This handout presents some simple examples of using Simulink to model dynamical systems.

1 Simple oscillator

(`oscillator.mdl`[1]) Here we obtain a Simulink model of the simple spring-mass-damper system described by

$$\boxed{m\ddot{y} + c\dot{y} + ky = 0}$$

To obtain a Simulink model, we first rearrange the equation as follows:

$$\ddot{y} = -\frac{c}{m}\dot{y} - \frac{k}{m}y$$

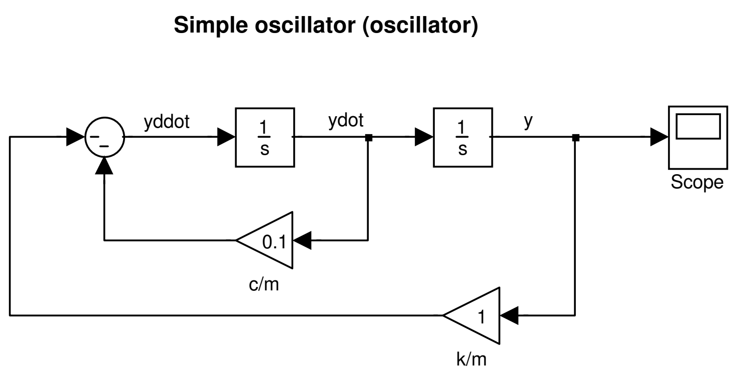

Figure 1: Simulink model of a simple oscillator with $c/m = 0.1$ and $k/m = 1$

Generating output and plots

```
[t x] = sim('oscillator')          %Runs simulation
plot(t,x(:,1))
```

Getting state info

```
[sizes xo states] = oscillator
sizes =
     2
     0
     0
     0
     0
     0
     1
xo =
     1
     0
```

[1]This is the name of the file associated with this example

```
states =
    'oscillator/y'
    'oscillator/ydot'
```

2 Pendulum

Here we consider the motion of a planar pendulum subject to a constant torque u:

$$\boxed{J\ddot{\theta} + c\dot{\theta} + Wl\sin\theta = u}$$

Note that this can be rewritten as:

$$\ddot{\theta} = \frac{1}{J}(u - c\dot{\theta} - Wl\sin\theta)\,.$$

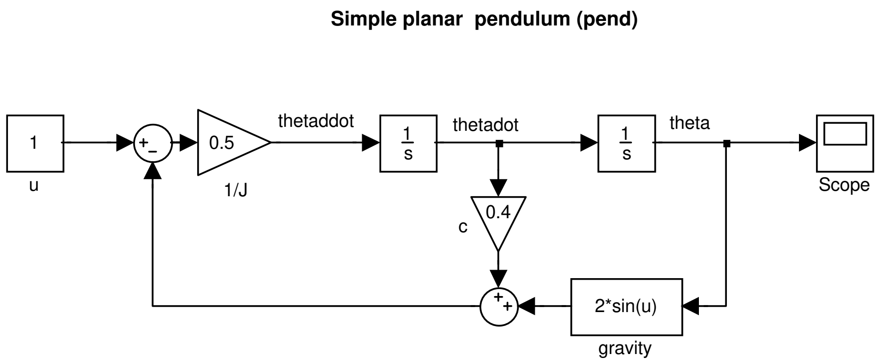

Figure 2: Simulink model of a simple planar pendulum with $Wl = 2$, $c = 0.4$, $J = 2$ and $u = 1$.

Finding equilibrium and trim conditions

```
trim('pend')

ans =

    0.0000
    0.5236
```

3 Pendulum on cart

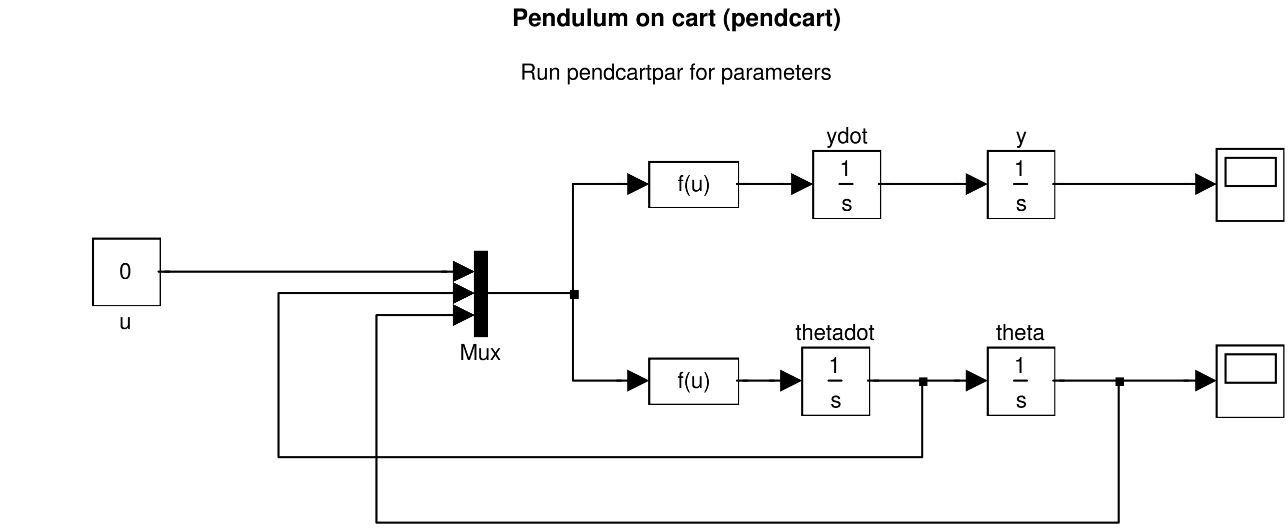

Figure 3: Simulink model of pendulum on cart

The motion of the pendulum on a cart can be described by

$$\begin{aligned}(M+m)\ddot{y} - ml\cos\theta\,\ddot{\theta} &+ ml\sin\theta\dot{\theta}^2 = u\\ -ml\cos\theta\,\ddot{y} + ml^2\ddot{\theta} &+ mlg\sin\theta = 0\end{aligned}$$

where M is the mass of the cart, m is the pendulum mass, l is distance from cart to pendulum mass, and g is the gravitational acceleration constant. The variables y and θ are the cart displacement and the pendulum angle, respectively.

Solving for $\ddot{\theta}$ and $\ddot{y}$ yields

$$\begin{aligned}\ddot{y} &= (u - ml\sin\theta\dot{\theta}^2 - mg\sin\theta\cos\theta)/(M + m\sin\theta^2)\\ \ddot{\theta} &= (\cos\theta u - ml\sin\theta\cos\theta\dot{\theta}^2 - (M+m)g\sin\theta)/(Ml + ml\sin\theta^2)\end{aligned}$$

Simulink model.

Top function block

```
(u[1]-m*l*sin(u[3])*u[2]*u[2]-m*g*sin(u[3])*cos(u[3]))/(M+m*sin(u[3])^2)
```

Bottom function block

```
(cos(u[3])*u[1]-m*l*sin(u[3])*cos(u[3])*u[2]*u[2]-(M+m)*g*sin(u[3])) /(M*l+m*l*sin(u[3])^2)
```

Specifying parameters. Note that in the above model the parameters were specified symbolically, for example, m and l. Before running a simulation, values must be assigned to the parameters which were assigned symbols. This can be done at the Matlab command line or by executing an M-file which assigns the parameters. Here I run the following file first before an initial simulation.

```
%
%pendcartpar.m
%
%Set parameters for pendulum on cart example
M=1
m=1
l=1
g=1
```

4 Cannonball

$$\begin{aligned}\dot{V} &= -g\sin\gamma - \kappa V^2/m \\ V\dot{\gamma} &= -g\cos\gamma\end{aligned}$$

and

$$\begin{aligned}\dot{p} &= V\cos\gamma \\ \dot{h} &= V\sin\gamma\end{aligned}$$

where $\kappa = \rho S C_D/2$.

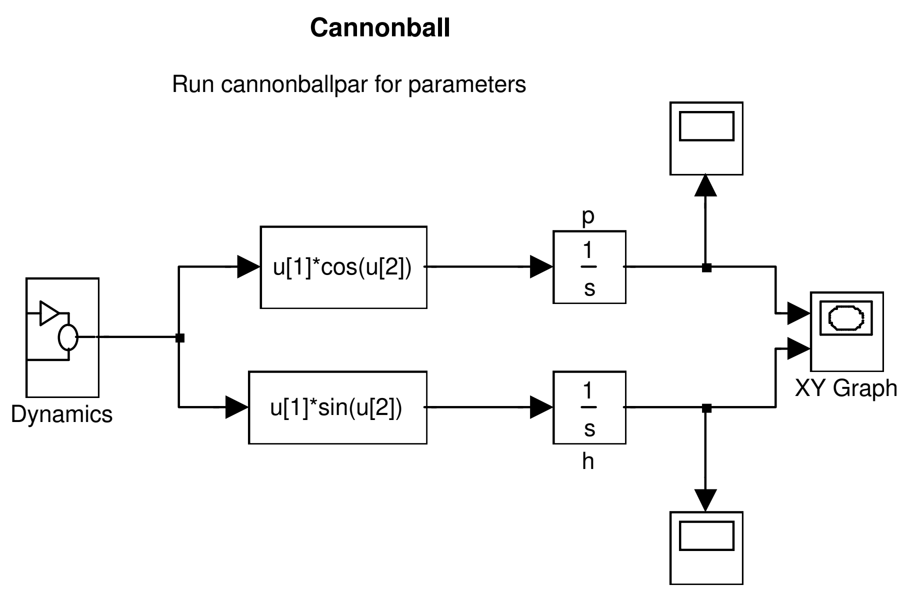

Figure 4: Simulink model of cannonball

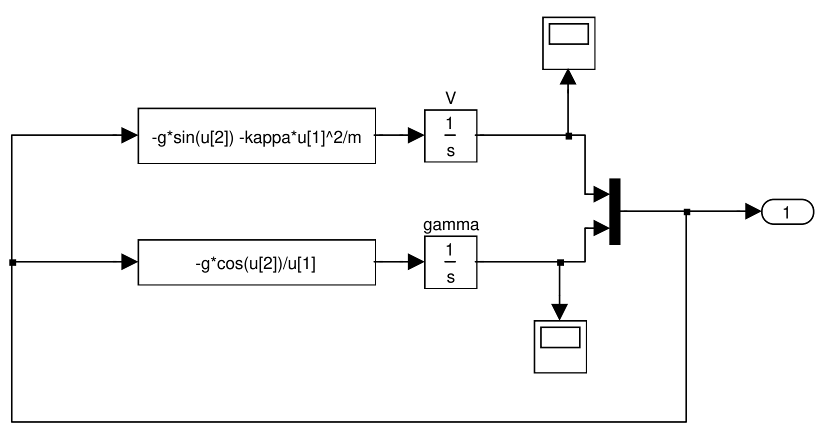

Figure 5: Dynamics subsystem

```
%cannonballpar.m
%
%Parameters for cannonball
g = 9.81
m= 1
rho=0.3809
Cd=0.4
S=0.1
kappa = rho*S*Cd/2
```

5 S-Functions

An *S-function* is useful when one wants to use equations to describe a dynamical system. One can use an S-function to completely describe an input-output system. Recall the simple pendulum example. Here we use an S-function to describe the pendulum dynamics.

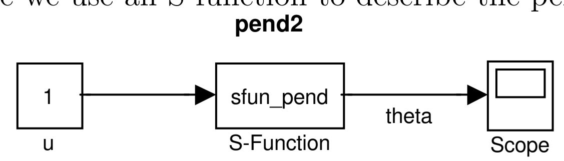

Figure 6: Simulink model of a simple planar pendulum with $Wl = 2$, $c = 0.4$ and $J = 2$.

The following Matlab M-file, called `sfun_pend.m`, generates a S-function for the simple pendulum with input u and output θ. You can use this file as a template for all your S-function M-files. You need only change the lines in the boxes.

```
% sfun_pend.m
% S-function to describe the dynamics of the simple pendulum
% Input is a torque and output is pendulum angle
function [sys,x0,str,ts] =sfun_pend(t,x,u,flag)

% t is time
% x is state
% u is input
% flag is a calling argument used by Simulink.
% The value of flag determines what Simulink wants to be executed.

switch flag

case 0          % Initialization
   [sys,x0,str,ts]=mdlInitializeSizes;

case 1          % Compute xdot
   sys=mdlDerivatives(t,x,u);

case 2         % Not needed for continuous-time systems

case 3          % Compute output
   sys = mdlOutputs(t,x,u);
```

```
case 4          % Not needed for continuous-time systems

case 9          % Not needed here

end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% mdlInitializeSizes
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
function [sys,x0,str,ts]=mdlInitializeSizes
%
% Create the sizes structure
sizes=simsizes;

sizes.NumContStates = 2;        % Set number of continuous-time state variables

sizes.NumDiscStates = 0;

sizes.NumOutputs = 1;           % Set number of output variables
sizes.NumInputs = 1;            % Set number of input variables

sizes.DirFeedthrough = 0;
sizes.NumSampleTimes = 1;       % Need at least one sample time
sys = simsizes(sizes);
%
x0=[0;0];                                 % Set initial state
%
str=[];                 % str is always an empty matrix
ts=[0 0];               % ts must be a matrix of at least one row and two columns
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% mdlDerivatives
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
function sys = mdlDerivatives(t,x,u)
%
% Compute xdot based on (t,x,u) and set it equal to sys
%
 sys(1) = x(2);
 sys(2) = (-2*sin(x(1))-0.4*x(2) + u)/2;
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% mdlOutput
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
function sys = mdlOutputs(t,x,u)
%
% Compute output  based on (t,x,u) and set it equal to sys
sys = x(1);
```

6 Using ODE files in S-functions

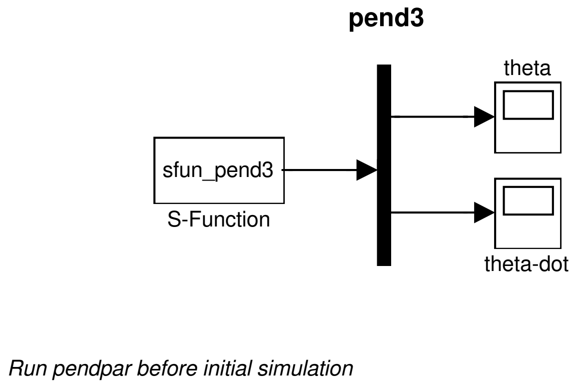

Figure 7: Another Simulink model of the simple planar pendulum

The following Matlab M-file generates a S-function for the simple planar pendulum with no input and two output variables θ and $\dot{\theta}$. The applied torque u is treated as a parameter. This example also illustrates how one can use ode files in an S-function.

You could also use this file as a template for your S-functions.

```
% sfun_pend3.m                                                          %CHANGE

% S-function to describe the dynamics of  a
% SIMPLE PLANAR PENDULUM                                                 %CHANGE


 function [sys,x0,str,ts] = sfun_pend3(t,x,u,flag)                       %CHANGE

% t is time
% x is state
% u is input
% flag is a calling argument used by Simulink.
% The value of flag determines what Simulink wants to be executed.

switch flag

case 0          % Initialization
   [sys,x0,str,ts]=mdlInitializeSizes;

case 1          % Compute xdot
   sys=mdlDerivatives(t,x,u);

case 2         % Not needed for continuous-time systems
```

```
case 3          % Compute output
   sys = mdlOutputs(t,x,u);

case 4          % Not needed for continuous-time systems

case 9          % Not needed here

end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% mdlInitializeSizes
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
function [sys,x0,str,ts]=mdlInitializeSizes
%
% Create the sizes structure
sizes=simsizes;
sizes.NumContStates = 2;    %Set number of continuous-time state variables %CHANGE
sizes.NumDiscStates = 0;
sizes.NumOutputs = 2;        %Set number of outputs                        %CHANGE
sizes.NumInputs = 0;         %Set number of inputs                         %CHANGE

sizes.DirFeedthrough = 0;
sizes.NumSampleTimes = 1;       %Need at least one sample time
sys = simsizes(sizes);
%
x0=[1;0];                       % Set initial state                         %CHANGE

str=[];                         % str is always an empty matrix
ts=[0 0];            % ts must be a matrix of at least one row and two columns
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% mdlDerivatives
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
function sys = mdlDerivatives(t,x,u)
%
% Compute xdot based on (t,x,u) and set it equal to sys
%
sys= pendode(t,x);                                                         %CHANGE
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% mdlOutput
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
function sys = mdlOutputs(t,x,u)
%
% Compute output  based on (t,x,u) and set it equal to sys

sys = x;                                                      %CHANGE
```

The above S-function uses the following ODE file

```
%pendode.m

function xdot = pendode(t,x)
global W l J c u
xdot(1) = x(2);
xdot(2) = (-W*l*sin(x(1)) -c*x(2) + u)/J;
```

This ODE file requires the following parameter file

```
%pendpar.m

global W l J c u
J = 2
c=0.4
W= 2
l=1;
```

7 Trim and Linearization

Recall that the simple planar pendulum is described by

$$J\ddot{\theta} + c\dot{\theta} + Wl\sin\theta = u$$

Linearization

$$J\delta\ddot{\theta} + c\delta\dot{\theta} + (Wl\cos\theta^e)\delta\theta = 0$$

Hence

$$A = \begin{bmatrix} 0 & 1 \\ -Wl\cos\theta^e/J & -c/J \end{bmatrix}$$

For the chosen parameters

$$A = \begin{bmatrix} 0 & 1 \\ -\cos\theta^e & -0.2 \end{bmatrix}$$

Trim

```
xe=trim('pend3')
xe =
    0.5236
   -0.0000
```

Linearization

```
A=linmod('pend3',xe)
A =
         0    1.0000
   -0.8660   -0.2000
```

More

```
u=0;

xe=trim('pend3')
xe =
  1.0e-023 *
   -0.6617
   -0.0009

A=linmod('pend3',xe)
A =
         0    1.0000
   -1.0000   -0.2000
```

8 Input output stuff

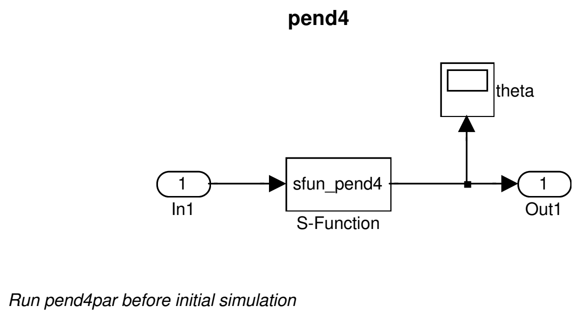

Figure 8: Yet another Simulink model of a simple planar pendulum

Recall that the simple planar pendulum is described by

$$J\ddot{\theta} + c\dot{\theta} + Wl\sin\theta = u$$

We will view this as an input output system with input u and output

$$y = \theta\,.$$

We choose $x_1 = \theta$ and $x_2 = \dot{\theta}$ as states.

Trim. Suppose that we wish that $y = y^e = \pi/6 \approx 0.5236$ when the system is in equilibrium. So, we need to compute u^e and x^e so that $y^e = \pi/6$. From the differential equation above, we obtain

$$u^e = Wl\sin(y^e) = (2)(1)\sin(\pi/6) = 1;$$

also

$$x^e = \begin{bmatrix} y^e \\ 0 \end{bmatrix} = \begin{bmatrix} 0.5236 \\ 0 \end{bmatrix}$$

Using the trim command, we obtain

```
[xe ue ye xdote] = trim('pend4',[],[],[pi/6],[],[],[1])
xe =
    0.5236
         0

ue =    1.0000

ye =    0.5236

xdote =
     0
     0
```

Linearization. The linearized system is described by

$$\begin{aligned}\delta\dot{x}_1 &= \delta x_2\\ \delta\dot{x}_2 &= -(Wl\cos y^e/J)\delta x_1 - (c/J)\delta x_2 + (1/J)\delta u\\ \delta y &= \delta x_1\end{aligned}$$

Hence,

$$A = \begin{bmatrix} 0 & 1\\ -Wl\cos y^e/J & -c/J\end{bmatrix}, \qquad B = \begin{bmatrix} 0\\ 1/J\end{bmatrix}, \qquad C = \begin{bmatrix} 1 & 0\end{bmatrix}, \quad D = 0\,.$$

For the chosen parameters and trim conditions:

$$A = \begin{bmatrix} 0 & 1\\ -0.8660 & -0.2\end{bmatrix}, \qquad B = \begin{bmatrix} 0\\ 1/J\end{bmatrix}, \qquad C = \begin{bmatrix} 1 & 0\end{bmatrix}, \quad D = 0\,.$$

MATLAB yields:

```
 [A B C D] = linmod('pend4',xe,ue)

A =
         0    1.0000
   -0.8660   -0.2000

B =
         0
    0.5000

C =
    1.0000         0

D =
     0
lambda =eig(A)

lambda =
  -0.1000 + 0.9252i
  -0.1000 - 0.9252i
```

Transfer function and poles and zeros.

```
[num den]=ss2tf(A,B,C,D)

num =         0         0    0.5000

den =    1.0000    0.2000    0.8660
```

```
poles=roots(den)

poles =
  -0.1000 + 0.9252i
  -0.1000 - 0.9252i

zeros=roots(num)

zeros =  Empty matrix: 0-by-1
```