Escuela de Ingeniería Industrial. Universidad de Vigo. Curso 2022-23.
Grado en Ingeniería en Electrónica Industrial y Automática

LABORATORIO de INGENIERÍA de CONTROL-2

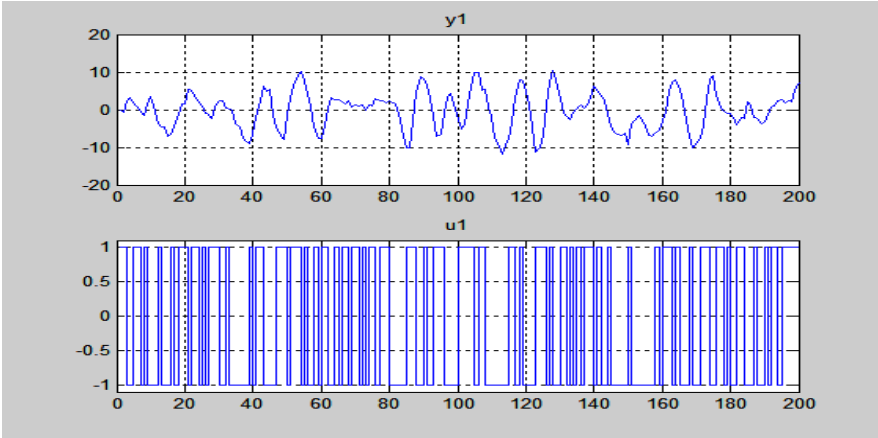
Práctica 6

Estimación paramétrica de la Función de Transferencia

En esta sesión práctica se introduce la "System Identification Toolbox" de Matlab, para identificación de modelos lineales en tiempo discreto, representados mediante Función de Transferencia, con estructuras ARX, ARMAX, Output-Error (OE), Box-Jenkins (BJ), etc.

Primero se hará una introducción a la Toolbox, mostrando cómo manejar modelos y señales (`idpoly, idinput, idplot`) órdenes (`na,nb,nk,nc,…`), identificación (`arx, armax,…`) y validación cruzada (`compare`).

Después de las explicaciones sobre las posibilidades de la Toolbox, se proporcionará a cada alumn@ una secuencia de datos {u(k),y(k)} de un sistema y se pedirá que realice el proceso de identificación, documentando los pasos y resultados en una breve memoria.



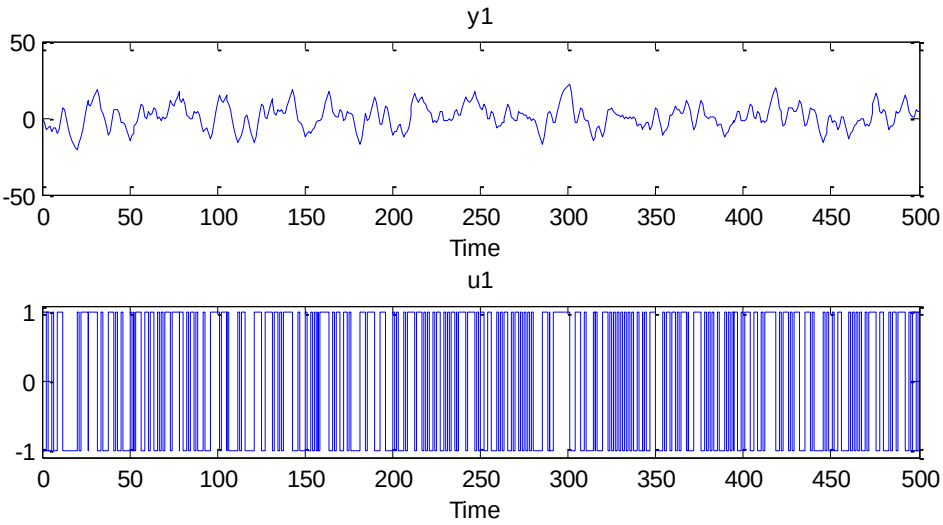

```
Las estructuras de modelo que maneja la Toolbox son:

    ARX:                A(q) y(t) = B(q) u(t-nk) + e(t)
    ARMAX:              A(q) y(t) = B(q) u(t-nk) + C(q) e(t)
    OE (Output Error):  y(t) = [B(q)/F(q)] u(t-nk) + e(t)
    BJ (Box-Jenkins):   y(t) = [B(q)/F(q)] u(t-nk) + [C(q)/D(q)] e(t)

donde los coeficientes polinómicos se codifican así:

A:    Polinomio A,  orden na, un vector fila  [1 a_1 a_2 ... a_na]
      Corresponde a:  A(q)y(t) = y(t) + a_1 y(t-1) + ... + a_na y(t-na)

B:    Polinomio B, órdenes [nb, nk], que corresponden a:
      B(q)u(t) = 0 u(t) + 0 u(t-1) +... + b_i u(t-i) + ... + b_j u(t-j)
            nk = i       (número de términos nulos iniciales)
            nb = j-i+1   (número de coeficientes b no nulos)

C,D:  Codificación similar a la de A
F:    Codificación similar a la de B
```

Ejemplo: Estimación de un proceso ARX. Para mostrar cómo funciona la *Toolbox*, primero generamos un modelo estable cualquiera, m0, y a partir del modelo (más ruido) simulamos datos de E/S (posteriormente estimaremos el modelo usando <u>*solo*</u> datos de E/S):

```
A = [ 1   -1.1   0.3];  % na=2:  n° de coeficientes del denominador
B = [ 0    3    1.5];  % nb=2/nk=1: n° coef.num./retardo u→y (n° ceros)
m0 = idpoly( A,B );

>> m0                                          % Modelo B(q)/A(q)
Discrete-time IDPOLY model: A(q)y(t) = B(q)u(t) + e(t)
A(q) = 1 - 1.1 q^-1 + 0.3 q^-2
B(q) =       3 q^-1 + 1.5 q^-2
```

A continuación simulamos 2*L datos de E/S, aplicando entrada 'u' aleatoria binaria y sumando ruido 'e' gaussiano

```
L= 250;
u = idinput( 2*L, 'rbs');
e = 0.25* randn(2*L, 1);
y = sim(m0, [u, e]);
z=[y,u];
figure(1); idplot(z);
```



La única información de la que se dispone son estos dos registros de las señales [y,u]=z. Debemos entonces tantear distintos órdenes del modelo ARX y <u>*hacer las estimaciones correspondientes utilizando una parte (p.ej., la mitad) de los datos:*</u>

```
% Estimacion modelos arx       (primeros L datos)
% [na, nb, nk]
% na = n° de coef. "a" en A
% nb = n° de coef. "b" en B
% nk = "retardo" E/S= n° de ceros iniciales en B
m221 = arx( z(1:L, 1:2), [2 2 1]);
m222 = arx( z(1:L, 1:2), [2 2 2]);
m111 = arx( z(1:L, 1:2), [1 1 1]);
```

Los modelos estimados salen:

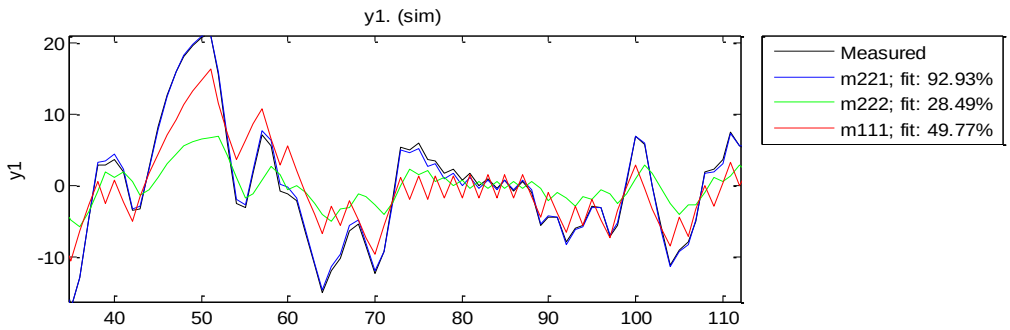
```
>> m221
Discrete-time IDPOLY model: A(q)y(t) = B(q)u(t) + e(t)
A(q) = 1 - 1.105 q^-1 + 0.3034 q^-2
B(q) =     3.005 q^-1 + 1.493 q^-2
Loss function 0.0564641 and FPE 0.058271

>> m222
Discrete-time IDPOLY model: A(q)y(t) = B(q)u(t) + e(t)
A(q) = 1 - 1.143 q^-1 + 0.35 q^-2
B(q) =                  1.318 q^-2 + 0.1124 q^-3
Loss function 8.93407 and FPE 9.21996

>> m111
Discrete-time IDPOLY model: A(q)y(t) = B(q)u(t) + e(t)
A(q) = 1 - 0.8961 q^-1
B(q) =      3.026 q^-1
Loss function 5.77859 and FPE 5.87104
```

Como estamos en simulación, la validez de los 3 modelos la podemos deducir comparando sus polinomios B(q)/A(q) con los polinomios originales B(q)/A(q) creados al principio.

Pero en una identificación práctica real, el modelo original no existiría. Para elegir el mejor de los modelos estimados podríamos fijarnos en los indicadores de error (Loss function, Final Prediction Error FPE); cuanto más pequeños, más exacto es el modelo.

Finalmente podemos hacer una validación comparativa de los modelos con la instrucción compare. Recordemos que *la validación se hace usando la otra parte (mitad) de los datos*.

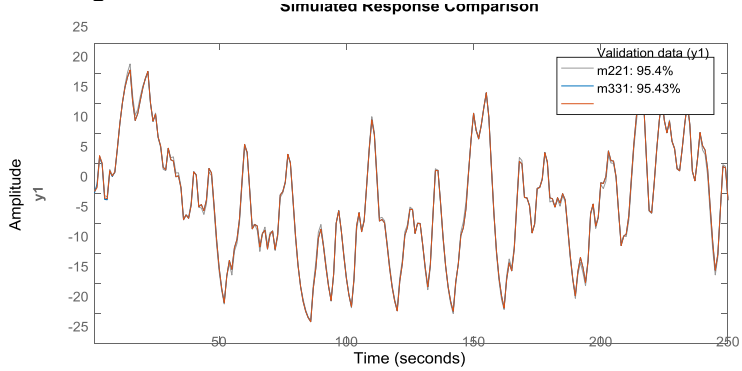
```
figure(2);                          % Validacion (ultimos L datos)
compare( z((L+1):(2*L), 1:2),    m221,m222,m111);
```



El gráfico muestra la salida registrada (negra, debida a la entrada aplicada) y la compara con las distintas salidas obtenidas pasando la misma entrada por los distintos modelos. La salida de modelo que más se parezca a la salida medida nos indica cuál es el mejor modelo. Esta mejor adecuación la confirma el porcentaje de fit (%). El mejor modelo es m221.

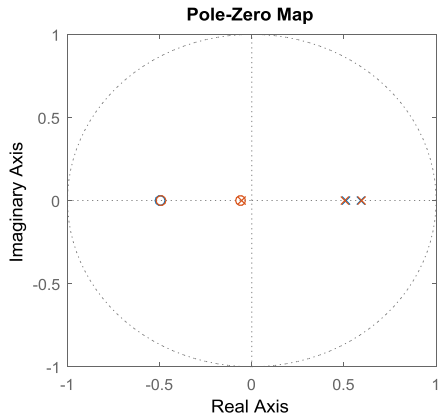
En una aplicación donde no tuviéramos ninguna información sobre el sistema podríamos intentar mejorar ese `fit` del 93% a base de probar otras estructuras (ARMAX, …) o/y otros órdenes. Por ejemplo si aumentamos en uno el grado de numerador y denominador, estimamos un ARX con esos ordenes (m331) y lo comparamos con el m221:

```
m331 = arx( z(1:L, 1:2), [3 3 1]);
figure(3);  compare( z((L+1):(2*L),1:2), m221, m331  );
```



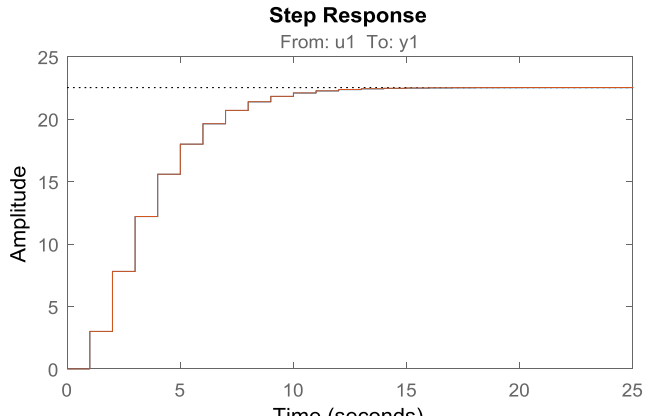
Vemos que la mejora del `fit` es insignificante, a costa de dos parámetros más, que parecen innecesarios. Podemos ver la distribución de raíces:

```
figure(4); pzmap(  m221, m331  )
```



El modelo m331 tiene un polo y cero adicional que son cancelables, generando m221, más simple. La equivalencia entre ambos se confirma también con las respuestas a escalón, que son indistinguibles.

```
figure(5); step(  m221, m331  )
```



Trabajo de Laboratorio

Se entregará a cada alumn@ un fichero .mat (dat11, dat12, …dat44) con muestras de E/S de un sistema desconocido. Al cargar el fichero (instrucción load) se crea una matriz z=[y,u], con dos columnas de 1000 muestras cada una, que representan la salida (y) y la entrada (u).

Se utilizará la primera mitad de datos para estimar y la segunda para validar

En una primera parte se estudiarán modelos ARX con rangos de órdenes

$$1 <= na <= 3, \qquad 1 <= nb <= 3, \qquad 0 <= nk <= 1$$

En la segunda parte se estudiarán modelos ARMAX con órdenes

$$1 <= na <= 3, \qquad 1 <= nb <= 3, \qquad 1 <= nc <= 2 \qquad nk = 1$$

En cada una de las dos partes se procede así:

- Se seleccionan los 2, 3 ó 4 modelos de mejor 'fit' tras la validación (primer criterio). Estos modelos tendrán un 'fit' prácticamente igual, por lo que el segundo criterio será el de simplicidad.

- El número de parámetros de un ARX es na+nb y el de un ARMAX es na+nb+nc. En igualdad de fit conviene elegir el modelo más sencillo (menos parámetros). Hacer una segunda selección y proponer el modelo final, en base a este criterio. Razonar brevemente la propuesta, usando si hace falta `pzmap` y/o `step` comparativos.

Hoja de resultados:

Apellidos y nombre: DNI=

r1= r2=

ARX: Modelos de mejor fit. Entre ellos, modelo(s) más sencillo(s). Breve discusión y propuesta de modelo final. Coeficientes de A(z), B(z).

ARMAX: Modelos de mejor fit. Entre ellos, modelo(s) más sencillo(s). Breve discusión y propuesta de modelo final. Coeficientes de A(z), B(z), C(z).

(máximo: 1 hoja/ 2 caras)