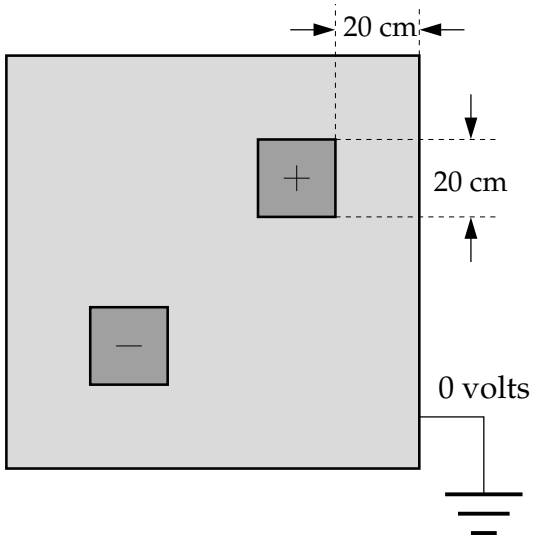
Physics 411: Homework 8

Note that you should do only one of questions 3 and 4, not both.

1. **Poisson's equation for the electrostatic potential:** Poisson's equation governs the electrostatic potential in the presence of a charge density ρ:

$$\nabla^2\phi = -\frac{\rho}{\epsilon_0}. \tag{1}$$

Here ϵ_0 is the permittivity of empty space (and we are assuming that we are in empty space). For simplicity, consider this equation in two dimensions, in a square box 1 meter along each side. All the walls of the box are at voltage zero, but there are two square charges in the box, one positive, one negative, thus:



The two charges are each 20 cm on a side and 20 cm from the walls of the box and have uniform charge density $\pm 1\,\mathrm{Cm}^{-2}$.

Write a Python program to solve for the potential ϕ on a grid of points within the box using the relaxation method and make a density plot of the result. Work in units where $\epsilon_0 = 1$ and continue the iteration until your solution for the electric potential changes by less than 10^{-6} V per step at every grid point.

☑ **For full credit** turn in a printout of your program and the plot it produces.

2. **Thermal diffusion in the Earth's crust:** A classic example of a diffusion problem with a time-varying boundary condition is the diffusion of heat into the crust of the Earth, as surface temperature varies with the seasons. Suppose the mean daily temperature at a particular point on the surface varies as:

$$T_0(t) = A + B\sin\frac{2\pi t}{\tau},$$

where $\tau = 365$ days, $A = 10°$C and $B = 12°$C. At a depth of 20 m below the surface almost all annual temperature variation is ironed out and the temperature is, to a good

approximation, a constant 11°C (which is higher than the mean surface temperature of 10°C—temperature increases with depth, due to heating from the hot core of the planet). The thermal diffusivity of the Earth's crust varies somewhat from place to place, but for our purposes we will treat it as constant with value $D = 0.1\,\mathrm{m}^2\,\mathrm{day}^{-1}$.

Write a program to calculate the temperature profile of the crust as a function of depth up to 20 m and time up to 10 years. Start with temperature everywhere equal to 10°C, except at the surface and the deepest point, choose values for the number of grid points and the time-step h, then run your program for the first nine simulated years, to allow it to settle down into whatever pattern it reaches. Then for the tenth and final year plot four temperature profiles taken at 3-month intervals on a single graph to illustrate how the temperature changes as a function of depth and time.

☑ **For full credit** turn in a printout of your program and the plot it produces.

3. **The Schrödinger equation and the Crank–Nicolson method:**

Do **only one of Questions 3 and 4**. The choice of which to do is up to you. They are both worth the same number of points.

Perhaps the most important partial differential equation, at least for physicists, is the Schrödinger equation. In this problem you will use the Crank–Nicolson method to solve the full time-dependent Schrödinger equation and hence develop a picture of how a wavefunction evolves over time.

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 $x = L$, so that the box has size 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 form 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 and in between these points we have grid points at a, $2a$, $3a$, and so forth. Let us arrange the values of ψ at these 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 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 earlier in the semester 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 $L = 10^{-8}\,\mathrm{m}$. At time $t = 0$ the wavefunction of the electron takes the form of a Gaussian wave packet thus:

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

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 $x = L$.

(a) Write a program to apply the Crank–Nicolson method to solve the Schrödinger equation for this electron and calculate 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

$$v_i = b_1\psi_i + b_2(\psi_{i+1} + \psi_{i-1}). \tag{2}$$

You will also need a value for the time-step h. A reasonable choice is $h = 10^{-18}$ 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 is to use the fast solver for banded matrices given in Appendix E of the book, which can be imported from the file `banded.py` (which you can find in the on-line resources).

Third, once you have the code in place to solve the equations, extend your program to perform the same operations repeatedly, and hence solve for ψ at a sequence of time-steps with separation h. 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 wavefunction 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.

There are various ways you could do the animation. A simple one would be to place a small sphere at each grid point whose vertical position represents the value of the real part of the wavefunction. A more sophisticated approach would be to use the `curve` object in the `visual` package—see the on-line documentation at `www.vpython.org` for details. A convenient feature of the curve object is that you can specify its set of x positions and y positions separately as arrays. In this exercise the x positions only need to specified once, since they never change, while the y positions will need to be specified anew each time you take a time-step.

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.

☑ **For full credit** turn in a printout of your final program and a snapshot of the animation it produces, along with your explanation of what you discovered upon running the program.

4. **The Schrödinger equation and the spectral method:** Do either this problem or the previous one, but not both—the choice of which one to do is yours.

   In this problem you'll write a program to solve the time-dependent Schrödinger equation

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

   using the spectral method for the same system as in problem 3, a single particle in one dimension in a box of length L with impenetrable walls. You may find it useful to read through problem 3, since it contains some discussion of the physics that is not repeated here.

   The wavefunction necessarily goes to zero on the walls of the box and hence one possible solution of the equation is

$$\psi_k(x,t) = \sin\left(\frac{\pi k x}{L}\right) \mathrm{e}^{\mathrm{i}Et/\hbar},$$

   where the energy E can be found by substituting into the Schrödinger equation, giving

$$E = \frac{\pi^2\hbar^2k^2}{2ML^2}.$$

   As with the vibrating string of Section 9.3.4 in the book, we can write a full solution as a linear combination of such individual solutions, which on the grid points $x_n = nL/N$ takes the value

$$\psi(x_n,t) = \frac{1}{N}\sum_{k=1}^{N-1} b_k \sin\left(\frac{\pi k n}{N}\right) \exp\left(\mathrm{i}\frac{\pi^2\hbar k^2}{2ML^2}t\right),$$

   where the b_k are some set of (possibly complex) coefficients that specify the exact shape of the wavefunction and the leading factor of $1/N$ is optional but convenient. (It makes the definition compatible with the standard discrete sine transform.)

   Since the Schrödinger equation (unlike the wave equation) is first order in time, we need only a single initial condition on the value of $\psi(x,t)$ to specify the coefficients b_k although, since the coefficients are in general complex, we will need to calculate both real and imaginary parts of each coefficient.

   As in problem 3 we consider an electron (mass $M = 9.109 \times 10^{-31}$ kg) in a box of length $L = 10^{-8}$ m. 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

$$x_0 = \frac{L}{2}, \qquad \sigma = 1\times10^{-10}\,\mathrm{m}, \qquad \kappa = 5\times10^{10}\,\mathrm{m}^{-1},$$

   and $\psi = 0$ on the walls at $x = 0$ and $x = L$.

(a) Write a program to calculate the values of the coefficients b_k, which for convenience can be broken down into their real and imaginary parts as $b_k = \alpha_k + i\eta_k$. Divide the box into $N = 1000$ slices and create two arrays containing the real and imaginary parts of $\psi(x_n, 0)$ at each grid point. Perform discrete sine transforms on each array separately and hence calculate the values of the α_k and η_k for all $k = 1 \ldots N-1$.

To perform the discrete sine transforms, you can use the fast transform function `dst` from the package `dcst`, which you can find in the on-line resources in the file named `dcst.py`. The function takes an array of N real numbers and returns the discrete sine transform as another array of N numbers.[1]

(b) Putting $b_k = \alpha_k + i\eta_k$ in the solution above and taking the real part we get

$$\operatorname{Re}\psi(x_n, t) = \frac{1}{N}\sum_{k=1}^{N-1}\left[\alpha_k \cos\left(\frac{\pi^2\hbar k^2}{2ML^2}t\right) - \eta_k \sin\left(\frac{\pi^2\hbar k^2}{2ML^2}t\right)\right]\sin\left(\frac{\pi k n}{N}\right).$$

This is an inverse sine transform with coefficients equal to the quantities in the square brackets. Extend your program to calculate the real part of the wavefunction $\psi(x, t)$ at an arbitrary time t using this formula with the inverse discrete sine transform function `idst`, also from the package `dcst`. Test your program by making a graph of the wavefunction at time $t = 10^{-16}$ s.

(c) Extend your program further to make an animation of the wavefunction over time, similar to that described in part (b) of problem 3 above. A suitable time interval for each frame of the animation is about 10^{-18} s.

(d) 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.

☑ **For full credit** turn in a printout of your final program and a snapshot of the animation it produces, along with your explanation of what you discovered upon running the program.

[1] Note that the first element of the input array should in principle always be zero for a sine transform, but if it is not the `dst` function will simply pretend that it is. Similarly the first element of the returned array is always zero, since the $k = 0$ coefficient of a sine transform is always zero. So in effect, the sine transform really only takes $N - 1$ real numbers and transforms them into another $N - 1$ real numbers.