InitPrinter=addline(port_io,2,2,'out')

Control =

       Strobe: [1x1 dioline]
     Autofeed: [1x1 dioline]
   InitPrinter: [1x1 dioline]
```

■ Se configura el bit 2 como salida digital.

```
>> Control.SelectPrinter=addline(port_io,3,2,'out')

Control =

       Strobe: [1x1 dioline]
     Autofeed: [1x1 dioline]
   InitPrinter: [1x1 dioline]
  SelectPrinter: [1x1 dioline]
```

■ Se configura el bit 3 como salida digital.

Aquí ya se tiene la configuración del puerto paralelo para la comunicación con la tarjeta, la forma de enviar y recibir los valores de los puertos se reduce solo a dos comandos que son:

putvalue(puerto,valor)

Envía el valor al puerto configurado como salida, el valor puede estar en binario o simplemente en decimal.

Ejemplo:

Putvalue(Data,10); %Envía un valor de 10 al puerto de Datos.

getvalue(puerto)

Lee el valor que hay en el puerto configurado como entrada.
Aquí hay una cosa muy importante que se debe tener en cuenta, y es que al leer un valor en el puerto, el getvalue devuelve el binario que hay en el puerto, pero este valor binario lo devuelve en espejo, o sea:
Si hay un binario de la siguiente forma en el puerto 00001010, esto es un 10 en decimal, al leer el puerto, el getvalue retorna el siguiente valor: 01010000.
Sin embargo para convertir este número en decimal, MATLAB posee un comando que tiene esto en cuenta, binvec2dec(valor_bin). Se utiliza de la siguiente forma:

```
>> valor_bin=getvalue(Data)
valor_bin =
   1   0   0   1   1   0   0   0
>> decimal=binvec2dec(valor_bin)
decimal =
   25
```

Se puede observar que el binario se puede interpretar normalmente como un 152 decimal, binvec2dec lo convierte en un 25 decimal, por lo del valor espejo que se explicó anteriormente.

La configuración del puerto se recomienda programarla como función para ser invocada al principio del programa de control y sólo trabajar con los objetos que contienen la dirección del puerto configurado, así:

Se crea una función de la siguiente forma, en este artículo se le dará el nombre de configlpt:

```
function [Data,Control,Status]=configlpt()
%------------ FUNCION QUE CONFIGURA EL PUERTO LPT ----------------
puertos=daqhwinfo('parallel');
existencia=puertos.InstalledBoardIds;
lpt=existencia{1};
port_io=digitalio('parallel',lpt);
Data=addline(port_io,[0:7],0,'out');
Status.Dato=addline(port_io,[1:4],1,'in');
Status.Error=addline(port_io,0,1,'in');
Control.Strobe=addline(port_io,0,2,'out');
Control.Autofeed=addline(port_io,1,2,'out');
Control.InitPrinter=addline(port_io,2,2,'out');
Control.SelectPrinter=addline(port_io,3,2,'out');
%-----------------------------------------------------------------------------------------------
```

Ahora bien, si se tienen los datos binarios en el puerto del computador, el software debe leerlo y hacer lógicamente la conversión matemática de la señal.

Por ejemplo, para este trabajo se utilizó una tarjeta que tiene como voltaje de referencia 5v dc, y hace la conversión a ocho bits, la resolución de la tarjeta será:

$$\frac{5v}{255} \quad 0.01960784313725(\text{v/bit})$$

Esto muestra la sensibilidad de voltaje del convertidor por cada bit.

Proceso de adquisición de datos

Para este proceso se deben tener preparados y calibrados los instrumentos de medición, el hardware y el software listos y probados. Ahora se debe elegir un período de muestreo, para lograr una reconstrucción adecuada de la señal análoga correspondiente a la respuesta del proceso. Además este período de muestreo interviene en el proceso de identificación y control del proceso. No es conveniente un período de muestreo muy pequeño que consuma demasiados ciclos de máquina innecesariamente, ni tan grande que se pierdan datos importantes de la respuesta de la planta.

Se le aplicará a la planta un estímulo escalón. La magnitud del escalón debe elegirse de acuerdo con la ganancia del sistema, si es un sistema con mucha ganancia, se debe elegir una magnitud pequeña del estímulo, para poder tomar suficientes datos de estímulos y respuestas de la planta, lo contrario se debe hacer si el sistema posee poca ganancia, se debe elegir un magnitud mayor para poder apreciar la respuesta. Pero como no se conoce cómo va a ser la respuesta de la planta por primera vez, en los ensayos hechos para probar y sincronizar el software con el hardware, se puede determinar la magnitud más apropiada para el sistema.

Para este proceso se diseñó un programa sencillo de adquisición, en las pruebas realizadas previamente para garantizar que todo funcionara bien, se pudo determinar que la planta posee una ganancia apreciable y, además una respuesta rápida, por lo que se decide para la toma de datos, darle un período de muestreo de 0.3 segundos y una magnitud para el estímulo de 2%.

Identificación del sistema

La selección de la mejor respuesta del sistema según el valor del estímulo debe tener los siguientes requisitos:

- La respuesta transitoria debe tener unos valores muy estables o continuos, no debe poseer muchas oscilaciones, puesto que en esta región de datos es de donde se medirán los valores para su identificación.
- Los valores en la región de estado estable deben ser muy constantes, porque la relación entre estos valores y el valor del estímulo determina la ganancia del sistema.
- Finalmente se debe elegir los datos de respuesta del sistema que posea una ganancia intermedia entre la más alta y la más baja, para que la identificación no posea los extremos dinámicos del sistema.

Teniendo en cuenta estos criterios, se seleccionó para este caso los valores de la respuesta del sistema en el escalón comprendido de 3% al 5 %. Los datos se organizaron para la identificación tal como se muestra en la figura 3. (Se desplazaron los ejes para que la respuesta de la señal y el tiempo reinicien en cero).

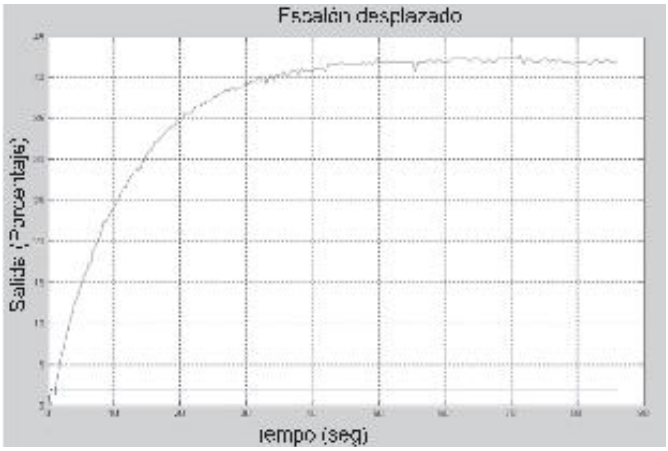


Figura 3. Datos desplazados para identificación

Para modelar el sistema se utilizó identificación no paramétrica, se aproximó el proceso a un modelo de primer orden y a un modelo de segundo orden.

Modelo de primer orden

El modelo de primer orden tiene por función de transferencia:

$$G(s)=\frac{Ke^{-\theta' s}}{\tau . S+1} \quad [1]$$

Donde:

$$k=\frac{\Delta y}{\Delta u}=\frac{Cambio\ en\ la\ Salida}{Cambio\ en\ la\ Entrada} \quad [2]$$

θ y T se calculan al resolver la ecuaciones:

$$\theta'+\frac{\tau}{3}=t_1$$
$$\theta'+\tau=t_2 \quad [3]$$

Siendo t1 y t2 los tiempos necesarios para que la respuesta alcance respectivamente el 28.3% y el 63.2% de su valor final.

De la figura 3 se obtiene:

Magnitud del estímulo:

$$\Delta u=2\%$$

Amplitud de respuesta:

$$\Delta y=41.5179\%$$

Ganancia del sistema:

$$k=\frac{\Delta y}{\Delta u}=\frac{41.5}{2}=20.75$$

Valor de Respuesta al 28.3%: $\Delta y1=11.74\%$
→ Tiempo correspondiente: $t_1=3.85$

Valor de Respuesta al 63.2%: $\Delta y2=26.2\%$
→ Tiempo correspondiente: $t_2=11.65$

Resolviendo las ecuaciones:
La constante de tiempo es:

$$\tau=1.5(t_2-t_1)=11.7$$

El retardo del sistema es:

$$\theta'=1.5(t_1-\frac{t_2}{3})=-0.050$$

Como θ' dio negativo se hace igual a cero.
Entonces el modelo del proceso,
aproximado a un modelo de primer orden es:

$$Gp(s)=\frac{20.7590}{11.7S+1} \quad [4]$$

Modelo de segundo orden

El modelo de segundo orden se tiene por función de transferencia:

$$G(s)=\frac{K.Wn^2e^{-\theta' S}}{S^2+2\xi.S+Wn^2} \quad \text{Para} \quad \xi<1 \quad [5]$$

$$G(s)=\frac{K.Wn^2e^{-\theta' S}}{(T_1S+1)(T_2S+1)} \quad \text{Para} \quad \xi\geq 1 \quad [6]$$

Con: $T_1=\frac{\xi+\sqrt{\xi^2-1}}{Wn}$

$T_2=\frac{\xi-\sqrt{\xi^2-1}}{Wn}$

Donde: $k=\frac{\Delta y}{\Delta u}=\frac{Cambio\ en\ la\ Salida}{Cambio\ en\ la\ Entrada}$

$$\xi=\frac{0.0805-5.547(0.475-X)^2}{X-0.356} \quad [7]$$

Con $X=\frac{t_2-t_1}{t_3-t_1}$

$$Wn=\frac{F_2(\xi)}{t_3-t_1} \quad [8]$$

Con: $F_2(\xi) = 0.708(2.811)^{\xi} \quad Para: \ \xi < 1$
$F_2(\xi) = 2.6\xi - 0.6 \quad Para: \ \xi \geq 1$

$$\theta' = t_2 - \frac{F_3(\xi)}{Wn} \qquad [9]$$

Con: $F_3(\xi) = 0.922(1.66)^{\xi}$

De la figura 3 se obtiene:

Valor de Respuesta al 15%: $\Delta y1 = 6.22\%$
→ Tiempo correspondiente: $t1 = 2.05$

Valor de Respuesta al 45%: $\Delta y2 = 18.67\%$
→ Tiempo correspondiente: $t2 = 7$

Valor de Respuesta al 75%: $\Delta y3 = 31.12\%$
→ Tiempo correspondiente: $t3 = 16.22$

Coeficiente de amortiguamiento: $\zeta = 1.0713$

Para $\zeta \geq 1$ se tiene:

$Wn = 0.1542 \quad \theta' = -3.2905$

→ El retardo hace igual a cero.

$$Gp(s) = \frac{20.7590}{(9.4397S + 1)(4.4553S + 1)} \qquad [10]$$

Es decir: $$Gp(s) = \frac{0.4936}{s^2 + 0.3304s + 0.0238} \qquad [11]$$

Aquí se pueden apreciar las funciones de transferencia de primero y segundo orden que modelan el sistema. Las dos funciones de transferencia representan la dinámica del sistema así que cualquiera de las dos sirve para calcular los controladores.

Para este caso se trabajará con la función de transferencia de segundo orden para el cálculo de los controladores.

Discretización de la FT de segundo orden

$$Gp(s) = \frac{0.4936}{s^2 + 0.3304s + 0.0238}$$

Se calculó el rango del período de muestreo por el método de ancho de banda:

$$0.4868 \leq T \leq 0.7301$$

Se toma: $T = 0.6085$ para discretizar la FT.
Se discretiza con retenedor de orden cero:

$$HG(z) = \frac{z-1}{z} \times \Im\left\{\frac{0.4936}{s\left(s^2 + 0.3304s + 0.0238\right)}\right\} \qquad [12]$$

$$HG(z) = \frac{0.0843z + 0.0811}{z^2 - 1.8099z + 0.8179} \qquad [14]$$

Con la Función de Transferencia de Pulso se inicia el diseño del controlador digital. Los controladores más utilizados para el control clásico son el Proporcional (P), Proporcional + Integral (PI) y el Proporcional + Integral + Derivativo (PID), para estos controladores hay muchos ajustes tales como: Ganancia Limite, Ziegler-Nichols y Criterios de error mínimo.

En el trabajo se calculó un controlador PID con ajuste por Ziegler-Nichols, asi:

De la ecuación [11] se tiene:

$Teq = 13.8774$ y $\theta' = 0$

Aplicando criterio Z - N

$$Kc = \frac{1.2\,Teq}{K\theta} \qquad [15]$$

$$Ti = 2\theta \qquad [16]$$

$$Td = 0.5\theta \qquad [17]$$

Con $$\theta = \theta' + \frac{T}{2} \qquad [18]$$

Se tienen los siguientes parámetros:

$Kc = 2.6387$	$Ti = 0.6085$	$Td = 0.1522$

La ecuación del controlador es la siguiente:

$$m(k) = q_0.e(k) + q_1.e(k-1) + q_2.e(k-2) + m(k-1)$$

q_0=4.66181 q_1=-2.6394 q_2=0.66

Con: $q_0 = K_c.\left(1 + \frac{T}{2.T_i} + \frac{T_d}{T}\right)$ [20]

$$q_1 = -K_c.\left(1 + \frac{2.T_d}{T} - \frac{T}{2.T_i}\right) \quad [21]$$

$$q_2 = \frac{(K_c.T_d)}{T} \quad [22]$$

Error: e(k) y Período de muestreo: *T*

La ecuación del controlador queda:

$$m(k) = 4.6181e(k) - 2.6394\ e(k-1) + 0.66\,e(k-2) + m(k-1) \quad [23]$$

La ejecución del controlador arrojó como resultado la siguiente respuesta del sistema:

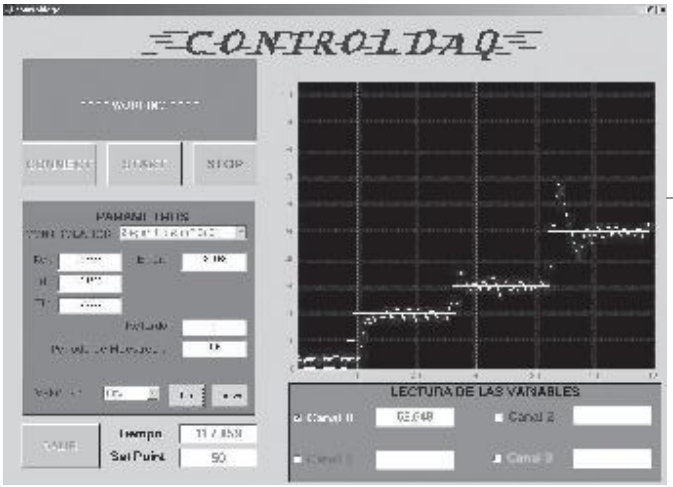


Figura 4.
Respuesta del sistema con el controlador PID por Ziegler Nichols.

El sistema con un controlador PID por Z-N presenta buena velocidad de respuesta y sobreimpulso, auque posee una variación alrededor de set point en el estado estable, es un controlador aceptable, puesto que está dentro de los límites permisibles para este sistema.

Se buscará entonces mejorar el comportamiento del sistema con un algoritmo de control moderno, como lo es el controlador por observador de estado de orden reducido.

Observador de estado de orden reducido

Después de realizar el procedimiento para determinar los parámetros del controlador por observador de estado de orden reducido se obtiene:

Se supone y(k)=X_1(k), entonces el sistema transformado es:

$$X(k+1) = \begin{bmatrix} -0.8502 & 0.3053 \\ -10.0851 & 2.6601 \end{bmatrix} X(k) + \begin{bmatrix} 0.0843 \\ 1 \end{bmatrix} U(k) \quad [24]$$

$$Y(k) = \begin{bmatrix} 1 & 0 \end{bmatrix} X(k) \quad [25]$$

La ecuación del observador de estado de orden reducido es:

$$\mathbf{q}_b(k+1) = (\mathbf{A}_{bb}-\mathbf{G}\,\mathbf{A}_{ab})\,\mathbf{q}_b(k)+\mathbf{G}\,\mathbf{y}(k+1) +(\mathbf{A}_{ba}-\mathbf{G}\,\mathbf{A}_{aa})\,\mathbf{y}(k)+(\mathbf{B}_b-\mathbf{G}\,\mathbf{B}_a)\mathbf{u}(k) \quad [26]$$

Con: $$G = \left(\phi A_{bb}\right)\begin{bmatrix} A_{ab} \\ A_{ab}A_{bb} \\ \cdots \\ A_{ab}A_{bb}^{\,n-2} \end{bmatrix}^{-1}\begin{bmatrix} 0 \\ 0 \\ \cdots \\ 1 \end{bmatrix} \quad [27]$$

$$\left(\phi A_{bb}\right) = A_{bb}^{\,n-1} + \alpha_1 A_{bb}^{\,n-2} + \ldots + \alpha_{n-2}A_{bb}^{\,n-1} + \alpha_{n-1}I \quad [28]$$

Los α_i son los coeficientes del polinomio:

$$Z^{n-1} + \alpha_1 Z^{n-2} + \ldots + \alpha_{n-2}Z + \alpha_{n-1}$$

que es resultado del ingreso de los polos deseados para el sistema.

La ley de control para el observador de estado de orden reducido es:

$$U(k) = -K_1\,y(k) - \mathbf{K}_b\,\mathbf{q}_b(k) \quad [29]$$

Se Calcula la matriz

K=[K_1 K_b] con polos en:

Z=0.9010+0.0784j y Z=0.9010-0.0784j

$$K = \begin{bmatrix} 0.001 & 0.007 \end{bmatrix} \quad [30]$$

Se calcula la matriz G con polo en Z=0.4990

$$G=[10.3476] \quad [31]$$

Entonces la ecuación del controlador es:

$$D(z) = \frac{-U(z)}{Y(z)} = K_1 + K_b\left[Z.I - A_{bb} + G.A_{ab} + (B_b - G.B_a)K_b\right]^{-1} * \left[G.Z + \{A_{ba} - G.A_{aa} - K_1(B_b - G.B_a)\}\right] \quad [32]$$

Con: A_{aa} = -0.8502; A_{ab} = 0.3053 ;
A_{ba} = -10.0851; A_{bb} = 2.6601;
B_a = 0.0843; B_b = 1; G=10.3476
K_0 = 0.575 K_1=0.001 K_b=0.007

Donde: $$Gw(z) = \frac{Nw(z)}{Dw(z)} = \frac{K_0 HG(z)}{1 + HG(z)D(z)} \quad [33]$$

La ecuación quedaría de la siguiente forma:

$$D(z) = 0.001 + 0.007\left[Z.I - 2.6601 + (10.3476)(0.3053) + (1 - (10.3476).(0.0843))0.007\right]^{-1} * \left[(10.3476).Z + \{-10.0851 - (10.3476).(-0.8502) - (0.001)(1 - (10.3476).(0.0843))\}\right] \quad [34]$$

La ejecución del controlador de Orden Reducido presentó la siguiente respuesta.

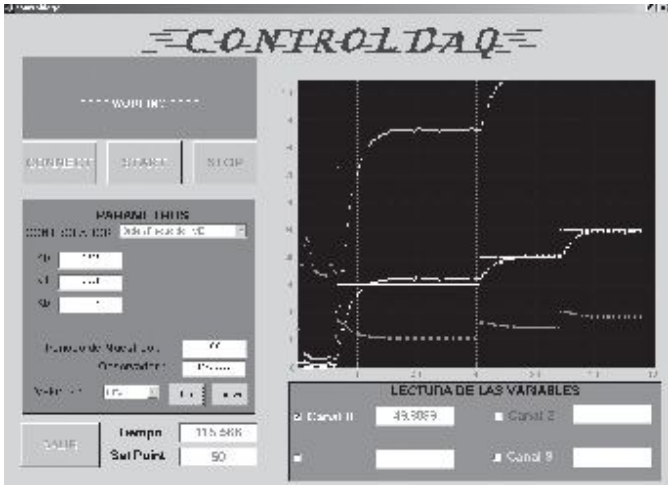


Figura 5.
Respuesta del sistema con el controlador Observador de Estado de Orden Reducido.

Numero	Señal
1	Set Point
2	Presión (Respuesta de la planta)
3	Salida de Control
4	Salida estimada por el Observador
5	Observador

Tabla 2 Señales del sistema controlado con el Observador de Estado

Se puede observar que el controlador por observador de estado de orden reducido, se ajusta mejor al set point establecido. Para una mejor visualización del funcionamiento del algoritmo, se graficaron también la salida de control (3), la salida del observador (5) y la salida estimada por el observador (4).

Este algoritmo de control requiere un poco más de procesamiento matemático, MATLAB brinda la facilidad de programar el algoritmo sin complicaciones a nivel matricial.

Conclusiones

Este trabajo muestra que el software Matlab 6.5, deja ya de ser un software de sólo cálculo y simulación, permitiendo trabajar con el puerto paralelo, al cual se le dedicó un gran espacio en esta publicación debido a la importancia que tiene el adecuado manejo de este puerto en la implementación de sistemas de control digital.

Para el sistema de presión controlado, el control clásico PID por Ziegler Nichols, presentó una respuesta aceptable, puesto que controlaba dentro del rango de la variable, aunque por su oscilación alrededor de ella y el sobreimpulso presentado, no sería recomendable si esta planta necesitara una operación con alto grado de precisión.

El control por Observador de estado de orden reducido, presentó una respuesta muy buena para esta planta. De los dos controladores empleados éste sería el ideal para manipular dicho sistema, auque el costo computacional es mucho más alto que el de Ziegler Nichols.

Pruebas como ésta, permiten al usuario, tener un buen criterio sobre la elección del controlador adecuado. Las dinámicas de los sistemas a controlar son muy diferentes, y buscar el controlador que más se acomode a éste y además garantizar un producto de alta calidad, exigen a los Ingenieros en Instrumentación y Control de hoy en día realizar investigaciones más precisas para el diseño de un control, que permitan ofrecer al mercado productos y procesos de alta competitividad.

Bibliografía

1. GARCÍA, Luis Eduardo. *Control Digital, Teoría y Práctica.* Medellín, Politécnico Colombiano Jaime Isaza Cadavid, 2004.
2. OGATA, Katsuhico. *Ingeniería de Control Moderna*. Prentice - Hall, 1996.
PHILLIPS, Charles, Nagle, Troy. *Digital control systems analysis and design.* Prentice - Hall. Englewood Cliffs. 1995.
3. STEFANI, Raymond T. *Design of Feedback Control System.* Prentice – Hall, Oxford, New York, 2002.
4. KUO, Benjamin. Sistemas de control automático. México, Prentice - Hall, 1996.
5. GRUPO de investigación en Sistemas de Control Digital (SCD) Universidad EAFIT. Guía básica de iniciación en control. http://www.control-systems.net/investigacion/control/guia_control.htm. Consultada en marzo de 2005
6. Saucedo Flores, Salvador. Página personal (ESIME Zacatenco-IPN)–2005.
http://www.prodigyweb.net.mx/saucedo8/ Consultada en marzo de 2005.