1. Introduction to Control System Toolbox

Control System Toolbox is a package for Matlab consisting of tools specifically developed for control applications. The package offers data structures to describe common system representations such as state space models and transfer functions, as well as tools for analysis and design of control systems. There are also tools for simulation of systems.

In this excercise you will get to know the basic commands of Control System Toolbox. When you have completed this excercise, you should be able to understand and use Control Systems Toolbox to create and analyze linear systems. Extensive use of the Matlab `help` command is recomended. It is also recomended that you create a script file (e.g. `myscript.m`) in which you write your commands. By running a script file instead of typing the commands directly at the Matlab prompt, it is easier to correct mistakes, and also, your work will be saved for later use.

The system you will use is

$$\dot{x} = Ax + Bu = \begin{bmatrix} 0 & 1 \\ -1 & -1 \end{bmatrix} x + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u$$
$$y = Cx = \begin{bmatrix} 1 & 0 \end{bmatrix}.$$

To enter a matrix in Matlab, for example matrix A, do

```
A = [0 1; -1 -1]
```

Creation and conversion of systems

Control System Toolbox supports several system representations of linear time invariant systems. In this excercise we will use two of the most common representations; state space models and transfer functions.

Define the system matrices A, B, C and D given above. (What is the value of D in the model?) Create a state space description of the system using `ss`, and name it `sys_ss`. Find out how to use `ss` by using the help function (`help ss`). At this stage, you should have obtained a state space description of the system.

Let us now create an equivalent transfer function model of the system above. This could, as you know, be done by using the formula $G(s) = C(sI - A)^{-1}B + D$. However, Matlab may also be used for the task. Use the command `tf` to convert the state space model to a tranfer function and name it `sys_tf`. Notice that `tf` may be used for creation of transfer functions as well as conversion. What is the syntax in the two cases respectively?

Stability analysis

Stability of a linear system is determined by the location of its poles in the complex plane. (What is the condition for stability?) Use the commands `ssdata` and `tfdata` to extract the necessary data from the models, and `eig` and `roots` to determine stability of the system. Verify that the roots of the characteristic polynomial of the transfer function are the same as the eigenvalues of the system matrix. What are the eigenvalues / poles? Is the system stable? You could also use the command `pole`, or for a graphical view, `pzmap`.

Time domain analysis

Use the command `step` to plot the step response of the system. Relate the characteristics of the step response to the location of the poles. If there is time, use `initial` and `lsim` to study the system response.

Some useful Matlab commands

plot	Linear plot.
subplot	Breaks the Figure window into small axes.
axis	Control axis and scaling appearance.
hold	Hold current graph.
grid	Grid lines.
title	Graph title.
xlabel, ylabel	Axes labels.
tf	Create a transfer function model.
tfdata	Extract numerator and denominator.
ss	Create a state-space model.
ssdata	Extract state-space matrices.
zpk	Create a zero/pole/gain/model.
step	Step response.
initial	Response of state-space model with given initial state.
pole	System poles.
zero	System zeros.
roots	Find polynomial roots.
pzmap	Pole-zero map.
eig	Compute eigenvalues and eigenvectors.
bode	Draw a bode frequency response.
lsim	Simulate time response of LTI models to arbitrary inputs.
simulink	Open the simulink browser.
sim	Use a Simulink model from a Matlab script
simset	Set options for the sim command
ictools	Interactive tools for control
pplane6	Phase plane analysis, download from our webside
linmod	Obtain the state-space linear model of the system of ordinary differential equations described in a simulink model, note that the model must contain a simulink block *out* from sinks and a simulink block *in* from sources

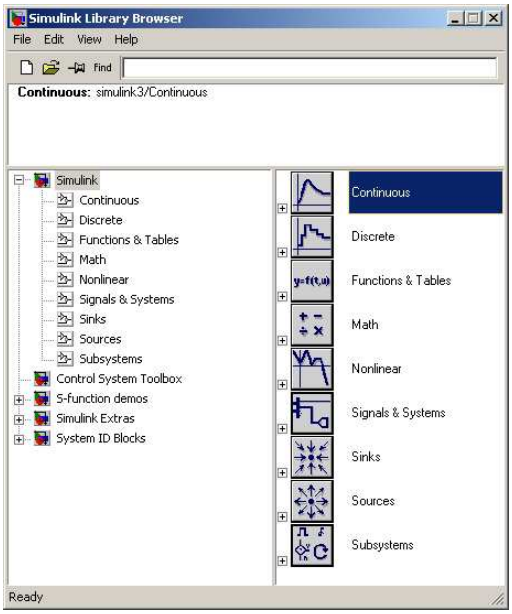
2. Introduction to Simulink[0]

Simulink is a simulation program based upon Matlab. There are several ways to define a model. One can work graphically and connect block-diagrams with predefined blocks. Alternatively one can give the mathematical description in forms of differential equations in an m-file (the format for programs written in the Matlab programming language). Matlab/Simulink supports both these representations as well as combinations. Furthermore one can use descriptions that include a hierarchy of connected subsystems.

To understand how models are described and simulated using block diagrams, it is best to run small examples on a computer. The rest of Section 2 shows some examples. If you are familiar with Simulink you can go directly to Section 3.

How to Start Simulink

Start Matlab-6. Then give the command `simulink` in Matlab. This gives a window with blocks as in Figure 1.



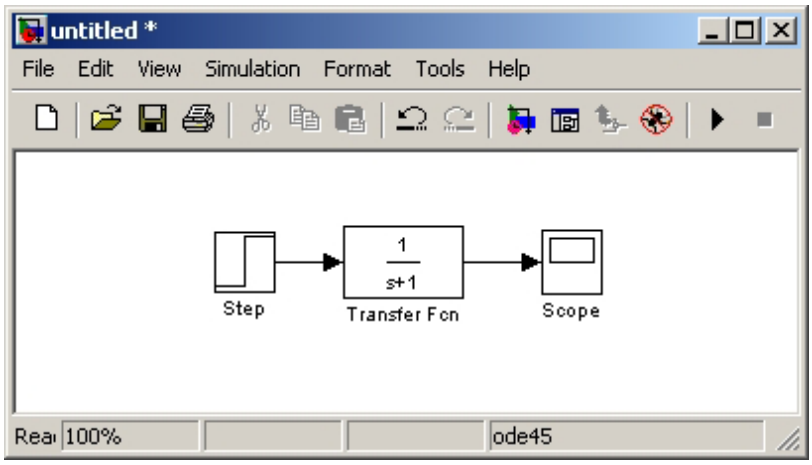
Figur 1 Available Simulink block diagram libraries

A Simple System

Click on `File` in the Simulink-window and choose `New->Model`. Click on the block `Continous` and move a `Transfer Fcn` to the new window called "Untitled". Do the same with `Source->Step Fcn` and `Sinks->Scope`. Draw arrows (left mouse button) and connect the ports on the block. You should now have a block diagram as in Figure 2.

Choose `Simulation->Parameters` in the window called "Untitled". Set `Stop time` to 5. Open the window `Scope` by double clicking on it. Put `Horizontal Range` to 6. Start a simulation by `Simulation->Start` (or by pressing Ctrl-t in the window called "Untitled").

[0]Written by Bo Bernhardsson and Erik Möllerstedt January 1999, revised by Johan Åkesson March 2002



Figur 2 A simple Simulink system

How to Change a System To change the system to

$$\frac{1}{s^2 + 0.5s + 2},$$

you double-click on the block `Transfer Fcn` and change `Denominator` to [1 0.5 2]. Simulate the new system (`Simulation->Start` or Ctrl-t). Change parameters in the `Simulation` menu and the scales in the block `Scope` until you are satisfied.

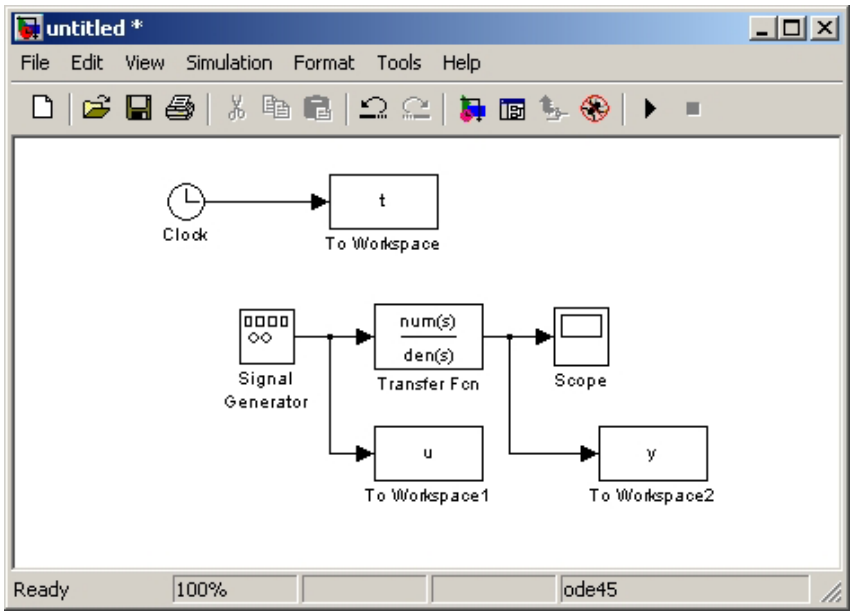
How to Change an Input Signal To change the input signal, start with removing the block `Step Fcn` by clicking on it and delete it by using `Edit->Cut` (or pressing Ctrl-x). Replace it by a `Sources->Signal Gen` block. Double-click on `Signal Gen` and select signal, amplitude and frequency. Also change `Simulation->Start->Stop Time` to 99999 and press `Simulation->Start`. This gives an "infinite" simulation that can be stopped by pressing `Simulation->Stop` (or Ctrl-t). Can the amplitude of the input signal be changed during simulation? Also try to change the block `Transfer Fcn` during simulation.

How to Use Matlab Variables in Blocks Note that variables defined in the Matlab environment can be used in Simulink. Define numerator and denominator by writing the following in the Matlab window.

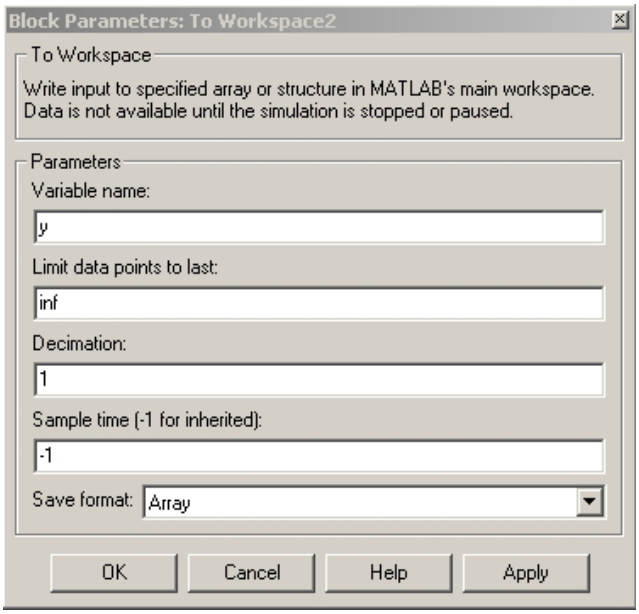
```
num=[1 1]
den=[1 2 3 4]
```

Change `Transfer Fcn->Numerator` to `num` and `Transfer Fcn->Denominator` to `den`.

How to Save Results to Matlab variables To save input and output move two copies of the block `Sinks->To Workspace`. Make sure that the "save format" option is set to "Array", se figure 2. Connect these with the input and output to the block `Transfer Fcn`. Also get a `Source->Clock` and connect it to a `Sinks->To Workspace`. Change the variable names to `u,y,t` respectively. The window should look something like the figure.



Figur 3 Model including "To Workspace" blocks.



Figur 4 The "Save format" should be set to "Array".

How to Use Simulation Results in Matlab Calculations Let the input signal be a sinusoidal with frequency 0.1 rad/s and amplitude 1. Do a simulation that is long enough for the output to become stationary. Compute the maximum value of y when the system has settled.

```
n=length(y)
max(y(n/2:n))
```

Using Simulink Models in Matlab Scripts

Often, it is convenient to work with Matlab scripts (m-files), in order to save a sequence of commands. It is possible to use Simulink models from within a Matlab script, using the command `sim`. By using the command `simset` options for the `sim` command may be specified.

Use the model from the previous example. Save the model, and name it "mymodel.mdl". Create a Matlab script named "mysim.m", and enter the following commands:

```
tfinal = 300;
options = simset('reltol',1e-5,'refine',10,'solver','ode45');
sim('mymodel',tfinal,options);

%plot results
figure(1)
clf
subplot(211)
plot(t,u);
ylabel('u')
subplot(212)
plot(t,y)
ylabel('y')
```

When you run the script, you should see a plot showing the input and the output of the transfer function. Use the `help` command to learn more about how to use the `simset` and `sim` commands.

How to Save Systems Use `File-Save As` or `File->Save`.

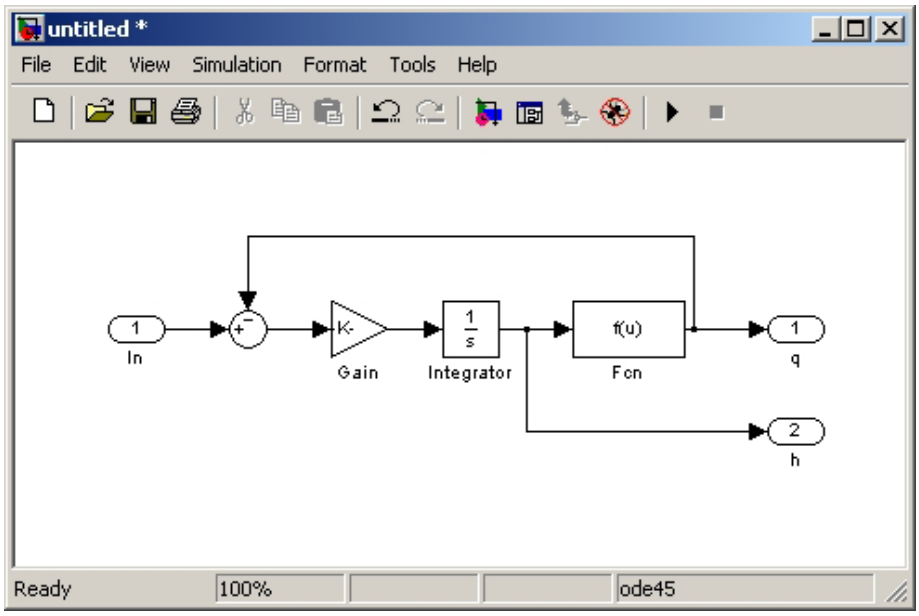
A Flow System

Consider a simple tank as in the basic control course

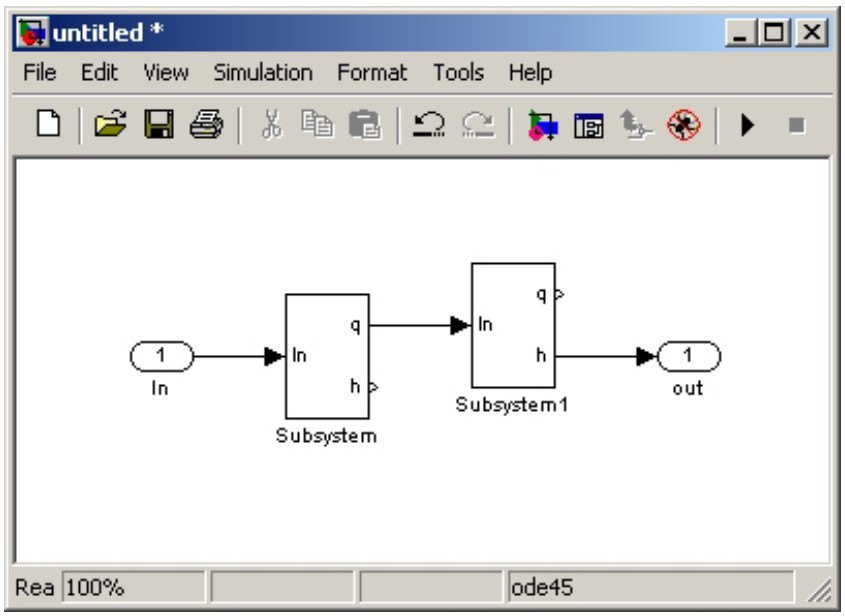
$$\begin{aligned} \dot{h} &= \frac{1}{A}(u-q) \\ q &= a\sqrt{2gh}. \end{aligned}$$

This can be implemented in Simulink as in Figure 3. The function $f(u)$ has the value `a*sqrt(2*g)*sqrt(u[1])`. The block `Sum` has been given two inputs with different signs by assigning the string "`-+|`" to `Sum->List of Signs`. The `Input` and `Output` blocks can be found under `Sources` and `Sinks` respectively. These blocks tell Simulink what should be considered inputs and outputs to this (sub)system. The block titles can be changed by clicking on them. Mark the entire system by holding the left mouse bottom pressed and drawing a square around it. Then choose `Edit->Create Subsystem`. The result is that the system is represented by one block. Use `Edit->Copy` to create the following double-tank system. Use the command `linmod` to find a linearized model of the double tank around $h_1^0 = h_2^0 = 0.1$. Use the parameters $A_1 = A_2 = 2.7 \times 10^{-3}$, $a_1 = a_2 = 7.0 \times 10^{-6}$, $g = 9.8$. Notice also in figure the simulink block *in* from sources and the simulink block *out* from sinks, which are necessary for the linmod commando.

```
>> A=2.7e-3;a=7e-6;g=9.8;
```



Figur 5 A tank system



Figur 6 Two tanks and some connections

```
>>x0 = [0.1000 0.1000]';
>>u0 = a*sqrt(q*g*x0(1));

>> [aa, bb,cc,dd]=linmod('flow',x0,u0);
```