RENSSELAER MECHATRONICS

How to convert m-file ODE to Simulink

Objectives:

- Convert an m-file model to a Simulink model

Summary of steps:

1. Copy m-file model contents to 'MATLAB Function' block
2. Specify input sized of 'MATLAB Function' block
3. Integrate $\dot{x}$, specify initial condition in integrator block

Optional steps:

4. Remove input time 't' from 'MATLAB Function' block if unused
5. Create a subsystem for model

Background Information:

M-files are typically used to quickly type out equations of motion and solve ODEs. When nonlinearities, such as friction, saturation and delays are considered, Simulink can be an easier tool to add these effects.

There are numerous ways to covert (port) a m-file ODE to Simulink. This tutorial shows one way to do this using the same ODE function m-file directly in Simulink.

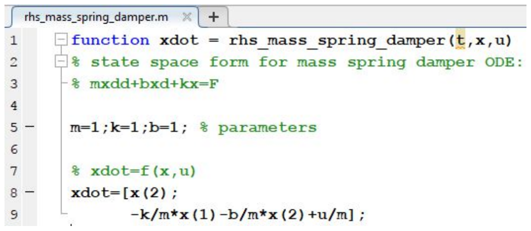
Standard m-file ODE solution:

Solve the mass spring damper ODE $m\ddot{z} + b\dot{z} + kz = F$ with initial condition $z_0 = [0.1\ 0]^T$ for 10 seconds.

1. Convert the ODEs to state space form $\dot{x} = f(x,u)$:

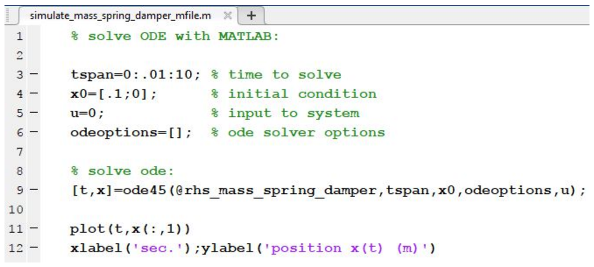
$$\dot{x} = \begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} x_2 \\ -k/m\, x_1 - b/m\, x_2 + F/m \end{bmatrix}$$

2. Write the m-file function that evaluates $\dot{x} = f(x,u)$. This could be any form to that solves $f(x,u)$ including coefficient matrix form (even though there are special solvers for this form).

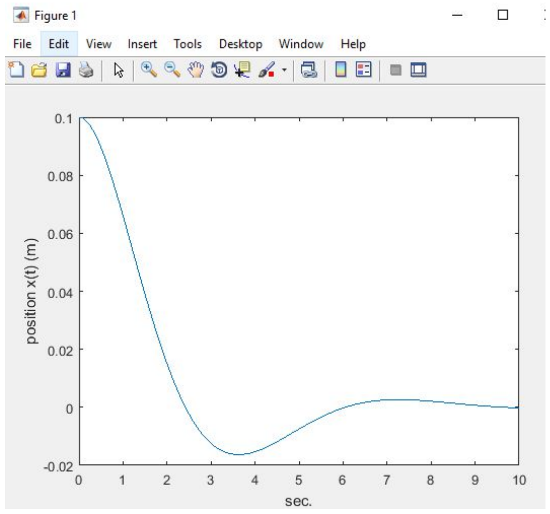


```
function xdot = rhs_mass_spring_damper(t,x,u)
% state space form for mass spring damper ODE:
% mxdd+bxd+kx=F

m=1;k=1;b=1; % parameters

% xdot=f(x,u)
xdot=[x(2);
      -k/m*x(1)-b/m*x(2)+u/m];
```

3. Write a m-file script that solves the ODE and plots the result:

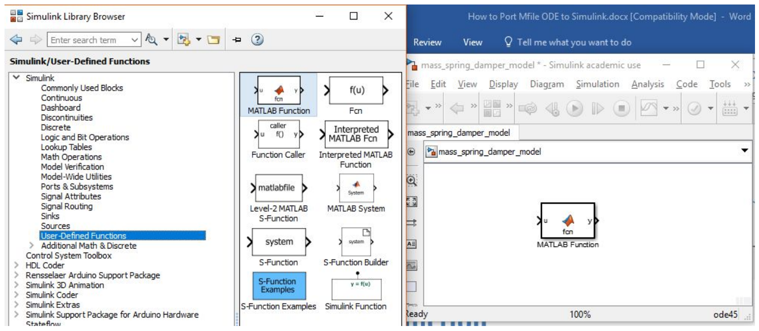


```
% solve ODE with MATLAB:

tspan=0:.01:10; % time to solve
x0=[.1;0];      % initial condition
u=0;            % input to system
odeoptions=[];  % ode solver options

% solve ode:
[t,x]=ode45(@rhs_mass_spring_damper,tspan,x0,odeoptions,u);

plot(t,x(:,1))
xlabel('sec.');ylabel('position x(t) (m)')
```

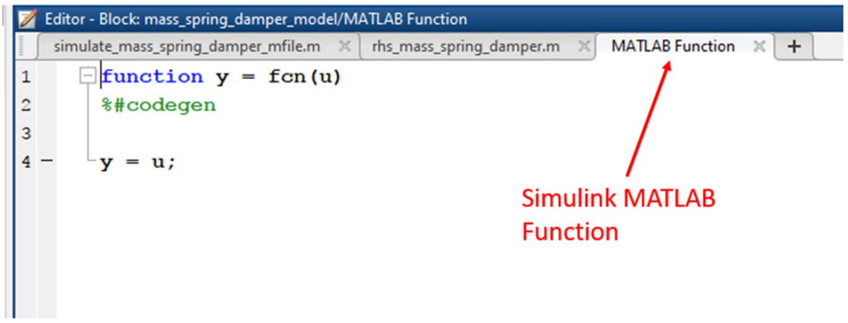


Simulink using m-file ODE function:

1. Create a Simulink diagram and add ‘MATLAB function’ block from ‘Simulink’, ’User-Defined Functions’:

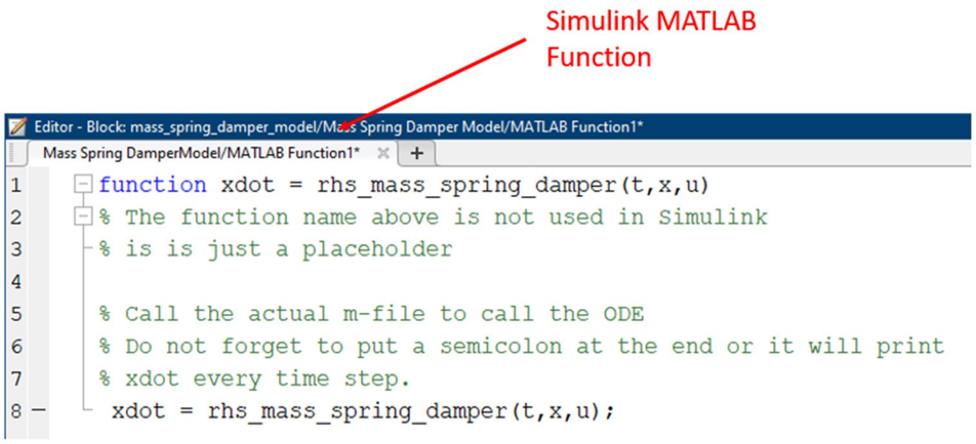


Double click the ‘MATLAB function block’


Editor - Block: mass_spring_damper_model/MATLAB Function
simulate_mass_spring_damper_mfile.m
rhs_mass_spring_damper.m
MATLAB Function
Simulink MATLAB Function

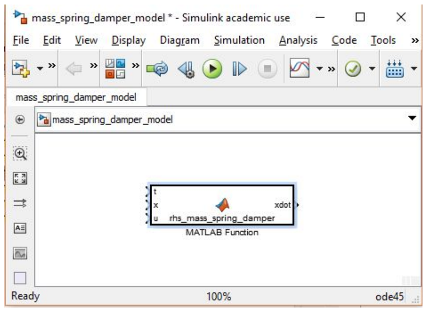
```
function y = fcn(u)
%#codegen

y = u;
```

Delete all the contents.   Copy and paste the first line of your m-file ODE file. Then copy this line again below but remove the “function” name in front. The actual function name is not important in this case, Simulink just used as a place holder.

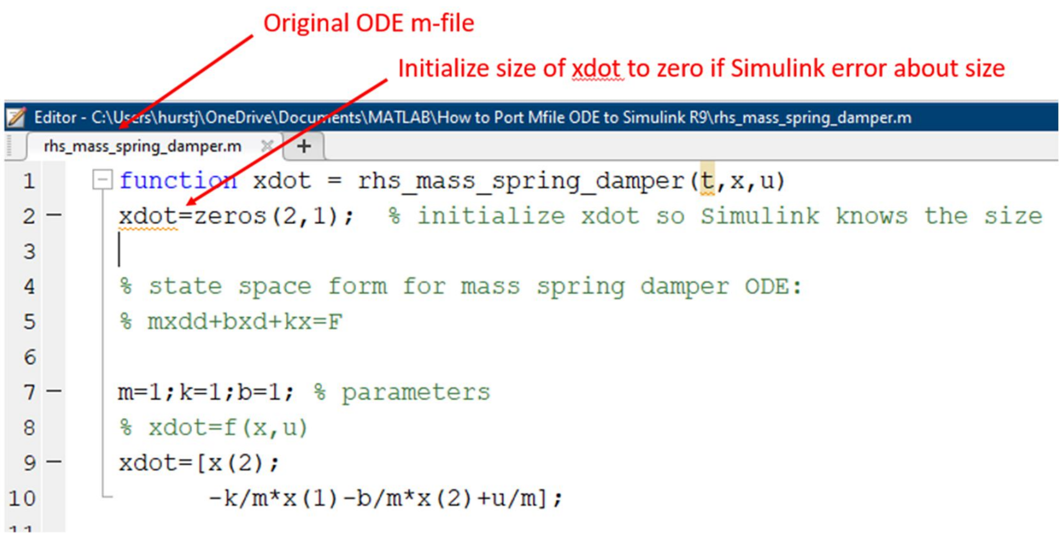

Simulink MATLAB Function
Editor - Block: mass_spring_damper_model/Mass Spring Damper Model/MATLAB Function1*
Mass Spring DamperModel/MATLAB Function1*

```
function xdot = rhs_mass_spring_damper(t,x,u)
% The function name above is not used in Simulink
% is is just a placeholder

% Call the actual m-file to call the ODE
% Do not forget to put a semicolon at the end or it will print
% xdot every time step.
 xdot = rhs_mass_spring_damper(t,x,u);
```

Notice how the MATLAB function block uses the **same exact code** as the m-file model. Save this file and the Simulink diagram will update the block. The resulting Simulink model is now:

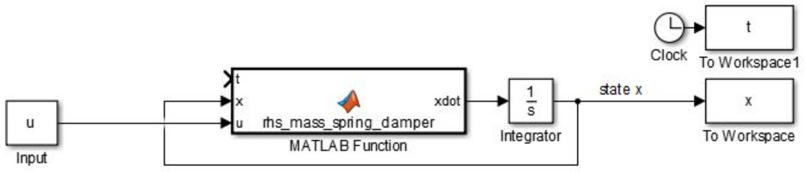

mass_spring_damper_model * - Simulink academic use
File Edit View Display Diagram Simulation Analysis Code Tools
mass_spring_damper_model
mass_spring_damper_model
t
x
u
xdot
rhs_mass_spring_damper
MATLAB Function
Ready
100%
ode45

2. Depending on how xdot is defined in your original ODE m-file Simulink *may* give an error bout sizes not being defined or known. To ensure the size of xdot is known initialize the xdot in your **original** m-file to the correct size:

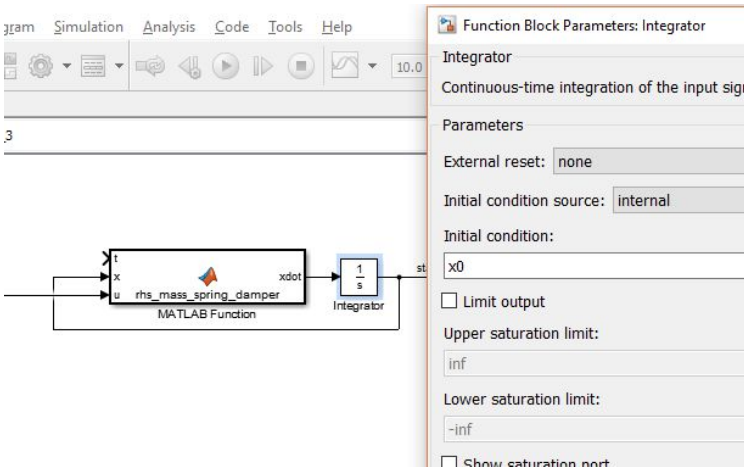

Original ODE m-file
Initialize size of xdot to zero if Simulink error about size
Editor - C:\Users\hurstj\OneDrive\Documents\MATLAB\How to Port Mfile ODE to Simulink R9\rhs_mass_spring_damper.m
rhs_mass_spring_damper.m

```
function xdot = rhs_mass_spring_damper(t,x,u)
xdot=zeros(2,1);  % initialize xdot so Simulink knows the size

% state space form for mass spring damper ODE:
% mxdd+bxd+kx=F

m=1;k=1;b=1; % parameters
% xdot=f(x,u)
xdot=[x(2);
      -k/m*x(1)-b/m*x(2)+u/m];
```

3. Finish ‘writing’ the equation $\dot{x} = f(x,u)$ by integrating $\dot{x}$, adding the input, and writing the output to the workspace (make sure the To Workspace bocks are set up as arrays)



Edit the integrator block to **specify the initial conditions:**



Function Block Parameters: Integrator

Integrator

Continuous-time integration of the input sig

Parameters

External reset: none

Initial condition source: internal

Initial condition:

x0

☐ Limit output

Upper saturation limit:

inf

Lower saturation limit:

-inf

☐ Show saturation port

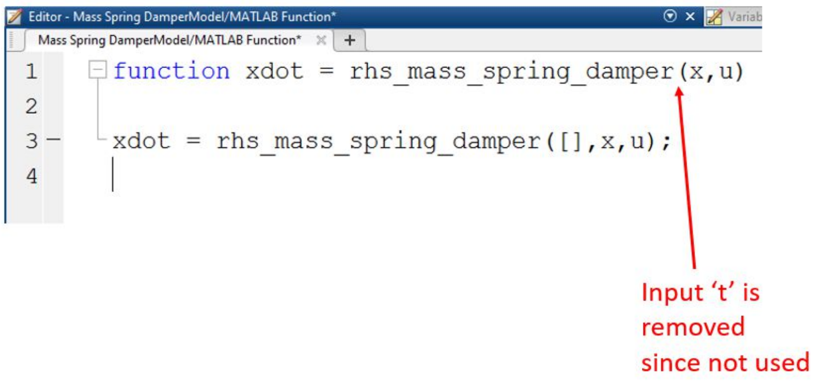
If you forget to specify the size of the initial conditions correctly Simulink will also give an error about size.

Simulate the Simulink diagram and plot the results:

```
% Solve the ODE with simulink
sim('mass_spring_damper_model.slx')
plot(t,x(:,1))
```

Optional steps – create model subsystem

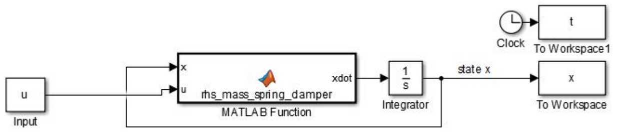
4. In most cases the input variable time ‘t’ is not used to evaluate xdot, but this was required by solver when using an m-file to solve an ode. You can edit this function and remove the time variable since it is not needed to evaluate xdot:



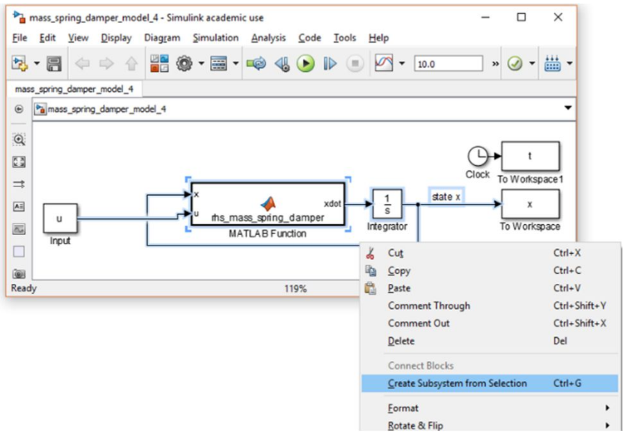
Editor - Mass Spring DamperModel/MATLAB Function*

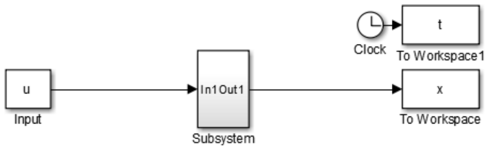
```
function xdot = rhs_mass_spring_damper(x,u)

xdot = rhs_mass_spring_damper([],x,u);
```

The final Simulink diagram with unnecessary input argument ‘t’ removed:

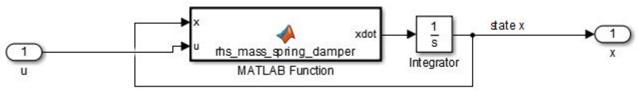


5. Select the ‘MATLAB Function’ and ‘Integrator’ block, right click, ‘Create Subsystem From Selection’

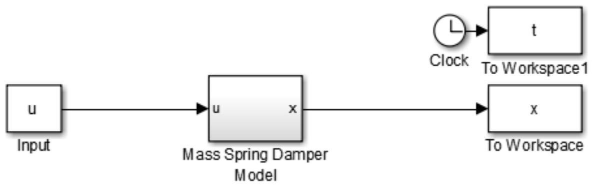

mass_spring_damper_model_4 - Simulink academic use
File Edit View Display Diagram Simulation Analysis Code Tools Help
mass_spring_damper_model_4
Cut Ctrl+X
Copy Ctrl+C
Paste Ctrl+V
Comment Through Ctrl+Shift+Y
Comment Out Ctrl+Shift+X
Delete Del
Connect Blocks
Create Subsystem from Selection Ctrl+G
Format
Rotate & Flip
Ready
119%



Open the subsystem and edit the input and output names, rename the subsystem to something meaningful. Now you have a model that is easy to use in other places.



1
u
x
u
rhs_mass_spring_damper
xdot
MATLAB Function
1/s
Integrator
state x
1
x



Clock
t
To Workspace1
u
Input
u
x
Mass Spring Damper Model
x
To Workspace