MATH 1070, Introduction to Computational Science

Lab Session 3 – calculations and problem Solving in MATLAB

Getting started

This lab runs on the PCs in the CLC, under the M/S Windows operating systems. If you are not already familiar with: finding and opening files, launching applications, etc. in windows; then you should take a few minutes to familiarize yourself with the computers in the lab – and feel free to ask for help.

Next, you should launch MATLAB, from the START -> APPLICATIONS menu on the lower LHS of the screen. Look for the MATLAB HELP menu, and browse around it, to get a feel for how its structured. This will be a key resource for you. Also look for, and use, the document file "SimpleMatLabCmds" on the MATH1070 web site. In Lab 1, you learned how to get started with MATLAB, while Lab 2 introduced some simple calculations and logical tests. As necessary review those exercise (*for*, *if*, etc).

Today's lab

In this tutorial you will attempt some applied calculations in MATLAB, including free fall motion and plot the results. You also will learn to put code modules into a special type of m-file, called a function.

1. A simple calculation

In his book , *Liber Abaci*, published in 1202, Leonardo Fibonacci posed the following problem"

> *A man put a pair of rabbits in a place surrounded on all sides by a wall. How many pairs of rabbits can be produced from that pair in a year it is supposed that every month each pair begets a new pair, which from the second month on becomes productive?*

The solution to this problem is known as the Fibonacci series. The Fibonacci numbers x_n are defined as follows $x_0 = x_1 = 1$ and $x_{n+1} = x_n + x_{n-1}$ for $n > 1$. Find the first 100 Fibonacci numbers and put them in a vector fib.

Exercise 2 Quadratic eqn.

a) Write a simple script file to solve a standard quadratic equation.

b) Now you will implement a Matlab **function** to do the same thing. You need to write two small codes: one is the main program, to set up the calculation, do I/O etc., and the other is the function or subroutine code. M-files can be either scripts or functions. So far you have used scripts which are simply files containing a sequence of MATLAB statements. Functions make use of their own local variables and accept input arguments: sin(x), cos(x), etc are examples that you already are familiar with. The name of a function, as defined in the first line of the M-file, should be the same as the name of the file, without the .m extension. For example, the existence of a file on your hard disk called **avg.m** defines a new function called **avg** that calculates the mean of a data vector:

```
function mean = avg(x,n)
mean = sum(x)/n;
```

The variables within the body of the function are all local variables – they are known only in the function code itself. You call this function (ie. the m-file) by a command that might look like:

```
average = avg(DataVector, NumberOfPoints)
```

You call this function from your "main" program just as you would call sin(x), etc. Functions normally return when the end of the function is reached. Use a **return** statement to force an early return – eg. after a logical test fails. The arguments (input) and output of a function can be either scalars or vectors. You can call your new function anything you like, so long as its not a Matalb "reserved name", like sin, cos, for, while, zeros, etc. Its best to make it descriptive, so later you don't forget what it does: eg. QuadraticRoots. Since there are three parameters and two roots you will need to use vectors. Use comments (%) to describe what it does – again, so that later you (or another user) knows what it does.

c) Put logic tests in the function code to test for exceptions (errors, bugs), like a = 0, etc. (*Anon. A bug is an exception you didn't think of yet*).

Exercise 3 Free Fall motion - Kittinger

On 16 August 1960, US Air Force Captain Joseph Kittinger jumped from the gondola of an helium balloon at an altitude of 31,354 m and took the longest skydive in history! Kittinger reports in his article in National Geographic that: *"No wind whistles or billows my clothing. I have absolutely no sensation of the increasing speed with which I fall. [The clouds] rushed up so chillingly that I had to remind myself they were vapor and not solid."* At this altitude the density of air is low therefore drag can be ignored but note that g = 9.72 m/s^2. Kittinger set the world's record (which remains unbroken) for the longest (1.95 miles) and fastest (4 minutes and 36 seconds) sky dive. He jumped from 102,800 feet (31.3 km). He was in freefall for ~4 minutes and reached a maximum speed of 614 mph (982 km/h) before opening his parachute at 18,000 feet (5,500 m).

1) Write the set of equations for velocity and position by applying Newton's second law to the and writing down the analytical solutions to the 1-D equations of motion (refer to Lecs 3 & 4, or any 1st yr Physics text). We assume that Kittinger starts his fall from rest (v=0) and that his mass is 150 kg. Write a MATLAB code to calculate his speed v(t) and height y(t). Hint: start by loading an array with the time values that you wish to use. Use the Matlab **PLOT** command (learn by reading the HELP menu and trying an example) to plot these functions. Learn about the label, axis & title options for PLOT and label your graphs. Ex 4 also has more information on plot.

2) Kittinger reports that he was in free fall from 31,354 m to 29,280 m and experienced no noticeable change in acceleration for an additional 1,830 m, despite having deployed his stabilization chute to avoid spinning. What would have been his speed and position then? Did Kittinger become supersonic? (speed of sound at 30,000 m altitude is 300m/s).

In subsequent labs we will solve the DEs for this problem, using ODE45, and then include drag forces, which slow the fall and make the model and calculations more realistic.

Exercise 4 CSI with MATLAB

(Courtesy A/Prof Ian Johnson, Physics, University of Sydney)
A body was found on the bottom of a disused quarry. The death is highly suspicious and the police has hired you as an expert witness to determine whether this was a suicide or there was foul play.

The wall of the quarry nearest the point where the body was found rises about 33 m, in a series of three distinct levels. The cliff face profile from the surveyor's drawing gives:

```
cliffProfile = [ 0  8 12 17.5 18 22 25 30 40;...
                   33 29 21 19 15 14  2  0  0];
```

where the two rows represent horizontal and vertical coordinates of points along the cliff face (all measurements are in metres). The two lower levels about halfway up the cliff. The body was found at the very base of the cliff, at the point (30,0).

NB: a Matlab statement can be continued to the next line with an ellipsis, or three dots: …

You must determine the following:
1) Could the body have reached the point where it was found if the person had jumped from the top of the cliff, or from either of the two lower levels?
2) Could the body have been thrown there from either the top of the cliff or the first ledge down?
From your physics classes you know that, if an object is thrown vertically upwards from the surface of the earth with a (vertical) velocity v_y, its vertical position y at any subsequent time t is given by:

$$y = v_y t - 1/2\, g\, t^2 \qquad (1)$$

Likewise, if the object is thrown horizontally with a (horizontal) velocity v_x, its horizontal position x at any subsequent time t is given by:

$$x = v_x t \qquad (2)$$

a) Write a short MATLAB script to draw the cliff face, ready for you to superimpose on it any trajectory you calculate. Use the **PLOT** command to draw the cliff face:

```
plot(cliffProfile(1,:), cliffprofile(2,:));
```

When you plot a trajectory, you do not want the axes of the graph to change from the values 0–40 m along the *x*-axis and the same along the *y*-axis, so use the command:

```
axis([0 40 0 40]);
```

b) Construct a vector representing the time from 0 to 10 seconds, in steps of 0.01 s. Then choose the following values for the components of the initial velocities and the acceleration due to gravity:

$v_x = 4$ m/s , $v_y = 3$ m/s, $g = 9.8$ m/s^2

Then construct two vectors to represent x and y at each of these times. Plot y vs. x superimposed on the drawing of the cliff. Your trajectory should lie close to the x-axis, starting at the origin. But at least you should be able to answer the question: are the values above reasonable values to choose for the initial velocities? You will need to use MATLAB's **HOLD ON** and **HOLD OFF** commands to superimpose graphs.

c) Repeat the calculation, this time adding constant initial displacements to the position vectors for x and y, so that the trajectory starts from a point on the cliff profile.

d) When you have got everything working properly, incorporate all the above statements into a MATLAB script which you can use over and over. You will want to re-run the script with many different values of v_x and v_y.
You may want to devise some way of reading in new values while the script is still running. You could use a couple of statements like,

```
vx = input('Enter the value of vx : ')
```

e) Now you are in a position to answer the questions asked above. By entering different (reasonable!) values of u and v, see if you can decide whether the body could have reached the point where it was found if the person had jumped from the top of the cliff, or from either of the two lower levels. Think of how fast you can walk or run.
How should you change your investigation to decide whether the body may have been thrown there from either the top of the cliff or the first ledge down?
We will do more on this problem in coming weeks ……

Conclusions. Obviously there were a lot of simplifying assumptions that went into this simple model of a falling body. Can you spell out what those assumptions were? How reasonable do you think your conclusions were? Would you be prepared to stand up in a court of law to defend the conclusions you drew above?

Acknowledgments: Dr. N Bordes, MATH2200; A/Prof I Johnson (Physics, USyd), CP2 & COSC1001.