

PONTIFICIA UNIVERSIDAD CATÓLICA DEL PERU

MAESTRIA EN INGENIERIA DE CONTROL Y AUTOMATIZACION



LABORATORIO DE IDENTIFICACIÓN DE SISTEMAS

**DOCENTE:**

**Dr. Juan Javier Sotomayor Moriano**

**PRESENTADO POR LOS ALUMNOS:**

1. **Luis Angel Arias Copacondori**
2. **Dimel Arturo Contreras Martínez**
3. **Elmer Calle Chojeda**

2015

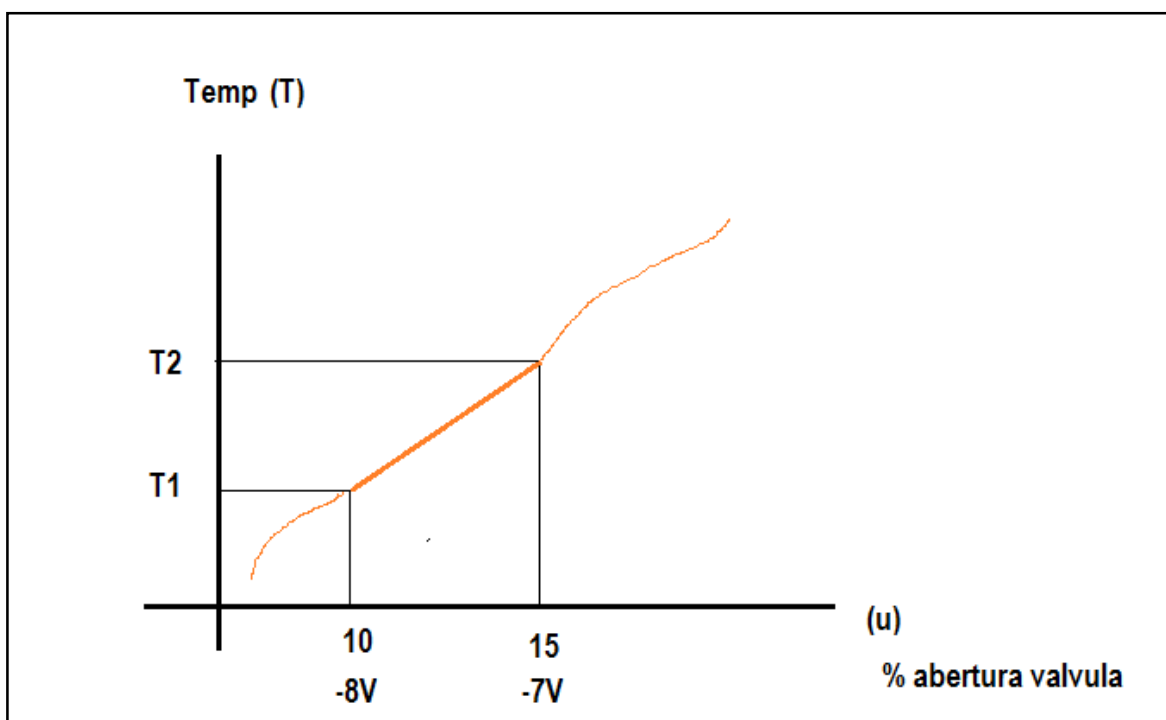
## INDICE

|   |    |
|---|----|
| Generación de señal PRBS.....                       | 3  |
| Adquisición de datos con señal Pseudoaleatoria..... | 6  |
| Objetivos.....                                      | 6  |
| Implementación del sistema.....                     | 6  |
| Equipos.....  | 6  |
| Conexiones.....                                     | 6  |
| Procedimiento software.....                         | 10 |
| Adquisición de datos.....                           | 10 |
| Identificación paramétrica de sistema.....          | 13 |
| Objetivos.....                                      | 13 |
| Procedimiento de identificación.....                | 13 |
| Utilizando Filtros Pasabajos.....                   | 26 |
| Informe.....  | 30 |
| Conclusiones.....                                   | 41 |
| Observaciones.....                                  | 42 |
| Bibliografía.....                                   | 43 |

**PARTE I****GENERACIÓN DE SEÑAL PRBS**

Es una señal binaria pseudoaleatoria, que posee excitación permanente para poder realizar la identificación paramétrica.

Los valores alto y bajo de la señal PRBS se obtienen a partir del rango de linealidad de la planta (intercambiador de calor), utilizando el mayor rango posible para disminuir la relación señal/ ruido.



Como la señal “ub” es de tipo binaria (0 y 1), para pasarlo a los valores de 0 y 15 se realiza la siguiente conversión:

$$u_p = 5 \cdot u_b + 10$$

Se escoge un PRBS de longitud N:

$$N = 2^n - 1$$

$$n = 7$$

$$N = 127$$

$$A = [1 \ 0 \ 0 \ 0 \ 0 \ 0 \ 1];$$

El periodo de conmutación:

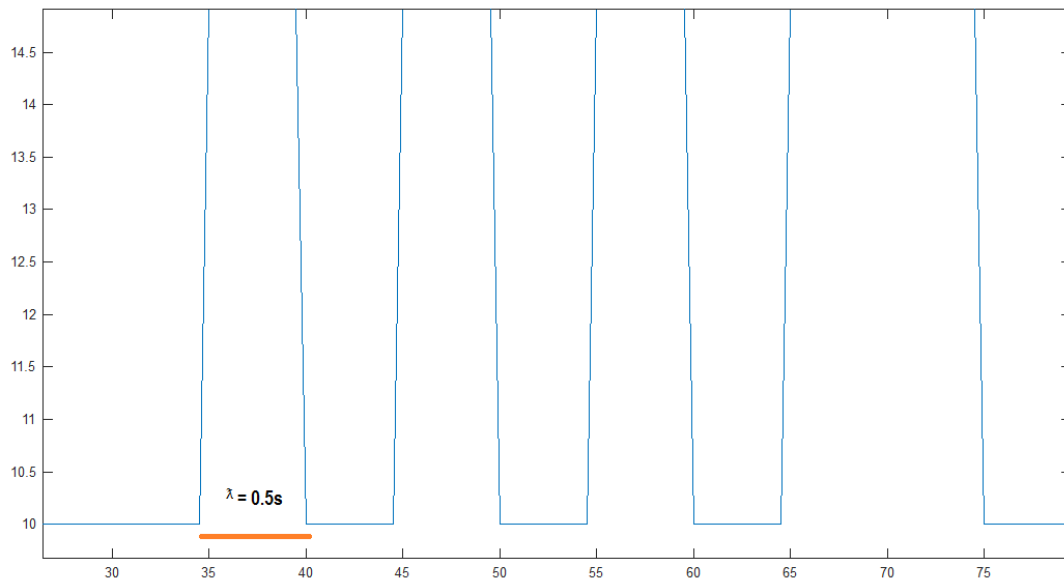
Se considera un  $\tau = 20s$  para la planta intercambiador de calor

$$\lambda = \frac{\tau}{4}$$

$$\lambda = 5s$$

El tiempo de muestreo es:

$$T_m = \frac{\lambda}{10} = 0.5s$$



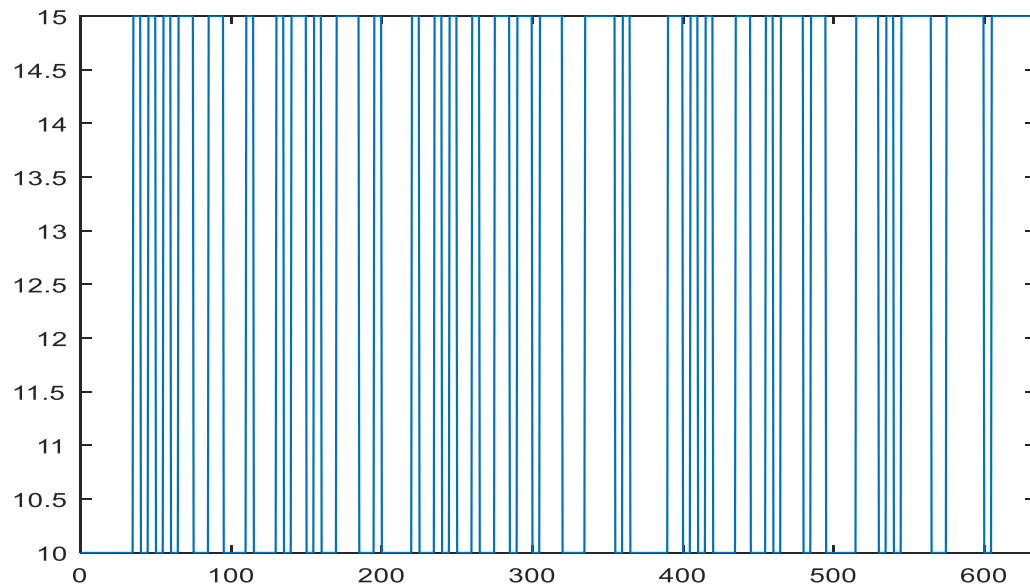
Código implementado:

```
%% Limited band
clear n A up k P
close all
n = 7;
P = 10;
A = [1 0 0 0 0 0 1]; % a6 a5 ... a1
up(1:(n+1)*P,1) = [zeros(n*P,1); ones(P,1)];

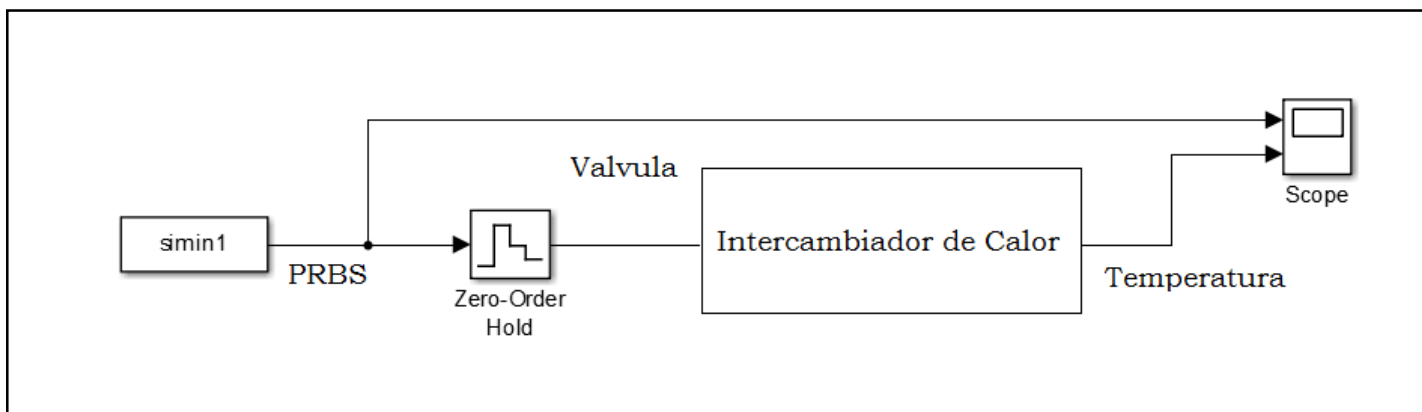
for k=n+2:2^n-1
    up((k-1)*P+1:k*P,1) = rem(A(1)*up((k-1)*P,1)+A(2)*up((k-2)*P,1)+A(3)*up((k-3)*P,1)...
        +A(4)*up((k-4)*P,1)+A(5)*up((k-5)*P,1)+A(6)*up((k-6)*P,1)+A(7)*up((k-7)*P,1)+1,2);
end
up = 5*up + 10;
figure(5)

t = (0:0.5:0.5*(length(up)-1))';
plot(t,up)
uprbs = [t up];
axis tight
```

Resultado:



Esquema de implementación en simulink:



## PARTE II

### ADQUISICION DE DATOS: EXCITACIÓN DE PROCESOS UTILIZANDO UNA SEÑAL PSEUDOALEATORIA

#### 1.-OBJETIVO

Aprender a realizar un proceso de adquisición de datos para una identificación paramétrica de una planta industrial. La presente guía describe: la implementación de las conexiones de hardware, las configuraciones en matlab y el uso del algoritmo desarrollado en el laboratorio anterior para generar una señal pseudoaleatoria.

#### 2.- IMPLEMENTACION DEL SISTEMA DE ADQUISICION DE DATOS

##### 2.1 EQUIPOS UTILIZADOS PARA LA ADQUISICION DE DATOS

Los equipos necesarios para el proceso de adquisición de datos serán los siguientes:

Tarjeta de adquisición de datos (DAQ) PCI-6229) de national instruments.

Bornera para la tarjeta de adquisición de datos.

Computador personal con MATLAB (para la tarjeta PCI-6229) incluyendo el módulo RT (Real Time) para usarlo en el entorno grafico SIMULINK.

Convertor voltaje/corriente (V/I), para transformar el voltaje de salida de la tarjeta DAQ en corriente necesaria para operar la válvula neumática, la tarjeta DAQ trabaja solamente con voltaje.

Resistencia de 250 ohmios, para transformar la salida del sensor en voltaje.

Cables de conexiones con las longitudes necesarias.

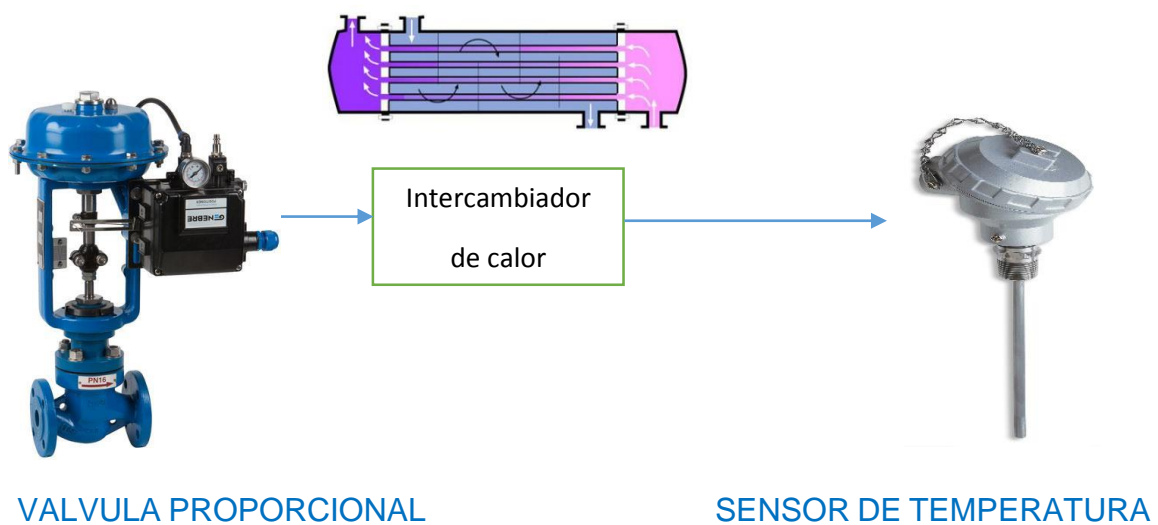
##### 2.2 CONEXIONES ENTRE EQUIPOS

La implementación requiere el empleo de herramientas como: destornilladores, alicates de corte y multímetros que cumplan con los estándares industriales.

#### Planta de control de temperatura (intercambiador de calor)

Esta planta regula el flujo de flujo caliente al intercambiador de calor para poder calentar un fluido a la temperatura deseada

El objetivo del laboratorio es hallar un modelo de la planta mediante identificación no paramétrica y identificación paramétrica



### ACTUADOR

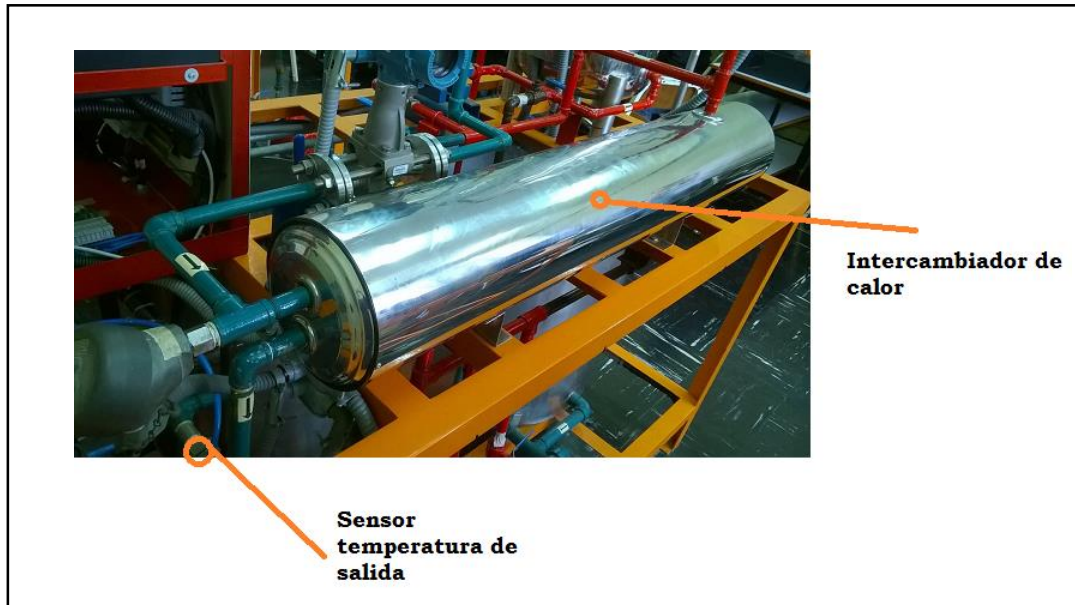
Es una válvula proporcional que regula el caudal de agua caliente hacia el intercambiador de calor. Se controla con señales de 4 a 20mA o de -10V a 10V.



Mediante la regulación de ésta válvula podemos controlar la temperatura del agua fría que pasa por el intercambiador de calor, hasta llegar a temperaturas cercanas al agua caliente.



SENSOR DE TEMPERATURA DEL AGUA DE SALIDA



PLANTA A IDENTIFICAR



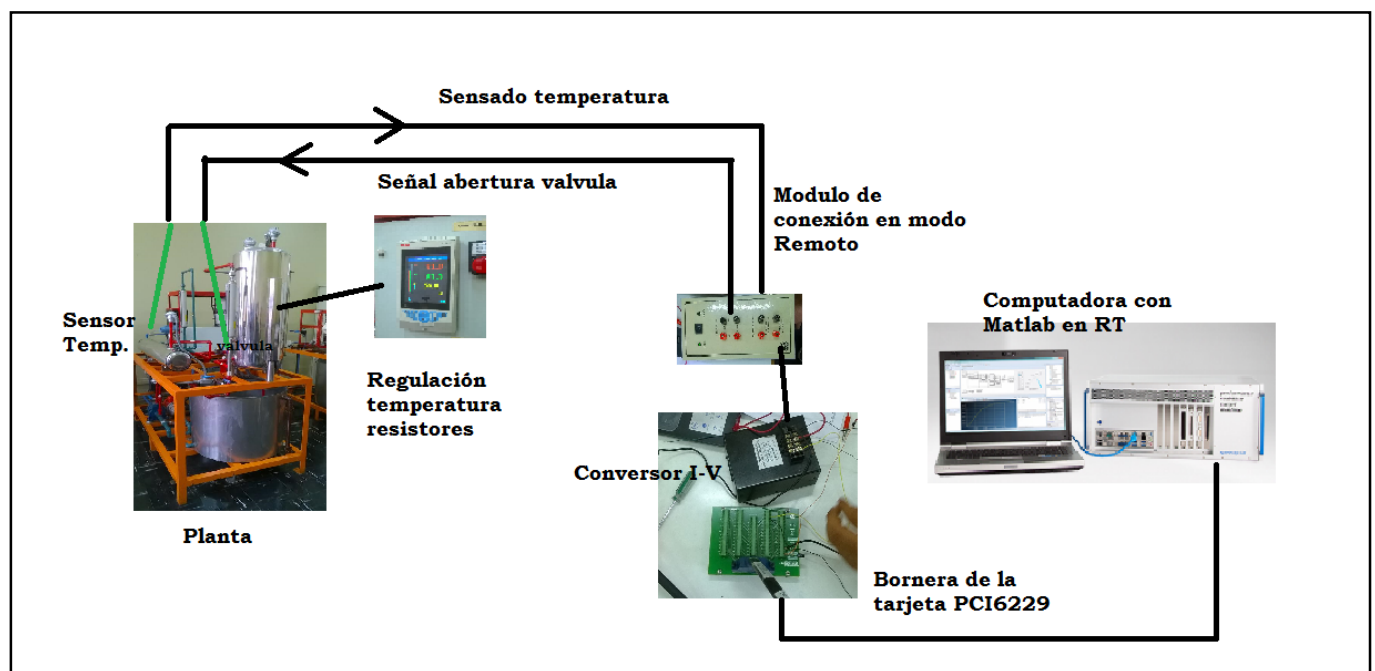
TABLERO DE CONTROL Y MONITOREO PROPIO DE LA PLANTA





En éste tablero se tienen que visualizar las variables, y principalmente controlar que el termo resistores no sobrepasen una temperatura límite.

## ESQUEMA DE IMPLEMENTACIÓN



En la experiencia se inició configurando el software Matlab en Tiempo Real, elaborando un programa SCRIPT para generar señal PRBS, y un esquema Simulink para enviar la señal PRBS a

la planta y recibir información del sensor de temperatura del agua calentada mediante el intercambiador de calor.

Las señales se envían y reciben a la planta desde la computadora utilizando:

**-Módulo de conexión en modo Remoto**

**-Conversor I-V**

**-Bornera de la tarjeta PCI 6229**

Además como parte del proceso se requiere controlar manualmente la temperatura de calentamiento de los resistores y además abrir la válvula de vaciado del tanque de agua calentada antes de que rebalse.

Con estos pasos realizados ya identificamos las variable a controlar y medir en la planta así como su funcionamiento, lo que sigue es configurar la estación de trabajo para la obtención de datos.

### 2.3 PROCEDIMIENTO PARA LA CONFIGURACION DEL SOFTWARE

En este laboratorio se utilizara el programa de simulink de matlab para enviar y recibir señales en tiempo real a través del toolbox de tiempo real.

Para poder ejecutar programas en tiempo real con matlab es necesario asegurarse de que el kernel Real-Time Windows target está instalado, para esto, en el workspace de matlab escriba lo siguiente:

```
>> rtwintgt -setup
```

Es necesario también el compilador de lenguaje C mediante la instrucción MEX.

Esta instrucción es capaz de enlazar y compilar archivos fuente en una biblioteca compartida llamada archivo mex que se ejecuta dentro de Matlab. Se instala de la siguiente manera:

```
>> mex -setup
```

### 2.4 ADQUISICION DE DATOS CON LA DAQ Y SIMULINK

-Para desarrollar la adquisición de datos se asume que ya se utilizó el algoritmo para generar la señal pseudoaleatoria y por consiguiente ya existe un vector bidimensional de esta señal en el espacio de trabajo con la forma:

$$\text{Simin1}=[t \ U] \text{ o simplemente } [t \ U]$$

Donde:

t=vector de tiempo

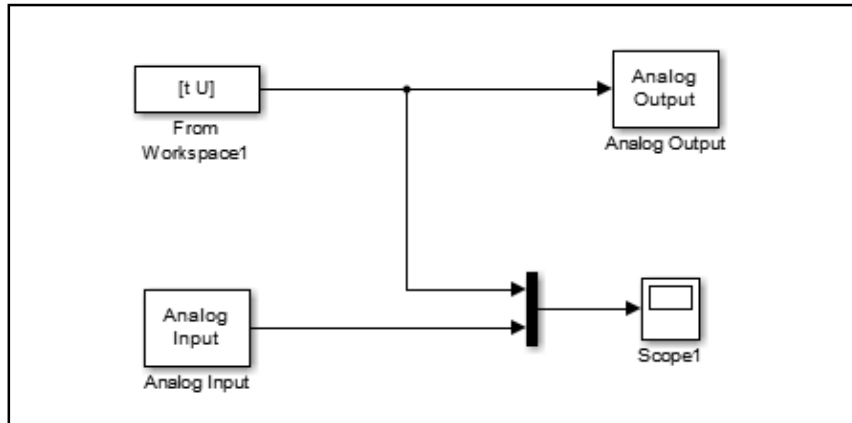
U=señal pseudoaleatoria

-Para crear un programa de envío y adquisición de las señales de respuesta de la planta en tiempo real con el simulink, se deberán utilizar los bloques analog output y

analog input de la librería de bloques: Real time Windows target, además se utilizara el bloque From Workspace del simulink. Luego se deberá crear el siguiente programa en el simulink:

## Modelamiento.mdl

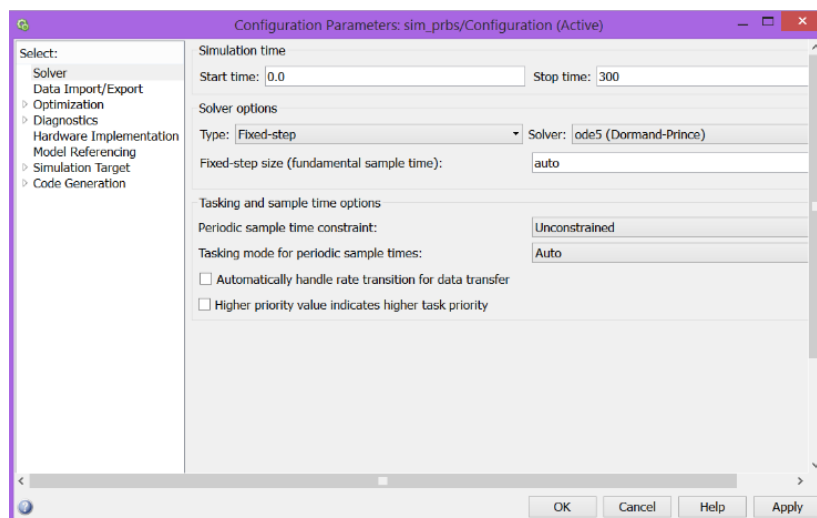
-Incluya los bloques correspondientes y conéctelos según:



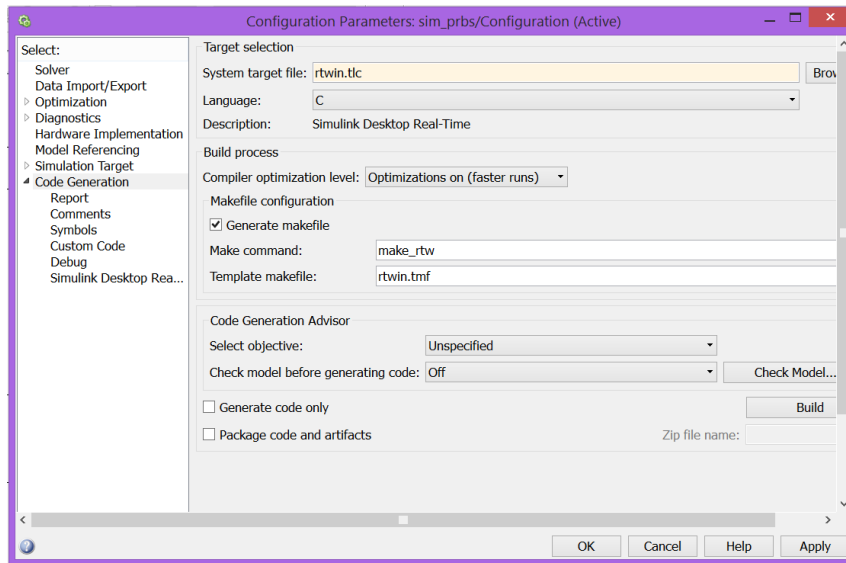
-Este programa lee señales de un canal de entrada analógica de la tarjeta y la gráfica. También envía un voltaje pseudoaleatorio a través de un canal de salida analógica de la tarjeta. Para que este programa funcione correctamente, tenemos que configura las opciones dentro de los boques analógicos. Para los bloques analog input y analog output es necesario configurar:

- La tarjeta a utilizar: National instruments PCI-6229
- El periodo de muestreo : pe 0.01
- El canal de entrada /salida a utilizar: es necesario considerar que matlab considera al canal 0 como 1 y así sucesivamente, por lo que se requiere sumarle una unidad al número del canal asignado en la tabla de National Instruments.

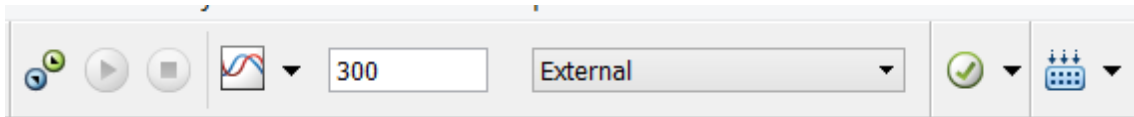
-Posteriormente se requieren configurar los parámetros de simulación del programa dentro del menú simulation en la opción model configuration Parameters, en el submenú solver.



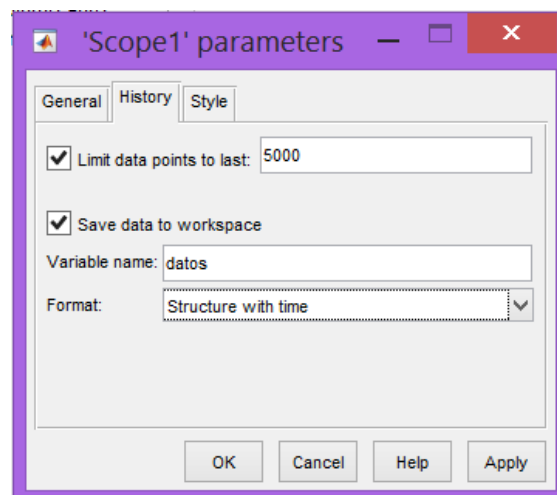
-Adicionalmente se requiere configurar el archivo objetivo del sistema y su compilador, dentro del submenú code generation.



-Finalmente se requiere compilar el programa y ejecutarlo en modo externo.



-Para el bloque Scope es necesario configurar una variable para almacenar la data, en la casilla DATA HISTORY: Los datos se almacenaran en una variable "datos" para su posterior procesamiento.



Con estos pasos realizados tenemos configurado Maltab para que trabaje en Real time usando la tarjeta de adquisición para poder obtener datos de la planta, lo siguiente que debemos hacer es usar los métodos de identificacion de sistemas con los datos obtenidos

## PARTE III

### IDENTIFICACION PARAMETRICA UTILIZANDO TECNICAS DE IDENTIFICACION DE SISTEMAS

#### 1. OBJETIVOS

- Preparar datos obtenidos experimentalmente para la adecuada modelación de un proceso.
- Realizar la identificación paramétrica de un proceso utilizando las técnicas de Identificación de Sistemas.

#### 2. PROCEDIMIENTO DE IDENTIFICACION PARAMETRICA

**1. Cargar los datos en el espacio de trabajo de Matlab:** Se puede utilizar el comando *load* o el comando *open* para abrir el archivo *identi.mat* donde fueron guardados los datos de entrada / salida.

```
>>load identi.mat
```

```
close all; clc;  
load identi.mat  
y1=datos1.signals.values(:,2);  
u1=datos1.signals.values(:,1);  
tt = datos1.time;
```

**2. Crear una estructura de datos de identificación, con los datos de entrada, salida y tiempo de muestreo utilizado, introducir los nombres de las variables respectivas.**

```
>>est_id = iddata (y1, u1, ts);
```

```
ts=0.5;  
Wnl=0;  
Wnh= 20;  
est_id=iddata(y1,u1,ts);
```

```
est_id =  
  
Time domain data set with 1201 samples.  
Sample time: 0.5 seconds  
Outputs    Unit (if specified)  
y1  
Inputs     Unit (if specified)G  
u1
```

Donde:

*est\_id* -estructura de datos de identificación.

y1      -salida o respuesta del proceso.  
u1      -entrada (señal pseudoaleatoria) del proceso.  
ts      -periodo de muestreo.

Para introducir los nombres de las variables:

```
>>est_id.InputName='flujo';  
>>est_id.OutputName='temperatura';
```

```
ts=0.5;  
Wnl=0;  
Wnh= 20;  
est_id=iddata(y1,u1,ts);  
est_id.InputName='Flujo_agua';  
est_id.OutputName='Temperatura';
```

De ser necesario, en este punto se pueden filtrar los datos obtenidos experimentalmente utilizando la función **idfilt**, de la siguiente forma:

```
>>est_idF = idfilt (est_id, [Wnl Wnh])
```

```
est_idF=idfilt(est_id,[Wnl Wnh]);
```

Donde:

est\_id -estructura que contiene los datos de entrada y de salida.

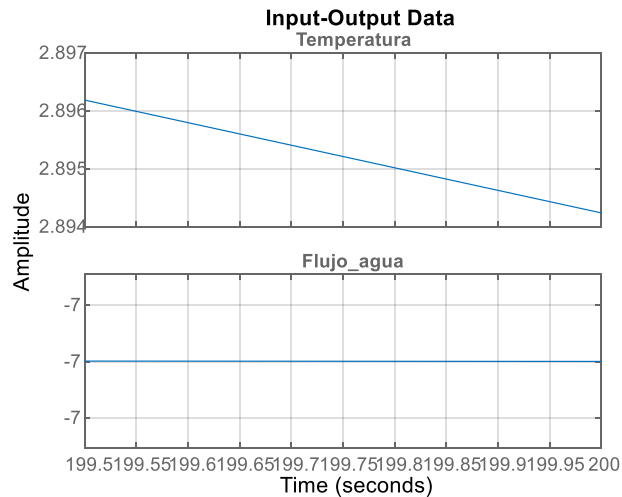
Wnl    -frecuencia inferior del filtro.

Wnh    -frecuencia superior de filtro.

**3. Seleccionar un rango de datos para obtener el modelo: el resto servirá para validar el mismo. En este caso el vector de entrada será:**

```
>>ze = est_idF(200:400);  
>>idplot(ze(200:300))
```

```
ze=est_idF(200:400);  
idplot(ze(200:300))
```



**4. Remove la varianza de los datos, para eliminar el offset de las señales o componentes continuas:**

```
>>ze = dtrend(ze);
```

```
ze=dtrend(ze)
```

```
ze =  
Time domain data set with 201 samples.  
Sample time: 0.5 seconds  
Outputs      Unit (if specified)  
  Temperatura  
Inputs       Unit (if specified)  
  Flujo_agua
```

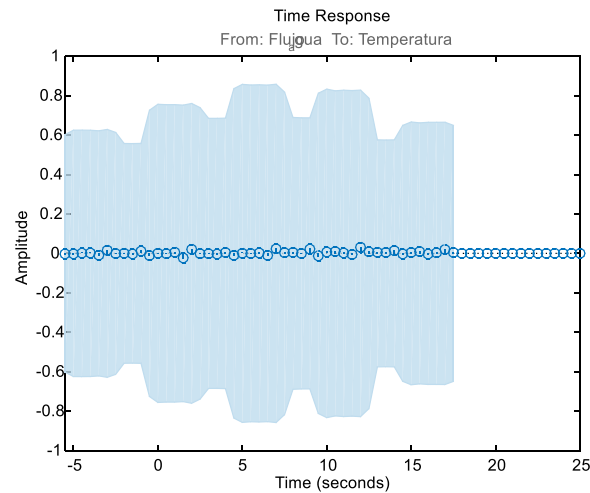
Es necesario restar los valores medios de cada señal debido a que, normalmente, se construyen modelos lineales que describen la respuesta del proceso para desviaciones de un equilibrio físico. Para los datos en estado de equilibrio, es razonable suponer que los niveles medios de las señales corresponden a este equilibrio. Es así, que es posible buscar modelos en torno a cero, sin modelar los niveles de equilibrio absoluto en unidades físicas.

**5. Obtener la respuesta al impulso del sistema, esto ayudará a tener una información preliminar importante para encontrar el modelo paramétrico.**

```
>>impulse(ze, 'sd', k)
```

```
impulse(ze,'sd',k)
```

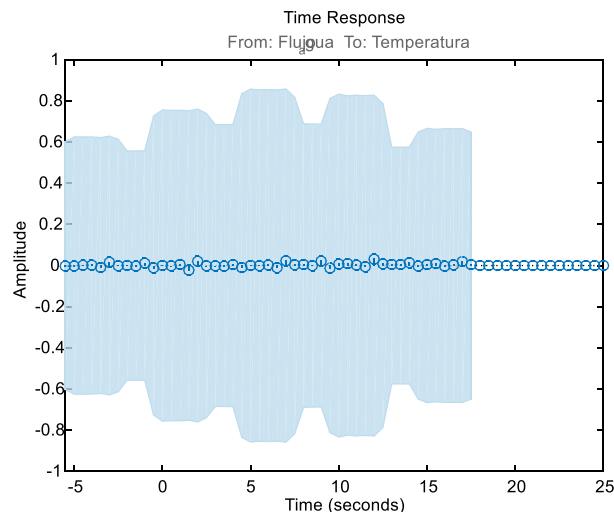




Esta instrucción grafica la respuesta impulsional del proceso, mostrando la región de confianza correspondientes a las desviaciones estándar  $k$ , como una región alrededor de cero. Cualquier respuesta fuera de esta región es así “significante”. Es posible añadir el argumento “FILL” después del modelo para mostrar la región o regiones de confianza como una banda de un color característico, para mayor facilidad de análisis de los resultados:

```
>> impulse(M, 'sd', k, 'fill')
```

```
impulse(ze,'sd',k,'fill');
```



De la gráfica resultante se puede encontrar una aproximación razonable del retardo de tiempo. Este retardo pudo haberse obtenido también de una id. no paramétrica.

**6. Utilizando el Método de Error de Predicción (PEM: prediction error method) encontrar una primera aproximación del modelo en espacio de estados. Matlab**

sintetiza este método en la instrucción *pem* que calcula un modelo en espacio de estados a partir de una estructura de datos.

>>m1 = pem(ze);

m1=pem(ze)

m1 =  
Discrete-time identified *state-space model*:  
 $x(t+Ts) = A x(t) + B u(t) + K e(t)$   
 $y(t) = C x(t) + D u(t) + e(t)$

A =  
          x1      x2  
x1 0.9387 0.02974  
x2 -0.1569 1.007

B =  
      Flujo\_agua  
x1 0.01265  
x2 -0.008795

C =  
          x1      x2  
Temperatura -0.2193 -0.08182

D =  
          Flujo\_agua  
Temperatura 0

K =  
      Temperatura  
x1 -1.4  
x2 -1.385

Sample time: 0.5 seconds

Parameterization:  
FREE form (all coefficients in A, B, C free).  
Feedthrough: none  
Disturbance component: estimate  
Number of free coefficients: 10  
Use "idssdata", "getpvec", "getcov" for parameters and their uncertainties.

Status:  
Estimated using PEM on time domain data "ze".  
Fit to estimation data: 77.05% (prediction focus)  
FPE: 2.375e-05, MSE: 2.2e-05

Las características de este modelo se encuentran, mediante:

```
>>get(m1)
```

```
get(m1)
```

```
a: [2x2 double]
b: [2x1 double]
c: [-0.2193 -0.0818]
d: 0
k: [2x1 double]
StateName: {2x1 cell}
StateUnit: {2x1 cell}
Structure: [1x1 pmodel.ss]
NoiseVariance: 2.2910e-05
Report: [1x1 idresults.ssest]
InputDelay: 0
OutputDelay: 0
Ts: 0.5000
TimeUnit: 'seconds'
InputName: {'Flujo_agua'}
InputUnit: {''}
InputGroup: [1x1 struct]
OutputName: {'Temperatura'}
OutputUnit: {''}
OutputGroup: [1x1 struct]
Name: ''
Notes: {}
UserData: []
SamplingGrid: [1x1 struct]
```

```
>>m1.EstimalInfo
```

```
m1.EstimationInfo
```

```
ans =

Status: 'Estimated using PEM with Focus = "prediction"'
Method: 'PEM'
LossFcn: 2.1998e-05
FPE: 2.3749e-05
DataName: 'ze'
DataLength: 201
DataTs: 0.5000
DataDomain: 'Time'
DataInterSample: 'bl'
WhyStop: 'Maximum number of iterations reached'
UpdateNorm: 1.8218
LastImprovement: 4.3435e-07
Iterations: 20
InitialState: 'estimate'
Warning: 'None'
N4Horizon: [15 10 10]
N4Weight: 'CVA'
```

Asimismo, para encontrar una de las matrices A, B, C o D del modelo en espacio de estados, basta con citarlas así:

```
>>m1.A
```

```
m1.A
```

```
ans =
```

```
0.9387 0.0297
-0.1569 1.0073
```

**7. Verificar la calidad del modelo obtenido, empleando el método de validación cruzada con un rango de datos diferentes a los utilizados en el proceso de modelación.**

Para esto se escoge un intervalo de datos no utilizados anteriormente. Por ejemplo:

```
>>zv = est_id(700:1000);
```

```
zv=est_id(700:1000);
```

```
zv =
Time domain data set with 301 samples.
Sample time: 0.5 seconds
Outputs      Unit (if specified)
  Temperatura
Inputs       Unit (if specified)
  Flujo_agua
```

Al igual que en los datos de modelamiento, se procede a eliminar la varianza:

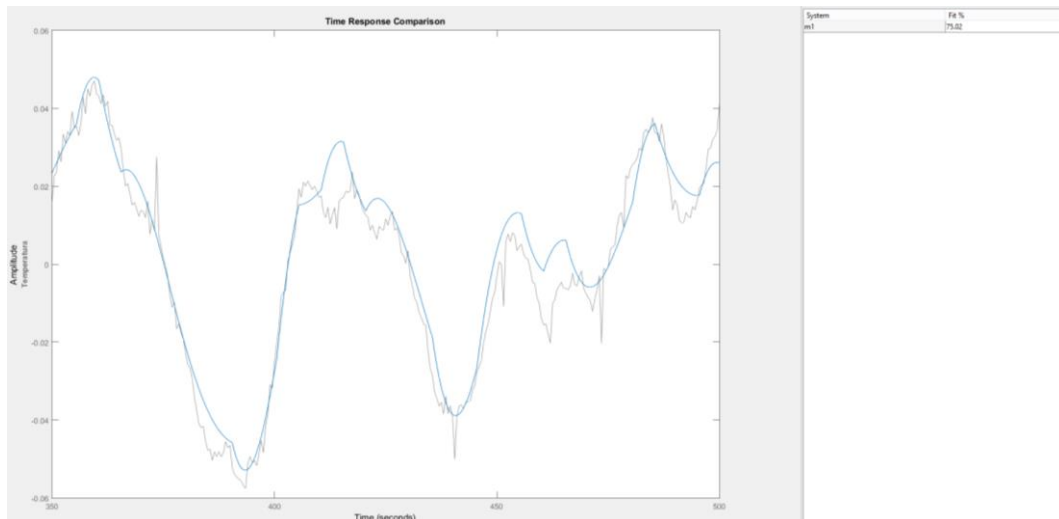
```
>>zv = detrend(zv);
```

```
zv=dtrend(zv);
```

Para la comparación propiamente dicha, se utiliza el comando **compare**, que tiene como parámetros de entrada: a la respuesta de un sistema (caracterizada en un intervalo de datos) y a un modelo en espacio de estados.

```
>>compare(zv, m1)
```

```
compare(zv,m1)
```



Se cuantifica la aproximación del modelo utilizando un “índice de performance”. Este índice es una medida cuantitativa de la calidad del modelo que puede obtenerse a partir de la norma de los errores residuales.

La fórmula que utiliza Matlab para calcular el índice de performance (FIT) es la siguiente:

$$FIT = \left( 1 - \frac{\|y - \hat{y}\|}{\|y - \bar{y}\|} \right) \times 100\%$$

Donde:

$y$  -salida medida.

$\hat{y}$  -salida estimada.

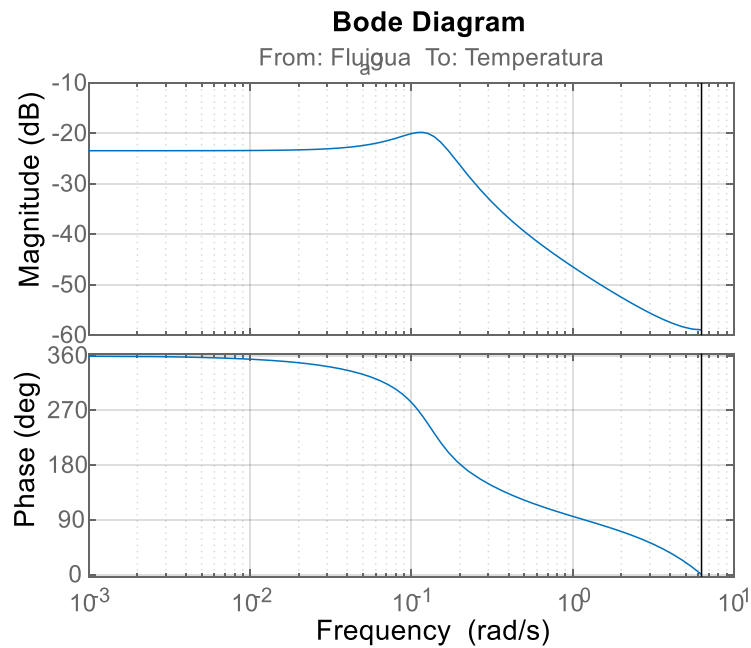
$\bar{y}$  -media de  $y$ .

Este índice corresponde a las variaciones de la salida real que son reproducidas por el modelo. Un número más cercano a 100% significa un mejor modelo.

**8. Para analizar las características frecuencias del modelo, se encuentra el diagrama de bode:**

```
>>bode(m1)
```

```
bode(m1);grid;
```



Alternativamente se puede utilizar un diagrama de Nyquist considerando regiones de incertidumbre (dentro de las elipses), correspondientes a  $k$  desviaciones estándar.

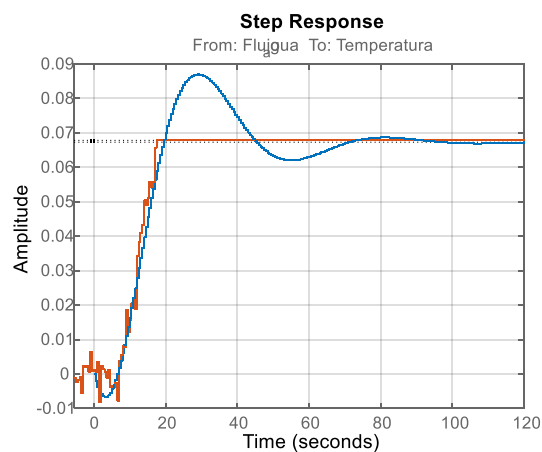
```
>>nyquist(m1, 'sd', k)
```

```
nyquist(m1,'sd',k);grid
```

**9. Para analizar el comportamiento temporal del modelo versus la respuesta real del proceso, se utiliza la función `step`.**

```
>>step(m1, ze)
```

```
step(m1,ze);grid
```



## 10. Creación de modelos paramétricos con estructuras pre-establecidas

El objetivo de este tipo de identificación es encontrar modelos polinomiales, por ejemplo para un modelo ARX se tiene la siguiente estructura:

$$y(t) = a_1 y(t - T) + a_2 y(t - 2T) \dots a_n y(t - nT) \\ = b_1 u(t - 3T) + b_2 u(t - 4T) \dots b_n y(t - nT)$$

Donde:

T -periodo de muestreo.

$a_n$  -número de polos.

$b_n$  -1-número de ceros.

10. La implementación de modelos ARX, se concibe según la función:

`>>m2=arx(ze,[na nb nk]);`

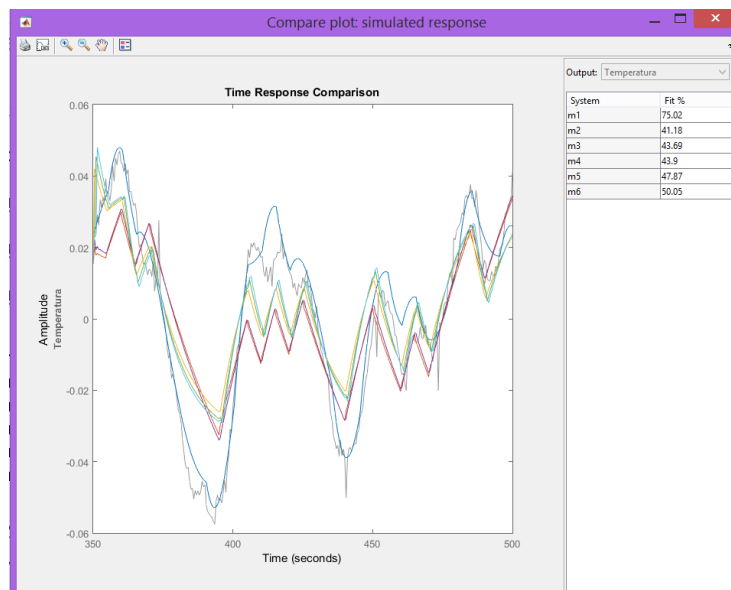
na -coeficiente de polos.

nb -coeficiente de ceros.

nk -coeficiente de periodos de retardo.

```
m2 = arx(ze,[2 1 10])
m3 = arx(ze,[1 2 10])
m4 = arx(ze,[2 2 10])
m5 = arx(ze,[1 3 10])
m6 = arx(ze,[1 4 10])
```

```
compare(zv,m1,m2,m3,m4,m5,m6)
```





Los coeficientes de polos y ceros se pueden ir iterando intuitivamente, comparando sus “índices de performance” hasta encontrar porcentajes de adecuación razonables.

Una herramienta de apoyo que introduce Matlab para encontrar estos coeficientes es la función **arxstruc** que computa funciones de pérdida para familias de modelos ARX; esto quiere decir que se introducen vectores que contienen menor función de pérdida entre un rango a valores para modelar: **ze** y un rango de valores a validar: **zv**. Para introducir los rangos de valores se hace uso de la función **struc**.

Un ejemplo de utilización de esta función es el siguiente:

```
>>V = arxstruc(ze, zv, struc(1:5,1:5,1:5))
```

```
V = arxstruc(ze, zv, struc(1:5,1:5,1:5));
```

Que calcula la función de pérdida para todas las posibles combinaciones de  $n_a$  entre [1:5] con  $n_b$  entre [1:5] y  $n_k$  entre [1:5].

El resultado del cálculo de la función de pérdida aparecerá en la primera fila para los coeficientes que se muestran en las filas restantes de cada columna, la última columna de V retorna el número de datos de cálculo.

**11. Calcular por lo menos 05 modelos de acuerdo a las familias encontradas con *arxstruc*, hasta encontrar el que presente mejor porcentaje de validación:**

```
>>m2 = arx (ze, [na2 nb2 nk2]);
```

```
>>m3 = arx (ze, [na3 nb3 nk3]);
```

```
>>m4 = arx (ze, [na4 nb4 nk4]);
```

```
>>m5 = arx (ze, [na5 nb5 nk5]);
```

```
>>m6 = arx (ze, [na6 nb6 nk6]);
```

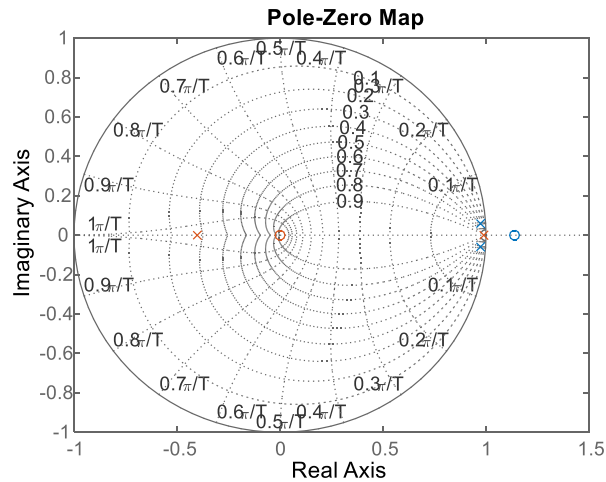
**12. Comparar los modelos encontrados, analizando su índice de performance:**

```
>>compare( zv, m1, m2, m3, m4, m5, m6)
```

**13. Comparar y graficar los polos y ceros de los modelos encontrados. Analizar los resultados, mediante la función:**

```
>>pzmap(m1, m2)
```

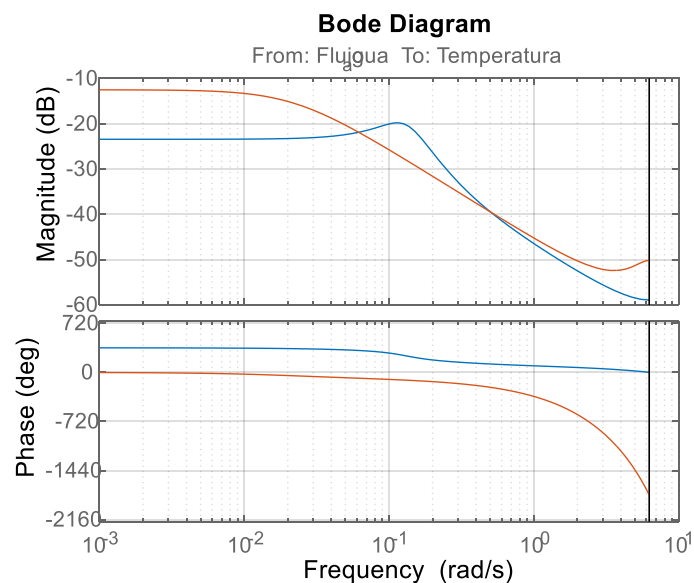
```
pzmap(m1,m2);grid
```



14. Igualmente para analizar las características en frecuencia de los modelos, se encuentra el diagrama de bode:

```
>>bode(m1, m2, m3, m4)
```

```
bode(m1,m2);grid
```



15. Implementar un modelo ARMAX de la forma:

$$A(q)y(t) = B(q)u(t) + C(q)e(t)$$

Donde:

$$A(q) = 1 + a_1q^{-1} + \dots + a_{na}q^{-na} \quad (\text{Polos})$$

$$B(q) = b_1 + b_2q^{-1} + \dots + b_{nb}q^{-nb} \quad (\text{Ceros})$$

$$C(q) = 1 + c_1q^{-1} + \dots + c_{nc}q^{-nc} \quad (\text{Perturbaciones})$$

Teniendo como referencia a los valores de los coeficientes obtenidos para el modelo ARX encontrado como el que mejor describe el comportamiento del proceso, se debe ir iterando tentativamente valores para la amplitud del vector de perturbaciones.

La función para implementar modelos ARMAX es la siguiente:

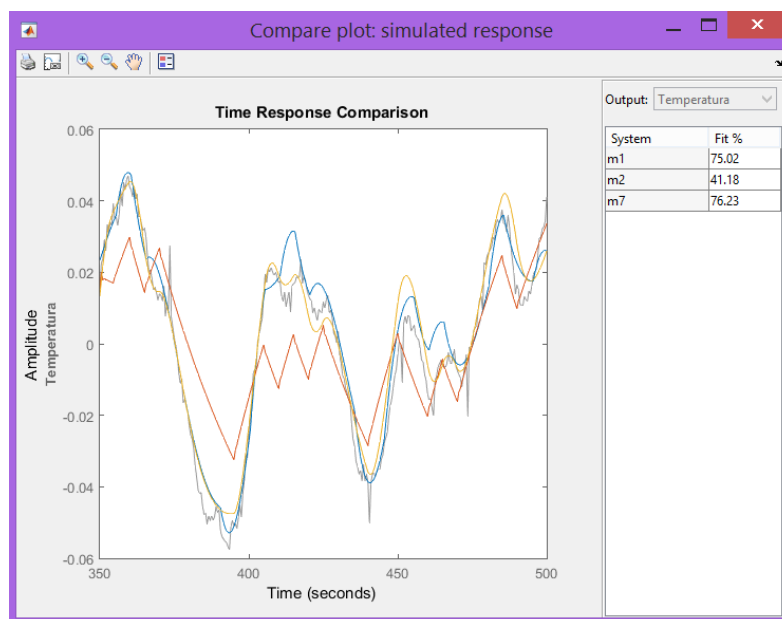
```
>>m7 = armax(ze,[na nb nc nk])
```

```
na = 2; %Mediante pruebas se obtuvo este valor
nb = 1; %que de aproxima mejor
nc = 2;
nk = 10;
m7 = armax(ze,[na nb nc nk])
```

### 16. Comparar los modelos encontrados, haciendo uso de la función *compare*.

```
>>compare(zv,m7)
```

```
compare(zv,m1,m2,m7)
```



### 17. Utilizando el mismo procedimiento se pueden calcular modelos Box-Jenkins, Output-Error, etc. Considerando sus respectivos parámetros iniciales y sus respectivas funciones de cálculo.

La función para implementar modelos Box-Jenkins es la siguiente:

```
>> m8 = bj(ze,[nb nc nd nf nk])
```

```
nb = 1;
nc = 2;
nd = 3;
nf = 5;
nk = 10;
m8 = bj(ze,[nb nc nd nf nk])
```

La función para implementar modelos Output-Error es la siguiente:

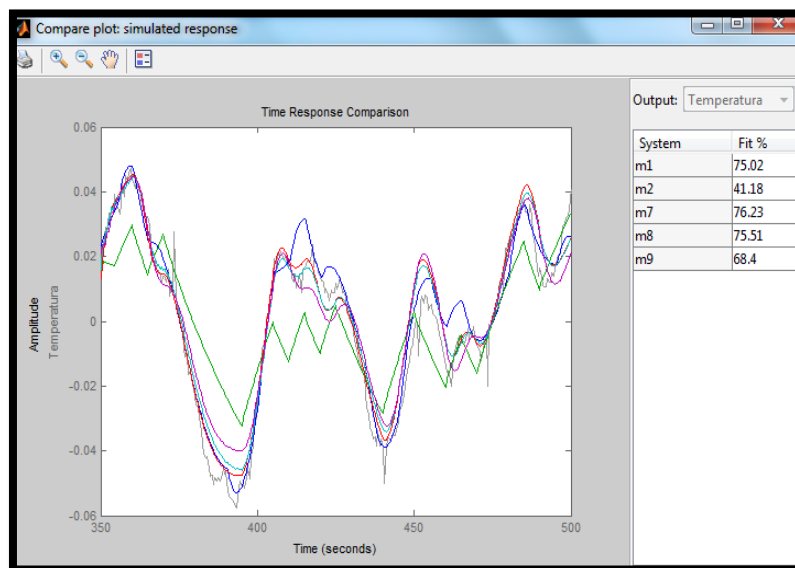
```
>> m9 = oe(ze,[nb nc nd nf nk])
```

```
nb = 1;
nf = 5;
nk = 10;
m9 = oe(ze,[nb nf nk])
```

Comparando los modelos encontrados, haciendo uso de la función **compare**.

```
>>compare(zv,m1,m2,m7,m8,m9)
```

```
compare(zv,m1,m2,m7,m8,m9)
```

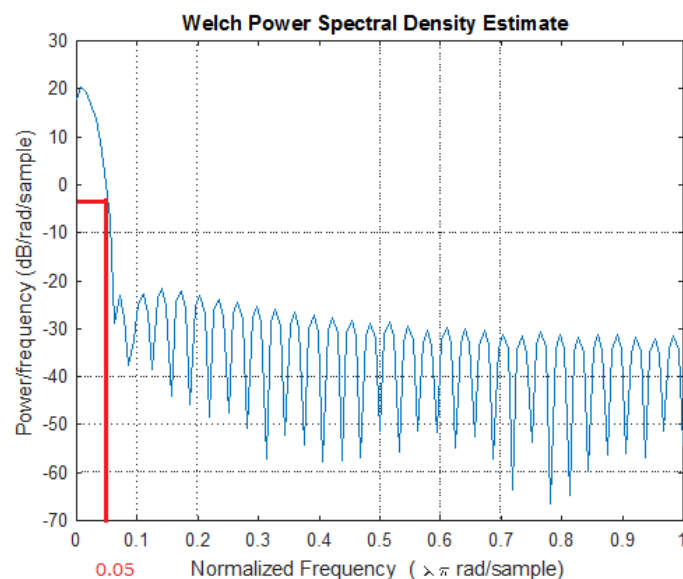
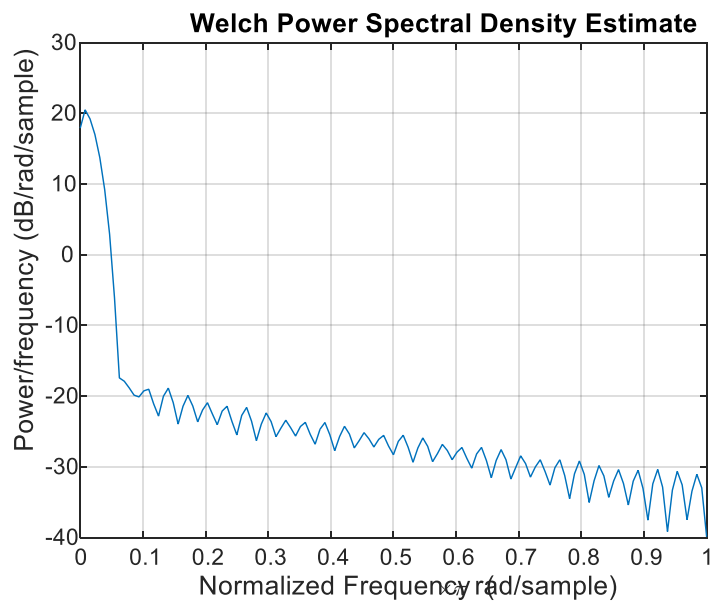


Realizado los pasos anteriores hemos obtenido modelos paramétricos de la planta, donde pudimos obtener mediante pruebas los parámetros que mejor se aproximan al sistema.

## UTILIZACION DE FILTROS PASABAJOS EN LA ADQUISICION

-Para diseñar el filtro adecuado, es necesario conocer la frecuencia de corte adecuada para atenuar las componentes de frecuencia superiores a esta. Para encontrar la frecuencia de corte se debe calcular la densidad espectral de potencia, esto se hará a través de la instrucción `psd.matlab`.

```
%%%%%%%%%%%%%% Grafica de la densidad Espectral de
Potencia%%%%%%%%
yys=spectrum.welch;
DEP=psd(yys,y1);
plot(DEP)
%%%%%%%%%%%%%%
```



$$F_{\text{normaliz}} \cdot \pi = F$$

$$F_{\text{normaliz}} = F_{\text{señal}} / F_{\text{muestreo}}$$

$$(F_{\text{señal}} / (F_{\text{muestreo}} \cdot \text{sample})) \cdot \pi = F$$

Donde  $F_{\text{muestreo}} = 1/0.5$

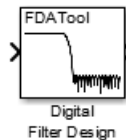
**Frecuencia de pase  $F_{\text{pass}}$  en la grafica es=0.01**

$$F_{\text{pass}} = 2.5736 \text{ rad/seg}$$

**Frecuencia de stop  $F_{\text{stop}}$  en la grafica es=0.1**

$$F_{\text{stop}} = 25.6153 \text{ rad/seg}$$

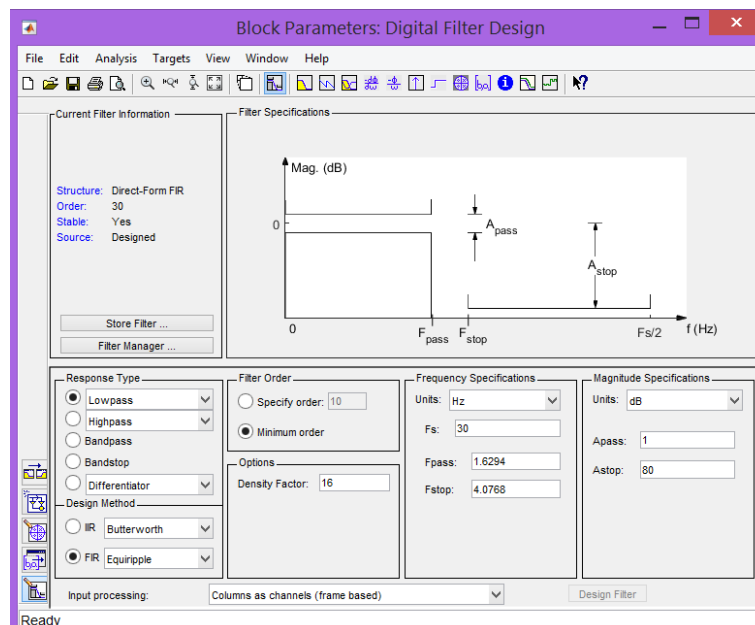
- Analizando la densidad espectral de potencia del grafico de DEP se obtendrá la frecuencia de corte que servirá para diseño del filtro pasabajo, en simulink se debe incluir el bloque: Digital Filter Design de menú: Signal Processing Blockset.



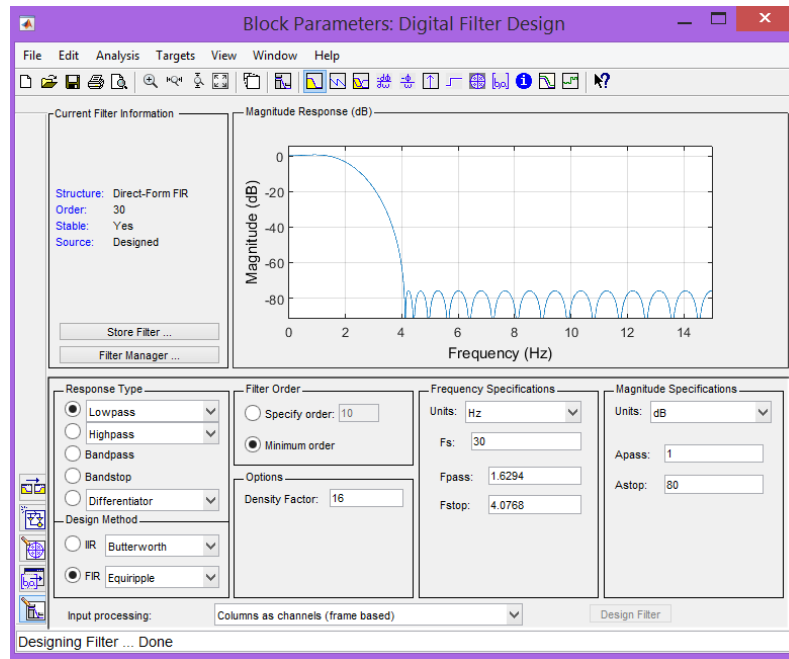
-Dentro de este bloque se utilizará la  $F_{\text{pass}}$  (Frec. Corte) para diseñar el filtro, se incluye un  $F_{\text{stop}}$ , además se pueden las especificaciones se pueden manejar las especificaciones de magnitud, el orden del filtro, etc.

$$F_{\text{pass}} = 1.62 \text{ Hz}$$

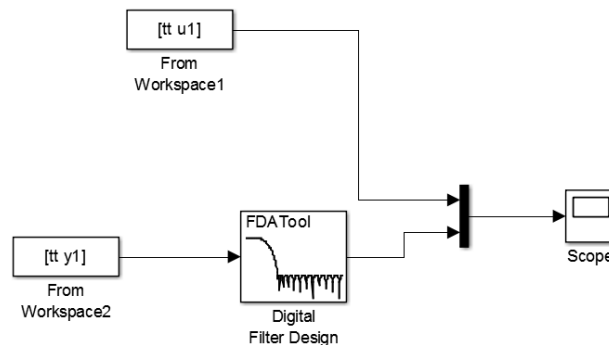
$$F_{\text{stop}} = 4.07 \text{ Hz}$$



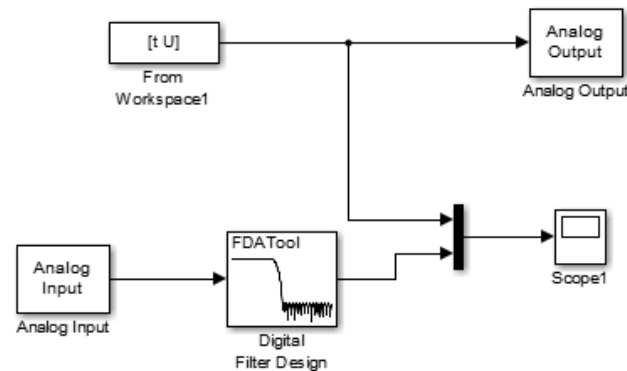
Frecuencia en Hz



-Después de introducir todos los parámetros necesarios se diseña el filtro mediante el botón design filter. Finalmente se introduce este bloque en el programa de adquisición de datos.



Incluyendo los bloques enunciados a lo largo de la guía, el modelo en simulink para enviar la señal pseudoaleatoria de excitación persistente al proceso a modelar, queda como se muestra en la siguiente figura



Hicimos la prueba con un filtro pasa bajos para la señal de entrada de temperatura, con cálculos realizados en su frecuencia de corte, obtuvimos datos semejantes.



## NOTA

En este punto analizamos las condiciones iniciales del sistema ante la respuesta que nos dio matlab.

Haciendo estos cambios (No trabajar con salida temperatura sino con la variación de temperatura y luego sumarle las condiciones iniciales)

$u1 = u1*5+50$ ; (para q vaya de 10 a 15 bar)

$y1 = y1-2.8$ ; (para que el sistema tenga salida variación de temperatura)  
(2.8 es el valor que entrega el sensor en  $T_o$ )

## INFORME

### Análisis de Variable de control y Variable de proceso

#### a. La entrada(u)

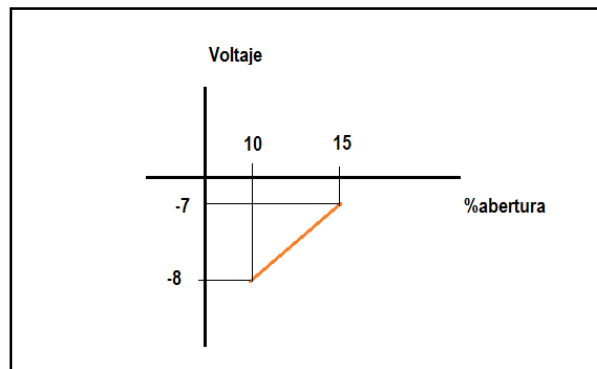
Es el flujo que se varía mediante una válvula.

Valores:

Eléctrico: -10V a +10V.

Físico: 0 -100% (Abertura)

Rango lineal utilizado: 10 – 15% / -8 a -7V



#### b. La salida(y)

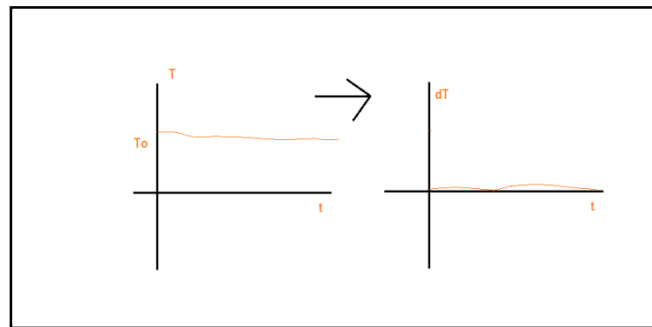
Es la temperatura del agua a la del intercambiador de calor.

Valores:

Eléctrico: 4 a 20mA. (Mediante un conversor se vuelve voltaje)

Físico: -----

Rango lineal utilizado: 40 – 60°C/ 2 a 4V (Considerado a partir de los datos)



Como la temperatura representa una salida con valor inicial diferente de cero, se tomará como variable de **salida la variación de temperatura, y como offset su valor inicial ( $T_o$ )**.

$$y(0)=T(o)=T_o$$

Si se considera la variación de temperatura en vez de la temperatura en sí:

$$y(0)=dT(o)=0 \text{ (sistema relajado) , Con offset } T_o$$

**1. Desarrollar el procedimiento completo de identificación del proceso estudiado, presentar un modelo ARX y un modelo ARMAX que mejor describa el comportamiento del proceso, aplicar análisis temporales y frecuenciales, sustentar los resultados con gráficos, tablas, etc. Dar conclusiones.**

Modelo ARMAX

Discrete-time ARMAX model:  $A(z)y(t) = B(z)u(t) + C(z)e(t)$

$$A(z) = 1 - 1.9 z^{-1} + 0.906 z^{-2}$$

$$B(z) = 0.0001144 z^{-10}$$

$$C(z) = 1 - 1.54 z^{-1} + 0.5401 z^{-2}$$

Sample time: 0.5 seconds

Parameterization:

Polynomial orders:  $n_a=2$   $n_b=1$   $n_c=2$   $n_k=10$

Number of free coefficients: 5

Use "polydata", "getpvec", "getcov" for parameters and their uncertainties.

Status:

Estimated using ARMAX on time domain data "ze".

Fit to estimation data: 76.79% (prediction focus)

FPE: 2.438e-05, MSE: 2.25e-05

m2 =  
Discrete-time ARX model:  $A(z)y(t) = B(z)u(t) + e(t)$   
 $A(z) = 1 - 0.5868 z^{-1} - 0.3976 z^{-2}$   
 $B(z) = 0.0007364 z^{-10}$   
Sample time: 0.5 seconds  
Parameterization:  
Polynomial orders: na=2 nb=1 nk=10  
Number of free coefficients: 3  
Use "polydata", "getpvec", "getcov" for parameters and their uncertainties.  
Status:  
Estimated using ARX on time domain data "ze".  
Fit to estimation data: 74.81% (prediction focus)  
FPE: 2.641e-05, MSE: 2.651e-05

Comprobación:

```
clc;
close all;

t=0:0.5:100000;
t=t(1:length(u1));
temp_init = 50; % Valor considerado
u_esc = u1;

figure(1)
% lsim(sys_dis,u1,t,'b');
subplot(211)
% lsim(sys_con,u_esc,t);
plot(t,u1,t,y1,'-g')
title('Variación de la temperatura')
legend('u Valvula PRBS' , 'y Variacion temperatura')

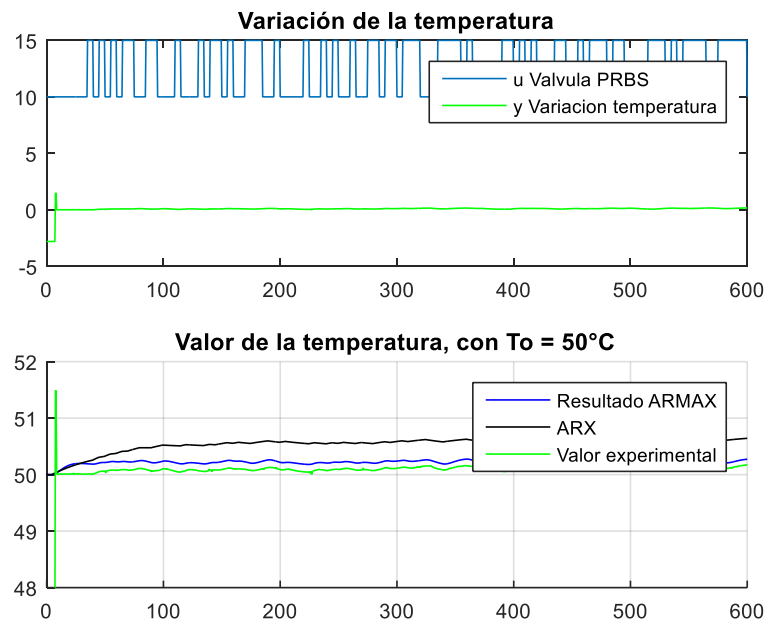
subplot(212)
hold on
y_m7=lsim(m7,u_esc,t)
plot(t,y_m7+temp_init,'-b')
legend('Resultado ARMAX')

y_m2 = lsim(m2,u_esc,t)
plot(t,y_m2+temp_init,'k')
legend('ARX')

plot(t,y1+temp_init,'-g')
legend('Resultado ARMAX','ARX','Valor experimental')

grid on
axis([0 600 38 42])
title('Valor de la temperatura, con To = 40°C')
```

Resultado:



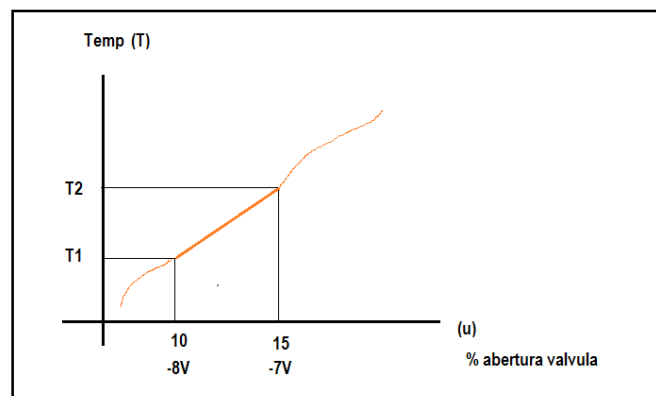
Se puede observar que el modelo ARMAX presenta mayor aproximación a la planta.

**2. Argumentar comparaciones y resultados entre el modelamiento no paramétrico temporal y frecuencial y el modelamiento paramétrico desarrollado en este laboratorio.**

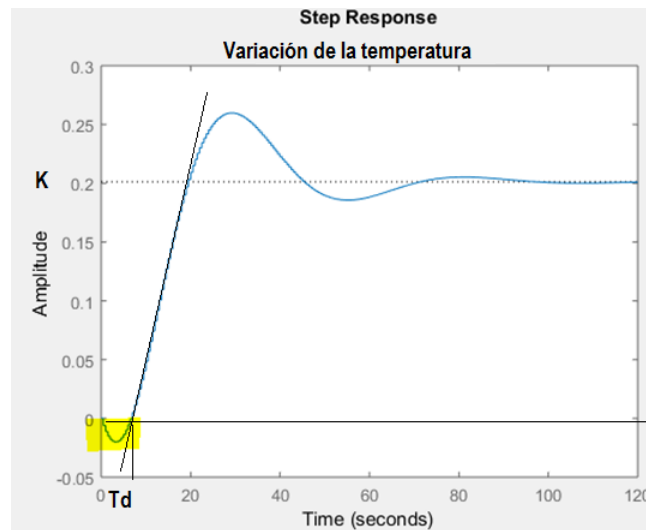
Mediante el modelo no paramétrico, se obtiene un retardo de:

$T_d = 5s$  (aproximadamente)

La región lineal:



Respuesta al escalón: (con amplitud A=15 – válvula abierta al 15%)



%% para el espacio de estados en tiempo continuo

```
clc
close all
A=m1.A;
B=m1.B;
C=m1.C;
D=m1.D;
t=0:0.5:100000;
t=t(1:length(u1));
% Compute the actual transfer function
[a b] = d2c(A,B,0.5)
[num den]=ss2tf(a,b,C,D)
sys=tf(num,den)
pole(sys)
lsim(sys,u1,t);grid
```

%%Estabilidad del sistema

```
eig(a)
```

%%calculo de la ecuacion en diferencias

```
sys =
```

$$\frac{-0.004511 s + 0.001166}{s^2 + 0.1022 s + 0.01732}$$

Continuous-time transfer function.

Donde obtenemos la ecuación en diferencias dado por:

$$Y[k] = -0.1022 \cdot Y[k-1] - 0.01732 \cdot Y[k-2] - 0.004511 \cdot X[k-1] + 0.001166 \cdot X[k-2]$$

### Del modelo ARMAX

Discrete-time ARMAX model:  $A(z)y(t) = B(z)u(t) + C(z)e(t)$

$$A(z) = 1 - 1.9z^{-1} + 0.906z^{-2}$$

$$B(z) = 0.0005721z^{-10}$$

$$C(z) = 1 - 1.54z^{-1} + 0.5401z^{-2}$$

Dada como ecuación en diferencias:

$$y[k] - 1.9y[k-1] + 0.906y[k-2] = 0.0005721u[k-10] + e[k] - 1.54e[k-1] + 0.5401e[k-2]$$

### Del modelo ARX

Discrete-time ARX model:  $A(z)y(t) = B(z)u(t) + e(t)$

$$A(z) = 1 - 0.5868z^{-1} - 0.3976z^{-2}$$

$$B(z) = 0.003682z^{-10}$$

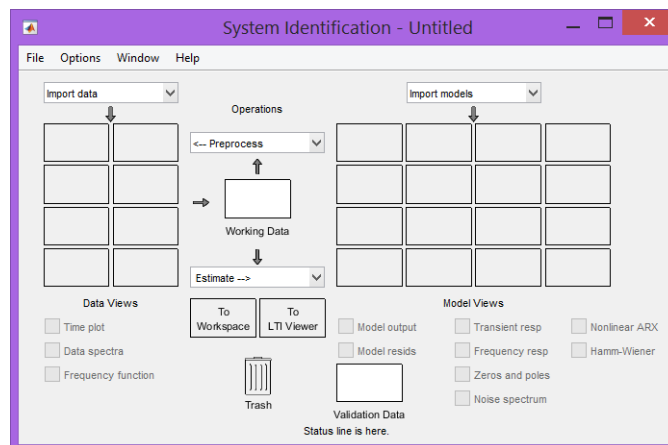
Dada como ecuación en diferencias:

$$y[k] - 0.5868y[k-1] - 0.3976y[k-2] = 0.003682u[k-10] + e[k]$$

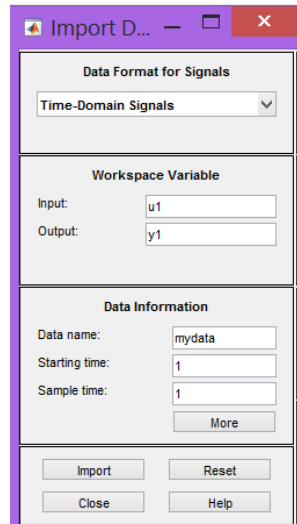
**3. presentar las diferencias encontradas entre la utilización de la interfaz gráfica ident de Matlab y el uso de la Línea de Comandos para la identificación de procesos utilizando el Toolbox de Identificación de Sistemas de Matlab.**

1.- Ejecutamos en Matlab toolbox "ident"

>>ident



2.- Ingresamos datos obtenidos en la planta "intercambiador de calor"-Valvula Apertura-Temperatura "import data"



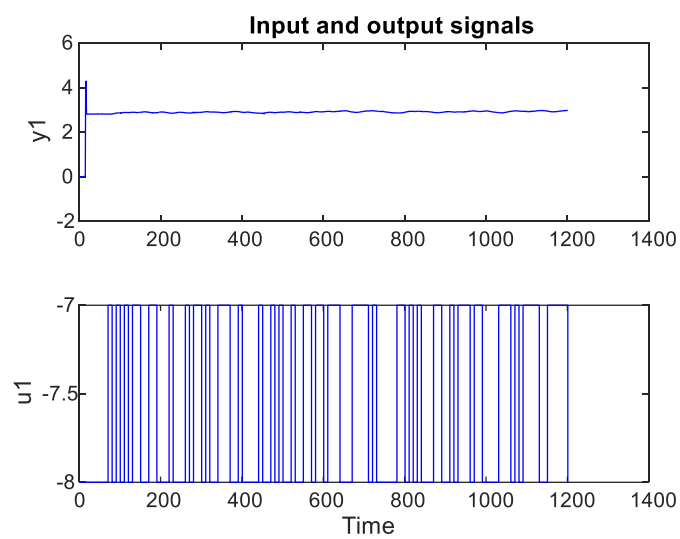
Import D...

Data Format for Signals  
Time-Domain Signals

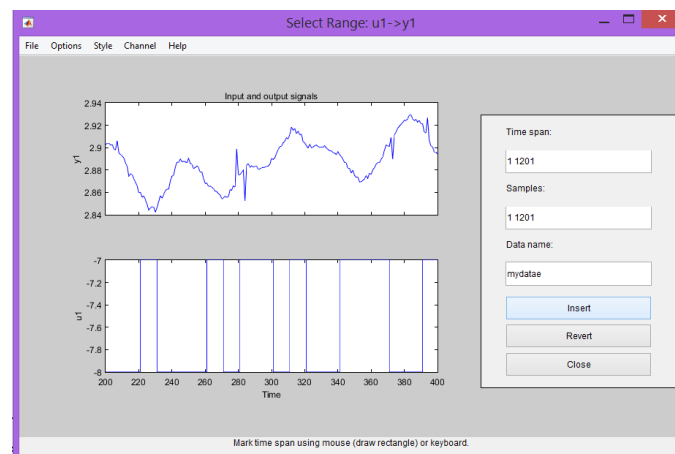
Workspace Variable  
Input: u1  
Output: y1

Data Information  
Data name: mydata  
Starting time: 1  
Sample time: 1  
More

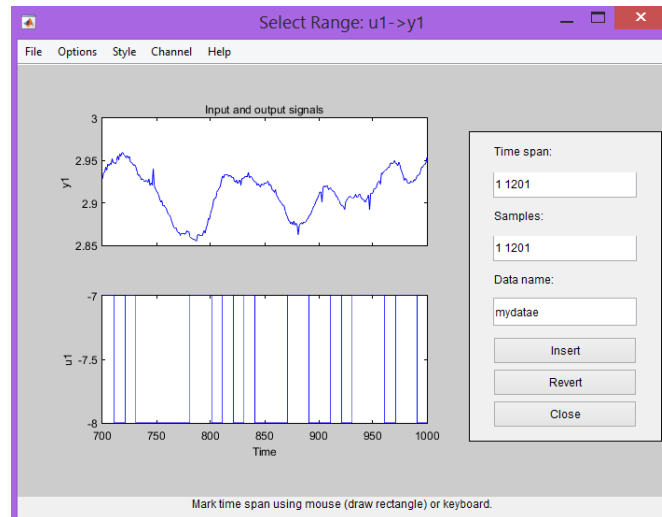
Import Reset  
Close Help



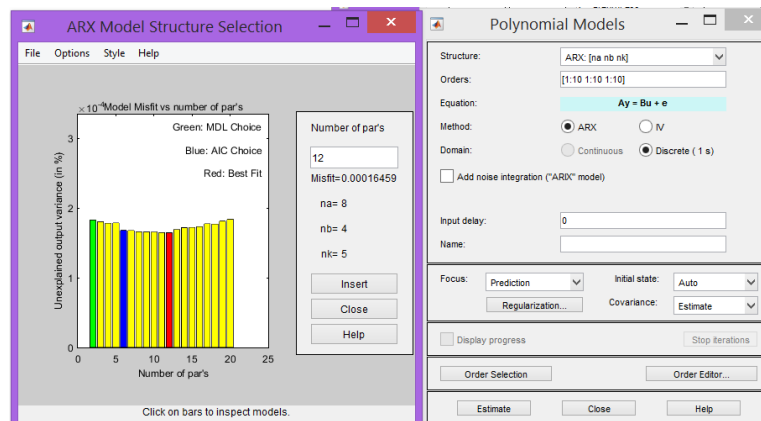
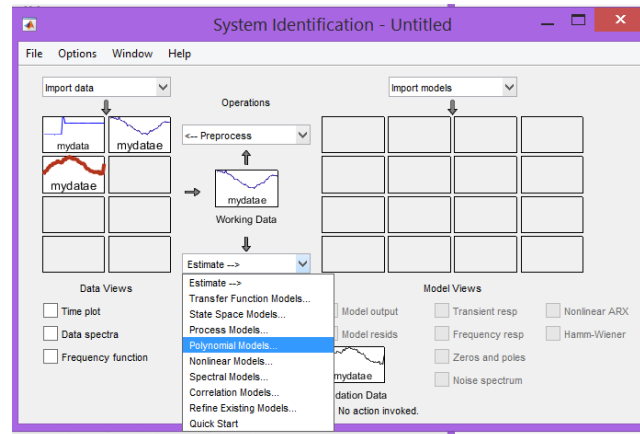
3.- Tomamos datos para la identificación [200-400] y validación [700-1000] los datos que da mejor estimación al sistema

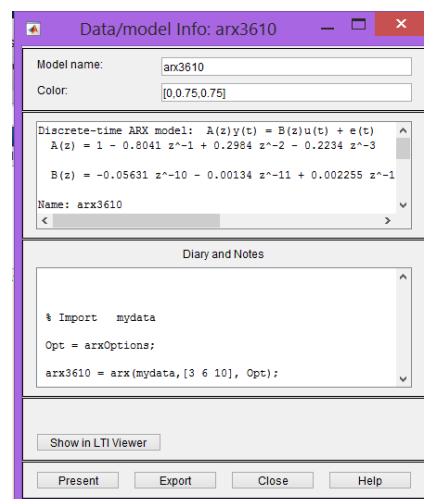
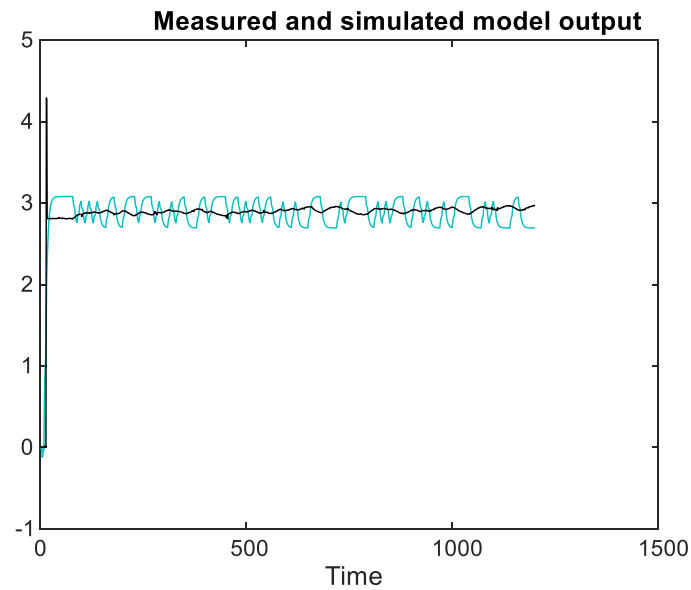




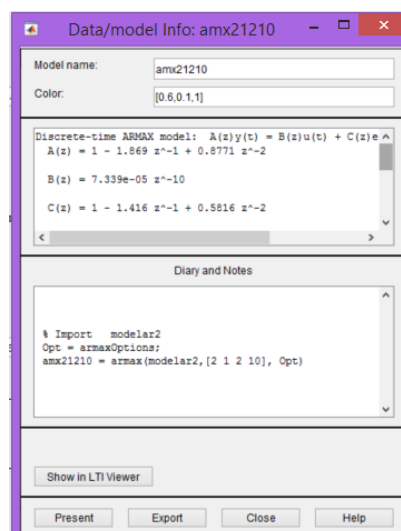


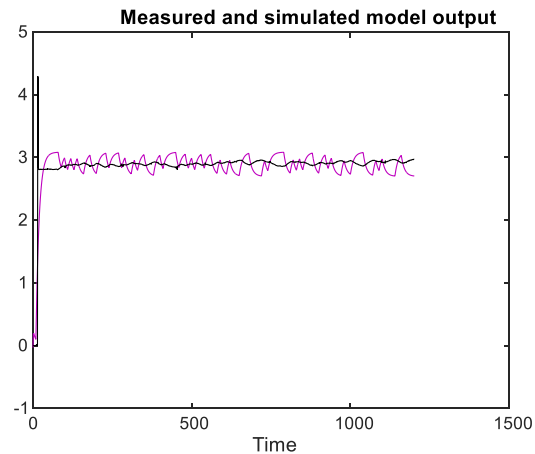
4.- Estimaremos el modelo ARX con la mejor estimación posible y el menor orden posible que nos da un mejor modelo





5.- Estimaremos el modelo ARMAX con la mejor estimación posible y el menor orden posible que nos da un mejor modelo





Del modelo ARMAX obtenido por el Ident gráfico :

$$A(z)y(t) = B(z)u(t) + C(z)e(t)$$

$$A(z) = 1 - 1.869 z^{-1} + 0.8771 z^{-2}$$

$$B(z) = 7.339 \cdot 10^{-5} z^{-10}$$

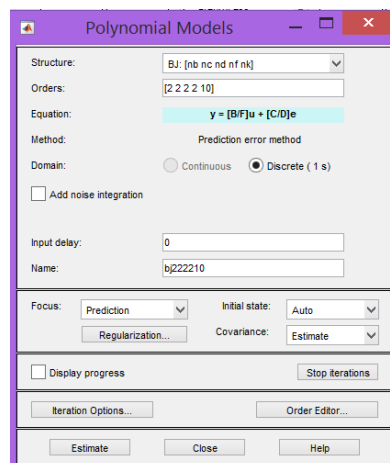
$$C(z) = 1 - 1.416 z^{-1} + 0.5816 z^{-2}$$

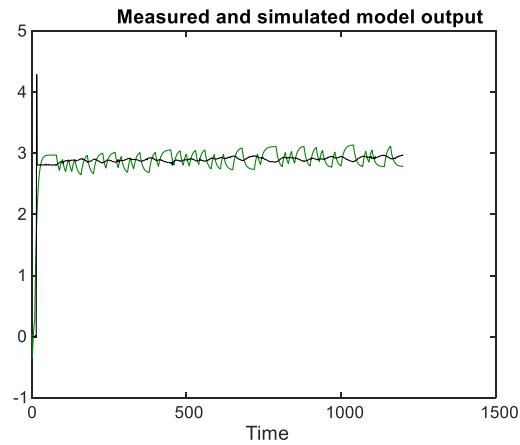
Dada como ecuación en diferencias:

$$y[k] - 1.869 y[k-1] + 0.8771 y[k-2] = 7.339 \cdot 10^{-5} u[k-10] + e[k] - 1.416 e[k-1] + 0.5816 e[k-2]$$

El modelo obtenido es muy semejante al modelo obtenido por ident código.

6.- Estimaremos el modelo BJ con la mejor estimación posible y el menor orden posible que nos da un mejor modelo





Data/model Info: bj222210

Model name:

Color:

Discrete-time BJ model:  $y(t) = [B(z)/F(z)]u(t) + [C(z)]$

$B(z) = -0.04081 z^{-10} + 0.04075 z^{-11}$

$C(z) = 1 - 0.1682 z^{-1} - 0.3564 z^{-2}$

$D(z) = 1 - 1.037 z^{-1} + 0.1193 z^{-2}$

Diary and Notes

```
% Import mydata
Opt = bjOptions;
nb = 2;
nc = 2;
nd = 2;
nf = 2;
```

Show in LTI Viewer

Present Export Close Help

## CONCLUSIONES

- I. La planta identificada, intercambiador de calor, resultó ser estable. Tanto en su modelo continuo como discreto.
- II. La planta resultó ser de fase no mínima, es decir posee un polo con parte real positiva. Por ello posiblemente sea más complicado de controlar que si hubiese resultado de fase mínima.
- III. El modelo identificado indica que la planta tiene un retardo de 5 segundos, ya que posee 10 muestras de retardo con tiempo de muestreo 0.5s.
- IV. El modelo ARMAX se aproxima a la planta más que el modelo ARX, tanto en estimación como en validación.
- V. El modelo ARMAX y el modelo en espacio de estados discreto, presentan aproximación a la planta prácticamente iguales.
- VI. La planta, al tener como salida la temperatura, resulta ser un sistema no relajado ya que  $y(0) = T_o \neq 0$ . Es por ello que se trabajó con la variación de temperatura como salida y con offset la temperatura inicial.
- VII. Se debe de seleccionar una señal PRBS con periodo de conmutación mayor al seleccionado, de tal manera que se produzcan mayores cambios de temperatura a la salida.
- VIII. Se trabajó con valores de apertura de la válvula de 10 a 15%, recomendados en el laboratorio. Estos valores representan los límites de la señal de control en la región lineal.
- IX. El desarrollo de la identificación obtuvo mejores resultados utilizando los comandos del **ident en código** respecto al **ident GUI**.

## OBSERVACIONES

- I. La temperatura de los termo resistores fue controlado manualmente por uno de los integrantes del grupo, éste representa una perturbación ya que no se mantiene constante en un solo valor sino que varía en un rango de valores dependientes del operador.
- II. Es preferible aumentar el tiempo de conmutación de la señal PRBS de tal manera que la variación de la temperatura sea más notoria.
- III. Hay que considerar que la velocidad de cambio de temperatura depende del sensor utilizado, es por eso que para otro tipo de sensor el sistema podría presentar una dinámica aparente más rápida o lenta.

## BIBLIOGRAFÍA

- I. Daniel Rodríguez Ramírez, Teodoro Álamo Cantarero, Ingeniería de Control - Identificación mediante el método de los mínimos cuadrados.
- II. Munther A. Dahleh. System Identification, Lecture6
- III. Ljung , Identification Systems
- IV. Jirka Roubal . Model identification, ARX ARMAX OE Model 2009