**PHY407 Computational Lab 7**
**Solving ODEs, Part 2**

# Computational Background

- **The shooting method for boundary value problems (and the secant method for finding roots) (for Q1):** The shooting method is an application of root finding for boundary value problems. A typical application is to look for correct values of parameters that lead to functions on the domain that match a given set of boundary conditions. You modify your guess until you get one that works. To do so, you typically have to solve a nonlinear equation for a root using methods such as the secant method.

  (We have mentioned the secant method in class, but have not used it yet. It is a version of Newton's method where you replace the derivative with its finite-difference approximation (compare Newman (6.96) to (6.104)). You use it instead of Newton's method when you do not have an analytic expression for the function you are trying to find the root of, or its derivative.)

  In Example 8.9, the shooting method is used with RK4 to find the ground state energy in a square well potential for the time independent Schrödinger equation. We'll talk about the physics of this problem in the Physics Background. But let's look over the code now and see if we can understand what it's doing: it is integrating a pair of ODEs (see the `solve(E)` function, which calls the function `f(r, x, E)`, which in turn calls the function `V(x)`). The solution to these equations depends on the parameter `E`. In the secant method loop, `E` is adjusted until a root is found for the variable `psi`, which the physics in the problem requires to be zero. Stated another way, the program does the following:

  - Initializes `E` with reasonable guesses.
  - Finds how `psi` depends on `E` using `solve(E)`.
  - Adjusts `E` using the secant method.
  - Repeats until `E` has converged to within a target accuracy.

  In the Physics background, we'll discuss why `psi` should be zero.

- **Energy-conserving methods (for Q2):** For conservative systems, certain time stepping methods are built to ensure energy conservation, or at least that energy is conserved for a complete cycle of motion due to the time-symmetry of the equations for the method. The text discusses four of these in Section 8.5:

  - The Leapfrog method
  - The Verlet method
  - The modified midpoint method
  - The Bulirsch-Stoer method

  In Q2a we will focus on the **Verlet** method for a conservative system. The algorithm is described in the text on pages 371-374. Starting from $a = F/m$, or equivalently

$$\frac{d^2\vec{r}}{dt^2} = \vec{f}(\vec{r}, t), \tag{1}$$

  and denoting the velocity as $\vec{v}$, here is the Verlet algorithm:

  For the very first step only:

$$\vec{v}(t + \frac{1}{2}h) = \vec{v}(t) + \frac{1}{2}h\vec{f}(\vec{r}(t), t) \tag{2}$$

  Then repeatedly apply the equations:

$$\begin{aligned} \vec{r}(t + h) &= \vec{r}(t) + h\vec{v}(t + \frac{1}{2}h) && (3)\\ \vec{k} &= h\vec{f}(\vec{r}(t + h), t + h) && (4)\\ \vec{v}(t + h) &= \vec{v}(t + \frac{1}{2}h) + \frac{1}{2}\vec{k} && (5)\\ \vec{v}(t + \frac{3}{2}h) &= \vec{v}(t + \frac{1}{2}h) + \vec{k} && (6) \end{aligned}$$

  This last equation for $\vec{v}(t + \frac{3}{2}h)$ becomes $\vec{v}(t + \frac{1}{2}h)$ for the next iteration of Equation (3). Note that Equation (5) is for the purposes of diagnosing energy but is not required for the algortithm to advance to the next time step.

In Q2b, we will focus on the **Bulirsh-Stoer** method (8.5.5- 8.5.6 of the text). It is a combination of the modified midpoint method and Richardson extrapolation. Although it is somewhat more complicated to program than the Runge-Kutta method, it can work significantly better even than the adaptive version of Runge-Kutta, giving more accurate solutions with significantly less work as long as your solutions are relatively smooth. Lectures from week 7 will provide the details on how this method works and the provided program "bulirsch.py" (Example 8.7) gives a demonstration of its use for the nonlinear pendulum.

# Physics Background

- **The hydrogen atom (for Q1):** The time independent Schrödinger equation is

$$-\frac{\hbar^2}{2m}\nabla^2\psi(\boldsymbol{x}) + V(\boldsymbol{x})\psi(\boldsymbol{x}) = E\psi(\boldsymbol{x}) \tag{7}$$

Where $\boldsymbol{x}$ is the spatial coordinate. For a central potential $V = V(r)$, we can assume the wavefunction $\psi$ can be separated according to $\psi(\boldsymbol{x}) = R(r)Y_l^m(\theta, \phi)$, where we are using standard notation with $Y_l^m$ being the spherical harmonic of degree $m$ and order $l$. In this case $R(r)$ satisfies a second order ODE

$$\frac{d}{dr}\left(r^2\frac{dR}{dr}\right) - \frac{2mr^2}{\hbar^2}\left[V(r) - E\right]R = l(l+1)R \tag{8}$$

and is subject to certain boundary conditions. For the particular case of a hydrogen atom with an electron interacting electrostatically with a proton, we can write

$$V(r) = -e^2/(4\pi\epsilon_0 r), \tag{9}$$

and in that case the solutions are well known. An online reference for the solution can be found below:

`http://hyperphysics.phy-astr.gsu.edu/hbase/quantum/hydwf.html`
`http://hyperphysics.phy-astr.gsu.edu/hbase/hyde.html#c2`

The energy levels are

$$E_n = -E_0/n^2,$$

where $E_0 \approx 13.6$ eV is the ionization energy of hydrogen and $n = 0, 1, ....$ The boundary condition is that $R(r) \to 0$ as $r \to \infty$. The ground state energy with $n = 1$ is about -13.6eV, the first excited state energy for $n = 2$ is at -3.4 eV, etc.. The wave functions corresponding to $n = 1, 2, 3$ and various values of $l$ can be found at the links above. This problem is well known and solved, and so provides a good test case for our home-built shooting method. In Q1 we will focus only on the radial part $R(r)$ but if you want you can also create plots of the three dimensional wave function of the different energy levels of the hydrogen atom.

# 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 your code, plots, written answers.
- Q2: Hand in plots and written answers only.
- Q3: Hand in written answers only.

# Lab Questions

1. **The hydrogen atom (40% of the lab):** In this exercise you will use shooting and RK4 to find the bound states of hydrogen for a couple of cases. The calculation will involve finding both the energy eigenvalues and the related eigenfunctions for the radial part of the Schrödinger equation. In your calculation, you will need to set initial conditions on $\psi$ and $\phi$. Because the RHS of (8) diverges at $r = 0$, you should carry out your integration starting at $r = h$, where $h$ is the step size, and set $\psi(h) = 0$ and $\phi(h)$ to a finite constant (1 is fine). This will cause the eigenfunction to go to zero at $r = 0$, which, it turns out, is not a good assumption for one of the cases below, but will not affect the energy eigenvalue calculation too much.

   Now do the following:

(a) You can answer this question by handing in your commented code.

Starting from (8), and with reference to Section 8.6.3, Example 8.9, and the code `squarewell.py`, write out the second order ODE in $r$ as a pair of coupled first order ODEs, and implement this in Python. Some notes:

- In the secant method loop in Example 8.9, the energy `E` is adjusted until the value `psi` is as close to zero as possible because `psi` represents the wavefunction at the right-hand boundary of the infinite square well. In the hydrogen atom there is no right wall; instead, your domain extends to "infinity", but you can get reasonable answers for a large value of $r$, and you need to set the wavefunction $R$ to zero there. This is similar to the issues raised in Exercise 8.14 in Newman.
- There are a few adjustable parameters in this exercise: one is the stepsize $h$ and the other is the maximum value of $r$, which is $r_\infty$. To start with, set $h = 0.002a$ and $r_\infty = 20a$, where $a \approx 5 \times 10^{-11}\,\mathrm{m}$ is the Bohr radius. You can make $r_\infty$ larger and make $h$ smaller to improve the solution. You will need to find out which one makes more of a difference in different situations.
- You will need to set a left boundary condition on $R$ near $r = 0$. For convenience, set $R(h) = 0$, even though this is not required by the mathematics and indeed contradicts the solution for n=1, l=0. This is a source of inaccuracy in the code.
- You will also need to set a target energy convergence. This is `e/1000` in `squarewell.py` but you should look at the impact of reducing this (it's not a major effect).
- Finally, you will need to initialize your eigen energies at some level for the secant method to work. It is easy to miss the energies if you aren't careful. I found that it was a good idea to bracket the energies so for different `n` I chose initial values of `E_1=-15e/n**2` and `E_2=-13e/n**2`.

(b) Calculate numerically the ground state energy ($n = 1$) and the first excited state energy ($n = 2$) for $l = 0$, and the ground state

energy for $l = 1$ (which is the same as the energy for $n = 2$ and $l = 0$. These can all be compared to the known solution for the hydrogen atom. Look at the effect of adjusting the various parameters discussed above.

(c) For the three cases in Part b), modify your program to plot the normalized eigenfunction $R$. Normalize by calculating $\int |R(r)|^2 dr$ over the range of $r$ and plot it as a function of $r$ over a finite range (this is similar to Exercise 8.14c). You can use either the trapezoidal or Simpson's rule to do the integration.

For some eigenvalues, you might obtain spurious large values of $R$ at the right-hand end ($r_\infty$) of the wave function. By definition, the wave function should go to 0 at the boundary. It does not because we don't have the energy EXACTLY right and it's extremely sensitive to the energy value. If the energy is off by even the tiniest bit then the value tends to diverge. This is not a problem for determining the energy itself. But it can give problems when normalizing and potting the wave function. To deal with this, you can restrict the vertical axis range (using ylim) to cut off the large spurious values at the end of the array.

(d) At the links above you can find explicit solutions for $R(r)$. Scale these solutions so they can be plotted as overlays on top of your numerically calculated solutions. How do the two solutions compare in terms of overall shape and zero crossings for the wave functions?

2. **Planetary dynamics (40% of the lab):**

   (a) **Using the Verlet method:** Do Newman Exercise 8.12.

   (b) **Using the Bulirsch-Stoer method:** Do Newman Exercise 8.13.

3. **Stability and instability (20% of the lab):** In Lecture 7 notes and the handout of Prof. Stanley's slides, we considered the concepts of stability of a function and stability of a numerical method. We showed that Euler's method:

$$y_{k+1} = y_k + h_k f(y_k) \tag{10}$$

for a specific test ODE:

$$\frac{dy}{dt} = \lambda y \tag{11}$$

had a growth factor $(1 + h_k\lambda)$. This meant that in order for Euler's method to be stable for this ODE, $|1 + h_k\lambda| \leq 1$. This ruled out any positive $\lambda$ and limited the step size $h_k$ to be within the range: $h_k \geq -2.0/\lambda$ to ensure stability. By looking at the local error we determined that Euler's method is 1st order accurate. Here you will do a similar analysis for 2 other methods for this same 1D test ODE:

(a) The backward Euler method: This method is very similar to Euler's method, the only difference is that you do the function evaluation at $y_{k+1}$ instead of at $y_k$. The formula looks like:

$$y_{k+1} = y_k + h_k f(y_{k+1}) \tag{12}$$

Notice that in order to determine $y_{k+1}$ you need to know $f(y_{k+1})$ for which you need to know $y_{k+1}$! This type of method is called an "implicit" method and in order to solve it you have to implement one of the nonlinear equation techniques you learned in Chapter 6. For now though, lets determine the stability and accuracy properties of the background Euler method. To do so, follow the same techniques as in the notes for the Euler method. (see hints below).

(b) The implicit Trapezoid method: This method combines the Euler method and the backward Euler method:

$$y_{k+1} = y_k + \frac{h_k}{2}[f(y_k) + f(y_{k+1})] \tag{13}$$

Hints: Here are the steps you need to follow:

- Plug in your test ODE's $f(y)$ into your method and take a single step of size $h_k$. Then rearrange your method in the form:

$$y_{k+1} = [???]y_k \tag{14}$$

The [???] is the growth factor which will depend on both $\lambda$ from your ODE and $h_k$ from your method.

- Restrict the absolute magnitude of your growth factor to be less than or equal to 1 and use this to determine conditions on $\lambda$ and $h_k$. This will determine your stability conditions.
- If you get a complicated growth factor, you may not be able to get nice expressions for stability conditions. In this case, you can plot the region of stability of $\mu = h_k\lambda$ in the complex plane and hand in a visualization of this region.
- To determine the accuracy: subtract your method expression for $y_{k+1}$ from the Taylor series expansion of the analytic solution of y after one step (i.e. $y(h_k)$). What you have left is your local error.
- From the local error determine the order of accuracy of your method.