Prithivi Rana. (11/30)

Ex 9.8: The schrodinger equation and the Crank-Nicolson method

This exercise uses the Crank-Nicolson method to solve the full time-dependent Schrodinger equation and hence develop a picture of how a wavefunction evolves over time.

We will look at the Schrodinger equation in one dimension. 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 to be zero at the walls, which we'll put at x=0 and x =L.

Problem 1. *Write a program to perform a single step of the Crank-Nicolson method for this electron, calculating the vector 1/J(t) of values of the wavefunction, given the initial wavefunction above and using N = 1000 spatial slices with a = L/N. Then extend your program to perform repeated steps and solve for a sequence of times.*

Solution. The first part of the problem asks us to write a program that performs repeated steps of the Crank-Nicolson method to get the wavefunction at ssubsequent timesteps. We start by declaring all the constants as noted in the problem:

```
#user defined variables
nsteps = 1000                 # number of timesteps
h = 1e-18                     # timestep in seconds

# constants
M = 9.109e-31                 # mass of electron in kg
L = 1e-8                      # length of the box in meters
N = 1000                      # number of spatial slices
a = L / N                     # Grid spacing
```

We will be considering an electron of mass `M` in a box of length `L`. We will be evaluating the wavefunction at `N` spatial slices between `x=0` and `x=L`, each distance `a` apart. This lets us express the wavefunction at a particular time as a vector:

$$\psi(t) = [\psi(0,t), \psi(a,t), \psi(2a,t) \ldots \psi(L,t)]$$

In the implementation of the program, the wavefunction is stored in a `(nsteps x N)` NumPy array of complex datatype where each row represents a ψ(t) vector at a timestep. This effectively looks like:

$$\Psi = \begin{bmatrix} \psi(0,0) & \psi(a,0) & \psi(2a,0) & \ldots & \psi(L,0) \\ \psi(0,1) & \psi(a,1) & \psi(2a,1) & \ldots & \psi(L,1) \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ \psi(0,\texttt{nsteps}) & \psi(a,\texttt{nsteps}) & \psi(2a,\texttt{nsteps}) & \ldots & \psi(L,\texttt{nsteps}) \end{bmatrix}$$

We start off with an empty array of this shape.

The problem also gives us an expression for the wavefunction at `t=0`:

$$\psi(x,0) = exp\left[-\frac{(x-x_0)^2}{2\sigma^2}\right]e^{ikx} \tag{1}$$

with the constants declared in the program as:

```
# constants for wavefunction calculation
x0 = np.float64(L / 2)      # x_0
sig = 1e-10                 # sigma in m
k = 5e10                    # k in m^-1
```

We evaluate this initial wavefunction at the `x` gridpoints (0, a, 2a, . . . L) using equation 1 and set this to the first row of our wavefunction matrix $\Psi[0]$. We also have the boundary conditions that dictate the particle's wavefunction at the walls is zero, i.e. $\psi(0,t) = 0$ and $\psi(L,t) = 0$. So, we explicitly set the first and last element of the first row to be 0 to avoid any potential errors in calculation.

Now we need to find the wavefunction at a later time. We have the Crank-Nicolson equation for the Schrodinger equation from the problem definition:

$$\begin{aligned}\psi(x,t+h) - h\frac{i\hbar}{4ma^2}\left[\psi(x+a,t+h) + \psi(x-a,t+h) - 2\psi(x,t+h)\right]\\ = \psi(x,t) - h\frac{i\hbar}{4ma^2}\left[\psi(x+a,t) + \psi(x-a,t) - 2\psi(x,t)\right]\end{aligned}$$

We can also express this equation as:

$$\mathbf{A}\psi(t+h) = \mathbf{B}\psi(t) \tag{2}$$

where,**A** and **B** are symmetric tridiagonal matrices with a_1 and b_1 in the main diagonals and a_2 and b_2 in the upper and lower diagonals (using values as defined in the problem). The matrix **A** was implemented as complex arrays of shape `(N x N)` and appropriate values were assigned to the main, upper, and lower diagonals.

If we find a vector **v**, such that $\mathbf{v} = \mathbf{B}\psi(t)$, we can redefine equation 2 as:

$$\mathbf{A}\psi(t+h) = \mathbf{v} \tag{3}$$

which resembles a linear system equation that can be solved using NumPy's `linalg.solve`.

So we start by first multiplying the vector ψ by the matrix $\mathbf{B}$ to get $\mathbf{v}$. Although this can be done using the matrix multiplication operator (`@`), the problem provides us with a faster method of computing the v_i component with the formula:

$$v_i = b_1\psi_i + b2(\psi_{i+1} + \psi_{i-1}) \tag{4}$$

The matrix $\mathbf{A}$ doesn't depend on time, so the same matrix can be used for all ψs. Once we obtain $\mathbf{v}$, we can solve equation 3 to get $\psi(t+h)$ which is the wavefunction at the next timestep. This describes one complete step of the Crank-Nicolson method.

A function to perform the computations listed above was defined:

```
def crank_nicolson(A,B=None,psi):
    psi_new = np.zeros(np.shape(psi),dtype=complex)

    # Calculating v with the formula given in the problem
    v = np.zeros(np.shape(psi),dtype=complex)
    v[1:-1] = b1 * psi[1:-1] + b2 * (psi[2:] + psi[0:-2])

    psi_new = np.linalg.solve(A, v)
    psi_new[0] = 0
    psi_new[-1] = 0
    return psi_new
```

The function first calculates the vector $\mathbf{v}$ with `v[1:-1] = b1 * psi[1:-1] + b2 * (psi[2:] + psi[0:-2])`. This assigns values for $\mathbf{v}$ at all the grid slices except at the boundaries, keeping those boundary values 0. It then uses `linalg.solve` on the matrix $\mathbf{A}$ and $\mathbf{v}$ to get the wavefunction at a new time.

The function then sets the boundaries (first and last element of the row) to 0 once again to ensure boundary conditions are satisfied. A loop that runs over `num_step` iterations calls this function repeatedly and saves each newly calculated ψ to a new row of the array Ψ.

Problem 2. *Extend your program to make an animation of the solution by displaying the real part of the wavefunction at each time-step.*

Solution. I made an animation of the wavefunction's time evolution using Matplotlib's `animation`. This was saved as an mp4, available in my GitHub repository for this course. The simulation was run for 5000 iterations with a single timestep of 10^{-18} s. A few slices of the plots at different timesteps are included in Figure 1.

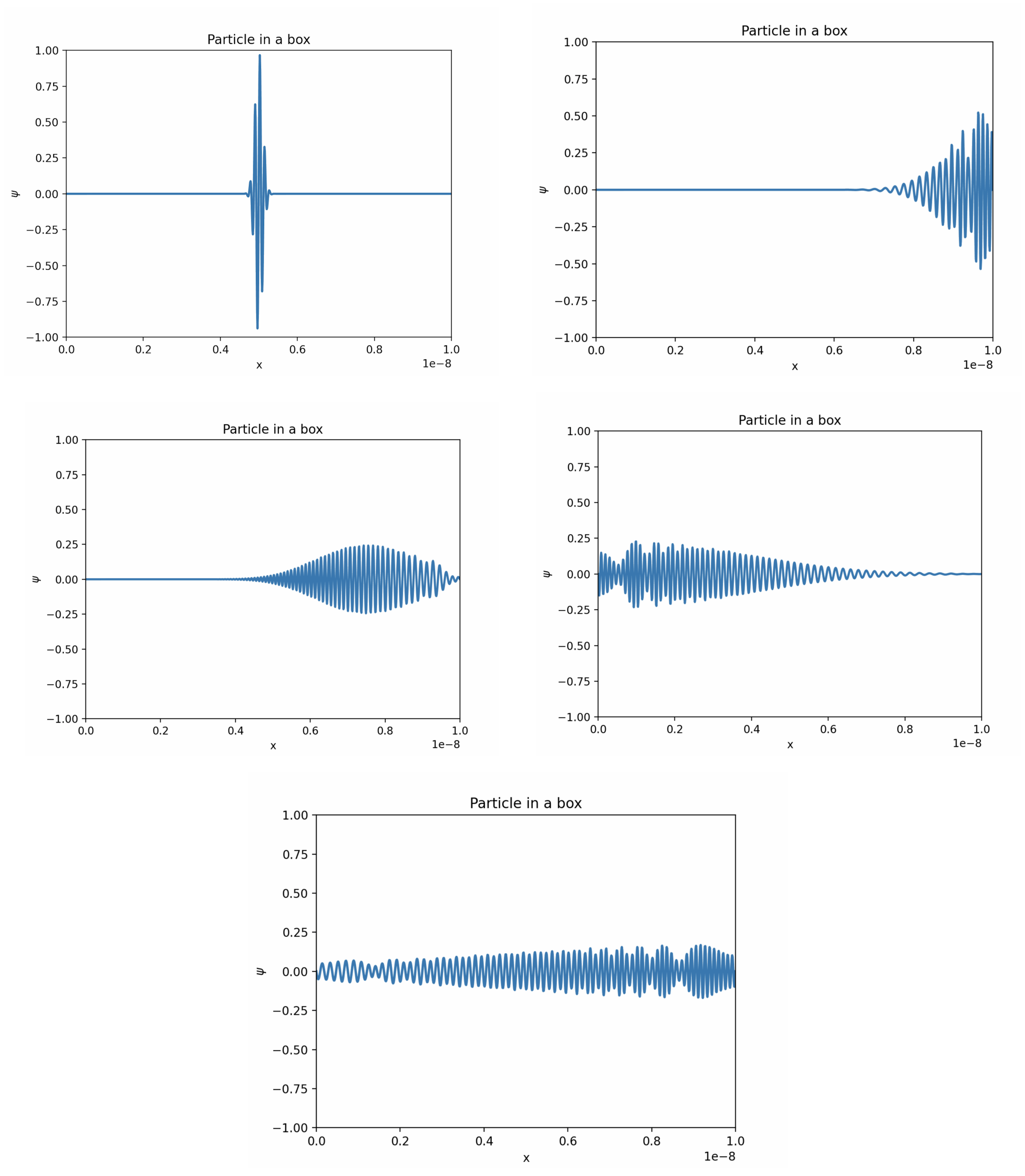

Figure 1: Slices of the wavefunction at different timesteps

Problem 3. *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.*

Solution. Looking at the animation, the waveform first starts in the middle of the box and propagates to the right. Initially, the wavefunction is localized in the middle of the box with strong narrow peaks. The localized wavefunction gives us a well defined position, which in turn means the momentum of the particle has greater uncertainty. As it propagates to the right, the peaks become less sharp and the waveform spreads out more. Upon hitting the right wall, the waveform is reflected back with what looks like an inverted profile (similar to how mechanical waves are reflected at fixed boundaries).

The reflected wave then seems to interfere with the wave still approaching the right wall and forms a sinusoidal shape as the entire wave is reflected. This occurs over mutiple reflections and the wave interferes with itself more and more leading to a waveform that is spread out over the entire length of the box with larger amplitudes in some regions (regions of constructive interference) and almost zero amplitudes in others (destructive interference). This shows that the wavefunction and hence the probability of finding the position of the particle behaves in response to the boundary conditions and shows the quantum nature of the particle.