EE317 / EE303 Computation and Simulation Assignment 2014-2015

February 3, 2015

1 Introduction

EE317/EE303 is a 5-credit module examined entirely by continuous assessment. You will work in teams of 2 or 3 to produce Matlab codes to solve a number of engineering problems outlined below. Please note that your team is expected to independently produce all such codes and **any form of plagiarism will be dealt with severely. Each member of each team must understand, and be able to explain all codes produced**. The assignments in this document are worth 30% of the total mark. Finalised codes should be emailed to me at brennanc@eeng.dcu.ie before Friday the 13th March at 5pm. These should be named as per the instructions below and commented sufficiently clearly that I can run and understand them.

2 Problem One: Modelling the solar system

For this problem you are asked to solve the ODEs governing the trajectory of a planet around the sun. Firstly we introduce an exact solution against which you can monitor the accuracy of your solver and then we discuss the equations of motion.

2.1 Preliminaries

According to Kepler's laws bodies orbit the sun in elliptical (rather than circular) orbits.An example is given in figure (1). The sun lies at one of the two focii of the ellipse and the planet orbits in a trajectory relative to the sun given by $(r(t), \theta(t))$. The (x, y) coordinates relative to the sun are thus

$$\begin{aligned} x(t) &= r(t)\cos\theta(t) \\ y(t) &= r(t)\sin\theta(t) \end{aligned}$$

A planet's location at any point in time must satisfy

$$r = \frac{p}{1 + \epsilon\cos\theta}$$

where p is the semi-latus rectum (see figure (1)) and ϵ is the eccentricity of the orbit (defined later). From this relationship we can deduce certain features of a planet's orbit. For example the planet is nearest the

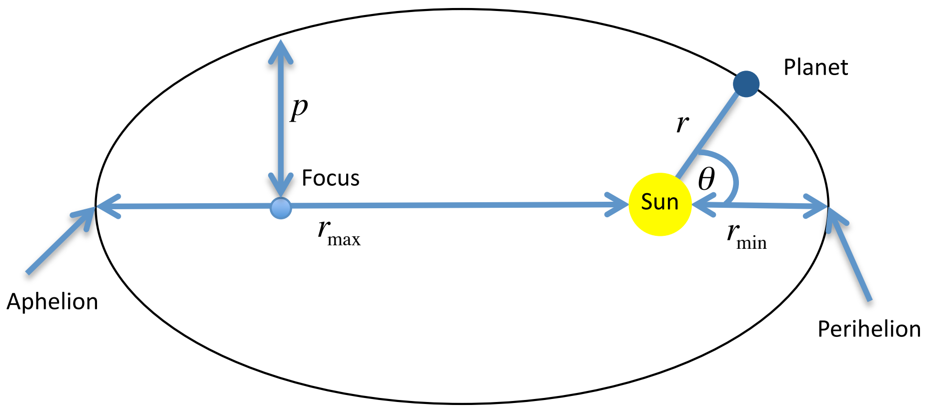

Figure 1: Planets orbit the sun in elliptical orbits

.

sun at perihelion when $\theta = 0$ and so

$$r_{min} = \frac{p}{1+\epsilon}$$

It is furthest from the sun at aphelion when $\theta = \pi$ and so

$$r_{max} = \frac{p}{1-\epsilon}$$

The eccentricity ϵ is given in terms of these minimum and maximum distances as

$$\epsilon = \frac{r_{max} - r_{min}}{r_{max} + r_{min}}$$

The semi-major axis a is the arithmetic mean of r_{max} and r_{min} and can be expressed as

$$a = \frac{p}{1-\epsilon^2}$$

Kepler's third law states that the square of the orbit period (i.e. the time taken to complete one full revolution around the sun) is proportional to the cube of the semi-major axis of the orbit.

$$P^2 = \frac{4\pi^2}{MG} a^3$$

where M is the mass of the sun and G is the gravitational constant ($6.67428 \times 10^{-11} N(m/kg)^2$). Accurate values for the relevant quantities (P,r_{min}, r_{max}, a, p *etc.*) can be readily found on the internet for all the planets of the solar-system (for example at http://en.wikipedia.org/wiki/Earth). Armed with these quantities you are now in a position to model the trajectory of the planets around the solar system.

2.2 Modelling the trajectories of the planets

Assume that at time $t = 0$ the planet is at perihelion ($\theta = 0$). The position in polar coordinates (r, θ) at time t since perihelion can be computed as follows. Compute the so-called *mean anomaly* M as follows

$$M = \frac{2\pi t}{P}$$

Now use M to compute the *eccentric anomaly* E.

$$M = E - \epsilon \sin E$$

Note that you will need to apply Newton Raphson, or some other root finding method to solve the above transcendental equation. Now having computed E compute θ according to

$$\tan\frac{\theta}{2} = \sqrt{\frac{1+\epsilon}{1-\epsilon}} \tan\frac{E}{2}$$

Finally compute the heliocentric distance r from

$$r = \frac{p}{1 + \epsilon \cos\theta}$$

Using this approach you can compute $(r(t), \theta(t))$ and hence $(x(t), y(t))$ for any given value of t. Your task is to create an animation modelling the trajectories of all the planets[1] in the solar system. Assume that at $t = 0$ all planets are at their perihelion and are arranged along the x axis with the sun at the origin. Relevant orbital parameters (such as r_{min} and r_{max}) can be found on the internet for each planet.
Deliverables:
1.1) You should produce a code, called "planet_motion.m" which prompts for the length of time (in earth years) the simulation should model and produces an animation showing the trajectory of the planets. Use a finite difference expression to estimate the velocity of each planet as it orbits and, in a separate figure, plot this against time for each planet.

[1]You can decide whether or not to include Pluto given its 2006 declassification to dwarf-planet status by the International Astronomical Union

2.3 Solving the equations of motion

Now your job is to validate the above exact solution of the equations of motion by instead **numerically** solving the governing equations of motion for Earth. Assume that at $t = 0$ Earth is at its perihelion and is travelling with velocity (v_x, v_y) where

$$\begin{aligned} v_x &= 0 \\ v_y &= \sqrt{GM\left(\frac{2}{r_{min}} - \frac{1}{a}\right)} \end{aligned}$$

The subsequent position of Earth $(x(t), y(t))$ depends on this initial velocity and also the acceleration experienced via the following equations

$$\begin{aligned} \frac{dx}{dt} &= v_x(t) \\ \frac{dy}{dt} &= v_y(t) \\ \frac{dv_x}{dt} &= a_x(t) \\ \frac{dv_y}{dt} &= a_y(t) \end{aligned}$$

If we know the acceleration we can compute the velocity and hence the position. What is the acceleration experienced by the planet? Well referring to figure (2) we see that the force on the planet is given by

$$\vec{F} = -\frac{GMm\vec{r}}{|\vec{r}|^3}$$

where m is the mass of the Earth, and so, using $\vec{F} = m\vec{a}$ we can say that

$$\vec{a} = -\frac{GM\vec{r}}{|\vec{r}|^3}$$

and so

$$\begin{aligned} a_x(t) &= -\frac{GMx(t)}{(x^2(t) + y^2(t))^{\frac{3}{2}}} \\ a_y(t) &= -\frac{GMy(t)}{(x^2(t) + y^2(t))^{\frac{3}{2}}} \end{aligned}$$

Deliverables:
1.2) You should produce a code, called "numerical_motion.m" which solves these equations of motion for planet Earth using a suitable ODE solver. An animation should be produced showing the computed orbit and comparing it against the exact solution of the previous section. Care should be taken to choose a suitable step size to ensure reasonable accuracy in realistic computation times. The code should

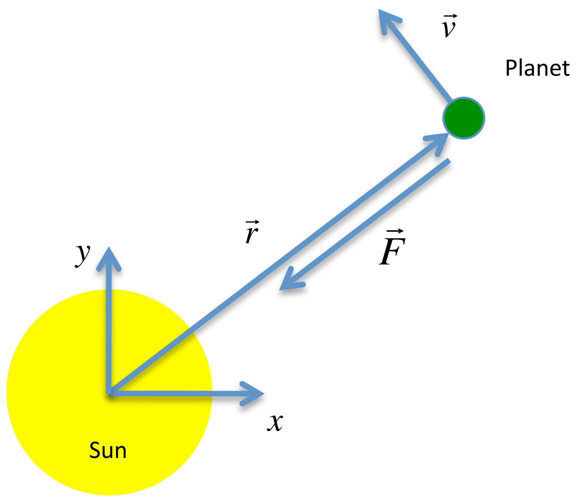

Figure 2: Equations of motion for planet

.

produce a separate graph illustrating the growth of the error in position incurred as the numerical solution progresses.
1.3) Develop some further innovation beyond what I have asked in 1.1) and 1.2). Perhaps you could investigate the effect of Jupiter on Earth's orbit? Perhaps you could model the voyage of a rocket from Earth to Mars? Perhaps you could statistically examine the likelihood of Earth being struck by an asteroid whose orbit intersects that of Earth? This code should be called "innovation.m"

3 Problem Two:

A magic square is a square array of *distinct* natural numbers such that the sum of elements along each row, each column and each diagonal (from top left to bottom right and from top right to bottom left) is the same. Write a Matlab code that will check if a given array is magic. Your code should read in the matrix from a file called "square.res". The file has a single column of $N^2 + 1$ numbers.
The first line is N - the number of rows (or columns). The next N lines are the values in row 1, the

next N are the values in row 2 etc. Your code should read in the matrix, perform the test and output answering whether the square is magic or not.
Example:

$$\begin{matrix} 2 & 7 & 6 \\ 9 & 5 & 1 \\ 4 & 3 & 8 \end{matrix}$$

is a magic square. It has 9 distinct numbers and each row column and diagonal sums to 15.
Deliverables
2.1) Write a Matlab code, called "magic.m" that will read in a square array from the file "square.res" and check whether it is magic or not.

4 Problem Three:

You need to lay a pipeline through a city. The city is represented by a square grid of $N \times N$ cells. Each cell has an associated natural number which represents the cost of digging up that cell to lay the pipe. Your task is to minimise the total cost of laying the pipe. The pipeline starts at any row of column 1 and ends at any row of column N. When the pipeline enters column i (at cell (j, i), row = j, column = i) the pipeline can progress to column $i+1$ in several ways:
• Lay the pipeline upward any number of cells (say d) within the city boundary and move to column $i+1$ at row $j-d$, i.e. $d+1$ cells in column i are dug up.
• Lay the pipeline downward any number of cells (say d) within the city boundary and move to column $i+1$ at row $j+d$, i.e. $d+1$ cells in column i are dug up.
• Lay the pipeline directly to column $i+1$ through the current cell. Only 1 cell in column i will be dug up in this case. You must write a code to identify the cheapest route that the pipeline should take across the city. The code should read in the following data from a file called "city.res". This file contains a single column of N^2+1 values.
The first line is N - the number of rows (or columns). The next N lines are the values in row 1, the next N values in row 2 etc.
Example: Consider the following 5×5 city.

$$\begin{matrix} 1 & 1 & 9 & 1 & 1 \\ 3 & 2 & 9 & 7 & 1 \\ 4 & 1 & 9 & 1 & 2 \\ 8 & 1 & 1 & 1 & 6 \\ 7 & 2 & 8 & 4 & 8 \end{matrix}$$

The cheapest route across this city is to start at entry $(1,1)$ (cost = 1), go to $(1,2)$ (cost = 1), travel downwards through cells $(2,2), (3,2), (4,2)$ (cost = 2,1,1), travel across through cells $(4,3), (4,4)$ (cost = 1,1), travel upwards through cell $(3,4)$ (cost = 1), and exit through cell $(3,5)$ (cost = 2). The total cost therefore is 11. Any other route would produce a higher cost.

Deliverables
3.1) Write a Matlab code, called "small_city.m" that will read in the values for a 3×3 city and compute the minimum cost.
3.2) Write a Matlab code, called "general_city.m", that will read in the values for a general $N \times N$ city and compute the minimum cost.

5 Marking

Marks will be awarded as follows:

Deliverable 1.1	5%
Deliverable 1.2	5%
Deliverable 1.3	5%
Deliverable 2.1	5%
Deliverable 3.1	5%
Deliverable 3.2	5%