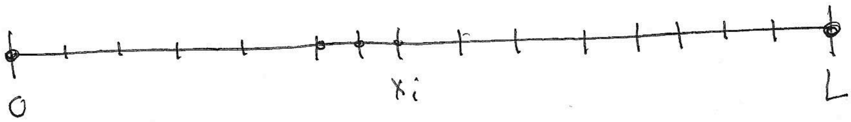
$$i\hbar \frac{\partial}{\partial t}\psi = \left[-\frac{\hbar^2}{2m}\frac{\partial^2}{\partial x^2} + V(x)\right]\psi$$

$$\psi(x,t) = a(x,t) + i\,b(x,t) \sim \begin{pmatrix} a(x,t) \\ b(x,t) \end{pmatrix} \to i\psi = \begin{pmatrix} -b(x,t) \\ a(x,t) \end{pmatrix}$$

$$\frac{\partial}{\partial t}\begin{pmatrix} -b(x,t) \\ a(x,t) \end{pmatrix} = -\frac{\hbar}{2m}\frac{\partial^2}{\partial x^2}\begin{pmatrix} a(x,t) \\ b(x,t) \end{pmatrix} + \begin{pmatrix} \frac{1}{\hbar}V(x)\,a(x,t) \\ \frac{V}{\hbar}(x)\,b(x,t) \end{pmatrix}$$



$i = 0, \ldots, n$ $\quad x_i = i\,\delta x \quad \delta x =$ lattice spacing $\quad \delta x = L/n$

$$a(x_i,t) = a_i(t) \qquad b(x_i,t) = b_i(t) \qquad V(x_i) = V_i$$

$$\left(\frac{\partial^2 a}{\partial x^2}\right)(x_i) \approx \frac{a_{i+1} + a_{i-1} - 2a_i}{(\delta x)^2}$$

$$\left(\frac{\partial^2 b}{\partial x^2}\right)x_i \approx \frac{b_{i+1} + b_{i-1} - 2b_i}{(\delta x)^2}$$

$$\begin{pmatrix} \partial_t(-b_i) \\ \partial_t(a_i) \end{pmatrix} = -\frac{\hbar}{2m\,\delta x^2}\begin{pmatrix} a_{i+1} + a_{i-1} - 2a_i \\ b_{i+1} + b_{i-1} - 2b_i \end{pmatrix} + \begin{pmatrix} \frac{V_i}{\hbar}\,a_i(t) \\ \frac{V_i}{\hbar}\,b_i(t) \end{pmatrix}$$

$$t = \overbrace{\left(\frac{2m\,\delta x^2}{\hbar}\right)}^{\delta t}\tau = (\delta t)\,\tau \leftarrow \text{dimensionless}$$

↓ time scale

$$\begin{pmatrix} -\frac{\partial}{\partial \tau} b_i(\tau) \\ \frac{\partial}{\partial \tau} a_i(\tau) \end{pmatrix} = \begin{pmatrix} 2a_i - (a_{i+1} + a_{i-1}) \\ 2b_i - (b_{i+1} + b_{i-1}) \end{pmatrix} + \begin{pmatrix} \frac{\delta t}{\hbar} V_i\, a_i(t) \\ \frac{\delta t}{\hbar} V_i\, b_i(t) \end{pmatrix}$$

$$V_i = \overset{\text{energy scale}}{V_0}\,\overset{\text{dimensionless}}{\tilde{V}_i} \qquad (2)$$

$$\underbrace{\begin{pmatrix} \frac{\partial}{\partial \tau} b_i(\tau) \\ \frac{\partial}{\partial \tau} a_i(\tau) \end{pmatrix} = \begin{pmatrix} a_{i+1} + a_{i-1} - 2a_i \\ 2b_i - (b_{i+1} + b_{i-1}) \end{pmatrix}}_{\text{Baseline model "Free Particle"}} + \left(\frac{\delta t}{\hbar} V_0\right) \begin{pmatrix} \tilde{V}_i\, a_i(\tau) \\ \tilde{V}_i\, b_i(\tau) \end{pmatrix}$$

Dimensionless constant

$$\beta := \frac{\delta t V_0}{\hbar} = \underbrace{\frac{2mL^2}{\hbar^3 n^2} V_0}$$

A dimensionless coupling constant on the lattice

Next: build an evolution scheme (algorithm) through small ($\delta\tau \ll 1$) dimensionless time steps.

Trade off: We want to do evolutions through enough time steps to see disturbances in the wave function propigate across the entire length of the lattice. This means if we step forward in time by $\delta\tau \sim 1$, we will need

total time $N\delta t$, $\quad N \approx \dfrac{L/\delta t}{(\delta x/\delta t)} = \dfrac{L}{\delta x}\dfrac{\cancel{\delta t}}{\cancel{\delta t}} = \underline{\underline{n}}$

If we choose $\Delta\tau \sim \frac{1}{m}$ (m some integer > 0) we will need between $n \cdot m$ and $\underline{2nm}$ total time steps at a minimum to see interactions that propigate accross the total lattice.

This means we prepare for 20,000 time steps on a 1000 point lattice (at the minimum) and probably more like 50,000 ~ 100,000.

This means that if we want to see real results in a real time T, a single evolution time step must run in a time less than $10^{-5}T$. (If e.g. $T = 100$ seconds then a single evolution time step must run in 10^{-3} s.)

This gives you a rule of thumb about the time your code can take to do an evolution time step for data on the lattice. Clearly, to be fast you need to get single time step evolution execution time under a millisecond. Since this scales with the lattice size, you will probably want to debug a lot of code on lattices with smaller number of points (like 200 - 500).

Boundary conditions: $a_0 = b_0 = a_n = b_n = 0$
Hold these values fixed for "box" boundary conditions
These values are not evolved during the program run.

SO FAR: We have translated a partial differential equation into a very large number of ordinary differential equations that are coupled together.

PDE → MANY ODEs

NEXT: We turn to the problem of approximating the time evolution in continuous time into an approximate series of time step evolutions.

You will want to encapsulate the time-evolution step in a single function or routine. I will suggest a baseline approach; however, there is room to upgrade or try out different evolution time step algorithms. Encapsulating this part of the program will facilitate experimentation.

General Structure of Your Code

Initialize State + Lattice → Evolution Code → Analysis + Display a History

Initialize State + Lattice

translate an initial state <u>specified by a formula</u> $\psi_0(x)$ into a lattice representation

prepare to evolve

Evolution Code

<u>Evolve</u> and <u>Record</u> a history of the lattice state and prop. information

⟨ Should produce an output file ⟩

Analysis + Display a History

Look at the lattice state - graphs and other quantities

WORKS ON OUTPUT FILES