PH4505/PAP723 Homework 2

AY17/18 Semester 2

Instructions for PH4505 students: You are to choose 2 of the 3 questions to submit as your graded assignment. Every plot produced by your programs should have clear x and y axis labels. If more than one curve is shown on a single plot, the two curves should be labeled clearly. For full marks, code must follow good programming style. Your code should be commented; there should be no cryptic variable and function names (like abc); and the program structure should be modular (e.g. numerical constants should be defined in one place instead of being scattered throughout the code).

Instructions for PAP723 students: You are to submit all 3 questions as your graded assignment. Every plot produced by your programs should have clear x and y axis labels. If more than one curve is shown on a single plot, the two curves should be labeled clearly. For full marks, code must follow good programming style. Your code should be commented; there should be no cryptic variable and function names (like abc); and the program structure should be modular (e.g. numerical constants should be defined in one place instead of being scattered throughout the code).

1. **The Initial-Value Problem**

In this problem, you will implement a couple of solvers for the initial-value problem, and study their performance. The initial-value problem has the standard form that given

$$\frac{\mathrm{d}y}{\mathrm{d}t} = f(y,t) \tag{1}$$

and

$$y(t = t_0), \tag{2}$$

compute

$$y(t_n). \tag{3}$$

(a) Implement a solver based on the fourth-order Runge-Kutta (RK4) method:

$$y_{n+1} \approx y_n + \frac{h_n}{6}(k_1 + 2k_2 + 2k_3 + k_4) \tag{4}$$

$$h_n = t_{n+1} - t_n \tag{5}$$

$$k_1 = f(y_n, t_n) \tag{6}$$

$$k_2 = f(y_n + \frac{h_n}{2}k_1, t_n + \frac{h_n}{2}) \tag{7}$$

$$k_3 = f(y_n + \frac{h_n}{2}k_2, t_n + \frac{h_n}{2}) \tag{8}$$

$$k_4 = f(y_n + h_n k_3, t_n + h_n) \tag{9}$$

The function should have the form: `def ode_int_rk4(fun, y0, t, args=())`. `fun` is the derivative function such that `fun(y,t,...)` returns $f(y,t,\ldots)$ where y is the state, t is the time and $\ldots$ is any other arguments needed. `y0` is the initial state, `t` is an array of time points and `args` is any extra arguments if needed. The function should return an array `y` where the entry `y[n]` is $y(t_n)$.

(b) Now consider the damped harmonic oscillator,

$$\ddot{x} + 2\gamma\dot{x} + \omega_0^2 x = 0\ . \tag{10}$$

Let $t = 0$ be the initial time. Define the error at time T by

$$\varepsilon \equiv \left| \frac{x_{\mathrm{num}} - x(T)}{x(T)} \right| \tag{11}$$

where $x(T)$ is the exact analytic value for x at time T and x_{num} is the numerical result. We will fix a value of T, and study how ε varies with N, the total number of times the solver evaluates the derivative function $f(y, t, \ldots)$ in order to compute its result. (Note that N is not just the number of time steps; higher-order methods evaluate f more times per step.)

Write a function which generates a log-log plot of the numerical error ε generated by the RK4 method, versus N. Use the oscillator parameters $\omega_0 = 1$, $\gamma = 0.02$, fixed final time $T = 20$. Let N vary between around 10^1 to 10^3, by choosing an appropriate number of discretization steps (with T kept fixed).

(c) You will now benchmark Scipy's built-in ODE solver. Modify the function that you have written in 1b so that it also plots the numerical error ε versus the number of derivative function calls N using the ODE solver that you have chosen.

2. **Pendulum Motion**

In this problem, we will study the motion of a driven pendulum. Its equation of motion is

$$\ddot{\theta} + 2\gamma\dot{\theta} + \omega_0^2 \sin\theta = F_d \cos(2\pi t), \tag{12}$$

where θ is the angle between the pendulum and the vertical, γ is the damping constant, ω_0^2 is the pendulum constant, and F_d is the magnitude of the driving force. The drive frequency is normalized to 2π (so that the drive period is 1).

(a) We are not interested in the transient (small t) behavior of the pendulum, but rather its behavior at long times. Write a function to sample the motion at long times. You can use Scipy's built-in ODE solvers to solve Eq. (12).

(b) Write a function which plots $\dot{\theta}$ versus t for parameters $\gamma = 0.1$ and $\omega_0 = 10$. Show the results for five different values of the drive magnitude, $F_d \in [40, 60, 65, 70, 80]$, in separate subplots. You should observe a transition from regular oscillation to chaos. Choose a sampling interval $T = 20$, and choose T_1 so that the non-chaotic plots exhibit steady-state oscillation (i.e. no transients). The reason we plot $\dot{\theta}$, instead of θ is an angular variable; angular variables are annoying to plot and to Fourier transform, because they are only unique modulo 2π.

(c) Write a function to test the sensitivity of the numerical results to perturbations in the initial conditions. In this function, plot $|\Delta\dot{\theta}(T)|$ versus $F_d \in [50, 55]$, where $\Delta\dot{\theta}(T)$ denotes the deviation in the angular velocity at final time T, when the initial angle is increased slightly from $\theta(0) = 1$ to $\theta(0) = 1 + 10^{-8}$. Take $\dot{\theta}(0) = 0$, $T = 50$, and all other oscillator parameters to be the same as before. *What do you observe? Discuss in code comments.*

(d) Write a function which displays a scatter plot of $\theta|_{\dot{\theta}=0}$ versus F_d. Here, $\theta|_{\dot{\theta}=0}$ refers to the turning points of the pendulum motion: the values of *theta* whenever $\dot{\theta} = 0$. Choose a horizontal range $50 < F_d < 80$, with all other pendulum parameters the same as the previous parts. When calculating $\theta|_{\dot{\theta}=0}$, you should "warm up" the simulation by running it sufficiently many time steps before hunting for the turning points, in order to remove the effects of the transients. It is up to you to devise a suitable way to locate these turning points.

3. **Quantum Harmonic Oscillator**

The equation for the quantum harmonic oscillator can be written as:

$$-\frac{\hbar}{2m}\frac{\mathrm{d}^2\psi(x)}{\mathrm{d}x^2}+\frac{1}{2}m\omega^2x^2\psi(x)=E\psi(x)\ . \tag{13}$$

The energy of level n for the quantum harmonic oscillator can be analytically solved to give

$$E=(n+\frac{1}{2})\hbar\omega \tag{14}$$

Taking m, $\hbar$ and ω to be equal to 1,

(a) Implement a solver based on the Numerov algorithm.

(b) Write a function that solves Eq. (13) using the Numerov algorithm. The function should compute the normalized wavefunctions for the first 3 energy levels i.e. $n=1,2,3$ and plot them in a single plot. The y-axis should be Energy and you need to include a plot of the potential well due to the harmonic oscillator.

(c) Write another function that does the same thing as 3b but this time solve Eq. (13) using the Scipy's inbuilt ODE solvers.

(d) Analytical solutions for quantum mechanical equations are not always guaranteed to be solvable. In some cases, you have to make use of other methods to guess for the correct energies to the Schrödinger equation. Write a function for the quantum harmonic oscillator problem and assume that we do not know the energy of the ground state. Use the shooter's method to find the correct energy for the ground state wavefunction. Plot the normalized wavefunction and compare the energy with the analytical solution.