UNIVERSITY OF SUSSEX
Scientific Computing
Tutor: Dr. Ilian Iliev, Office: Pev III 4C5

Assignment 3
Deadline: 12pm on Thursday, December 11th, 2014.
Penalties will be imposed for submissions beyond this date.
Final submission date: Friday, December 12th, 2014
No submissions will be accepted beyond this date.

1. (a) Write a code to solve for the motion of the anharmonic oscillator described by the equation

$$\frac{d^2x}{dt^2} = -\omega^2 x^3 \tag{1}$$

Take $\omega = 1$ and initial conditions $x = 1$ and $dx/dt = 0$ and make a plot of the motion of the oscillator from $t = 0$ to 20. Increase the amplitude to 10. You should observe that the oscillator oscillates faster at higher amplitudes. (You can try lower amplitudes too if you like, which should be slower.)

(b) Modify your program so that instead of plotting x against t, it plots dx/dt against x, i.e., the velocity of the oscillator against its position. Such a plot is called a phase space plot. Make sure you say **pylab.axis('equal')** to ensure the plot axes have the same length for the phase space plots.

(c) How does the behaviour in (a) and (b) differ (quantitatively) from a normal harmonic oscillator? (Just run your code again after modifying the equation appropriately.) [30]

2. Consider the differential equation (called van der Pol oscillator):

$$x''(t) = -x - \epsilon(x^2 - 1)x'(t), x(0) = 0.5, x'(0) = 0,$$

where ϵ is a positive constant. As the value of ϵ increases, this equation becomes increasingly stiff.

(a) Convert this equation to two first-order equations.

(b) Solve these equations using the Runge-Kutta method of 4th order provided in class with $\epsilon = 10$, and $[a, b] = [0, 10\pi]$ and different choices of step-size: $h = 0.02, 0.05$ and 0.1. Plot all solutions (one-by-one, in the above order), along with the solution obtained by using the in-build `scipy.integrate.odeint` using linear axes, and (in a separate figure) plot the absolute errors of all solutions using semi-log axes. What do you observe? **Note that the in-build method has a different interface to the function defining the equation(s) compared to the supplied functions (the arguments are reversed). Result is also formatted differently, check the help.**.

(c) Make a phase-space (coordinate vs. velocity) plot of the `scipy.integrate.odeint` solution for $\epsilon = 1, 4$ and 10. Make sure you use a small enough value of the time interval h to get a smooth, accurate phase space plot and again use `pl.axis('equal')`.

[40]

3. **Fourier filtering and smoothing**

On Study Direct you'll find a file called `dow.txt`. It contains the daily closing value for each business day from late 2006 until the end of 2010 of the Dow Jones Industrial Average, which is a measure of average prices of the largest companies of the US stock market.

Write a program to do the following:

(a) Read in the data from `dow.txt` and plot them on a graph.

(b) Calculate the coefficients of the discrete Fourier transform of the data using the function `rfft` from `numpy.fft`, which produces an array of $\frac{1}{2}N + 1$ complex numbers.

(c) Now set all but the first 10% of the elements of this array to zero (i.e., set the last 90% to zero but keep the values of the first 10%).

(d) Calculate the inverse Fourier transform of the resulting array, zeros and all, using the function `irfft`, and plot it on the same graph as the original data. You may need to vary the colors of the two curves to make sure they both show up on the graph. Comment on what you see. What is happening when you set the Fourier coefficients to zero?

(e) Modify your program so that it sets all but the first 2% of the coefficients to zero and run it again.