PHY407 Computational Lab 6
Solving ODEs, Part 1

Computational Background

- **RK4:** You've previously seen the Euler and Euler-Cromer methods for integrating systems of ODEs with initial values. A fourth-order method, which is probably the most widely used method for solving systems of ODEs, is the 4th order Runge-Kutta method (or RK4 for short).

  As discussed in the text, this method invokes calculating the RHS vector of $d\mathbf{r}/dt = \mathbf{f}(\mathbf{r}, t)$ at various intermediate points between steps. Full implementation requires coding the following 5 lines which are iterated over t values:

$$\begin{aligned}
\mathbf{k}_1 &= h\mathbf{f}(\mathbf{r}, t) && (1)\\
\mathbf{k}_2 &= hf(\mathbf{r} + \frac{1}{2}\mathbf{k}_1, t + \frac{1}{2}h) && (2)\\
\mathbf{k}_3 &= hf(\mathbf{r} + \frac{1}{2}\mathbf{k}_2, t + \frac{1}{2}h) && (3)\\
\mathbf{k}_4 &= hf(\mathbf{r} + \mathbf{k}_3, t + h) && (4)\\
\mathbf{r}(t+h) &= \mathbf{r}(t) + \frac{1}{6}(\mathbf{k}_1 + 2\mathbf{k}_2 + 2\mathbf{k}_3 + \mathbf{k}_4) && (5)
\end{aligned}$$

- **Varying the step size:** Choosing your step size (h in the above formulas) is always a balance between speed and accuracy. If your RHS vector $\mathbf{f}$ varies rapidly, it is usually very beneficial to alter the step size during the calculation to accommodate this.

  The basic philosophy in choosing a step size is that we want the error at each step to be similar. If the function is changing slowly, then in those regions we can take bigger steps. If the function is changing rapidly, then in those regions we should take smaller steps. A code that does this implements an "adaptive step size" method.

  In order to ensure we are keeping the error roughly constant at each step, we need to be able to estimate the error at each step. If the error is larger (smaller) than we want, then we decrease (increase) the step size accordingly. Section 8.4 in the text discusses how to implement an adaptive step size in RK4.

Physics Background

- **Vibrations in a one-dimensional system (for Question 2):** Exampe 6.2 starting on p.235 of Newman's text describes the dynamics of a system of masses coupled by springs. The masses oscillate longitudinally and the mass on one end is acted on by a force oscillating with frequency ω. In Example 6.2, the system is solved by looking for solutions that oscillate at the frequency ω (which means that the initial transients have died down in this system). In that example, the system is solved by Gaussian elimination taking advantage

of the banded structure of the matrix in the problem. In Problem 2 (Newman Exercise 8.9) you will solve the related problem of a set of driven coupled ODEs. To do this you will need to convert the second order ODEs into equivalent pairs of first order ODEs. The standard way to do this is to define a velocity type variable along the lines of the following: if the original equation is

$$m\frac{d^2\zeta_i}{dt^2} = k(\zeta_{i+1} - \zeta_i) + k(\zeta_{i-1} - \zeta_i) + F_i$$

at each i, the equivalent pair of first order equations is

$$\dot{\xi}_j = v_j, \dot{v}_j = \frac{k}{m}(\zeta_{i+1} - \zeta_i + \zeta_{i-1} - \zeta_i) + \frac{F_i}{m}.$$

- **Cometary Orbits (for Question 3):** When comets are very far from the sun, the gravitational force is weak and they therefore orbit much slower than when they are near the sun. This makes using adaptive step size routines a good idea for implementing a cometary orbital dynamics code.

Lab Instructions

For all codes mentioned, it is always recommended to write pseudocode first!

For grading purposes, only hand in the following parts. Ensure that your codes are well commented and readable by an outsider:

- Q1: Hand in the output you generate, your written answers, and the `.wav` file.
- Q2: Hand in your code that can generate the animation, the outputs, and written answers.
- Q3: Hand in your plots and the code that does the adaptive time stepping.

Lab Questions

1. **A low-pass filter - RK4 applied to a one variable system (30% of the lab):** Do Newman Exercise 8.1. Then extend the exercise as follows:

   In Lab05 Q1, you created a low pass filter for a sound signal using the Fourier Transform. Now, instead of filtering an electrical signal as in Exercise 8.1, you will process the signal through a differential equation to achieve the same objective.

   Adapt the code in Exercise 8.1 to filter the time series. Here, interpret V_{in} variable from the exercise as the original input signal, and V_{out} as the output signal. The constant $1/RC$ has units of frequency (Hz), and can be set to $1/RC = 880$ Hz to approximate the filtering in Lab05 Q1.

   Now carry out the same activity as in Lab05: plot the original and filtered time series for the sound file, plot the original and filtered spectra, and create an output `.wav` file.

The results won't be exactly the same as in Lab05, so briefly comment on similarities and differences in the results.

2. **RK4 for a multiple variable system (30% of the lab):** Do Newman Exercise 8.9. In Part c), you are asked to create an animation. The file `moving_sinewave.py` gives you Python code that creates a moving sine wave. You can adapt it to display your numerical solutions (note you may not want to refresh the figure at every single timestep if you choose a very small timestep).

3. **Using adaptive step sizes with RK4 (40% of the lab):** Newman Exercise 8.10 on pp. 361-363.