Exercises Unit 5. Simulink

Working period: Weeks 9 and 10
Due date: 5 May 2013

Submit one file **`my_name_E5.pdf`** containing (1) Simulink models and submodels that solve the proposed exercises, and (2) the resulting plots. Exercise 1 may not be included in the solutions file. If you decide to solve any of the optional exercises, then you must include all the generated files (*.mdl, *.m,…) in a *.zip or *.rar file.

1. Simulink. Basic exercises

Exercise 1. Introduction to Simulink.

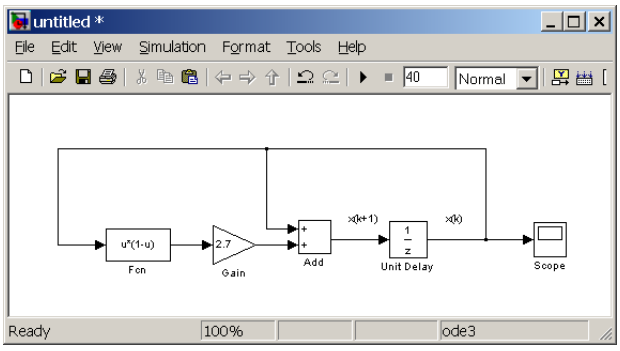
1) Take a look to the blocks that can be found in the Simulink (sub)libraries. Take a look to the available options in the model window menu and toolbar.

2) Look for blocks named "PID Controller" (corresponding to a proportional-integral-derivative controller).

3) Execute the demo file `sldemo_househeat.mdl`. (or `sldemo_househeat.slx` in v8). To open it, type `>> sldemo_househeat` in the Matlab command window. Change the different parameters and see what happens. □

Exercise 2. Non linear difference equation: Dynamics of a population.

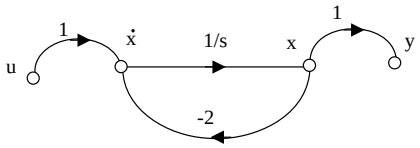
1) Build the model corresponding to the discrete system defined as $x_{k+1} = x_k + rx_k(1-x_k)$. Hint:



2) Plot its behavior for the cases r = 0.2 and r = 2.7. Take initial condition 0.5, sampling period 0.8s and final time for the simulation 40s.

3) Parameterize the `Scope` block so that it passes the result to the MATLAB *workspace*, run the model from the command window (function `sim`) and plot $x(k)$ with `stairs`. □

Exercise 3. Differential equation. The next figure shows the state flux diagram corresponding to the differential equation $\dot{x}(t) = -2x(t) + u(t)$. The input to the flux diagram is u and the output is the state $y=x$. Note that, for implementation reasons, we prefer to use integrators ($1/s$) rather than derivations (s).

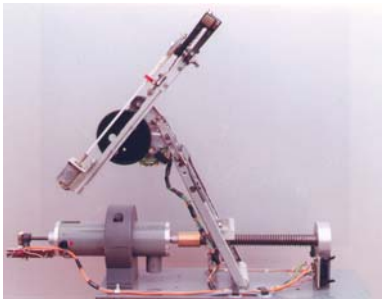


1) Build the equivalent Simulink model (blocks: `Integrator`, `Sum`, `Gain`, `Scope`). Choose the appropriate block in the `Sources` library to generate a square signal $u(t)$ of amplitude 1 and frequency 1rad/s.

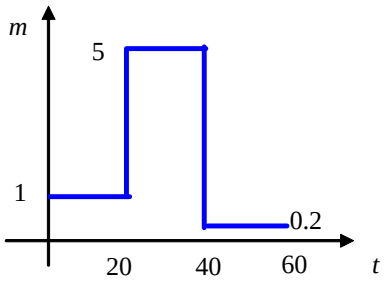
2) Plot in the same `Scope` the two signals $x(t)$ and $u(t)$. To do so, put a multiplexor block (`Mux`) before the `Scope`. □

2. Simulink. Application exercises

Exercise 4. Robot arm. Open loop behavior. Consider the two degrees of freedom manipulator in the figure (a). The load is modeled as a mass m that varies with time between 0.2 and 5kg as shown in the figure (b).



(a)



(b)

With no control (*open loop*), the positioning element (*plant*) presents the following transfer function

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

We want to study the time evolution of the output $y(t)$ when the input $r(t)$ is a square signal of amplitude 1 and frequency 0.05Hz. To do so, follow the steps listed below:

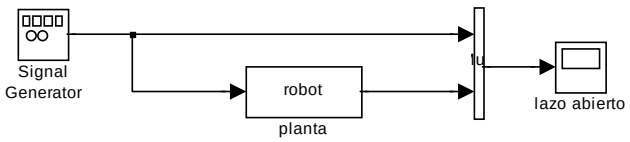
1) Look for the template file named **`csfunc.m`** (continuous time systems for use in S functions). Copy it to the working directory <work> and change its name (call it **`robot.m`**, for instance). Modify it with the data corresponding to the considered plant $P(s)$ (try to understand the code below):

```
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 cont time state vars (x=[x1;x2])
sizes.NumDiscStates  = 0;
sizes.NumOutputs     = 1;   %only one output (y)
sizes.NumInputs      = 1;   %only one input (r)
sizes.DirFeedthrough = 1;
sizes.NumSampleTimes = 1;
sys = simsizes(sizes);
x0  = zeros(sizes.NumContStates,1);   %zero initial conditions
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;
```

Note that the numerical simulation of dynamic systems is always performed by means of the state equations (for this reason matrices **A**, **B**, **C**, **D** have appeared). Note also that S functions are able to simulate nonlinear and/or time-varying systems.

2) Build the following Simulink model (specify the file `robot.m` in the *S-function* block). Run the simulation up to 60s.

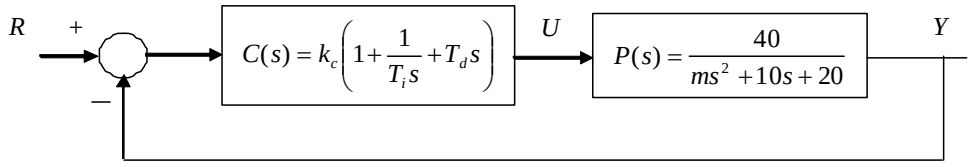


□

Exercise 5. Robot arm. PID control. Since we do not like the robot arm behavior in open loop, we are going to try different closed loop control configurations in order to improve the system performance.

We would like a controlled (*closed loop*) behavior corresponding to the following transfer function $M(s)=\dfrac{4}{s+4}$.

Next block diagram shows the control configuration that has to be translated into a Simulink model. The controller $C(s)$ is a PID since it provides proportional, integrative and derivative actions.



1) Cancellation PID: Plot in a same Scope the output $y(t)$ and the input $r(t)$ (square signal) for the following selection of the PID controller parameters:

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

(Note 1: Use the Simulink block that implements the ideal PID (*Simulink Extras → additional Linear*), but be careful with the parameters: You do not have to enter k_c, T_i, T_d, but the parameters P, I, D instead. See the definition of such parameters in the PID block)

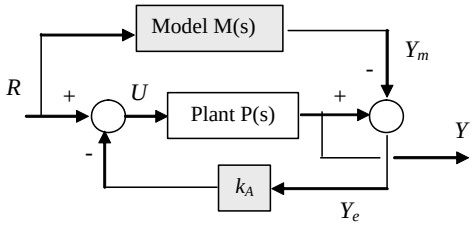
(Note 2: To understand the term "cancellation", note that the controller numerator is the plant denominator for m=1:

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

Hence the loop transfer function is $L(s)=0.1\frac{s^2+10s+20}{s}\frac{40}{s^2+10s+20}=\frac{4}{s}$ and the closed loop transfer function is the desired one, $T(s)=\frac{L(s)}{1+L(s)}=\frac{4}{s+4}$. We can design cancellation PIDs for the cases m=5 and m=0.2 as well).

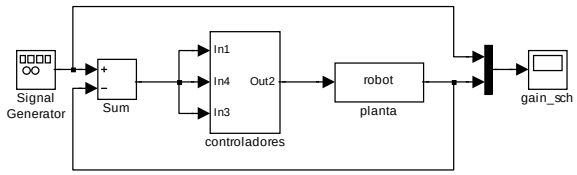
2) PID with anchoring zeros and distant pole (robust controller): Repeat the exercise for the controller given by $C(s)=\frac{s^2+12s+70}{s^2+150s}$. Compare and comment the obtained results. □

Exercise 6. Robot arm. Model reference adaptive control (MRAC). Now implement in a Simulink model the following control configuration. The reference model is $M(s)=\frac{4}{s+4}$, the plant is the robot arm $P(s)=\frac{40}{ms^2+10s+20}$, and the loop gain k_A must be large enough (try different values and comment the obtained results).



□

Exercise 7. Robot arm. Gain scheduling. Since the time evolution of parameter *m* is *a priori* known, it is possible to design three cancellation PIDs each one for each value of parameter *m* and enable the corresponding controller depending of the particular time instant. Implement in Simulink this control strategy.



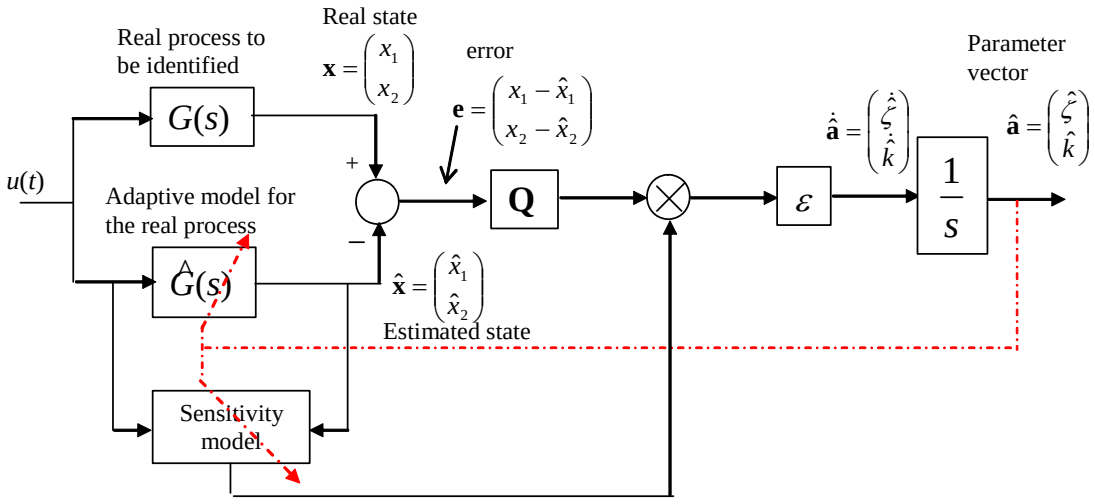
Hint: Use subsystems and "enable" blocks. Use a "clock" block to know the time instant and implement the necessary Boolean comparisons in order to activate one or another controller.

□

Exercise 8. On-line parameter identification. Consider a process that can be modeled as a second order low pass filter $G(s) = \dfrac{k}{s^2 + 2\zeta\omega_n s + \omega_n^2}$. We know that its natural frequency is $\omega_n = 3$, but we do not know the exact value for its gain k and damping coefficient ζ.

In this exercise we are going to implement an on-line parameter identification scheme in order to identify k and ζ. Assume that $\zeta = 0.3$ and $k = 9$ are the "unknown" values we are searching for.

The scheme to be implemented is the MIT rule shown in next figure:



MIT rule: Since the error depends on the estimated parameters $\hat{\mathbf{a}}$, $\mathbf{e} = \mathbf{e}(t, \hat{\mathbf{a}})$, we will vary $\hat{\mathbf{a}}$ until $\mathbf{e}$=0 (the error will be zero when $\hat{\mathbf{a}} = \mathbf{a}$).

Consider the positive definite Lyapunov function of error $V(\mathbf{e}) = \frac{1}{2}\mathbf{e}^T\mathbf{Q}\mathbf{e}$, where $\mathbf{Q}$ is a positive definite symmetric matrix. Since $V(\mathbf{e})$ is positive definite, if the gradient $\dot{V}$ is negative,

e will tend asymptotically to zero (its minimum value). Therefore, what we need is that $\hat{\mathbf{a}}$ vary as the gradient of *V*, $\dot{\hat{\mathbf{a}}} = -\varepsilon\left(\frac{\partial V}{\partial \hat{\mathbf{a}}}\right)^T$, ε>0. This can be expressed as $\dot{\hat{\mathbf{a}}} = \varepsilon \mathbf{S}^T \mathbf{Q}\mathbf{e}$, where $\mathbf{S} = -\frac{\partial \mathbf{e}}{\partial \hat{\mathbf{a}}} = \frac{\partial \hat{\mathbf{x}}}{\partial \hat{\mathbf{a}}}$ is the sensitivity matrix. Parameter ε affects the convergence properties.

Note: The gradient is $\frac{\partial V}{\partial \hat{\mathbf{a}}} = \frac{\partial V}{\partial \mathbf{e}}\frac{\partial \mathbf{e}}{\partial \hat{\mathbf{a}}}$ where, if the matrix **Q** is square and symmetric, we have $\frac{\partial V}{\partial \mathbf{e}} = \frac{1}{2}\left(\frac{\partial \mathbf{e}^T}{\partial \mathbf{e}}\mathbf{Q}\mathbf{e} + \mathbf{e}^T\frac{\partial(\mathbf{Q}\mathbf{e})}{\partial \mathbf{e}}\right) = \frac{1}{2}\left(\mathbf{Q}\mathbf{e} + \mathbf{e}^T\mathbf{Q}^T\right) = \mathbf{Q}\mathbf{e} = (\mathbf{Q}\mathbf{e})^T$. Here we have used the vector derivation expressions: $\frac{\partial(\mathbf{c}^T\mathbf{x})}{\partial \mathbf{x}} = \mathbf{c}$ y $\frac{\partial(\mathbf{A}\mathbf{x})}{\partial \mathbf{x}} = \mathbf{A}^T$. And the sensitivity matrix is

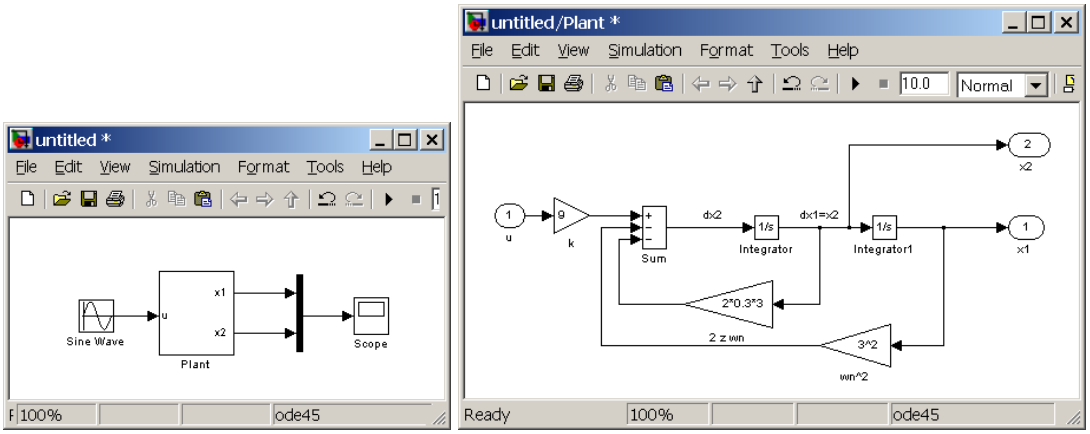
$$\mathbf{S} = -\frac{\partial \mathbf{e}}{\partial \hat{\mathbf{a}}} = -\frac{\partial(\mathbf{x}-\hat{\mathbf{x}})}{\partial \hat{\mathbf{a}}} = \frac{\partial \hat{\mathbf{x}}}{\partial \hat{\mathbf{a}}} = \begin{bmatrix} \frac{\partial \hat{x}_1}{\partial \hat{\zeta}} & \frac{\partial \hat{x}_1}{\partial \hat{k}} \\ \frac{\partial \hat{x}_2}{\partial \hat{\zeta}} & \frac{\partial \hat{x}_2}{\partial \hat{k}} \end{bmatrix}.$$

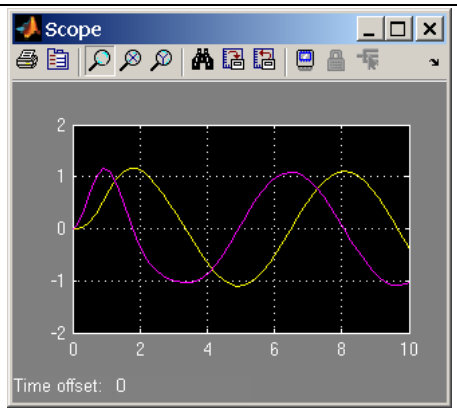
1) Real process: Implement in a Simulink subsystem the real process state equations $\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, u, \mathbf{a})$, where **x** is the state vector, *u* is the process input signal, and $\mathbf{a} = \begin{bmatrix} \zeta \\ k \end{bmatrix}$ is the parameter vector:

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

(Note that we are not going to use *Transfer Function* blocks. We are going to directly implement the system state equations using the blocks: *Integrator*, *Sum*, *Gain*)

The two outputs of the subsystem must be x_1 and x_2. The input *u* must be a unit amplitude sinusoid of frequency 1rad/s.





2) Adaptive model: To estimate parameters ζ and k we are going to build an adaptive model with the same structure than the real process $\dot{\hat{\mathbf{x}}} = \mathbf{f}(\hat{\mathbf{x}}, u, \hat{\mathbf{a}})$, but here $\hat{\mathbf{x}}$ and $\hat{\mathbf{a}}$ are estimates of **x** and **a**.

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

In order to vary the parameters $\hat{\zeta}$ and $\hat{k}$, insert two additional inputs: $\hat{\zeta}$ and $\hat{k}$. Fix the additional inputs to the "unknown" real values $\hat{\zeta}$ = 0.3 y $\hat{k}$ = 9 (use blocks Constant or Step) and excite the submodel with the same *u* as before. Check that the output is the same than the real process.

3) Use blocks Sum to obtain the error signal between the two submodels (real plant and adaptive model), $\mathbf{e} = \mathbf{x} - \hat{\mathbf{x}}$. Let the excitation signal be a sinusoid *u*(*t*) of unit amplitude and frequency ω = 1 *rad/s.*

4) Sensitivity model: The sensitivity model for the adaptive model is computed below:

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

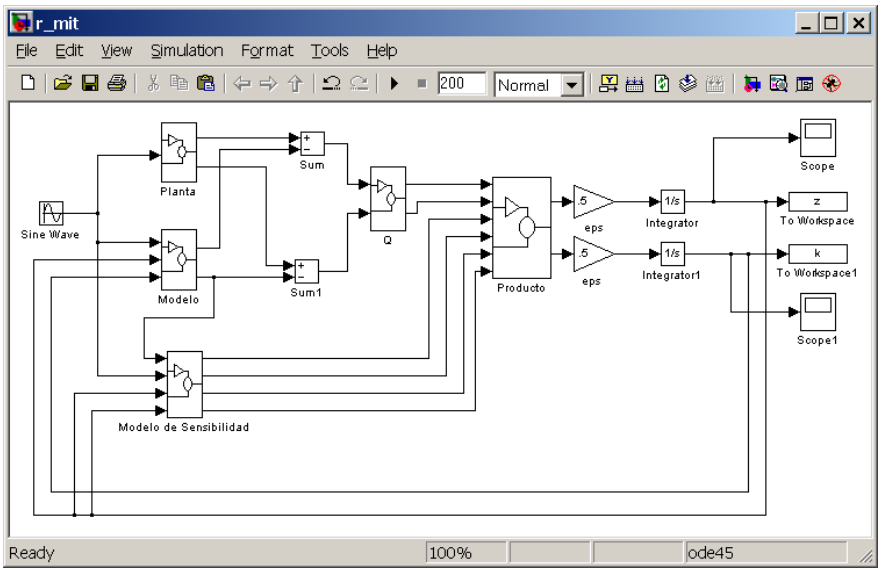
$$\begin{bmatrix} \frac{\partial \dot{\hat{x}}_1}{\partial \hat{\zeta}} & \frac{\partial \dot{\hat{x}}_1}{\partial \hat{k}} \\ \frac{\partial \dot{\hat{x}}_2}{\partial \hat{\zeta}} & \frac{\partial \dot{\hat{x}}_2}{\partial \hat{k}} \end{bmatrix} = \begin{bmatrix} \frac{\partial f_1}{\partial \hat{x}_1}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

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

(Note that the sensitivity model consists of 4 state equations, one for each of its 4 state variables S_{ij}). Build a SIMULINK subsystem to implement the sensitivity model (Blocks: *Integrator*, *Sum*, *Gain*). Note that the model consists of 4 differential equations. Subsystem inputs are parameter estimates $\hat{\zeta}$ and $\hat{k}$, excitation *u*, and state estimate $\hat{\mathbf{x}}$. Outputs are sensitivities S_{11}, S_{12}, S_{21}, and S_{22}.

5) To obtain the parameter vector $\hat{\mathbf{a}}$ from $\mathbf{S}^T$ and **e**, implement a subsystem to perform the following product: $\dot{\hat{\mathbf{a}}} = \varepsilon \mathbf{S}^T \mathbf{Q} \mathbf{e}$ (take $\mathbf{Q} = \mathbf{I}$ and ε = 2). Then, put two integrator blocks to

obtain $\hat{\mathbf{a}}$ from $\dot{\hat{\mathbf{a}}}$. Initial values for $\hat{\zeta}$ and $\hat{k}$ must be included inside the two integrator blocks. Connect all the subsystems and put Scope blocks. Next figure shows a possible solution. It is suggested to use masks and different block formats.



6) Plot the time evolution of $\hat{\zeta}$ and $\hat{k}$ (take final simulation time 150*s* and initial conditions $\hat{\zeta}$ =1, $\hat{k}$ =5). To which values do the estimates converge?

7) Plot the time evolution of the state vector **x** and its estimate $\hat{\mathbf{x}}$.

8) Study the effect of ε in the convergence of parameter estimates *k* y ζ. Simulate with ε = 0.7 and ε = 2. □

Exercise 9. Stateflow (optional). Propose and solve a simple Stateflow example (for instance, some application of sequential control, semaphore,…). See the example in the Unit 5. □

Exercise 10. Animation (optional). Implement an animation effect by using an S-function block and GUI tools (for instance, a pendulum movement, a cart-spring-damper system,…). See the example in the Unit 5. □