AMS 27L **LAB #7** **Winter 2009**

Inline Functions and Function M-Files

Objectives:

1. To learn how to write functions
2. To solve differential equations using the "dsolve" command

Today, we'll learn how to write function M-files, subfunctions, and use Matlab functions to examine solutions to diff-EQs.

1 Solutions to Diff-EQs

Let's get some more practice using the `dsolve` command to examine solutions to differential equations. Consider the first order initial value problem

$$xy' + y = 2e^{2x} \quad with \;\; y(1) = 0$$

If we enter the differential equation, without the initial value into the `dsolve` command, Matlab gives us the general solution

```
>> dsolve('Dy + y/x = (2/x)*exp(2*x)','x')
ans =
(exp(2*x) + C1)/x
```

where `C1` is the integration constant. Instead, if we include the initial value $y(1) = 0$ in the `dsolve` command, Matlab solves for the unique solution, which is $y = \frac{e^{2x}-e^2}{x}$.

Problem #1:

(a) Calculate the unique solution to the initial value problem using `dsolve`.

(b) Let's pretty-up the solution by entering:

```
>> z=simple(y)
>> pretty(z)
```

Note that the `simple` command, isn't that simple! In fact, the `simple` command executes a number of Matlab commands that simplify symbolic expressions, and returns the simplest form. The `pretty` function prints symbolic output in a format similar to typeset.

Problem #2: Consider an object thrown in the air obeys the initial value problem

$$\frac{d^2y}{dt^2} = -9.8 - \frac{dy}{dt}, \quad with \;\; y(0) = 0 \;\; and \;\; y'(0) = 120.$$

where y is the height in meters of the object above ground level after t seconds. Use the Matlab function `dsolve` to solve the linear equation and show that

$$y = \frac{-49}{5}t + \frac{649}{5}(1 - e^{-t}).$$

Problem #3: Verify the solution to the differential equation by showing that $\frac{d^2y}{dt^2} + 9.8 + \frac{dy}{dt} = 0$, using the definition of $y(t)$.

2 Function M-Files

By programming a function M-file, we will be able to predict the height of the ball at $t = 5$ and for any value of t. Open a new **function M-file** in the MATALB editor, and enter

```
function y=height(t)
    y = -(49/5)*t + (649/5) *(1 - exp(-t));
end
```

and save it as `height.m`. To find the height at $t = 5$ seconds, we simply enter

```
>>   y=height(5)
```

Problem #4: Predict the height of the ball after 5 seconds, using a function M-file.

Note: If your search path does not include the directory that **height.m** is saved in, you will get an error that says your function is undefined. You will need to add your working directory into the search path.

Writing Subfunctions. We can combine more than one function into a file, by using subfunctions. Subfunctions are a relatively new addition to Matlab. For practice, we'll plot the function, `height`, using a subfunction. Open a new **function M-file** in the MATALB editor, and enter

```
function plotheight
 close all
 t = linspace(0,15,200);
 y = height(t);
 plot(t,y)
 grid on
 xlabel('time in seconds')
 ylabel('height in meters')
 title('Solution of y'''' = -9.8 - y'',y(0)=0,y''(0) = 120')
end


function y=height(t)
         y = -(49/5)*t + (649/5) *(1 - exp(-t));
end
```

and save it as `plotheight.m`. The command `close all` will close all open figure windows. Be careful and make sure that's what you want to do! You can select **Debug→Run** from the editor menu to execute the commands. Alternatively, you can use the **F5** key to save and execute the file. We have just written the single M-file `plotheight` with `height` included as a subfunction.

Problem #5: Program the subfunction `plotheight` to plot the trajectory of the ball through time.

3 Functions of Functions

Matlab has a large number of functions that act on functions. If you type

```
>>   help funfun
```

you will see a list of MATLAB routines for finding zeros and extrema of functions, tools for numerical integration, and more. We can use the function `fzero` to find the time it takes the ball to reach the ground. Using the help page for `fzero` we see that the command takes two inputs, the first is a *function handle* and the second is a value near the zero. The expression `@height` is the function handle for the function `height`.

```
>>   t=fzero(@height, 13)
```

Problem #6: How long does it take the object to return to ground level?

In the list of functions provided by `help funfun`, we see that Matlab doesn't have a function to find the maximum of a function. However, we can find the maximum by finding a local minimum of the negative of the function. The function `fminbnd` finds a local minimum by supplying the function handle and the beginning and endpoints of the interval in which to find the minimum.

We can write a function for the negative of our height function and save it as an M-file, although Matlab provides us with a shortcut: *inline functions!* We can do this in two possible ways

```
>>   f = inline('(49/5)*t - (649/5)*(1 - exp(-t))','t')
```

However, since we've already created the function, `height`, we can create the function, f, as

```
>>   f = inline('-height(t)','t')
```

We can estimate that the maximum height occurs between $t = 0$ and $t = 5$, so the command

```
>>  t = fminbnd(f,0,5)
```

finds the time at which the maximum occurs. Notice that there are no single quotes around `f` in `fminbnd(f,0,5)`, because inline functions are treated differently from function M-files as inputs.

Another way to calculate the time it takes to reach the maximum height by combining the previous two commands into a single command, by

```
>>   t = fminbnd(inline('-height(t)'),1,5)
```

Problem #7: What is the maximum height reached?

A plot can contain many subplots by using the Matlab command `subplot`. The following codes will create a $2x3$ matrix of tiled plots

```
>>  subplot(2,3,1), plot(t,y)              Plot in the first 2x3 tiled location.
>>  subplot(2,3,2), plot(t,y)              Plot in the second 2x3 tiled location.
>>  subplot(2,3,3), plot(t,y)              Plot in the third 2x3 tiled location.
```

and so on.

Problem #8: A simple RC-circuit with emf $V(t) = 3\cos(\omega t)$ is modeled by the initial value problem

$$RCV_C' + V_C = 3\,cos(\omega t), \quad V_C(0) = 0,$$

where R is the resistance, C the capacitance, ω is the driving frequency, and V_C is the voltage response across the capacitor. Show that the solution is

$$V_C(t) = \frac{RC\,\omega\sin\omega t + \cos\omega t - e^{-t/(RC)}}{1 + R^2C^2\omega^2}$$

Create an M-file to plot the voltage response for $0 \le t \le 2\pi$ for $\omega = 1, 2, 4$, and 8, keeping $R =$ 1.2 ohms and $C = 1$ farad constant. Use a subfunction to evaluate the voltage response solution, with four parameters: t, R, C and w, as we did for the subfunction `height` in `plotheight.m`. Plot the voltage response solutions for the different values of the driving frequency on the same graph. Use the function `subplot` to plot a $2x2$ tiled matrix of plots on one page.

Problem #9: Why do you think that the simple RC-circuit modeled in the previous problem is called a *low pass filter*?

Extra Credit: Suppose we have a population of 100 individuals at time $t = 0$ and that the population is correctly modeled by the logistic equation. Suppose that at time $t = 2$ there are 200 individuals in the population, and that the population approaches steady state at a population of 1000. Plot the population over the interval [0,20]. What is the population at time $t = 10$?

Quit MATLAB by clicking on the `File` menu in the upper left corner and choosing `Exit`. Please remember to **Log Off** (from the "Start" menu in the lower left of the screen).