Laboratorio de control

Identificación de modelos de orden reducido a partir de la curva de reacción del módulo de presión

Mauricio Duque Quintero, Yorguin Jose Mantilla Ramos, Santiago Carmona Benavides

Departamento de Ingeniería Electrónica
Universidad de Antioquia
Medellín, Colombia

mauricio.duque@udea.edu.co, yorguinj.mantilla@udea.edu.co, santiago.carmonab@udea.edu.co

Resumen – Se encuentra la respuesta al escalón unitario de una máquina que controla la presión de un tanque con el fin de caracterizar el sistema mediante la curva de reacción. De esta manera se buscan varios modelos matemáticos de 1er y 2do orden que se ajusten al punto de operación trabajado. Finalmente se seleccionan los mejores modelos de cada orden a partir de criterios de error.

Palabras clave – modelo experimental, comparación, parámetros, escalón, orden, control, dinámico, error, polos, tiempo muerto, estabilidad.

I. INTRODUCCION

Para poder realizar un control por realimentación es necesario caracterizar el sistema a controlar; esto se logra con diversos modelos de distinta complejidad. La caracterización se puede lograr experimentalmente al obtener su curva de reacción ante una excitación arbitraria. Este método no tiene en cuenta parámetros internos pero permite modelar satisfactoriamente algunos sistemas, siendo así ventajoso respecto a otros métodos de modelamiento más complejos.

En esta práctica se identifican varios modelos de 1er y 2do orden con tiempo muerto para representar el comportamiento de la presión de un tanque, el cual se pretende controlar en un futuro. En particular la región de operación que se busca describir corresponde a la desviación por pequeña señal alrededor de un punto de operación, de tal manera que el comportamiento del sistema sea lineal. Los modelos hallados así corresponden a modelos ENTRADA-SALIDA que se pueden representar fácilmente mediante una función de transferencia. Luego de hallado los modelos se realiza una comparación tanto visual como cuantitativa entre ellos para así encontrar el que mejor represente al sistema.

II. OPERACIÓN DEL SISTEMA DE PRESIÓN

A. Condiciones de experimentación

Para poner en funcionamiento el sistema, es necesario abrir las dos válvulas que le proveen aire. Luego de manera interna se abre solo una válvula de control que permite el flujo de aire hacia al tanque. El conversor de corriente a presión usado es el primero de los dos que tiene el sistema.

La señal de entrada del sistema corresponde a la apertura de la válvula de control. Se caracterizó el rango de dicha variable siendo el 100% correspondiente a una corriente estable de 19.2 mA en la señal de salida, la cual expresa en corriente la presión del tanque. El valor usado para estabilizar la respuesta del sistema sobre un punto de operación fue una apertura de válvula del 29%, la cual entregó una lectura de 9.88 mA a la salida. Luego se aplicó un incremento en la entrada hasta llegar al 34% de apertura, obteniendo un valor estable en la señal de salida de 10,9 mA.

La necesidad de mantener las mismas condiciones de operación a lo largo del trabajo en el laboratorio es debido a querer obligar que el sistema sea representado por el mismo modelo lineal creado en esta práctica.

B. Adquisición de datos

El computador se encuentra conectado con el sistema y posee una herramienta denominada *SoftControl* que va a permitir controlar manualmente la apertura de la válvula de entrada. A partir de allí se obtienen los datos de entrada (out1 o % de apertura de la válvula) y de salida (PV1 o presión en el sistema expresado en términos de corriente).

El programa mientras va registrando los datos permite crear una gráfica donde se muestra el comportamiento de las señales de entrada y salida del sistema. Dicha gráfica se observa en la figura 1, donde la señal azul es la entrada y la señal rosa es la salida. Observe que la entrada es de tipo escalón.

Fig 1. Señales de entrada y salida del sistema creadas en SoftControl.

También, el software *SoftControl* genera un archivo de Excel en base a las gráficas, del cual es posible obtener la amplitud de las dos señales contra el tiempo. En total se obtuvieron 992 datos durante un tiempo aproximado de 7 minutos.

III. DEPURACIÓN DE LOS DATOS

El archivo de Excel que guarda el programa posee en repetidas ocasiones muchas mediciones de amplitud para un mismo segundo de tiempo, de manera que, es necesario fraccionar los intervalos de tiempo. Lo anterior se logró mediante Google Sheets de la siguiente forma: 1) A cada segundo se le cuenta cuantos datos se obtuvieron y se le asigna una frecuencia de muestreo particular. 2) Se construye un vector de tiempo cuya frecuencia de muestreo varía en cada segundo. Finalmente los datos se guardan en un archivo csv que se importa a Matlab para realizar la depuración.

En Matlab lo primero que se hizo fue re-escalar el vector del escalón de tal manera que sus valores queden en el rango de 4 a 20 mA, con la equivalencia de que el 0% de apertura fuera 4mA y el 100% correspondiera a 19.2 mA. Lo anterior se hizo con una función que mapea datos en el rango (a,b) al rango (c,d):

```
% map range [a,b] to range [c,d]
function [x] = mapScale(x,a,b,c,d)
x = (x-a)*(d-c)/(b-a) + c;
end
```

Al acercar suficientemente la gráfica de los datos "crudos", como se observa en la Fig. 2, se puede apreciar que la señal posee saltos discontinuos, por lo que usando Matlab, se realiza el suavizado de los datos mediante un filtro de tipo 'moving average' el cuál se configura para tomar en cada iteración el 10% de la longitud de los datos. El código de suavizado se puede observar en lo que sigue:

```
% Smoothing
% span of 10% of the total number of data
points
span = 0.1 * length(time);
pv_smooth = smooth(time,pv,span,'moving')';
```

Sin embargo el suavizado no se realiza inmediatamente, sino después de cortar los datos al punto donde comienza el escalón. Lo anterior, debido a que si se realiza con los datos completos el filtro, al hacer el promedio, tiende a adelantar la respuesta del sistema con respecto a la aplicación del escalón. El proceso anterior corresponde al código que sigue:

```
% find when the step stimuli starts
start = find(vop_all>=max(vop_all), 1,
'first');
time = time_all(start:end);
% remove offset to start from zero
time = time - time_all(start);
pv = pv_all(start:end);
vop = vop_all(start:end);
```

La comparación entre la respuesta real y la respuesta suavizada se puede apreciar en la Fig. 3, mientras que en la Fig. 4 se observa la señal de entrada y de salida con offset.

Para quitar los offsets de los datos, se resta a cada señal su valor mínimo, obteniendo así datos con un nivel de referencia nulo como sigue:

```
offset.input = min(vop_all);
offset.output = min(pv_smooth);
% remove offset
plant.in = plant.in - offset.input;
plant.out = plant.out - offset.output;
plant.in_all = plant.in_all - offset.input;
plant.out_all = plant.out_all -
offset.output;
```

Las señales sin offset se pueden apreciar en la Fig. 5. Hasta este punto el vector de tiempo de los datos no es equi-espaciado, lo cual dificulta el uso de algunas funciones en análisis posteriores; por esta razón se realizó una interpolación de los datos sobre un vector de tiempo equi-espaciado generado a partir de la frecuencia de muestreo promedio entre todos los segundos y que va desde cero hasta el último segundo del vector de tiempo original. La interpolación se realiza de la siguiente manera:

```
% make time uniform by interpolation
% mean sampling frequency was 2.2 samples/sec
time.fs = 2.2;
time.Ts = 1/time.fs;
time.ending = 600;
interpolated.time =
min(time.t):time.Ts:max(time.t);
interpolated.out =
interp1(time.t,plant.out,interpolated.time);
interpolated.in =
interp1(time.t,plant.in,interpolated.time);
interpolated.time_all =
min(time.all):time.Ts:max(time.all);
interpolated.out_all =
interp1(time.all,plant.out_all,interpolated.ti
me_all);
interpolated.in_all =
interp1(time.all,plant.in_all,interpolated.tim
e_all);
interpolated.time_ext =
min(time.all):time.Ts:time.ending;
```

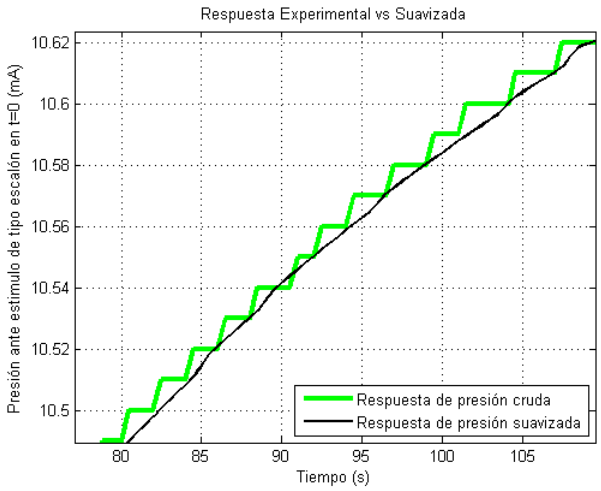

Fig 2. Saltos discontinuos en la señal "cruda".

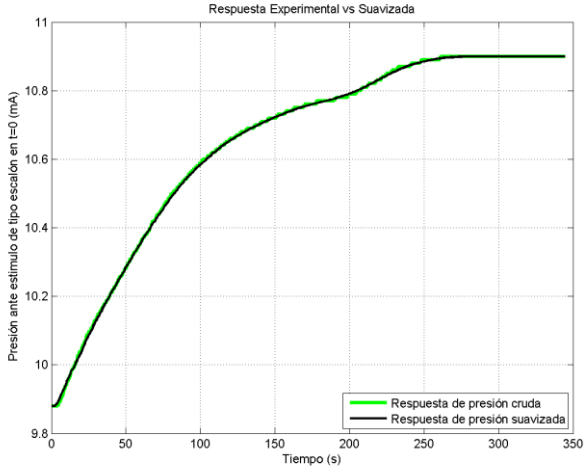

Fig 3. Comparación señal "cruda" de salida con señal suavizada de salida.

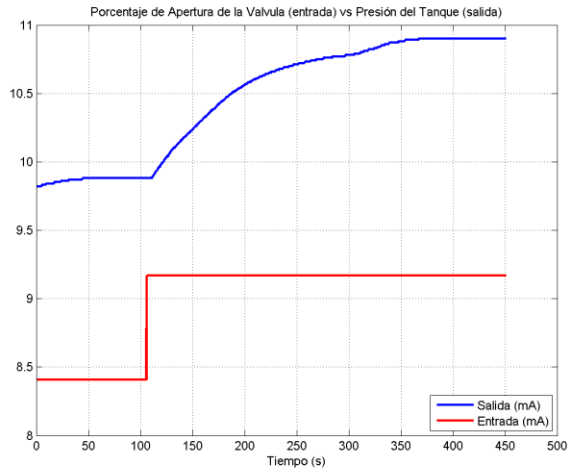

Fig 4. Señales de entrada y salida depuradas con Matlab.

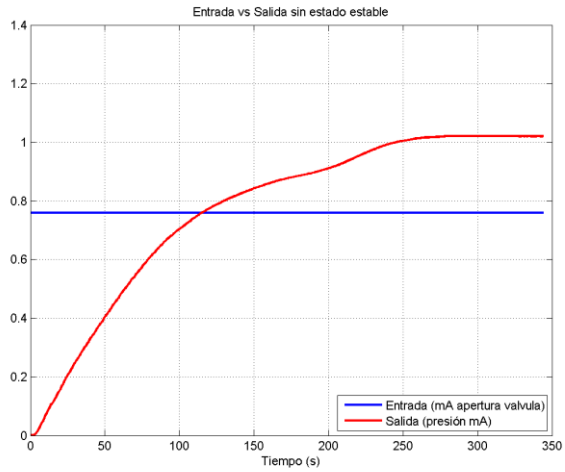

Fig 5. Señales de entrada y salida sin offset.

IV. IDENTIFICACIÓN DE MODELOS DE PRIMER ORDEN

Los modelos de primer orden, no son muy recomendados ya que no capturan el comportamiento de mayor complejidad de los sistemas reales, sin embargo son sencillos por lo que si no se necesita mayor exactitud pueden resultar más apropiados. La función de transferencia que caracteriza a los modelos de 1er orden son de la siguiente forma:

$$G(s) = \frac{k_p\, e^{-t_m s}}{\tau\, s + 1} \quad (\mathbf{1})$$

Note en (1) que siempre deben hallarse tres parámetros: Kp, t_m y τ. La Kp es asociada con la ganancia estática del sistema, t_m es el tiempo muerto, al cual asocian con la cantidad de tiempo que le llevó responder al sistema antes de alterar su salida ante el cambio en la entrada y finalmente como τ la constante de tiempo aparente del sistema.

Primero se hará el cálculo de la ganancia Kp, ya que todos los modelos presentados tienen la misma definición para ella. La ganancia será el cambio total en la salida divida por el cambio total de la entrada.

$$Kp = \frac{\Delta Y}{\Delta U} = 1.34 \quad (\mathbf{2})$$

En código se encuentra primero cuando la señal se estabiliza usando la siguiente función:

```
function [stableTime,stableVal,stableIndex] =
findStablePoint(time,signal,timeThresh,frac)
    tol = frac*(max(signal) - min(signal));
    for i = 1:length(signal)
        for j = i:length(signal)
            delta = signal(j) - signal(i);
            if abs(delta) >= tol
                break
            end
            if time(j)-time(i)>= timeThresh
                stableVal = signal(j);
                stableTime = time(j);
                stableIndex = j;
                return
            end
        end
    end
    stableVal = false;
    stableTime = false;
    stableIndex = false;
end
```

Así la ganancia se calcula de la siguiente forma:

```
% taking into account perturbation
[plant.stableTime,plant.stableVal,plant.stableI
ndex] =
findStablePoint(time.t,plant.out,40,0.005);
 % find gain (notice that offset was removed so
no need to do a delta)
plant.kp =
plant.out(plant.stableIndex)/plant.in(plant.sta
bleIndex);
```

A. Método de la tangente de Ziegler-Nichols

El método usa en primer lugar una recta, que se traza en la gráfica experimental ya filtrada. Esta recta es la recta tangente al punto de máxima pendiente, que presente la señal de respuesta. Para encontrarlo se deriva el vector de la respuesta numéricamente, mediante la función diff y se ubica el valor máximo.

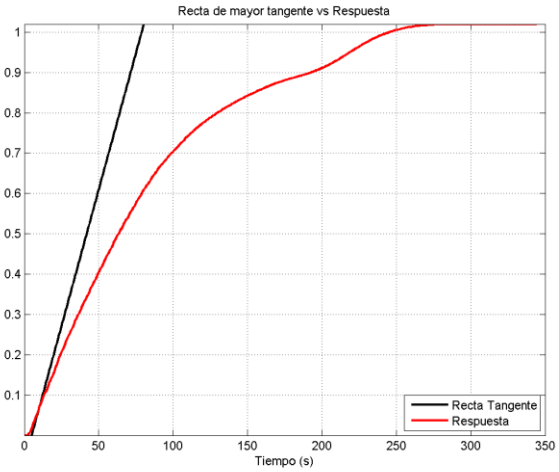

Fig 6. Recta tangente.

Conocido el valor de la máxima pendiente y el tiempo en que se da, solo es necesario evaluar la amplitud en dicho tiempo en la señal de respuesta para crear la ecuación de la recta presentada en (3).

$$y = m_{max}(t - t_{max}) + y_{max}$$

Esta ecuación se puede reducir a la forma:

$$y = m_{max}t + b \qquad (3)$$

Donde $b = -m_{max}t_{max} + y_{max}$

Se entiende que las coordenadas 'max' son las del punto de máxima pendiente. En la Fig. 6 puede observar la tangente resultante, de la cual el método hará uso para obtener ciertos valores necesarios.

Por las facilidades que se han realizado con las señales al comenzar desde cero cuando el escalón cambia, se puede decir que el tiempo muerto en este método se halla buscando el tiempo en que la recta tangente corta con el eje temporal. En (3) se despeja t y 'y' se iguala a cero, así se obtiene (4) para hallar con exactitud el valor.

$$t_m = t_{max} - \frac{y_{max}}{m_{max}} \qquad (4)$$

Para hallar la constante de tiempo del sistema con este método solo basta con hallar el instante en que la recta tangente es igual al valor máximo de la señal de salida y restarle el tiempo muerto. Haciendo despejes similares a lo anterior en
(3) se puede obtener (5):

$$\tau = \frac{\max(y) - y_{max}}{m_{max}} + t_{max} - t_m \qquad (5)$$

Nota: Es importante recordar que las definiciones aquí planteadas son posibles gracias a que se corrieron las señales hasta el momento en que el escalón cambia, sino se deberá considerar en las ecuaciones que debe restarse el tiempo que tarda el escalón en cambiar.

Como resultado se obtiene la siguiente función de transferencia:

$$G(s) = \frac{1.34\, e^{-5.05\, s}}{75.4\, s + 1} \qquad (6)$$

Este método se implementa mediante el siguiente código:

```
% Method: Ziegler Nichols
zn.name = 'Ziegler - Nichols';
zn.slope = diff(plant.out)./diff(time.t);
[zn.maxSlope,zn.maxSlope_index] =
max(zn.slope);
 zn.m = zn.maxSlope;
zn.b = plant.out(zn.maxSlope_index) -
zn.m*time.t(zn.maxSlope_index);
zn.tangent = zn.m * time.t + zn.b;

% tm : time between stimuli start and cut
of the x axis by the tangent
zn.y = 0;
zn.tm = (zn.y-zn.b)/zn.m;
```

```
% tau : time between tm and where the
tangent reaches the final value of
% the response
zn.y = plant.out(plant.stableIndex);
zn.tau = (zn.y-zn.b)/zn.m - zn.tm;
% Create the system
s = tf('s');
zn.sys = (plant.kp/(zn.tau*s + 1))*exp(-
zn.tm*s);
zn.out = step(zn.sys,time.ext)';
```

B. *Método de la tangente modificada de Miller*

El tiempo muerto en este método, se halla igual que en el de Ziegler-Nichols, por lo tanto se mantiene el mismo resultado hallado en el anterior.

El valor que cambia es la constante de tiempo τ, debido a que Miller propone corregir el error de estimación dado en Ziegler-Nichols. Miller calcula la constante τ en el tiempo donde la respuesta alcanza el 63,2% de su valor estable y a este valor se le resta el tiempo muerto. Partiendo de la ecuación de la recta tangente de Ziegler-Nichols se obtiene la siguiente ecuación:

$$\tau = \frac{y_{EE} * (0.632) - b}{m_{max} - t_m} \qquad (7)$$

La expresión real usada es $1 - e^{-1}$, la cual aproximada da 63,2%, pero como Matlab puede trabajar con varios decimales se optó por dejar la expresión para ser más exacto en el valor.

Como resultado se obtiene la siguiente función de transferencia:

$$G(s) = \frac{1.34\, e^{-5.05\, s}}{47.7\, s + 1} \qquad (8)$$

Este método se implementa mediante el siguiente código:

```
% Method: Miller Modification
miller.name = 'Miller';
miller.tm = zn.tm;
miller.y = plant.out(plant.stableIndex)*(1-
exp(-1));
miller.tau = (miller.y - zn.b)/zn.m -
miller.tm;
miller.sys = (plant.kp/(miller.tau*s +
1))*exp(-miller.tm*s);
miller.out = step(miller.sys,time.ext)';
```

Método general de dos puntos para identificar 1er orden más tiempo muerto.

Los siguientes 3 métodos se logran basados en obtener dos ecuaciones con dos incógnitas utilizando dos puntos sobre la curva de reacción. De este modo se garantiza que la respuesta del modelo coincida con la del sistema real en estos dos puntos como mínimo. Es posible establecer ecuaciones generales para dicho método:

$$\tau = a\, t_{1x} + b\, t_{2y} \qquad (9)$$

$$t_{m=} c\, t_{1x} + d\, t_{2y} \qquad (10)$$

Las constantes a, b, c, d están definidas para cada método. Por otro lado t_{1x} corresponde al instante donde la salida alcanza x% de su valor en estado estable. De igual manera se hace t_{2y}.

Al ser baja la probabilidad de encontrar los valores exactos por tener un conjunto de datos, simplemente se verifican cada uno de los valores de la respuesta del sistema, de forma que se encuentre el índice de tiempo en el cual la función alcanzó la amplitud más cercana a la esperada.

Dicho método de forma general se implementa con la siguiente función:

```
function [tm,tau] =
model2points(time,signal,stableVal,p1,p2,a,b,c
,d)
    [~,t1] = findClosest(signal,stableVal*p1);
    t1 = time(t1);
    [~,t2] = findClosest(signal,stableVal*p2);
    t2 = time(t2);

    tau = abs(a*t1 + b*t2);
    tm = abs(c*t1 + d*t2);

end
```

La function findClosest( ) se encarga de encontrar en el vector el valor más cercano al pedido, y se implementa como sigue:

```
function [closest,ix] = findClosest(x,val)
[ d, ix ]  = min( abs( x-val ) );
closest = x(ix);
end
```

C. *Método de Smith*

Este método corresponde a los instantes donde la respuesta alcanza el 63,2% y el 28,3% de su valor de estado estable. Con a = -1.5, b = 1.5, c = 1.5, d =-0.5.

Como resultado se obtiene la siguiente función de transferencia:

$$G(s) = \frac{1.34\, e^{-9.09\, s}}{77.7\, s + 1} \qquad (11)$$

Se puede implementar este método de la siguiente forma:

```
% Method: Smith
smith.name = 'Smith';
[~,time.t28] =
findClosest(plant.out,plant.out(plant.stableIn
dex)*0.283);
time.t28 = time.t(time.t28);
[~,time.t63] =
findClosest(plant.out,plant.out(plant.stableIn
dex)*0.632);
time.t63 = time.t(time.t63);
smith.tau = 1.5*(time.t63-time.t28);
smith.tm = time.t63 - smith.tau;

smith.sys = (plant.kp/(smith.tau*s + 1))*exp(-
smith.tm*s);
smith.out = step(smith.sys,time.ext)';
```

D. Método de "1/4 - 3/4" de Alfaro

Este método corresponde a los instantes donde la respuesta alcanza el 25% y el 75% de su valor de estado estable. Con a = -0.91, b = 0.91, c = 1.262, d = -0.262.

Como resultado se obtiene la siguiente función de transferencia:

$$G(s) = \frac{1.34\, e^{-8.4\, s}}{78.2\, s + 1} \qquad (\mathbf{12})$$

Con la función general model2points( ) hecha se puede implementar este método como sigue:

```
% Method: Alfaro
alfaro.name = 'Alfaro 1/4 3/4';
m2p.p1 = 0.25;
m2p.p2 = 0.75;
m2p.a = -0.91;
m2p.b = -1*m2p.a;
m2p.c = 1.262;
m2p.d = -1*(m2p.c - 1);
[alfaro.tm,alfaro.tau] =
model2points(time.t,plant.out,plant.out(plant.sta
bleIndex),m2p.p1,m2p.p2,m2p.a,m2p.b,m2p.c,m2p.d);
alfaro.sys = (plant.kp/(alfaro.tau*s + 1))*exp(-
alfaro.tm*s);
alfaro.out = step(alfaro.sys,time.ext)';
```

E. Método de Ho et al.

Este método corresponde a los instantes donde la respuesta alcanza el 35% y el 85% de su valor de estado estable. Con a = -0.67, b = 0.67, c = 1.3, d = -0.29.

Como resultado se obtiene la siguiente función de transferencia:

$$G(s) = \frac{1.34\, e^{-9.01\, s}}{81\, s + 1} \qquad (\mathbf{13})$$

Con la función general model2points( ) hecha se puede implementar este método como sigue:

```
% Method Ho et al
ho.name = 'Ho et al';
m2p.p1 = 0.35;
m2p.p2 = 0.85;
m2p.a = -0.670;
m2p.b = -1*m2p.a;
m2p.c = 1.300;
m2p.d = -0.290;
[ho.tm,ho.tau] =
model2points(time.t,plant.out,plant.out(plant.sta
bleIndex),m2p.p1,m2p.p2,m2p.a,m2p.b,m2p.c,m2p.d);
ho.sys = (plant.kp/(ho.tau*s + 1))*exp(-ho.tm*s);
ho.out = step(ho.sys,time.ext)';
```

V. IDENTIFICACIÓN DE MODELOS DE SEGUNDO ORDEN

Los modelos de segundo orden, que se trabajaron en esta práctica, se caracterizan por la función de transferencia que se observa a continuación:

$$G(s) = \frac{Kp * e^{-t_m s}}{(\tau_1 s + 1)(\tau_2 s + 1)} \qquad (\mathbf{14})$$

Note que (14) presenta una sola variación con respecto a (1), donde el polinomio característico es de grado dos, lo que implica una constante de tiempo adicional denominada τ_2. Este parámetro extra, aumenta un poco la complejidad del modelado de los diferentes métodos con respecto a los de primer orden; aunque sigue siendo muy simple en comparación a un caso analítico. De esta forma, los modelos de segundo orden aquí presentados necesitan de cuatro parámetros para poder sacar la función de transferencia. La ganancia Kp para estos modelos sigue siendo la misma que se halló en (2), ya que es una constante en el sistema; así que solo se enfocará en encontrar el tiempo muerto y las dos constantes de tiempo según lo indique el método seleccionado.

A. Método Alfaro general 123c

Alfaro realizó una extensión del método *"1/4 - 3/4"*, al añadir un punto más de coincidencia entre la respuesta del modelo y la respuesta real del sistema, permitiendo esto disminuir el margen de error generado entre ambos. Este punto se ubica en el instante donde la respuesta alcanza el 50% de su valor de estado estable

Al ser un método de segundo orden, se requieren hallar las dos constantes de tiempo que en este caso vienen dadas por:

$$\tau_1 = \tau'' \qquad (\mathbf{15})$$

$$\tau_2 = a\tau'' \qquad (\mathbf{16})$$

Donde los valores de a y τ'' vienen dados por:

$$a = \frac{-0.6240 t_{25} + 0.9866 t_{50} - 0.3626 t_{75}}{0.3533 t_{25} - 0.7036 t_{50} + 0.3503 t_{75}} \qquad (\mathbf{17})$$

$$\tau'' = \frac{t_{75} - t_{25}}{0.9866 + 0.7036 a} \qquad (\mathbf{18})$$

Y el tiempo muerto t_m viene dado por :

$$t_m = t_{75} - (1.3421 + 1.3455 a)\tau'' \qquad (\mathbf{19})$$

Como resultado se obtiene la siguiente función de transferencia:

$$G(s) = \frac{1.34\, e^{-10.71\, s}}{(67.5\, s + 1)(27.4\, s + 1)} \qquad (\mathbf{20})$$

Este método se puede implementar como sigue:

```
% Method: Alfaro General 123c
g123c.name = 'Alfaro General 123c';
[~,time.t25] =
findClosest(plant.out,plant.out(plant.stableIn
dex)*0.25);
time.t25 = time.t(time.t25);
[~,time.t50] =
findClosest(plant.out,plant.out(plant.stableIn
dex)*0.5);
```

```
time.t50 = time.t(time.t50);
[~,time.t75] =
findClosest(plant.out,plant.out(plant.stableIn
dex)*0.75);
time.t75 = time.t(time.t75);
g123c.a = (-0.6240*time.t25 + 0.9866*time.t50
- 0.3626*time.t75)/(0.3533*time.t25 -
0.7036*time.t50 + 0.3503*time.t75);
g123c.a = abs(g123c.a);
g123c.tauPP = (time.t75-
time.t25)/(0.9866+0.7036*g123c.a);
g123c.tauPP = abs(g123c.tauPP);
g123c.tau1 = g123c.tauPP;
g123c.tau2 = g123c.a*g123c.tauPP;
g123c.tm = time.t75 -
(1.3421+1.3455*g123c.a)*g123c.tauPP;
g123c.tm = abs(g123c.tm);
g123c.sys = (plant.kp * exp(-
g123c.tm*s))/((g123c.tau1*s+1)*(g123c.tau2*s+1
));
g123c.out = step(g123c.sys,time.ext)';
```

B. Método "simétrico"

Este método recoge igualmente la información de 3 instantes de tiempo según las amplitudes en la salida del sistema. La diferencia con el de *123c* se debe a que dichos puntos no están fijos, sino que pueden variar. El único tiempo que mantienen es aquel en que la magnitud alcanza el 50% de su valor de estado estable, puesto que luego se va a elegir un tiempo *x* entre el 0% y el 50% de la amplitud final de salida. La simetría se presenta cuando el tercer tiempo elegido será entonces aquel que posea dicho porcentaje igual a el 100% menos el porcentaje elegido para el tiempo *x*, es decir 1- x.

La elección de dicho porcentaje para obtener el tiempo *x* es que la predicción de la sumatoria del error cuadrático debe ser minimizada. De esta manera, era necesario hacer en Matlab una función que encontrara el rango donde según el porcentaje elegido para el tiempo *x* se obtuvieran los errores mínimos. Una vez se obtiene el rango, que en esencia debe ser de diferencia aproximadamente decimal, se encontraba el mejor valor de forma iterativa.

En vista de que los parámetros de este método se calculan con la misma estructura del método general *123c* entonces no se volverá a repetir las ecuaciones. Simplemente se dirá que se tiene el tiempo t_x y el tiempo t_{1-x}, en donde las ecuaciones (17), (18) y (19) se cambian el t_{75} por t_{1-x}, y el t_{25} por t_x.

Como resultado se obtiene la siguiente función de transferencia:

$$G(s) = \frac{1.34 * e^{-2.61s}}{(39.54\,s + 1)(39.52\,s + 1)} \qquad (21)$$

Este método se puede implementar como sigue en la siguiente función:

```
function [tau1,tau2,tm,x] =
symmetricModel(numberOfTests,plant_kp,plant_in,
plant_out,plant_stableIndex,time,start)
% start 0 < x < 0.5
% start 1e-3 works, realmin does not, nor 1e-10
x = linspace(start,0.5-start,numberOfTests);
errors = zeros(length(x),1);
s = tf('s');
taus1 = zeros(length(x),1);
taus2 = zeros(length(x),1);
tms = zeros(length(x),1);
for i = 1:length(x)
[~,tx] =
findClosest(plant_out,plant_out(plant_stableIn
dex)*x(i));
tx = time(tx);
[~,t50] =
findClosest(plant_out,plant_out(plant_stableIn
dex)*0.5);
t50 = time(t50);
[~,t1mx] =
findClosest(plant_out,plant_out(plant_stableIn
dex)*(1-x(i)));
t1mx = time(t1mx);
a = (-0.6240*tx + 0.9866*t50 -
0.3626*t1mx)/(0.3533*tx - 0.7036*t50 +
0.3503*t1mx);
a = abs(a);
tauPP = (t1mx-tx)/(0.9866+0.7036*a);
tauPP = abs(tauPP);
tau1 = tauPP;
tau2 = a*tauPP;
tm = t1mx - (1.3421+1.3455*a)*tauPP;
tm = abs(tm);
sys = (plant_kp * exp(-
tm*s))/((tau1*s+1)*(tau2*s+1));
sys_out = step(sys,time)';
errors(i) =
immse(sys_out,plant_out/max(plant_in));
taus1(i) = tau1;
taus2(i) = tau2;
tms(i) = tm;
end

[~,index] = min(errors);

tau1 = taus1(index);
tau2 = taus2(index);
tm = tms(index);
x = x(index);

end
```

Así el modelo usado se consigue de la siguiente forma:

```
% Method : Simetrico
sym.name = 'Simetrico';
sym.numberOfTests = 200;
[sym.tau1,sym.tau2,sym.tm,sym.x] =
symmetricModel(sym.numberOfTests,plant.kp,plan
t.in,plant.out,plant.stableIndex,time.t,1e-3);
sym.sys = (plant.kp * exp(-
sym.tm*s))/((sym.tau1*s+1)*(sym.tau2*s+1));
sym.out = step(sym.sys,time.ext)';
```

C. Método de Stark

Es un método de tres puntos, los cuales corresponden a los tiempos donde la respuesta del sistema alcance el 75%, 45% y 15% de su valor de estado estable. Para encontrar los parámetros que se están buscando de (14) se debe operar en el siguiente orden que se muestra:

$$\eta = \frac{t_{45} - t_{15}}{t_{75} - t_{15}} \qquad (22)$$

$$\varsigma = \frac{0.0805 - 5.547(0.475 - x)^2}{x - 0.356} \quad (23)$$

$$f_2(\varsigma) = 2.6\varsigma - 0.6 \quad (24)$$

$$\omega_n = \frac{f_2(\zeta)}{t_{75} - t_{15}} \quad (25)$$

$$f_3(\zeta) = 0.922(1.66)^{\varsigma} \quad (26)$$

$$t_m = t_{45} - \frac{f_3(\zeta)}{\omega_n} \quad (27)$$

$$\tau_1 = \frac{\zeta + \sqrt{\zeta^2 - 1}}{\omega_n} \quad (28)$$

$$\tau_2 = \frac{\zeta - \sqrt{\zeta^2 - 1}}{\omega_n} \quad (29)$$

Nota: Este procedimiento sirve porque el sistema trabajado entrega una respuesta del tipo sobreamortiguada, por lo que ζ será mayor que 1 (en particular 1.17). Los parámetros ς y ω_n se asocian con lo que se conoce como factor de amortiguamiento y frecuencia natural de oscilación del sistema.

Como resultado se obtiene la siguiente función de transferencia:

$$G(s) = \frac{1.34 * e^{-8.69\,s}}{(70.8\,s + 1)(22.4\,s + 1)} \quad (30)$$

Este método también permite hallar modelos a curvas del tipo subamortiguado haciendo algunos cambios en el procedimiento mostrado anteriormente.

El modelo se puede implementar como sigue:

```
% Method: Stark
[~,time.t15] =
findClosest(plant.out,plant.out(plant.stableInd
ex)*0.15);
time.t15 = time.t(time.t15);
[~,time.t45] =
findClosest(plant.out,plant.out(plant.stableInd
ex)*0.45);
time.t45 = time.t(time.t45);
[~,time.t75] =
findClosest(plant.out,plant.out(plant.stableInd
ex)*0.75);
time.t75 = time.t(time.t75);
stark.name = 'Stark';
stark.x = (time.t45-time.t15)/(time.t75-
time.t15);
stark.damp = (0.0805-5.547*(0.475-
stark.x)^2)/(stark.x-0.356);
if stark.damp <= 1
    stark.f2 = 0.708*(2.811)^stark.damp;
else
    stark.f2 = 2.6*stark.damp -0.6;
end

stark.wn = stark.f2/(time.t75-time.t15);
stark.f3 = 0.922 * 1.66^(stark.damp);
stark.tm = time.t45 - stark.f3/stark.wn;
stark.tm = abs(stark.tm);
stark.tau1 = (stark.damp + sqrt(stark.damp^2
- 1))/(stark.wn);
stark.tau2 = (stark.damp - sqrt(stark.damp^2
- 1))/(stark.wn);

stark.sys = (plant.kp * exp(-
stark.tm*s))/((stark.tau1*s+1)*(stark.tau2*s+
1));
stark.out = step(stark.sys,time.ext)';
```

D. Método de Jahanmiri- Fallahi – Polo Doble

Este método está basando en los tiempos para alcanzar el 2% (t_2) o el 5% (t_5), el 70% (t_{70}) y 90% (t_{90}) del valor de estado estable. Para elegir los valores entre el 2% y el 5% se realizó una función en Matlab, que se encarga de analizar cual valor es más óptimo, es decir, que brinda un menor índice aproximado de error y así determinar el tiempo muerto (t_m).

Las ecuaciones que se usan para identificar el modelo son:

$$\eta = \frac{t_{90} - t_{70}}{t_{90} - t_m} \quad (31)$$

si $\eta \leq 0.4771$ entonces:

$$\varsigma = \sqrt{\frac{0.4844651 - 0.75323499\eta}{1 - 2.0946444\eta}} \quad (32)$$

Si $\eta \geq 0.4771$ entonces:

$$\varsigma = 13.9352$$

$$\tau = \frac{t_{90} - t_m}{0.424301 + 4.62533\varsigma - 2.65412e^{-\varsigma}} \quad (33)$$

Como resultado se obtiene la siguiente función de transferencia:

$$G(s) = \frac{1.34 * e^{-5.45\,s}}{(3.1\,s + 1)^2} \quad (34)$$

Así este método se puede implementar como sigue:

```
% Method: Jahanmiri - Fallahi
jf.name = 'Jahanmiri - Fallahi';
[~,time.t2] =
findClosest(plant.out,plant.out(plant.stableIn
dex)*0.02);
time.t2 = time.t(time.t2);
[~,time.t5] =
findClosest(plant.out,plant.out(plant.stableIn
dex)*0.05);
time.t5 = time.t(time.t5);
[~,time.t90] =
findClosest(plant.out,plant.out(plant.stableIn
dex)*0.9);
time.t90 = time.t(time.t90);
[~,time.t70] =
findClosest(plant.out,plant.out(plant.stableIn
dex)*0.7);
time.t70 = time.t(time.t70);
```

```
jf.tms = [time.t2;time.t5];
jf.errors = zeros(length(jf.tms),1);

for aux_i = 1:length(jf.tms)
    [jf.out,~] =
jf_model(time.t,plant.kp,jf.tms(aux_i),time.t7
0,time.t90);
    jf.errors(aux_i) =
immse(jf.out,plant.out/max(plant.in));
end

[~,jf.index] = min(jf.errors);
jf.tm = jf.tms(jf.index);
[jf.out,jf.sys,jf.tau] =
jf_model(time.ext,plant.kp,jf.tm,time.t70,time
.t90);
```

E. Ho et al – Polo doble

Este método se calcula de la misma manera que el de Ho et al de primer orden, pero con la diferencia de que el polo de la función de transferencia es doble, el cual permite aumentar la estabilidad del sistema.

Como resultado se obtiene la siguiente función de transferencia:

$$G(s) = \frac{1.34 * e^{-25.77\,s}}{(56\,s+1)^2} \qquad (35)$$

Este método se puede implementar como sigue:

```
% Method: Ho et al - Polo Doble
ho2.name = 'Ho et al - Polo Doble';
m2p.p1 = 0.35;
m2p.p2 = 0.85;
m2p.a = -0.463;
m2p.b = -1*m2p.a;
m2p.c = 1.574;
m2p.d = -1*(m2p.c - 1);
[ho2.tm,ho2.tau] =
model2points(time.t,plant.out,plant.out(plant.s
tableIndex),m2p.p1,m2p.p2,m2p.a,m2p.b,m2p.c,m2p
.d);
ho2.sys = (plant.kp/((ho2.tau*s + 1)^2))*exp(-
ho2.tm*s);
ho2.out = step(ho2.sys,time.ext)';
```

Modelos de Matlab

Los modelos estimados por Matlab son obtenidos gracias a una herramienta llamada System *Identification Tool*, con la cual es posible encontrar nuevas funciones de transferencia a partir de los datos experimentales. Las funciones de transferencia conseguidas de primer y segundo orden respectivamente, con esta herramienta fueron:

$$G(s) = \frac{1.4\,e^{-3.64\,s}}{92.5\,s+1} \qquad (36)$$

$$G(s) = \frac{1.39\,e^{-1.36\,s}}{(4.1\,s+1)(89.2\,s+1)} \qquad (37)$$

Note que (36) y (37) también manejan el estilo de función de transferencia que se trabajó con los modelos de primer y segundo orden. Además, los parámetros de ganancia, tiempo muerto y constantes de tiempo obtienen valores muy similares a algunos modelos que se realizaron.

Dicha identificación de matlab se implementa de la siguiente forma:

```
% Matlab Identification 1st Order
id1.name = 'Matlab 1er Orden';
id1.data =
iddata(plant.out_all',plant.in_all',time.Ts);
%id1.data =
iddata(plant.out',plant.in',time.Ts);
% sys = tfest(data,np,nz,iodelay); if NaN
estimate
id1.sys = tfest(id1.data,1,0,NaN);
id1.out = step(id1.sys,time.ext)';
id1.tau = get_taus(id1.sys.den);
id1.kp = id1.sys.num;
for aux_i = 1:length(id1.tau)
id1.kp = id1.kp * id1.tau(aux_i);
end
id1.tm = id1.sys.ioDelay;

% Matlab Identification 2nd Order
id2.name = 'Matlab 2do Orden';
id2.data =
iddata(plant.out_all',plant.in_all',time.Ts);
%id2.data =
iddata(plant.out',plant.in',time.Ts);
%sys = tfest(data,np,nz,iodelay); if NaN
estimate
id2.sys = tfest(id2.data,2,0,NaN);
id2.out = step(id2.sys,time.ext)';
id2.taus = get_taus(id2.sys.den);
id2.kp = id2.sys.num;
for aux_i = 1:length(id2.taus)
id2.kp = id2.kp * id2.taus(aux_i);
end
id2.tm = id2.sys.ioDelay;
```

VI. ANÁLISIS

En las figuras 7 y 8 se logran observar los resultados obtenidos de los modelos al excitarse con un escalón unitario. Además se incluye la respuesta del sistema ante un escalón unitario obtenida a partir de dividir la respuesta original por la magnitud del escalón original. Visualmente se logra apreciar unos muy buenos resultados, con curvas muy similares a la respuesta que sacó el sistema.

A. Selección de modelos

Para la evaluación y selección de un modelo se recurrió a dos criterios, uno visual y otro cuantitativo. La inspección visual permite ver las características generales de cada modelo de forma simultánea con los demás. Por otra parte, el criterio cuantitativo premia aquel modelo cuyo comportamiento sea más cercano al experimental, es decir, que sus amplitudes para cada valor de tiempo del modelo y del resultado experimental sean más cercanas a la respuesta real. Dicho proceso se logra comparando mediante el error cuadrático medio, cuyo error corresponde a la suma de los cuadrados de la distancia entre los valores del modelo y el experimental. Este error al ser cuadrático les otorga más peso a los valores que estén más alejados, teniendo un criterio

mayor para determinar el error asociado de cada modelo.

$$Error = \frac{\sum_{K=1}^{N}(e_k)^2}{N} \quad \textbf{(38)}$$

Para hacer esto, primero se toman los vectores que representan las amplitudes de la respuesta experimental y del modelo en el tiempo, en una función en Matlab. Estos dos vectores se restan con el fin de encontrar la distancia entre los valores para cada instante de tiempo. Luego se elevan los valores al cuadrado y después se suman todos los valores de error para cada valor de tiempo. Finalmente se divide entre el número total de muestras o valores y se obtiene el error cuadrático como se sintetiza en Ec (38), donde *ek* corresponde al error en cada instante siendo la resta entre el valor del modelo y el valor de la respuesta experimental y N es el número de muestras. Este valor corresponde a un número diferente para cada modelo de primer y segundo orden.

Así el error cuadrático medio se implementa como sigue:

```
function err = immse(x, y)
if isinteger(x)
    x = double(x);
    y = double(y);
end
err = (norm(x(:)-y(:),2).^2)/numel(x);
```

La comparación entre los modelos de 1er orden se logra entonces como:

```
% MSE for 1st order models
models_1.names =
{alfaro.name,broida.name,cy.name,ho.name,miller
.name,smith.name,viteckova.name,zn.name,id1.nam
e};
models_1.outs=
[alfaro.out;broida.out;cy.out;ho.out;miller.out
;smith.out;viteckova.out;zn.out;id1.out];
models_1.number = 9;
models_1.MSE = zeros(models_1.number,1);
%remember to compare against scaled out for
unit step
for aux_i = 1:models_1.number
    models_1.MSE(aux_i) =
immse(models_1.outs(aux_i,:),plant.out_ext/max(
plant.in));
end
[models_1.error,models_1.idx] =
min(models_1.MSE);
models_1.best = models_1.names(models_1.idx);
```

De forma similar se aplica en los modelos de 2do orden.

En la figura 7 se puede apreciar la comparación de los modelos de 1er orden; en ella se observa que los únicos modelos que se desvían mucho con respecto a la respuesta real del sistema son el de Miller y el de Matlab de 1er orden. Mientras que en el primero antes mencionado su respuesta se desvía de la respuesta transitoria experimental, en el otro se desvía en cuánto a la respuesta experimental de estado estable. De resto, visualmente todas las curvas se aprecian como muy buenas opciones. Siendo más detallistas, la curva de Ziegler-Nichols se despega también un poco en la respuesta transitoria. Esto deja tres modelos (Smith, Alfaro y Ho et al) entre los cuáles es muy difícil elegir por inspección visual. Por ello se corroborará dicha información a partir del Error Cuadrático Medio por medio de la siguiente tabla.

Métodos 1er Orden	ECM
Ziegler-Nichols	0.00086177
Miller	0.01470748
Smith	0.00040725
"1/4 – 3/4" de Alfaro	0.00039507
Ho et al.	0.00048736
Matlab 1er Orden	0.00098199

Tabla 1. ECM de 2 orden

Vemos entonces que el mejor modelo de 1er orden corresponde al de Alfaro "1/4 – 3/4".

En la figura 8 podemos observar la comparación entre modelos de 2do orden para facilitar así la selección del modelo que más se ajusta a la curva real. De esta manera se puede descartar rápidamente el modelo de Polo doble de Ho et al. Así mismo, podemos descartar el método 123c de Alfaro en favor del de Stark ya que es muy semejante pero con ligeramente mejores resultados. El modelo de segundo orden identificado por Matlab presenta nuevamente una sobre-estimación en la parte estable de la respuesta. Esto nos deja con 3 modelos posibles: Simétrico, Jahanmiri-Fallahi y el de Stark. A pesar de que es difícil decidir entre el Simétrico y el de Stark, el método de Jahanmiri-Fallahi se aproxima más a la curva que esos otros dos.

Se proseguirá ahora con el análisis cuantitativo de los modelos de la misma forma como se hizo con los de 1er orden por medio del Error Cuadrático Medio, mostrado en la siguiente tabla.

Métodos 2do Orden	ECM
Alfaro General 123c	0.00744777
Simétrico	0.00207254
Stark	0.00491352
Jahanmiri - Fallahi	0.00055585
Ho et al - Polo Doble	0.03667915
Matlab 2do Orden	0.00074361

Tabla 2. ECM de 2 orden

En la tabla 2 se puede observar que el menor ECM se presenta en el modelo de Jahanmiri-Fallahi, sin embargo este método proporciona un modelo de polo doble. El siguiente mejor valor lo brinda el modelo dado por el método simétrico, pero la función de transferencia correspondiente tiene la particularidad de poseer proximidad entre los polos al variar en pocas décimas entre sí, por lo que podría considerarse que el efecto de estos dos polos es similar al de un polo doble. Debido a que la guía sugiere elegir un modelo de polos diferentes se opta por tomar el modelo hallado por medio del método de Stark.

Al comparar los modelos de 1er y 2do orden mediante ambas tablas se puede observar que el modelo de 1er orden de Alfaro "1/4-3/4" tiene el menor error entre todos los métodos explorados.

VII. FIGURAS

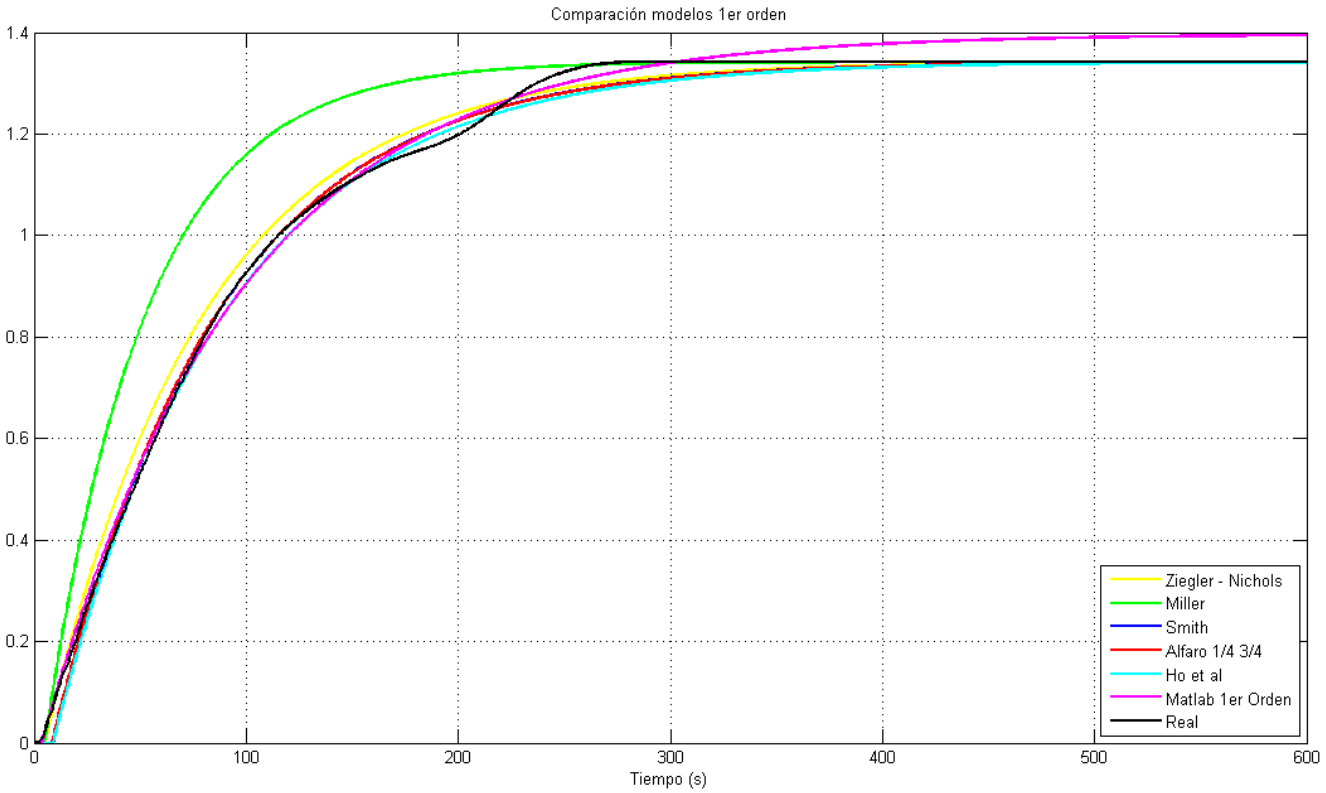

Fig 7. Modelos de primer orden descritos comparados con la respuesta real del sistema.

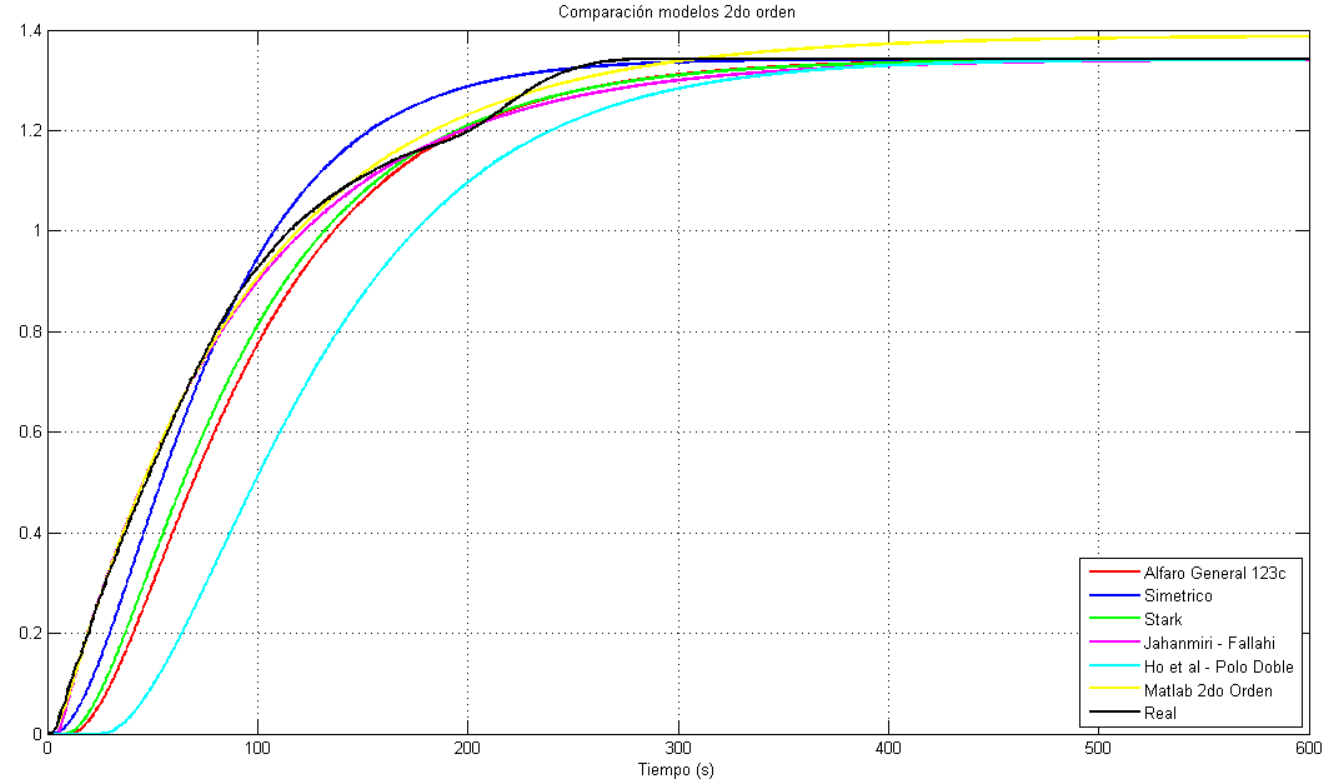

Fig 8. Modelos de segundo orden descritos comparados con la respuesta real del sistema.

VIII. CONCLUSIONES

1. Una primera conclusión no tan intuitiva es el hecho de que un mayor orden en los modelos no garantiza una mejor aproximación al sistema real. De igual manera, ciertos métodos presentan una mayor complejidad matemática en sus procedimientos (métodos iterativos, etc), pero estos no necesariamente se ajustan mejor a la curva real. Se puede observar en los resultados que, algunos modelos de primer orden realizan una curva más similar a la respuesta del sistema que modelos de segundo orden, no solo visualmente, sino también en el análisis matemático donde algunos obtienen menos error que los de segundo orden. De manera que, a la hora de identificar modelos para el sistema es recomendable desarrollar tanto de primer como de segundo orden para tener una mejor diversidad y mayor número de opciones al momento de elegir el modelo a trabajar.

2. Las metodologías de identificación buscan suplir la necesidad de encontrar un modelo que represente la dinámica del sistema que se desea controlar. Exactamente cuan fiel debe ser el modelo es competencia del ingeniero al analizar el error que puede ser aceptado en su caso particular. El ingeniero debe ser entonces capaz de discernir qué modelo usar valiéndose de criterios como la complejidad del modelo, la facilidad de uso y el error obtenido.

3. En general los sistemas reales poseen comportamientos no lineales, por lo que para un modelado menos complejo se linealizan en torno a un punto de operación, dado el supuesto de variaciones de pequeña señal. Así el sistema debe operarse respetando las condiciones iniciales presentadas durante la toma de datos que alimentaron los modelos hechos. Por otra parte es posible que un sistema varíe sus características durante el tiempo, obligando a que constantemente se realicen modelos. En el trabajo de laboratorio se asume que el sistema es invariante en el tiempo o por lo menos que su variación es mínima.

4. Varios de los modelos trabajados cuentan con diferencias en cuanto a su desarrollo matemático, pero se sustentaban en la misma idea de elegir puntos de la curva de respuesta del sistema para desarrollar el modelo. Según el sistema es posible que algunos puntos sean mejores que otros. Los métodos iterativos proveen entonces formas más generales de obtener los mejores puntos. De modo que, el método que mejor resultó en este sistema no implica ser el mejor en otro sistema.

5. Las perturbaciones ocurridas durante la caracterización del sistema son un problema particularmente difícil de resolver al momento del modelado ya que modifican el comportamiento tanto en amplitud como en el tiempo; afectando así parámetros como la ganancia y el tiempo de estabilización. Es provechoso entonces hacer múltiples caracterizaciones para observar la verdadera tendencia del sistema, y así obtener un modelo más robusto.

6. El tratamiento de los datos obtenidos durante la experimentación del sistema necesita de una depuración de los datos y una suavización en la respuesta con el fin de tener una información más clara del comportamiento del sistema, ya que se eliminan ciertos picos que constituyen una dificultad, por ejemplo, a la hora de obtener la derivada o el punto de inflexión de la función, que son de gran utilidad para encontrar el modelo adecuado.

7. A pesar de que se tiene una gran variedad de métodos para obtener los parámetros del sistema, algunos de estos no pueden ser aplicados siempre. En nuestro caso el método de Harriott no pudo ser desarrollado ya que no se cumplía con el rango establecido en uno de sus parámetros.

8. Para la comparación de los modelos es necesario que el vector de tiempo sea lo suficientemente largo para que cada uno de los modelos hallados se logren estabilizar. Si lo anterior no se asegura es posible que medidas como el ECM se vean comprometidas y arrojen resultados que produzcan conclusiones erróneas. En los modelos producidos por Matlab se puede observar que se apegan muy bien a la curva en el período transitorio pero que en estado estable sobrepasan el valor esperado. Esto se debe a que poseen una mayor ganancia. En particular si el ECM es calculado sin suficientes muestras en el periodo estable, este puede mostrar que el modelo es el más apropiado a pesar de la discrepancia en la estabilización.

9. El modelado a partir de la curva de reacción ante la respuesta al escalón unitario constituye un método simple de caracterización experimental de los sistemas. Sin embargo es importante recordar que existen otras señales de excitación que permiten caracterizar los sistemas, como lo son la rampa, la sinusoide, el ruido blanco o incluso una aproximación experimental del impulso.

REFERENCIAS

I. Tavera, A; Modelos de procesos. Curso de Control Semestre 0219, Universidad de Antioquia.

II. Murillo, I. (2004). Comparación de las características de desempeño de los modelos de primer y segundo orden más tiempo muerto. Universidad de Costa Rica.

III. Alfaro, V. (s.f). Identificación de procesos sobreamortiguados utilizando técnicas de lazo abierto. San José, Costa Rica.

IV. De La Cruz, F.; Camacho O. (2015) Controlador de Modos Deslizantes basado en Predictor de Smith y Modelo de Segundo Orden para Procesos con Elevado Retardo. Venezuela, Ecuador.