MATH 1070, Introduction to Computational Science

Lab Session 4 – calculations and problem Solving in MATLAB

Getting started (still)

This lab runs on the PCs in the CLC, under the M/S Windows operating systems. You 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 it's structured. Also use the document file "SimpleMatLabCmds" on the MATH1070 web site. In Lab 1, you got started with MATLAB, while Lab 2 introduced some simple calculations and logical tests; then in Lab 3, you solved a quadratic equation using a function call and applied the above techniques to free fall and projectile motion, with plotting of your results. Solutions for selected exercises are included at the end.

Today's lab

In this tutorial you will continue some calculations in MATLAB, You also will learn to put code modules into a special type of m-file, called a function.

Exercise 1. DE – exponential growth & decay

Solve a simple ODE, dx/dt = a x(t), numerically using Matlab's ODE45. You know that the analytic solution is x(t) = x(0) * exp(a*t), so you can compare the accuracy of your numerical solution. You need *two* m-files: one main program, to set up the problem, to call ode45 and to plot the results; and another function to calculate the RHS of your DE.

The *first* file, called "whateveryoulike.m"

```
%  MAIN program
% simple ODE - ex 0 (9/02)    dx / dt = xprime(x,t)
global a  % so function can access value of a

t = [0.0 : 0.1 : 2.0];  % set up time axis array, here h =0.1
a = -1.0;                % rate parameter  - decay
x0 = 1.0;                % initial condition, x(t = 0)

 % RHS of the DE is in the function xprime = a*x.

 [t,x] = ode45('xprime1', [0, 2.0], x0);
         % Runge-Kutta method  ode45 – nb. Time range and IC.
 plot (t, x, '-')

% compare to know analytic (exact) solution
 %  for i = 1:length(t)                % use a for loop to do it "long hand"
%     exact(i) = x0*exp( a*t(i) );    %  load array with exact solution to DE
%  end
exact = exp(a*t) ;  % vectorised calculation – more compact!

plot (t, x, 'o', t , exact, '-')  % plot 2 curves to compare numerical & exact solutions
xlabel('time,  t'); ylabel(' x(t)  ,  exact(t)' ); title ('simple DE,  a = -1,  x0 = 1')
```

The *second* file, called "xprime1.m" - the name must match that in '....' in the ode45 call.

```
function xprim = xprime1(t, x)
global a
xprim =  a*x;          % RHS of the DE
```

Experiment with the step size, h (look for it above!) to check the accuracy of the numerical solutions. Vary a, both negative and positive (for exp growth).

Try the non-linear DEs, dx / dt = x^2 ; dx / dt = -2x -y.

Exercise 2 Free Fall motion - Kittinger (cont'd from Lab 3)

Recall from Lab 3, the 1960 Kittinger jump from an altitude of 31,354 m where the low air density causes zero drag and g = 9.72 m/s^2; his mass =150 kg and he started his fall from rest, v(t=0) = 0. You used analytical equations for velocity and position to follow his 1-D motion; ie. directly calculated his speed v(t) and height y(t).

1) Now you will numerically solve the DEs for this problem, so that later you can include drag forces, which slow the fall and make the model and calculations more realistic. Newton's second law can be written as two simultaneous **1st** order DEs:

$$m\frac{dv}{dt} = mg$$

$$\frac{dy}{dt} = v$$

As they stand these two equations are decoupled. So you can start by solving them separately. Numerically solve for v(t) only using the 1st eqn – just like you did in Ex 1. How do you find the height, y(t)? Now solve the two DEs for <u>both</u> y(t) and v(t) using the Matlab function **ODE45** and compare to the analytic results (cf. Lab 3). Now xprim (cf. above) will be a vector with <u>*two*</u> components, y & v; you need an IC for each in the call to ODE45.

2) As Kittinger reaches altitudes where the air is thicker (denser), his body experiences a drag force due to the air resistance and his velocity decreases. Last week, you found that altitude to be 27,450 m, so that y(t ~28 s) = 3,904 m & v = 275.5 m/s; use these as your ICs. The drag force is proportional to the velocity (linear), so the total force is F = mg – k*v(t), where k is the drag constant: k = 0.408 (N s/m). We now assume that g=9.8 m/s^2 and remains constant for the rest of the fall. Newton's equations become:

$$\frac{dy}{dt} = v$$

$$m\frac{dv}{dt} = mg - kv$$

Or

$$\frac{dy}{dt} = v$$

$$\frac{dv}{dt} = g - \frac{k}{m}v$$

These can be solved analytically however we will use Matlab to do it numerically. Modify your MATLAB code to calculate v(t), subject to this drag. What is the speed and trajectory of Kittinger? Explain whether the model makes sense.

This model is <u>not</u> realistic: Kittinger hits the Earth in some 50 s, with a terminal velocity of about 3500 m/s. The assumption of linear drag is more appropriate for water then air.

3) Now let's assume that the drag force is proportional to the <u>square</u> of the velocity (k = 0.408 N s^2 /m^2 - nb. New units). Newton's equations now become:

$$\frac{dy}{dt} = v$$

$$m\frac{dv}{dt} = mg - kv^2$$

Or

$$\frac{dy}{dt} = v$$
$$\frac{dv}{dt} = g - \frac{k}{m}v^2$$

These also can be solved analytically; however we still use Matlab to solve them numerically. Modify your MATLAB code above to calculate this new drag law. What is the new speed and trajectory of Kittinger? Does this new model makes sense?

Kittinger would hit the Earth in about 7 minutes (without a parachute) with a terminal velocity of about 60m/s:

4) Kittinger opened his large chute at 4 min 49 sec after jumping, when he had reached an altitude of about 5334 m. Does the model discussed in 3), above fit? Explain why.

Advanced

5/ Several skydivers have been planning to beat Kittinger's record by jumping from an altitude of 40,000 m. If we assume g=1.72 m/s^2, would the skydiver break the sound barrier (300m/s)? If yes, at what altitude?
What would be a possible maximum speed at 3,904 m, as you calculated Kittinger's speed in 2) ?

TO CONFIRM: altitude would be 35,360 m and speed 494 m/s

Exercise 3 CSI with MATLAB (cont'd).

(Courtesy A/Prof Ian Johnson, Physics, University of Sydney)
Recall your cliff face profile from last week:

```
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 (measurements in metres).

You considered 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 Newton's equations of motion, 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:

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

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:

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

The combined (x,y) path is the trajectory of a projectile. You started by constructing a vector representing the time from 0 to 10 seconds, in steps of 0.01 s and then choose some values for the components of the initial velocities and the acceleration due to gravity, eg:

$v_x = 4$ 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. 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.

a) Work through the above, if you did not finish it last week.

b) Now you are in a position to answer the questions asked above. By entering different (reasonable!) values of v_x and v_y, 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; and how high you could jump. Discuss this with your neighbour(s) in the lab – and experiment!
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?

c) In the above you approximated the body as a point particle (do you feel like a point?); whereas in fact the man was 160 cm high and weighed 75 kg. Does the fact that the person's centre of mass is above the ground to start with affect the conclusions you drew? Modify y(0) to correct for this and repeat the calculations.

d) Advanced: Could the body have reached the point it did if it simply toppled from the edge of either level 1 or level 2? (By "toppling" we mean standing on the edge and falling forward keeping your body rigid, so that your feet stay in contact with the ground for some time). If the body could have reached that spot by toppling, would it have hit its head on the side of the cliff on the way down?

Is air drag likely to be important for this problem (cf. Ex 2).

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.

Solution: Ex 2 (lab 3)

```
% solve a quadratic equation
% Main program to call a function
% a*x^2 +b*x + c = 0
%   watch for instabilities (1, -100, 1);
%    (1, 2.9, 2);  try  b = big
a = 1;
b = 30;
c = 2;
roots = zeros(1,2)    % set up array
roots = Fquadratic(a,b,c)  % call the function

function roots = Fquadratic(a,b,c)
% Function to solve a quadratic equation
% a*x^2 +b*x + c = 0
 % do the tests – eg:   if(a= =0) "do something else"!
term = sqrt( b*b - 4*a*c);
div = 2*a;
% the 2 roots:
roots(1) = (-b + term)/div
roots(2) = (-b - term)/div
% nb.    a 1x2 array is returned
return;  % often optional
```

CHECK: case a=0 : Does code default to the linear case?

Solution: Ex 3:

Part 1). Assuming that the origin of the coordinate system is the altitude, y where Kittinger jumps from, and the y axis is positive in the down direction, Newton's second law gives:

$$m\frac{dv}{dt} = mg$$

$$\frac{dy}{dt} = v$$

Solving the equation, we get:

$$v = gt + v_0$$

$$y = g\frac{t^2}{2} + v_0 t + y_0$$

At $t=0$, $v_0=0$ hence $v = 9.72t$

$$\frac{dy}{dt} = 9.72t$$

Solving the equation, we get:

$$y = 4.86t^2 + y_0$$

At $t=0$, $y_0=0$ hence $y = 4.86t^2$

Matlab Program

```
 % free fall motion
 % load arrays first
time=[0:28];
yvzero=[0,0];
 % calculated values
calc_v=9.72.*time;
calc_y=(9.72.*time.^2)./2;
% plot results & label axes, etc.
plot(time, calc_y), title('trajectory'), xlabel('time'), ylabel('y'),  axis([0 50 0 3900])
```

```
plot(time,calc_v, 'rx'), title('v'),  ylabel('velocity'),  axis([0 50 0 300])

% note: the above uses "vector instructions" – you could use a for loop to iterate though the
% array index – eg. for i=1:28  etc…
```

Part 2).

31354 – 29280 + 1830= 3904 (m) (for an observer on the ground, 27,450 m)

t ≈28s

v= 275.5 (m/s) - not quite supersonic (~300 m/s).

Solution: Ex 4 (CSI – cliff face):

A possible implementation Code:

```
% Function: cliffinvestigate
%
% Answers to MATH 1070 lab ex.  Plotting the cliff and the initial guess trajectory.
% This function allows various initial points to be tried and calculates the
% required velocities.  The enables the user to check the validity of a given initial
% velocity vector as required by the question.  Run the function as shown below and follow
% the prompts.
%
% Usage:  initialxvelocity = cliffinvestigate(cliffx,cliffy)
%
% Ben Gladwin (Maths, UQ) 2003.

function initialxvelocity = cliffinvestigate(cliffx,cliffy)
close all;
%Gravitation constant
g = 9.81;

% set up the cliff face boundary condition

cliffProfile = [ 0  8 12 17.5 18 22 25 30 40;...
                 33 29 21 19 15 14  2  0  0];
%Plot the cliff face.
figure(1)
plot(cliffProfile(1,:), cliffprofile(2,:));
axis([0 40 0 40]);
xlabel('Distance from base of cliff.'); ylabel('Height.');
title('Plot of Cliff Face');

%Construct the time vector.
time = 0:0.01:10;

%Prompt user for data.
initx = input('What is the initial x coordinate of the body?');
inity = input('What is the initial y coordinate of the body?');
velx = input('What is the velocity in the x direction (the value of u)?');
vely = input('What is the velocity in the y direction (the value of v)?');

% Construct the X and Y vectors.
X = velx*time + initx;
Y = vely*time - (g * time.^2)/2 + inity;

%Check the trajectory.
figure(1)
hold on;  % allows tradjectory plot over the cliff face
plot(X,Y,'r');
legend('Trajectory of Body');
hold off
```