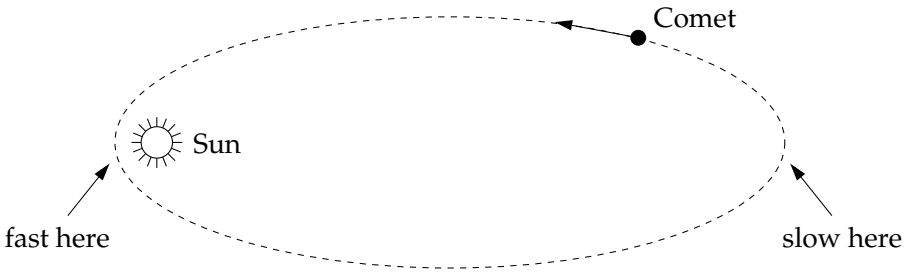
PHYS 7411 Spring 2015
Computational Physics
Homework 5

Due by 3:00pm in Nicholson 447 on 13 April 2015

Exercise 1: Cometary orbits

Many comets travel in highly elongated orbits around the Sun. For much of their lives they are far out in the solar system, moving very slowly, but on rare occasions their orbit brings them close to the Sun for a fly-by and for a brief period of time they move very fast indeed:



This is a classic example of a system for which an adaptive step size method is useful, because for the large periods of time when the comet is moving slowly we can use long time-steps, so that the program runs quickly, but short time-steps are crucial in the brief but fast-moving period close to the Sun.

The differential equation obeyed by a comet is straightforward to derive. The force between the Sun, with mass M at the origin, and a comet of mass m with position vector $\mathbf{r}$ is GMm/r^2 in direction $-\mathbf{r}/r$ (i.e., the direction towards the Sun), and hence Newton's second law tells us that

$$m\frac{\mathrm{d}^2\mathbf{r}}{\mathrm{d}t^2} = -\left(\frac{GMm}{r^2}\right)\frac{\mathbf{r}}{r}.$$

Canceling the m and taking the x component we have

$$\frac{\mathrm{d}^2x}{\mathrm{d}t^2} = -GM\frac{x}{r^3}\,,$$

and similarly for the other two coordinates. We can, however, throw out one of the coordinates because the comet stays in a single plane as it orbits. If we orient our axes so that this plane is perpendicular to the z-axis, we can forget about the z coordinate and we are left with just two second-order equations to solve:

$$\frac{\mathrm{d}^2x}{\mathrm{d}t^2} = -GM\frac{x}{r^3}\,, \qquad \frac{\mathrm{d}^2y}{\mathrm{d}t^2} = -GM\frac{y}{r^3}\,,$$

where $r = \sqrt{x^2+y^2}$.

a) Turn these two second-order equations into four first-order equations, using the methods you have learned.

b) Write a program to solve your equations using the fourth-order Runge–Kutta method with a *fixed* step size. You will need to look up the mass of the Sun and Newton's gravitational constant G. As an initial condition, take a comet at coordinates $x = 4$ billion kilometers and $y = 0$ (which is somewhere out around the orbit of Neptune) with initial velocity $v_x = 0$ and $v_y = 500\,\mathrm{m\,s^{-1}}$. Make a graph showing the trajectory of the comet (i.e., a plot of y against x).

Choose a fixed step size h that allows you to accurately calculate at least two full orbits of the comet. Since orbits are periodic, a good indicator of an accurate calculation is that successive orbits of the comet lie on top of one another on your plot. If they do not then you need a smaller value of h. Give a short description of your findings. What value of h did you use? What did you observe in your simulation? How long did the calculation take?

c) Make a copy of your program and modify the copy to do the calculation using an adaptive step size. Set a target accuracy of $\delta = 1$ kilometer per year in the position of the comet and again plot the trajectory. What do you see? How do the speed, accuracy, and step size of the calculation compare with those in part (b)?

d) Modify your program to place dots on your graph showing the position of the comet at each Runge–Kutta step around a single orbit. You should see the steps getting closer together when the comet is close to the Sun and further apart when it is far out in the solar system.

Calculations like this can be extended to cases where we have more than one orbiting body—see Exercise 8.16 for an example. We can include planets, moons, asteroids, and others. Analytic calculations are impossible for such complex systems, but with careful numerical solution of differential equations we can calculate the motions of objects throughout the entire solar system.

Exercise 2: The three-body problem

If you mastered Exercise 8.10 on cometary orbits, here's a more challenging problem in celestial mechanics—and a classic in the field—the *three-body problem*.

Three stars, in otherwise empty space, are initially at rest, with the following masses and positions, in arbitrary units:

	Mass	x	y
Star 1	150	3	1
Star 2	200	−1	−2
Star 3	250	−1	1

(All the z coordinates are zero, so the three stars lie in the xy plane.)

a) Show that the equation of motion governing the position $\mathbf{r}_1$ of the first star is

$$\frac{\mathrm{d}^2\mathbf{r}_1}{\mathrm{d}t^2} = Gm_2\frac{\mathbf{r}_2 - \mathbf{r}_1}{|\mathbf{r}_2 - \mathbf{r}_1|^3} + Gm_3\frac{\mathbf{r}_3 - \mathbf{r}_1}{|\mathbf{r}_3 - \mathbf{r}_1|^3}$$

and derive two similar equations for the positions $\mathbf{r}_2$ and $\mathbf{r}_3$ of the other two stars. Then convert the three second-order equations into six equivalent first-order equations, using the techniques you have learned.

b) Working in units where $G = 1$, write a program to solve your equations and hence calculate the motion of the stars from $t = 0$ to $t = 2$. Make a plot showing the trails of all three stars (i.e., a graph of y against x for each star).

c) Modify your program to make an animation of the motion on the screen from $t = 0$ to $t = 10$. You may wish to make the three stars different sizes or colors (or both) so that you can tell which is which.

To do this calculation properly you will need to use an adaptive step size method, for the same reasons as in Exercise 8.10—the stars move very rapidly when they are close together and very slowly when they are far apart. An adaptive method is the only way to get the accuracy you need in the fast-moving parts of the motion without wasting hours uselessly calculating the slow parts with a tiny step size. Construct your program so that it introduces an error of no more than 10^{-3} in the position of any star per unit time.

Creating an animation with an adaptive step size can be challenging, since the steps do not all correspond to the same amount of real time. The simplest thing to do is just to ignore the varying step sizes and make an animation as if they were all equal, updating the positions of the stars on the screen at every step or every several steps. This will give you a reasonable visualization of the motion, but it will look a little odd because the stars will slow down, rather than speed up, as they come close together, because the adaptive calculation will automatically take more steps in this region.

A better solution is to vary the frame-rate of your animation so that the frames run proportionally faster when h is smaller, meaning that the frame-rate needs to be equal to C/h for some constant C. You can achieve this by using the `rate` function from the `visual` package to set a different frame-rate on each step, equal to C/h. If you do this, it's a good idea to not let the value of h grow too large, or the animation will make some large jumps that look uneven on the screen. Insert extra program lines to ensure that h never exceeds a value h_{max} that you choose. Values for the constants of around $C = 0.1$ and $h_{\text{max}} = 10^{-3}$ seem to give reasonable results.

Exercise 3: The Schrödinger equation and the Crank–Nicolson method

Perhaps the most important partial differential equation, at least for physicists, is the Schrödinger equation. This exercise uses the Crank–Nicolson method to solve the full time-dependent Schrödinger equation and hence develop a picture of how a wavefunction evolves over time. The following exercise, Exercise 9.9, solves the same problem again, but using the spectral method.

We will look at the Schrödinger equation in one dimension. The techniques for calculating solutions in two or three dimensions are basically the same as for one dimension, but the calculations take much longer on the computer, so in the interests of speed we'll stick with one dimension. In one dimension the Schrödinger equation for a particle of mass M with no potential energy reads

$$-\frac{\hbar^2}{2M}\frac{\partial^2\psi}{\partial x^2} = \mathrm{i}\hbar\frac{\partial\psi}{\partial t}.$$

For simplicity, let's put our particle in a box with impenetrable walls, so that we only have to solve the equation in a finite-sized space. The box forces the wavefunction ψ to be zero at the walls, which we'll put at $x = 0$ and $x = L$.

Replacing the second derivative in the Schrödinger equation with a finite difference and applying Euler's method, we get the FTCS equation

$$\psi(x,t+h) = \psi(x,t) + h\frac{\mathrm{i}\hbar}{2ma^2}\bigl[\psi(x+a,t) + \psi(x-a,t) - 2\psi(x,t)\bigr],$$

where a is the spacing of the spatial grid points and h is the size of the time-step. (Be careful not to confuse the time-step h with Planck's constant $\hbar$.) Performing a similar step in reverse, we get the implicit equation

$$\psi(x,t+h) - h\frac{\mathrm{i}\hbar}{2ma^2}\bigl[\psi(x+a,t+h) + \psi(x-a,t+h) - 2\psi(x,t+h)\bigr] = \psi(x,t).$$

And taking the average of these two, we get the Crank–Nicolson equation for the Schrödinger equation:

$$\psi(x,t+h) - h\frac{\mathrm{i}\hbar}{4ma^2}\bigl[\psi(x+a,t+h) + \psi(x-a,t+h) - 2\psi(x,t+h)\bigr] \\ = \psi(x,t) + h\frac{\mathrm{i}\hbar}{4ma^2}\bigl[\psi(x+a,t) + \psi(x-a,t) - 2\psi(x,t)\bigr].$$

This gives us a set of simultaneous equations, one for each grid point.

The boundary conditions on our problem tell us that $\psi = 0$ at $x = 0$ and $x = L$ for all t. In between these points we have grid points at a, $2a$, $3a$, and so forth. Let us arrange the values of ψ at these interior points into a vector

$$\boldsymbol{\psi}(t) = \begin{pmatrix} \psi(a,t) \\ \psi(2a,t) \\ \psi(3a,t) \\ \vdots \end{pmatrix}.$$

Then the Crank–Nicolson equations can be written in the form

$$\mathbf{A}\boldsymbol{\psi}(t+h) = \mathbf{B}\boldsymbol{\psi}(t),$$

where the matrices $\mathbf{A}$ and $\mathbf{B}$ are both symmetric and tridiagonal:

$$\mathbf{A} = \begin{pmatrix} a_1 & a_2 & & & \\ a_2 & a_1 & a_2 & & \\ & a_2 & a_1 & a_2 & \\ & & a_2 & a_1 & \\ & & & & \ddots \end{pmatrix}, \qquad \mathbf{B} = \begin{pmatrix} b_1 & b_2 & & & \\ b_2 & b_1 & b_2 & & \\ & b_2 & b_1 & b_2 & \\ & & b_2 & b_1 & \\ & & & & \ddots \end{pmatrix},$$

with

$$a_1 = 1 + h\frac{\mathrm{i}\hbar}{2ma^2}, \qquad a_2 = -h\frac{\mathrm{i}\hbar}{4ma^2}, \qquad b_1 = 1 - h\frac{\mathrm{i}\hbar}{2ma^2}, \qquad b_2 = h\frac{\mathrm{i}\hbar}{4ma^2}.$$

(Note the different signs and the factors of 2 and 4 in the denominators.)

The equation $\mathbf{A}\boldsymbol{\psi}(t+h) = \mathbf{B}\boldsymbol{\psi}(t)$ has precisely the form $\mathbf{A}\mathbf{x} = \mathbf{v}$ of the simultaneous equation problems we studied in Chapter 6 and can be solved using the same methods. Specifically, since the matrix $\mathbf{A}$ is tridiagonal in this case, we can use the fast tridiagonal version of Gaussian elimination that we looked at in Section 6.1.6.

Consider an electron (mass $M = 9.109 \times 10^{-31}\,\mathrm{kg}$) in a box of length $L = 10^{-8}\,\mathrm{m}$. Suppose that at time $t = 0$ the wavefunction of the electron has the form

$$\psi(x,0) = \exp\left[-\frac{(x-x_0)^2}{2\sigma^2}\right]\mathrm{e}^{\mathrm{i}\kappa x},$$

where

$$x_0 = \frac{L}{2}, \qquad \sigma = 1 \times 10^{-10}\,\mathrm{m}, \qquad \kappa = 5 \times 10^{10}\,\mathrm{m}^{-1},$$

and $\psi = 0$ on the walls at $x = 0$ and $x = L$. (This expression for $\psi(x,0)$ is not normalized—there should really be an overall multiplying coefficient to make sure that the probability density for the electron integrates to unity. It's safe to drop the constant, however, because the Schrödinger equation is linear, so the constant cancels out on both sides of the equation and plays no part in the solution.)

a) Write a program to perform a single step of the Crank–Nicolson method for this electron, calculating the vector $\boldsymbol{\psi}(t)$ of values of the wavefunction, given the initial wavefunction above and using $N = 1000$ spatial slices with $a = L/N$. Your program will have to perform the following steps. First, given the vector $\boldsymbol{\psi}(0)$ at $t = 0$, you will have to multiply by the matrix $\mathbf{B}$ to get a vector $\mathbf{v} = \mathbf{B}\boldsymbol{\psi}$. Because of the tridiagonal form of $\mathbf{B}$, this is fairly simple. The ith component of $\mathbf{v}$ is given by

$$v_i = b_1\psi_i + b_2(\psi_{i+1} + \psi_{i-1}).$$

You will also have to choose a value for the time-step h. A reasonable choice is $h = 10^{-18}\,\mathrm{s}$.

Second you will have to solve the linear system $\mathbf{A}\mathbf{x} = \mathbf{v}$ for $\mathbf{x}$, which gives you the new value of $\boldsymbol{\psi}$. You could do this using a standard linear equation solver like the function `solve` in `numpy.linalg`, but since the matrix $\mathbf{A}$ is tridiagonal a better approach would be to use the fast solver for banded matrices given in Appendix E, which can be imported from the file `banded.py` (which you can find in the on-line resources). Note that although the wavefunction of a particle in principle has a complex value, in this case the wavefunction is always real—all the coefficients in the equations above are real numbers so if, as here, the wavefunction starts off real, then it remains real. Thus you do not need to use a complex array to represent the vector $\boldsymbol{\psi}$. A real one will do the job.

Third, once you have the code in place to perform a single step of the calculation, extend your program to perform repeated steps and hence solve for ψ at a sequence of times a separation h apart. Note that the matrix $\mathbf{A}$ is independent of time, so it doesn't change from one step to another. You can set up the matrix just once and then keep on reusing it for every step.

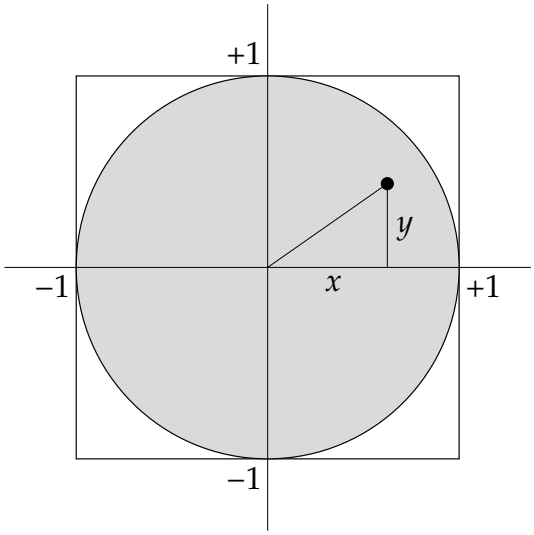
b) Extend your program to make an animation of the solution by displaying the real part of the wavefunction at each time-step. You can use the function `rate` from the package `visual` to ensure a smooth frame-rate for your animation—see Section 3.5 on page 117.

There are various ways you could do the animation. A simple one would be to just place a small sphere at each grid point with vertical position representing the value of the real part of the wavefunction. A more sophisticated approach would be to use the `curve` object from the `visual` package—see the on-line documentation at `www.vpython.org` for details. Depending on what coordinates you use for measuring x, you may need to scale the values of the wavefunction by an additional constant to make them a reasonable size on the screen. (If you measure your x position in meters then a scale factor of about 10^{-9} works well for the wavefunction.)

c) Run your animation for a while and describe what you see. Write a few sentences explaining in physics terms what is going on in the system.

Exercise 4: Volume of a hypersphere

This exercise asks you to estimate the volume of a sphere of unit radius in ten dimensions using a Monte Carlo method. Consider the equivalent problem in two dimensions, the area of a circle of unit radius:



The area of the circle, the shaded area above, is given by the integral

$$I = \int\int_{-1}^{+1} f(x,y)\,\mathrm{d}x\,\mathrm{d}y,$$

where $f(x,y) = 1$ everywhere inside the circle and zero everywhere outside. In other words,

$$f(x,y) = \begin{cases} 1 & \text{if } x^2+y^2 \le 1, \\ 0 & \text{otherwise.} \end{cases}$$

So if we didn't already know the area of the circle, we could calculate it by Monte Carlo integration. We would generate a set of N random points (x,y), where both x and y are in the range from -1 to 1. Then the two-dimensional version of Eq. (10.33) for this calculation would be

$$I \simeq \frac{4}{N}\sum_{i=1}^{N} f(x_i,y_i).$$

Generalize this method to the ten-dimensional case and write a program to perform a Monte Carlo calculation of the volume of a sphere of unit radius in ten dimensions.

If we had to do a ten-dimensional integral the traditional way, it would take a very long time. Even with only 100 points along each axis (which wouldn't give a very accurate result) we'd still have $100^{10} = 10^{20}$ points to sample, which is impossible on any computer. But using the Monte Carlo method we can get a pretty good result with a million points or so.