PRÁCTICA 5. Simulink

Periodo de realización: Semanas 7 y 8 del curso
Fecha límite de entrega: 29 de abril de 2012

Subir un único fichero **`apellido_P5.pdf`** con los modelos y submodelos que resuelven cada uno de los ejercicios y (2) las gráficas obtenidas en la simulación. Del Ejercicio 1 no hay que entregar nada. Si se resuelven los dos ejercicios opcionales, subir un *.zip con los ficheros generados.

1. Simulink. Ejercicios básicos

Ejercicio 1. Familiarización con el entorno.

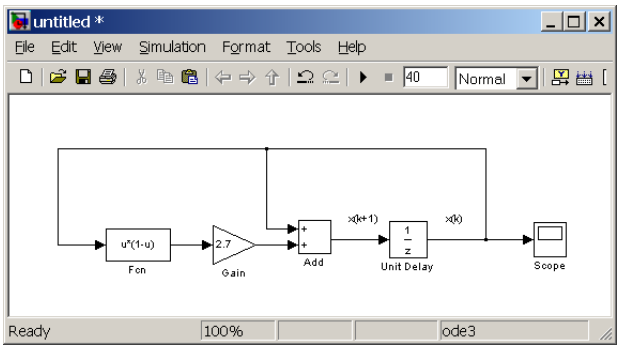
1) Dedicar unos minutos a ver qué bloques hay en cada una de las (sub)librerías de SIMULINK y qué opciones hay en la barra de menú de la ventana del modelo.

2) Buscar bloques de nombre "PID Controller" (correspondiente a un controlador proporcional-integral-derivativo)

3) Ejecutar el modelo de demostración `thermo.mdl`. Para abrirlo, basta con escribir `>>thermo` en la ventana de comandos de MATLAB. Investigar las diferentes opciones. □

Ejercicio 2. Ecuación en diferencias no lineal: Dinámica de una población.

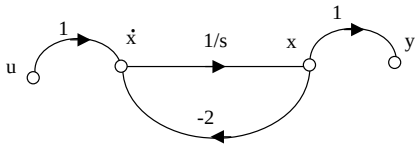
1) Construir el modelo del sistema discreto definido por $x_{k+1} = x_k + rx_k(1-x_k)$. Pista:



2) Representar su comportamiento para los casos r = 0.2 y r = 2.7. Escoger como condición inicial 0.5, como tiempo de muestreo 0.8 y como tiempo final de simulación 40*s*.

3) Parametrizar el `Scope` a fin de que cargue el resultado en el *workspace* de Matlab, simular el modelo desde la ventana de comandos (función `sim`) y representar $x(k)$ con ayuda de la función `stairs`. □

Ejercicio 3. Ecuación diferencial. La figura muestra el flujograma de estado correspondiente a la ecuación diferencial $\dot{x}(t) = -2x(t) + u(t)$. La entrada del flujograma es *u* y la salida es *y*=*x*. Notar que, por razones de implementabilidad, se prefiere trabajar con integradores (1/*s*) en vez de con derivadores. Se pide:

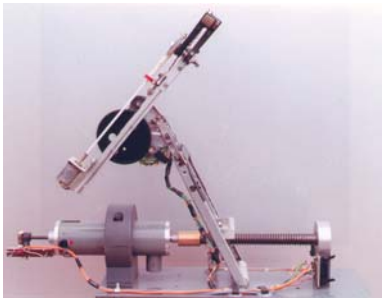


1) Construir el modelo equivalente en Simulink (bloques: `Integrator`, `Sum`, `Gain`, `Scope`). Elegir un bloque de la librería `Sources` que permita generar una *u*(*t*) del tipo señal cuadrada de amplitud 1 y frecuencia 1rad/s.

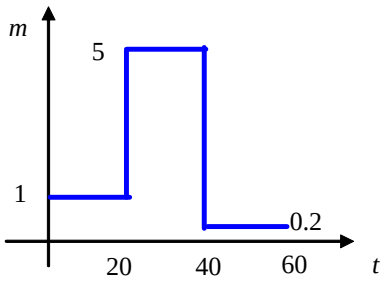
2) Simular y visualizar, en un mismo `Scope`, las señales *x*(*t*) y *u*(*t*). Para ello utilizar un bloque multiplexor (`Mux`) antes del `Scope`. □

2. Simulink. Ejercicios de aplicación

Ejercicio 4. Brazo robótico. Comportamiento en lazo abierto. Se dispone del manipulador de 2 grados de libertad de la Figura (a). Su carga resulta variable y se modela por medio de una masa *m* que varía con el tiempo entre 0.2 y 5kg tal y como muestra la Figura (b).



(a)



(b)

Sin ningún tipo de control (*lazo abierto*), el elemento posicionador (*planta*) presenta la siguiente función de transferencia

$$P(s) = \frac{40}{ms^2 + 10s + 20}.$$

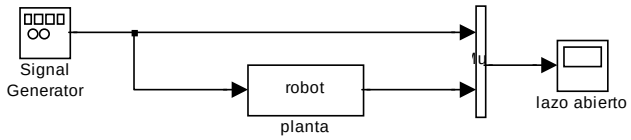
Se desea representar, vía Simulink, la evolución de la salida *y*(*t*) cuando la entrada *r*(*t*) es un tren de pulsos de amplitud 1 y frecuencia 0.07Hz. Para ello, seguir los pasos que se indican a continuación:

1) Buscar en qué directorio de Matlab se encuentra el fichero plantilla **`csfunc.m`** (descripción de sistemas continuos para usar dentro de una función S). Copiarlo al directorio de trabajo <work> y cambiarle el nombre (llamarlo por ejemplo **`robot.m`**). Modificarlo con los datos de la planta considerada *P*(*s*) tal y como se indica a continuación (intentar entender qué hace cada parte del programa):

```
function [sys,x0,str,ts] = robot(t,x,u,flag)

m=1;
if t>=20;m=5;end
if t>=40;m=0.2;end
A=[0 1;-20/m -10/m];B=[0;40/m];C=[1 0];D=0;

switch flag,
case 0,    [sys,x0,str,ts]=mdlInitializeSizes(A,B,C,D);
case 1,    sys=mdlDerivatives(t,x,u,A,B,C,D);
case 3,    sys=mdlOutputs(t,x,u,A,B,C,D);
case { 2, 4, 9 },    sys = [];
otherwise    error(['Unhandled flag = ',num2str(flag)]);
end

function [sys,x0,str,ts]=mdlInitializeSizes(A,B,C,D)
sizes = simsizes;
sizes.NumContStates  = 2;  %2 var de estado continuas (x=[x1;x2])
sizes.NumDiscStates  = 0;
sizes.NumOutputs     = 1;  %una única salida (y)
sizes.NumInputs      = 1;  %una única entrada (r)
sizes.DirFeedthrough = 1;
sizes.NumSampleTimes = 1;
sys = simsizes(sizes);
x0  = zeros(sizes.NumContStates,1); %condiciones iniciales nulas
str = [];
ts  = [0 0];

function sys=mdlDerivatives(t,x,u,A,B,C,D)
sys = A*x + B*u;

function sys=mdlOutputs(t,x,u,A,B,C,D)
sys = C*x + D*u;
```

Notar que para realizar la simulación numérica de sistemas dinámicos siempre hay que expresar éstos por medio de ecuaciones de estado (de ahí la aparición de las matrices **A**, **B**, **C**, **D**). Notar asimismo que, gracias a las funciones S, Simulink es capaz de simular el comportamiento de sistemas no lineales y/o variantes con el tiempo.

2) Construir el siguiente modelo Simulink (especificar el fichero **`robot.m`** dentro del bloque *S-function*) y simular la respuesta hasta 60s:

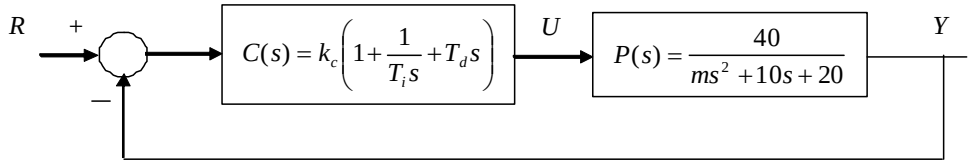


□

Ejercicio 5. Brazo robótico. Control PID. Puesto que el comportamiento del sistema sin controlar del **Ejercicio 4** no nos convence, probaremos distintas configuraciones de control a ver si podemos mejorarlo.

Se desea controlarlo de manera que su comportamiento (*en lazo cerrado*) corresponda a la función de transferencia $M(s) = \frac{4}{s+4}$.

El esquema de bloques de la figura muestra la configuración de control que habrá que traducir a Simulink. El controlador $C(s)$ es un PID puesto que presenta acción proporcional, derivativa e integral.



1) PID con cancelación . Representar en un mismo Scope la salida $y(t)$ y la entrada $r(t)$ (tren de pulsos de amplitud 1 y frecuencia 0.07Hz) para el caso en que los parámetros del controlador PID son los siguientes:

$$C(s)=k_c\left(1+\frac{1}{T_i s}+T_d s\right) \text{ con } k_c = 1,\ T_i = 0.5,\ T_d = 0.1$$

Representar también el esfuerzo de control (señal $u(t)$) en otro Scope.

(Nota: Utilizar el bloque predefinido de Simulink que implementa el PID ideal (librería *Simulink Extras → Additional Linear*), pero vigilar la entrada de parámetros: En Simulink no se entran k_c, T_i, T_d sino los parámetros P, I, D. En el mismo bloque está la definición)

(Nota: Para entender el nombre "cancelación" notar que el controlador resultante es tal que su numerador coincide con el denominador de la planta para m=1:

$$C(s)=k_c\frac{T_d s^2+s+1/T_i}{s}=0.1\frac{s^2+10s+20}{s}$$

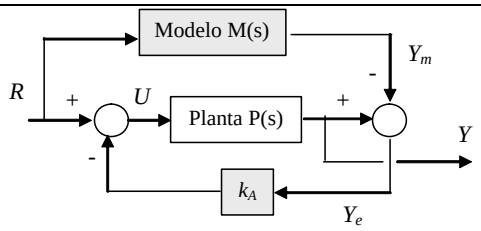
con lo cual, la función de transferencia del lazo resultante queda como $L(s)=0.1\frac{s^2+10s+20}{s}\frac{40}{s^2+10s+20}=\frac{4}{s}$ y la función de transferencia en lazo cerrado es la deseada, $T(s)=\frac{L(s)}{1+L(s)}=\frac{4}{s+4}$. Notar que podríamos haber diseñado $C(s)$ para cancelar el caso m=5 o el caso m=0.2 también).

2) PID con ceros de anclaje y polo lejano (controlador robusto): Repetir el apartado anterior para la siguiente elección de parámetros. Comparar y comentar los resultados:

$$C(s)=\frac{s^2+12s+70}{s^2+150s}$$

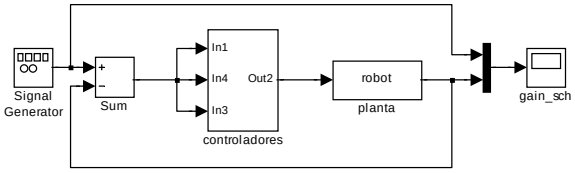
□

Ejercicio 6. Brazo robótico. Control adaptativo por modelo de referencia. Implementar ahora en Simulink esta otra configuración de control. El modelo es $M(s)=\frac{4}{s+4}$, la planta es el robot del **Ejercicio 4** y la ganancia del lazo k_A tiene que ser grande (probar diversos valores y comentar el resultado).



Ejercicio 7. Brazo robótico. Programación de la ganancia (*gain scheduling control*). Puesto que la evolución temporal de *m* es conocida *a priori* (ver **Ejercicio 4**) es posible diseñar 3 controladores PID con cancelación del denominador de la planta, uno para cada valor de *m* (ver **Ejercicio 5**), y hacer que entre en funcionamiento uno u otro dependiendo del instante *t*. Se pide implementar esta estrategia de control.

Pistas: Hay que usar un bloque *clock* para conocer el instante temporal y los bloques de comparaciones booleanas necesarios. Hay que usar subsistemas y elementos que activen (*enable*) estos subsistemas cuando sea necesario. El modelo general tendrá esta apariencia:



Ejercicio 8. Identificación *on-line* de parámetros. Un determinado proceso puede modelarse como un filtro pasa bajas de segundo orden de la forma $G(s)=\dfrac{k}{s^2+2\zeta\omega_n s+\omega_n^2}$. Se sabe que su frecuencia natural es $\omega_n = 3$, pero se desconoce el valor exacto de su ganancia *k* y su coeficiente de amortiguamiento ζ.

Se trata de implementar un esquema que identifique en línea estos dos parámetros. Suponer que $\zeta = 0.3$ y $k = 9$ son los valores desconocidos.

En concreto se implementará la regla del MIT.

Regla del MIT. Puesto que el error depende de los parámetros estimados $\hat{\mathbf{a}}$, $\mathbf{e}=\mathbf{e}(t,\hat{\mathbf{a}})$, se trata de variar $\hat{\mathbf{a}}$ hasta conseguir $\mathbf{e}$=0, momento en que $\hat{\mathbf{a}}=\mathbf{a}$.

Se considera la función de Lyapunov (definida positiva) del error $V(\mathbf{e})=\frac{1}{2}\mathbf{e}^T\mathbf{Q}\mathbf{e}$, donde $\mathbf{Q}$ es una matriz simétrica definida positiva.

Puesto que $V(\mathbf{e})$ es una función definida positiva, si su gradiente $\dot{V}$ es negativo, ello significa que $\mathbf{e}$ tenderá asintóticamente al origen (su valor mínimo). Por ello, se fuerza a que el vector de parámetros evolucione según el sentido de descenso del gradiente, $\dot{\hat{\mathbf{a}}}=-\varepsilon\left(\dfrac{\partial V}{\partial\hat{\mathbf{a}}}\right)^T$, ε>0. Ello

nos lleva a $\dot{\hat{\mathbf{a}}} = \varepsilon \mathbf{S}^T \mathbf{Q}\mathbf{e}$, siendo $\mathbf{S}$ la matriz de sensibilidad. El parámetro ε influye en la velocidad de convergencia.

Nota: El gradiente es $\frac{\partial V}{\partial \hat{\mathbf{a}}} = \frac{\partial V}{\partial \mathbf{e}} \frac{\partial \mathbf{e}}{\partial \hat{\mathbf{a}}}$ donde, si la matriz $\mathbf{Q}$ es cuadrada y simétrica, tenemos $\frac{\partial V}{\partial \mathbf{e}} = \frac{1}{2}\left(\frac{\partial \mathbf{e}^T}{\partial \mathbf{e}} \mathbf{Q}\mathbf{e} + \mathbf{e}^T \frac{\partial (\mathbf{Q}\mathbf{e})}{\partial \mathbf{e}} \right) = \frac{1}{2}\left(\mathbf{Q}\mathbf{e} + \mathbf{e}^T \mathbf{Q}^T\right) = \mathbf{Q}\mathbf{e} = (\mathbf{Q}\mathbf{e})^T$. Aquí hemos usado las siguientes expresiones de derivación vectorial: $\frac{\partial (\mathbf{c}^T \mathbf{x})}{\partial \mathbf{x}} = \mathbf{c}$ y $\frac{\partial (\mathbf{A}\mathbf{x})}{\partial \mathbf{x}} = \mathbf{A}^T$. Y la matriz de sensibilidad es $\mathbf{S} = -\frac{\partial \mathbf{e}}{\partial \hat{\mathbf{a}}} = -\frac{\partial (\mathbf{x} - \hat{\mathbf{x}})}{\partial \hat{\mathbf{a}}} = \frac{\partial \hat{\mathbf{x}}}{\partial \hat{\mathbf{a}}} = \begin{bmatrix} \frac{\partial \hat{x}_1}{\partial \hat{\zeta}} & \frac{\partial \hat{x}_1}{\partial \hat{k}} \\ \frac{\partial \hat{x}_2}{\partial \hat{\zeta}} & \frac{\partial \hat{x}_2}{\partial \hat{k}} \end{bmatrix}$.

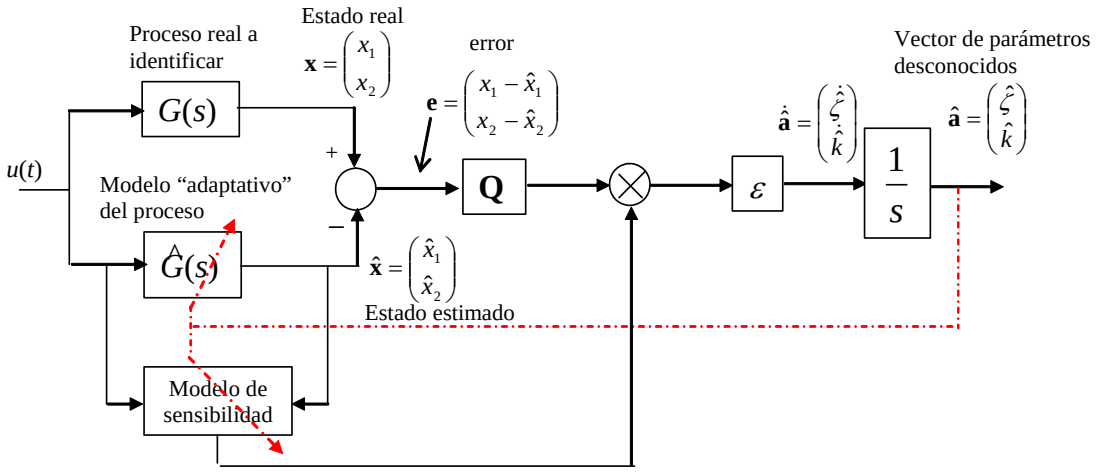


Fig. 1. Esquema de la regla del MIT

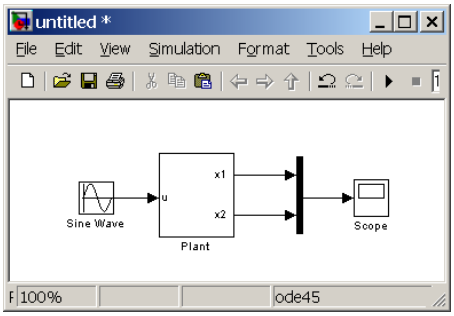
Se pide:

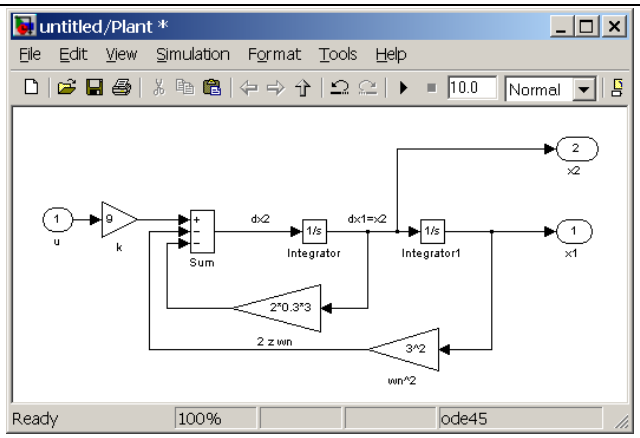
1) Proceso real: Construir un subsistema en SIMULINK que implemente el flujograma de señal del proceso real, $\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, u, \mathbf{a})$. Tendrá que tener dos salidas: x_1 y x_2. Para ello, hacer lo siguiente y verificar que funciona. Tomar como excitación un seno $u(t)$ de amplitud unidad y frecuencia $\omega = 1$ *rad/s*.

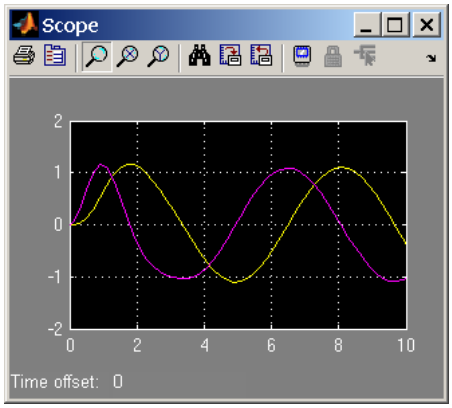
Notar que no se va a trabajar con bloques *Transfer Function* sino que se implementan directamente las ecuaciones de estado $\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, u, \mathbf{a})$ del sistema

$$\left.\begin{aligned} \dot{x}_1 &= x_2 \\ \dot{x}_2 &= -\omega_n^2 x_1 - 2\zeta\omega_n x_2 + ku \end{aligned}\right\}$$

mediante bloques *Integrator*, *Sum*, *Gain*.







2) Modelo adaptativo: Construir un subsistema en SIMULINK que implemente un modelo adaptativo con la misma estructura del proceso, $\dot{\hat{\mathbf{x}}} = \mathbf{f}(\hat{\mathbf{x}}, u, \hat{\mathbf{a}})$, pero que permita variar los valores de los parámetros $\hat{\zeta}$ y $\hat{k}$. Tendrá que tener dos entradas adicionales $\hat{\zeta}$ y $\hat{k}$. Verificar que funciona excitando con la misma $u(t)$ del apartado anterior y con sendos escalones (*Step*) de valor $\hat{\zeta}$ = 0.3 y $\hat{k}$ = 9, y comprobando que el resultado es el mismo que en el apartado anterior (sistema real).

Pista: Las ecuaciones de estado del modelo adaptativo son

$$\left.\begin{aligned} \dot{\hat{x}}_1 &= f_1(\hat{\mathbf{x}}, u, \hat{\mathbf{a}}) = \hat{x}_2 \\ \dot{\hat{x}}_2 &= f_2(\hat{\mathbf{x}}, u, \hat{\mathbf{a}}) = -\omega_n^2 \hat{x}_1 - 2\hat{\zeta}\omega_n \hat{x}_2 + \hat{k}u \end{aligned}\right\}$$

(Posteriormente lo que se hará será obtener la diferencia entre el estado del sistema real y el estado del modelo adaptativo, $\mathbf{e} = \mathbf{x} - \hat{\mathbf{x}}$)

3) Modelo de sensibilidad: La matriz de sensibilidad $\mathbf{S} = \dfrac{\partial \hat{\mathbf{x}}}{\partial \hat{\mathbf{a}}}$ también varía con el tiempo.

Puesto que $\dot{\hat{\mathbf{x}}} = \mathbf{f}(\hat{\mathbf{x}}, u, \hat{\mathbf{a}})$, las ecuaciones dinámicas del modelo de sensibilidad son las siguientes:

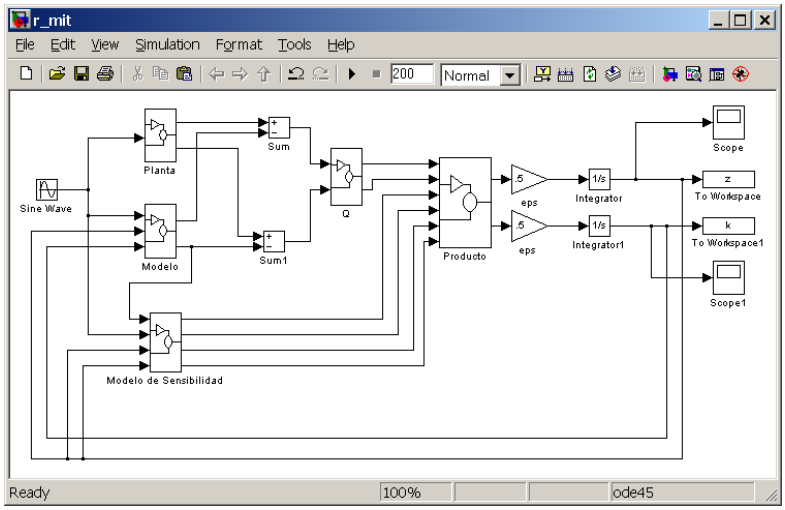
$$\frac{\partial \dot{\hat{\mathbf{x}}}}{\partial \hat{\mathbf{a}}} = \frac{\partial \mathbf{f}}{\partial \hat{\mathbf{x}}}\frac{\partial \hat{\mathbf{x}}}{\partial \hat{\mathbf{a}}} + \frac{\partial \mathbf{f}}{\partial \hat{\mathbf{a}}} \quad \Rightarrow \quad \dot{\mathbf{S}} = \frac{\partial \mathbf{f}}{\partial \hat{\mathbf{x}}}\mathbf{S} + \frac{\partial \mathbf{f}}{\partial \hat{\mathbf{a}}}$$

$$\begin{bmatrix} \frac{\partial \dot{\hat{x}}_1}{\partial \hat{\zeta}} & \frac{\partial \dot{\hat{x}}_1}{\partial \hat{k}} \\ \frac{\partial \dot{\hat{x}}_2}{\partial \hat{\zeta}} & \frac{\partial \dot{\hat{x}}_2}{\partial \hat{k}} \end{bmatrix} = \begin{bmatrix} \frac{\partial f_1}{\partial \hat{x}_1} & \frac{\partial f_1}{\partial \hat{x}_2} \\ \frac{\partial f_2}{\partial \hat{x}_1} & \frac{\partial f_2}{\partial \hat{x}_2} \end{bmatrix} \cdot \begin{bmatrix} \frac{\partial \hat{x}_1}{\partial \hat{\zeta}} & \frac{\partial \hat{x}_1}{\partial \hat{k}} \\ \frac{\partial \hat{x}_2}{\partial \hat{\zeta}} & \frac{\partial \hat{x}_2}{\partial \hat{k}} \end{bmatrix} + \begin{bmatrix} \frac{\partial f_1}{\partial \hat{\zeta}} & \frac{\partial f_1}{\partial \hat{k}} \\ \frac{\partial f_2}{\partial \hat{\zeta}} & \frac{\partial f_2}{\partial \hat{k}} \end{bmatrix}$$

$$\begin{bmatrix} \dot{S}_{11} & \dot{S}_{12} \\ \dot{S}_{21} & \dot{S}_{22} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -\omega_n^2 & -2\hat{\zeta}\omega_n \end{bmatrix} \cdot \begin{bmatrix} S_{11} & S_{12} \\ S_{21} & S_{22} \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ -2\omega_n \hat{x}_2 & u \end{bmatrix}$$

Se pide construir un subsistema que las implemente. Notar que al tener 4 variables de estado deberá contener 4 integradores. Las entradas al subsistema serán u, $\hat{x}_2$ y $\hat{\zeta}$ y las salidas serán las 4 variables de estado S_{ij}.)

4) Realizar las conexiones necesarias entre todos los subsistemas y añadir visualizadores. Pista: La figura muestra una posible configuración. Se sugiere personalizar los subsistemas mediante máscaras. En este ejercicio, por simplicidad, tomaremos **Q** = **I**.



5) Representar la evolución temporal de $\hat{\zeta}$ y $\hat{k}$ (tomar tiempo final 150*s* y condiciones iniciales $\hat{\zeta}$ =1 y $\hat{k}$ =5). ¿A qué valores converge la estimación?

6) Representar la evolución temporal del vector de estado **x** y su estimación $\hat{\mathbf{x}}$.

7) Estudiar el efecto de ε en la convergencia de la estimación de los parámetros k y ζ. Simular con $\varepsilon = 0.7$ y $\varepsilon = 2$. □

Ejercicio 9. Stateflow (opcional). Proponer y resolver con Stateflow un ejemplo sencillo (por ejemplo, alguna aplicación de control secuencial, semáforo…) □

Ejercicio 10. Animación (opcional). Implementar un efecto de animación sencillo usando una función S y las herramientas GUI (por ejemplo, movimiento de un péndulo, de un carrito con muelle y amortiguamiento,...) Ver ejemplo en los apuntes. □