MatLab Function Example for Numeric Solution of Ordinary Differential Equations

This handout demonstrates the usefulness of Matlab in solving both a second-order linear ODE as well as a second-order nonlinear ODE.

Example 1 - A Generic ODE

Consider the following ODE:

$$\ddot{x} + b\dot{x} + cx = f(t)$$

where $b = c = 2, \quad \dot{x}(0) = x(0) = 0, \quad f(t) = u(t-1)$

The ODE needs to be re-written as a system of first-order differential equations:

Let $x1(t) = x(t)$
Then $x2(t) = \dot{x}(t) = \dot{x}1(t)$
And $\dot{x}2(t) = \ddot{x}(t) = f(t) - b\dot{x}(t) - cx(t)$

To Summarize:

$$\dot{x}(t) = x2(t)$$
$$\dot{x}2(t) = f(t) - cx1(t) - bx2(t)$$

Which can be rewritten as:

$$\underline{x}(t) = \begin{bmatrix} x1(t) \\ x2(t) \end{bmatrix} \quad \text{and} \quad \dot{x}(t) = \begin{bmatrix} 0 & 1 \\ -c & -b \end{bmatrix} \underline{x}(t) + \begin{bmatrix} 0 \\ 1 \end{bmatrix} f(t)$$

MatLab "solves" this by calculating the numerical approximation of the following integral.

$$\underline{x}(t) = \underline{x}(t0) + \int_{t0}^{t} \underline{\dot{x}}(\tau) d\tau$$

To accomplish this, MatLab needs to have a way of knowing what $\underline{\dot{x}}(\tau)$ is at any time τ. We provide this by writing an M-file function which fits the calling sequence expected by MatLab's integrating routines, *ode23* and *ode45*. The first routine, *ode23*, integrates a system of ordinary differential equations using 2nd and 3rd order Runge-Kutta formulas. [T,Y] = ODE23('yprime', [T0 Tfinal], Y0) integrates the system of ordinary differential equations described by the M-file YPRIME.M, over the interval T_0 to Tfinal, with initial conditions Y_0. [T, Y] = ODE23(F, [T0 Tfinal], Y0, TOL, 1) uses tolerance TOL and displays status while the integration proceeds. The other routine, *ode45*, uses 4th and 5th order Runge-Kutta formulas. [T,Y] = ODE45('yprime', [T0 Tfinal], Y0) integrates the system of ordinary differential equations described by the M-file YPRIME.M, over the interval T_0 to T_{final}, with initial conditions Y_0. [T, Y] = ODE45(F, [T0 Tfinal], Y0, TOL, 1) uses tolerance TOL and displays status while the integration proceeds.

To begin, open the MatLab application. This is done by clicking **Start** -> **All Programs->Math Programs** -> **MATLAB R2015a**. In the menu bar of the MatLab Command Window, select the **New Script** button in the **File** pane of the **Home** tab. The **Matlab Editor** will appear. Type the following:

```
function xdot = exmplode(t,x)
% EXMPLODE Evaluation of ODE Derivative
% For Second order ODE
% x'' + bx' + cx = f(t)
% b=c=2
% f(t)= u(t-2)
xdot = [0 1;-2 -2]*x + [0;1]*(t>=2);
% end of exmplode.m
```

**Note: % denotes a comment line*
**Note that (t>=2) is just MatLab's way of duplicating the delayed unit step*

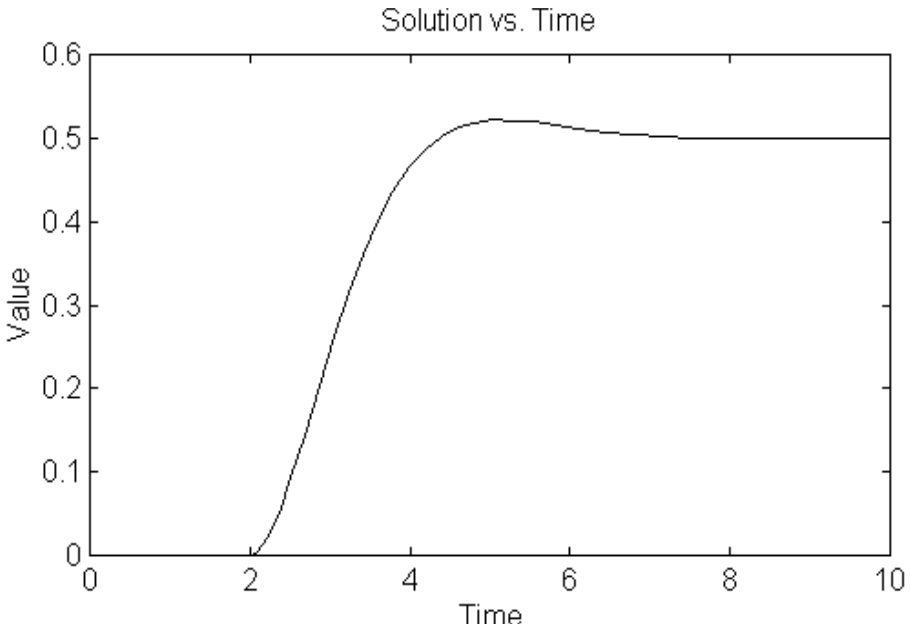
$$u(t-2)=\begin{cases}1, t\geq 2\\ 0, t<2\end{cases}$$

Save the file as **exmplode.m**.

The steps required to carry out the integration of this system are best placed in an M-file script. Follow the previous steps to create a script called **runexmpl.m**, with the following script:

```
% Script which drives ODE solution for ODE in exmplode.m
  xphi = [0;0];           % initial condition
  tphi = 0;       % Initial time
  tfin = 10;      % Final time
  [t,x] = ode45('exmplode',[tphi tfin],xphi);
  plot(t,x(:,1))      % Plot of solution vs. time.
```

Execute the first example by typing **runexmpl** at the MatLab command prompt and the plot will appear.



To add x and y axis labels and a title to the plot, go to the menu in the graph window under **Insert** click **X label,** once you have added the x label text do the same for the **Y label** and **Title**.

Your plot should now resemble the one above.

Example 2 - A Nonlinear ODE, The Classic Pendulum Problem.

Take the equation for the pendulum:

$$ml^2\ddot{x} + cl\dot{x} + mgl\sin(x) = 0$$

where: m = mass = 1kg
l = length = 0.5m
c = damping constant = 0.1
x = angle from vertical in radians $x(0) = \pi/6$ and $\dot{x}(0) = 0$
g = 9.81 m/s^2

As in example 1, the equation needs to be re-written as a system of first-order differential equations. After this is done, we are left with:

$$\ddot{x} + \frac{c}{ml}\dot{x} + \frac{g}{l}\sin(x) = 0$$

For this example, let the following be true:

$$x1 = x\text{, and } x2 = \dot{x} = \dot{x}1.$$

Assuming this, we end up with:

$$\dot{x}2 = \ddot{x} = -\frac{c}{ml}\dot{x} - \frac{g}{l}\sin(x) = -\frac{c}{ml}x2 - \frac{g}{l}\sin(x1).$$

The important parts of this are:

$$\dot{x}1 = x2\text{, and } \dot{x}2 = -\frac{c}{ml}x2 - \frac{g}{l}\sin(x1)$$

To get the above equation into MatLab, it is once again easiest to write an M-file Script. Follow the steps above, and create the following Script.

```
function xdot=pendemo(t,x)
%  PENDEMO Pendulum ODE derivative evaluation
xdot(1,1) = x(2,1);
xdot(2,1) = -0.1/(1*0.5)*x(2,1) - 9.81/0.5*sin(x(1,1));
% End of pendemo.m
```

Save the file as **pendemo.m**. And, just like before, it is easiest to create a script and place it in an M-file to carry out the numerical analysis. Create yet another Script, entering the text below:

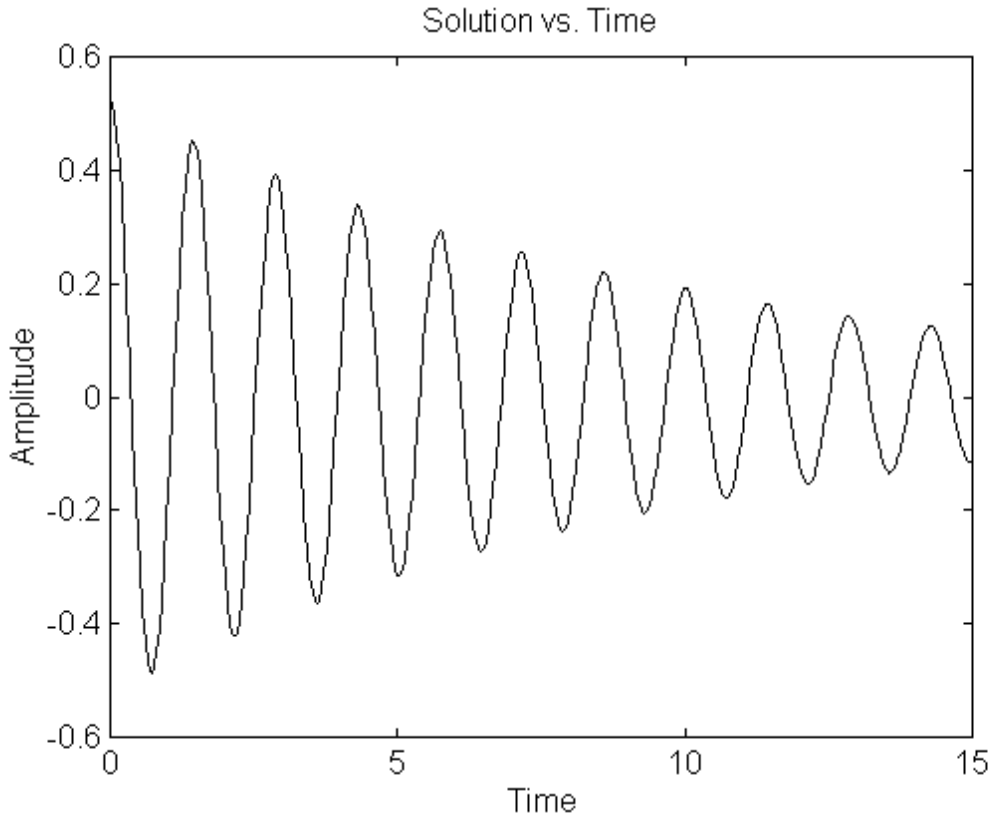
```
% Script file which drives ODE solution of ODE in pendemo.m
```

```
xphi = [pi/6;0];
tphi = 0;
tfin = 15;
[t,x] = ode45('pendemo',[tphi tfin],xphi);
plot(t,x(:,1))
```

Save the file as **runpend.m**, then execute this example by typing **runpend** at the MatLab command prompt. The plot window will appear; to label the plot, use the following commands within the MatLab Command window:

```
title('Solution vs. Time')
xlabel('Time')
ylabel('Amplitude')
```

The finished plot should look somewhat like the following:



Note: If you are going to import a MatLab plot into MS Word, you **must** use a special procedure in order for the plot to display and print correctly in Word. To get instructions for this, type **help word** from the MatLab command prompt.