

PHYS 580

FALL 2010

due **Midnight**, Monday, Dec 13, 2010

PROJECT 6

Time-dependent Schrodinger Equation in a Potential

In this project you will solve the *time-dependent* Schrodinger equation

$$i\hbar \frac{\partial}{\partial t} \Psi(x,t) = -\frac{\hbar^2}{2m} \frac{\partial^2}{\partial x^2} \Psi(x,t) + V(x)\Psi(x,t) \text{ using the Crank-Nicolson method.}$$

We start by introducing the Hamiltonian operator $\hat{H} = -\frac{\hbar^2}{2m} \frac{\partial^2}{\partial x^2} + V(x)$, which is the same we had in Project 4. Then we rewrite the Schrodinger equation

$$\frac{\partial}{\partial t} \Psi(x,t) = -\frac{i}{\hbar} \hat{H} \Psi(x,t). \text{ The next step is to discretize } \Psi(x,t) \text{ as a vector: } \Psi(x,t_n) \rightarrow \vec{\psi}^{(n)},$$

where $\Psi(x_j, t_n) = \psi_j^{(n)}$. We then discretize the Hamiltonian as a matrix, exactly as in Project 4 (in fact, if you want to reuse code from Project 4, please do). The Crank-Nicolson method is to split up the Hamiltonian, half explicit and half implicit:

$$\frac{\partial}{\partial t} \Psi(x,t) = \frac{\vec{\psi}^{(n+1)} - \vec{\psi}^{(n)}}{\Delta t} = -\frac{i}{\hbar} \hat{H} \Psi(x,t) = \frac{1}{2} \left(-\frac{i}{\hbar} \hat{H} \vec{\psi}^{(n+1)} - \frac{i}{\hbar} \hat{H} \vec{\psi}^{(n)} \right) \text{ or}$$

$$\left(1 + \frac{i\Delta t}{2\hbar} \hat{H} \right) \vec{\psi}^{(n+1)} = \left(1 - \frac{i\Delta t}{2\hbar} \hat{H} \right) \vec{\psi}^{(n)} \text{ or } \vec{\psi}^{(n+1)} = \left(1 + \frac{i\Delta t}{2\hbar} \hat{H} \right)^{-1} \left(1 - \frac{i\Delta t}{2\hbar} \hat{H} \right) \vec{\psi}^{(n)}.$$

As you can see, this problem requires complex variables. One can either code the problem using complex variables, or by recasting as a purely real problem. I will outline the real variable approach. You may use either one.

Real variable approach: We rewrite $\left(1 + \frac{i\Delta t}{2\hbar} \hat{H} \right) \vec{\psi}^{(n+1)} = \left(1 - \frac{i\Delta t}{2\hbar} \hat{H} \right) \vec{\psi}^{(n)}$ as

$$\left(1 + \frac{i\Delta t}{2\hbar} \hat{H} \right) [\text{Re} \vec{\psi}^{(n+1)} + i \text{Im} \vec{\psi}^{(n+1)}] = \left(1 - \frac{i\Delta t}{2\hbar} \hat{H} \right) [\text{Re} \vec{\psi}^{(n)} + i \text{Im} \vec{\psi}^{(n)}] \text{ which becomes two}$$

$$\text{coupled equations, } \text{Re} \vec{\psi}^{(n+1)} - \frac{\Delta t}{2\hbar} \hat{H} \text{Im} \vec{\psi}^{(n+1)} = \text{Re} \vec{\psi}^{(n)} + \frac{\Delta t}{2\hbar} \hat{H} \text{Im} \vec{\psi}^{(n)} \text{ and}$$

$$\text{Im} \vec{\psi}^{(n+1)} + \frac{\Delta t}{2\hbar} \hat{H} \text{Re} \vec{\psi}^{(n+1)} = \text{Im} \vec{\psi}^{(n)} - \frac{\Delta t}{2\hbar} \hat{H} \text{Re} \vec{\psi}^{(n)}. \text{ We can combine these into a}$$

$$\text{"supermatrix" or } \begin{pmatrix} 1 & -\frac{\Delta t}{2\hbar} \hat{H} \\ \frac{\Delta t}{2\hbar} \hat{H} & 1 \end{pmatrix} \begin{pmatrix} \text{Re} \vec{\psi}^{(n+1)} \\ \text{Im} \vec{\psi}^{(n+1)} \end{pmatrix} = \begin{pmatrix} 1 & \frac{\Delta t}{2\hbar} \hat{H} \\ -\frac{\Delta t}{2\hbar} \hat{H} & 1 \end{pmatrix} \begin{pmatrix} \text{Re} \vec{\psi}^{(n)} \\ \text{Im} \vec{\psi}^{(n)} \end{pmatrix}. \text{ Therefore, if the}$$

wavefunction is discretized as a (complex) vector of length N , then we replace it with purely real vectors of length $2N$ and purely real $2N \times 2N$ matrices.

To evolve in time from time t_n to time t_{n+1} , start with the $2N$ -dimension vector $\begin{pmatrix} \text{Re}\bar{\psi}^{(n)} \\ \text{Im}\bar{\psi}^{(n)} \end{pmatrix}$

and then multiply $\begin{pmatrix} 1 & \frac{\Delta t}{2\hbar}\hat{H} \\ -\frac{\Delta t}{2\hbar}\hat{H} & 1 \end{pmatrix} \begin{pmatrix} \text{Re}\bar{\psi}^{(n)} \\ \text{Im}\bar{\psi}^{(n)} \end{pmatrix}$. Finally one must solve

$$\begin{pmatrix} 1 & -\frac{\Delta t}{2\hbar}\hat{H} \\ \frac{\Delta t}{2\hbar}\hat{H} & 1 \end{pmatrix} \begin{pmatrix} \text{Re}\bar{\psi}^{(n+1)} \\ \text{Im}\bar{\psi}^{(n+1)} \end{pmatrix} = \begin{pmatrix} 1 & \frac{\Delta t}{2\hbar}\hat{H} \\ -\frac{\Delta t}{2\hbar}\hat{H} & 1 \end{pmatrix} \begin{pmatrix} \text{Re}\bar{\psi}^{(n)} \\ \text{Im}\bar{\psi}^{(n)} \end{pmatrix} \text{ to get } \begin{pmatrix} \text{Re}\bar{\psi}^{(n+1)} \\ \text{Im}\bar{\psi}^{(n+1)} \end{pmatrix}. \text{ Because}$$

everything is real, one can use the `matinv.f` package and use LU decomposition, and

either solve the linear equation or invert $\begin{pmatrix} 1 & -\frac{\Delta t}{2\hbar}\hat{H} \\ \frac{\Delta t}{2\hbar}\hat{H} & 1 \end{pmatrix}$. If the latter, it is more efficient

to invert the matrix once and store. Finally, for the projects below, you will need to compute the probability density $\rho(x_j, t_n) = |\Psi(x_j, t_n)|^2 = (\text{Re}\psi_j^n)^2 + (\text{Im}\psi_j^n)^2$.

Note: I updated the `matinv.f` file so that it can handle larger arrays (up to 1000), so download the revised file. However, I find using more than 100 or 200 lattice points very slow, and seems unnecessary. *Be sure your results are stable against changing the # of lattice points while keeping L fixed; when I coded up my solution, I accidentally used $1/dx$ rather than $1/dx^{**2}$ for the second derivative. Your code should be stable under changes in dt as well (see below for some hints on choice of dt).*

Basic project: Put your wavefunction in a box from $-L$ to $+L$. By using the same discretization of the second derivative as in Project 4, we implicitly have a boundary condition that the wavefunction vanishes at the boundaries. You should set $\hbar = m = 1$ (which is different from Project 4.)

Input: size of box L ; # of lattice points N ; time step dt ; # of time steps.

Create a Gaussian wavefunction somewhere in your box. **The user must be able to input the center and width of the Gaussian.** You will also input the size L of the box, the number of lattice points N for your wavefunction (so that $dx = 2 * L / N$), as well as the time step dt . Now let your wavefunction evolve a certain number of time steps (which must be input by the user).

Your code should print to separate files at each time t :

1: the normalization $\sum_{j=1}^N \rho_j dx$ which should be constant;

2. The expectation value of x : $\langle x \rangle = \sum_{j=1}^N x_j \rho_j dx / \sum_{j=1}^N \rho_j dx$

3. The width $\sigma = \langle x^2 \rangle - \langle x \rangle^2$, where $\langle x^2 \rangle = \sum_{j=1}^N x_j^2 \rho_j dx / \sum_{j=1}^N \rho_j dx$

So you should have 3 files; the first will have t and the normalization (this checks that the normalization remains constant); the second, $t \langle x(t) \rangle$, and so on. Finally, at the end (after the last time step) have your code print to a file (make it clear to the user what the filename is) of x and $|\Psi(x)|^2$.

In quantum mechanics, we can actually compute analytically the spreading of a Gaussian. Suppose we started with a Gaussian at position x_0 and width σ :

$\Psi(x, t = 0) = (2\pi\sigma^2)^{-1/4} \exp\left(-\frac{(x - x_0)^2}{4\sigma^2}\right)$. One can show in quantum mechanics that under

the Schrodinger equation, at time t this wavefunction is now

$$\Psi(x, t) = (2\pi\sigma^2)^{-1/4} \frac{1}{\sqrt{1 + i \frac{\hbar^2 t}{2m\sigma^2}}} \exp\left(-\frac{(x - x_0)^2}{4\sigma^2 \left(1 + i \frac{\hbar^2 t}{2m\sigma^2}\right)}\right),$$

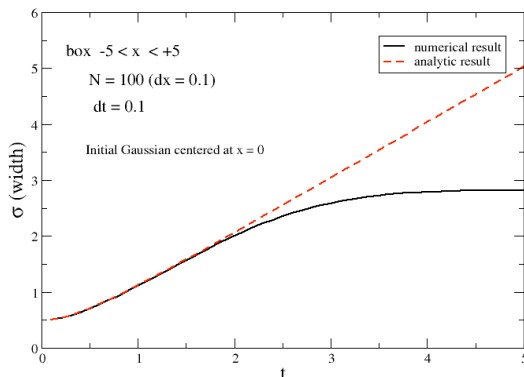
which has a density

$$\rho(x, t) = |\Psi(x, t)|^2 = \frac{1}{\sqrt{2\pi \left(\sigma^2 + \frac{\hbar^4 t^2}{4m^2\sigma^4}\right)}} \exp\left(-\frac{(x - x_0)^2}{2 \left(\sigma^2 + \frac{\hbar^4 t^2}{4m^2\sigma^4}\right)}\right).$$

As a function of time,

one finds the width grows $\langle (x - x_0)^2 \rangle = \sigma^2 + \frac{\hbar^4 t^2}{4m^2\sigma^2}$. (I've actually left out the speed of light c which we always set = 1.)

Validating your code. Confirm that your width grows, initially, according to the analytic prescription. After some time it will deviate because of the wall. Here is an example run in a box with $L = 5$, $N = 100$ (so $dx = 0.1$), an initial Gaussian with $\langle x \rangle = 0$. and initial width of 0.5, with a time step of 0.1



(figure 1)

As mentioned, the deviation (which starts around $t = 2$) is expected. We can also see this by snapshotting the wavefunction at different times.

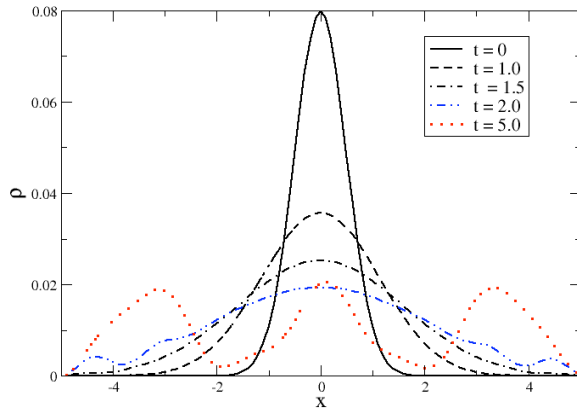
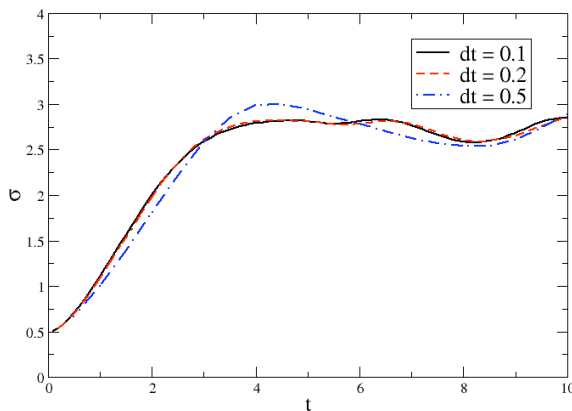


Figure 2

Produce and submit a plot like figure 1 (you do not need the analytic solution). Although you do not have to submit it, your code should write to a file the density as a function of x as well (you do not, and should not, write at all time steps; instead make it write at the last time step). That is, do *not* submit the equivalent of Figure 2, but be sure your code can produce the output so that one could easily generate Figure 2.

To check dependence on dt I looked at both the stability of the normalization (which should be 1) and on the width. For the same parameters described above, the normalization was very stable, out to $t = 10.0$ at least up to $dt = 2.0$. (This is expected for the Crank-Nicolson method, which is unitary and thus supposed to conserve the normalization.) The width, however, was more sensitive; the $dt = 0.1$ and 0.2 results were very similar, but started to change for $dt = 0.5$:



If you have trouble, solve using the simpler explicit algorithm, i.e.,

$$\begin{pmatrix} \text{Re}\vec{\psi}^{(n+1)} \\ \text{Im}\vec{\psi}^{(n+1)} \end{pmatrix} = \begin{pmatrix} 1 & \frac{\Delta t}{\hbar} \hat{H} \\ -\frac{\Delta t}{\hbar} \hat{H} & 1 \end{pmatrix} \begin{pmatrix} \text{Re}\vec{\psi}^{(n)} \\ \text{Im}\vec{\psi}^{(n)} \end{pmatrix}. \text{ You'll find that even for small } dt \text{ the normalization}$$

will quickly deviate from 1; even for $dt = 0.005$, I found the normalization blew up after 50 time steps, which is a total time of only 0.25.

Advanced project. Now put your wavefunction in a harmonic well, $V(x) = \frac{1}{2} kx^2$; the user should input k . Now you want to follow $\langle x \rangle$ as a function of t . Classically, one expects $\langle x(t) \rangle = x_0 \cos \sqrt{\frac{k}{m}} t$. Do you get this? What happens if you change your initial width? Note: I recommend a relatively small value of k (say around 0.01 or 0.02). I also find that even with a small dt the normalization grows with time; do not worry about this. Your code should have exactly the same input as the basic project, but also asks for k . You should give me a plot for $\langle x(t) \rangle$ for some value of x_0 for some k (your choice, but put your parameters on the plot), and for two different starting widths. Make sure that x_0 is not too close to the boundaries of the box, and also that the width is small enough that the tail of the wave packet is small at the boundaries.

Extra credit (even for the basic project): The Crank-Nicolson method should conserve the energy of the system. Check that it does. (Part of the assignment is to deduce, for yourself, *how* to compute the energy of the system.)

Due by midnight, Monday, Dec 13, 2010. Early submission deadline: midnight, Thursday Dec 9, 2010.