Project 2

Problem 1

Background

We are solving the Schrödinger equation, subject to the boundary and initial conditions shown below:

$$\begin{cases} i\psi(x,t)_t = -\psi_{xx} + V(x,t)\psi, & 0 \le x \le 1, \quad 0 \le t \le t_{\max} \\ \psi(x,0) = \psi_0(x) \\ \psi(0,t) = \psi(1,t) = 0 \end{cases} \tag{1}$$

where for this problem we consider $\psi_0(x) \in \mathbb{R}$. ψ however will be complex, and we will separate it into real and imaginary components, $\psi = \psi_{\text{Re}} + \psi_{\text{Im}}$.

We will be using the integral of the probability density, $\rho = |\psi|^2 = \psi\psi^*$, as a check of our implementation. We have that

$$P(x,t) = \int_0^x \rho(\tilde{x},t)\mathrm{d}\tilde{x} = \int_0^x \psi(\tilde{x},t)\psi^*(\tilde{x},t)\mathrm{d}\tilde{x}, \tag{2}$$

which, when we integrate of the entire domain $0 \le x \le 1$ (by evaluating $P(1,t)$), should be equal to 1 if our solution ψ is normalized. If it is not normalized, we will simply expect $P(1,t)$ to remain constant to some level of error, $P(1,t) \approx \text{const}$. We could also plot $\sqrt{\rho} = |\psi|$ as a disgnostic, but we will lose information by only looking at the magnitude of ψ.

The family of solutions to (1) is:

$$\psi(x,t) = e^{-im^2\pi^2 t}\sin(m\pi x), \qquad m = 1,2,3,... \tag{3}$$

Implementation

Our implementation is completed in the function `sch_1d_cn()`, which has the following header:

```
function [x t psi psire psiim psimod prob v] = sch_1d_cn(tmax, level,
    lambda, idtype, idpar, vtype, vpar)
% Solves 1D time-dependent Schrodinger equation via the Crank-
    Nichelson method
% Inputs
%
% tmax: Maximum integration time
% level: Discretization level
% lambda: dt/dx
% idtype: Selects initial data type
% idpar: Vector of initial data parameters
% vtype: Selects potential type
% vpar: Vector of potential parameters
%
% Outputs
%
% x: Vector of x coordinates [nx]
% t: Vector of t coordinates [nt]
% psi: Array of computed psi values [nt x nx]
% psire Array of computed psi_re values [nt x nx]
% psiim Array of computed psi_im values [nt x nx]
% psimod Array of computed sqrt(psi psi*) values [nt x nx]
% prob Array of computed running integral values [nt x nx]
% v Array of potential values [nx]
```

Discretization – Crank-Nicolson

We will discretize the domain to a level l, with the time and spatial spacings defined by

$$\lambda = \frac{\Delta t}{\Delta x}. \tag{4}$$

From this we have the standard discretization,

$$n_x = 2^l + 1 \tag{5}$$

$$\Delta x = 2^{-l} \tag{6}$$

$$\Delta t = \lambda \Delta x \tag{7}$$

$$n_t = \text{round}(t_{max}/\Delta t) + 1 \tag{8}$$

The Crank-Nicolson (CN) discretization of (1) gives:

$$i\frac{\psi_j^{n+1} - \psi_j^n}{\Delta t} = -\frac{1}{2}\left(\frac{\psi_{j+1}^{n+1} - 2\psi_j^{n+1} + \psi_{j-1}^{n+1}}{\Delta x^2} + \frac{\psi_{j+1}^{n} - 2\psi_j^{n} + \psi_{j-1}^{n}}{\Delta x^2}\right) + \frac{1}{2}V_j^{n+\frac{1}{2}}\left(\psi_j^{n+1} + \psi_j^n\right) \tag{9}$$

for $j = 2, 3, ..., n_x - 1, \quad n = 1, 2, ..., n_t - 1$. We also have boundary conditions implemented as follows:

$$\psi_1^{n+1} = \psi_{n_x}^{n+1} = 0, \quad n = 1, 2, ..., n_t - 1 \tag{10}$$

$$\psi_j^1 = \psi_0(x_j), \quad j = 1, 2, ..., n_x. \tag{11}$$

Equation (9) can be rearranged into the form $\underline{\underline{A}}\,\underline{\psi}^{n+1} = \underline{f}$, which can be solved by left division. This is done below, by isolating all of the $n+1$ and n terms on the left and right hand sides, respectively.

$$\frac{i}{\Delta t}\psi_j^{n+1} + \frac{1}{2\Delta x^2}\psi_{j+1}^{n+1} - \frac{1}{\Delta x^2}\psi_j^{n+1} + \frac{1}{2\Delta x^2}\psi_{j-1}^{n+1} - \frac{1}{2}V_j^{n+\frac{1}{2}}\psi_j^{n+1} =$$
$$\frac{i}{\Delta t}\psi_j^{n} - \frac{1}{2\Delta x^2}\psi_{j+1}^{n} + \frac{1}{\Delta x^2}\psi_j^{n} - \frac{1}{2\Delta x^2}\psi_{j-1}^{n} + \frac{1}{2}V_j^{n+\frac{1}{2}}\psi_j^{n}$$

which we can organize into:

$$\left(\frac{1}{2\Delta x^2}\right)\psi_{j+1}^{n+1} + \left(\frac{i}{\Delta t} - \frac{1}{\Delta x^2} - \frac{1}{2}V_j^{n+\frac{1}{2}}\right)\psi_j^{n+1} + \left(\frac{1}{2\Delta x^2}\right)\psi_{j-1}^{n+1} =$$
$$\left(-\frac{1}{2\Delta x^2}\right)\psi_{j+1}^{n} + \left(\frac{i}{\Delta t} + \frac{1}{\Delta x^2} + \frac{1}{2}V_j^{n+\frac{1}{2}}\right)\psi_j^{n} + \left(-\frac{1}{2\Delta x^2}\right)\psi_{j-1}^{n}$$

Simplify further by writing

$$C_+\psi_{j+1}^{n+1} + C_0\psi_j^{n+1} + C_-\psi_{j-1}^{n+1} = f_j$$

where

$$C_+ = \left(\frac{1}{2\Delta x^2}\right)$$
$$C_0 = \left(\frac{i}{\Delta t} - \frac{1}{\Delta x^2} - \frac{1}{2}V_j^{n+\frac{1}{2}}\right)$$
$$C_- = \left(\frac{1}{2\Delta x^2}\right) = C_+$$
$$f_j = \left(-\frac{1}{2\Delta x^2}\right)\psi_{j+1}^{n} + \left(\frac{i}{\Delta t} + \frac{1}{\Delta x^2} + \frac{1}{2}V_j^{n+\frac{1}{2}}\right)\psi_j^{n} + \left(-\frac{1}{2\Delta x^2}\right)\psi_{j-1}^{n}$$

C_+, C_0, C_- form the tridiagonal inner elements of the matrix $\underline{\underline{A}}$. Including the boundaries from (10), we

have:

$$\underline{\underline{A}} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & ... \\ C_- & C_0 & C_+ & 0 & 0 & 0 & ... & 0 \\ 0 & C_- & C_0 & C_+ & 0 & ... & 0 & 0 \\ 0 & 0 & C_- & C_0 & ... & 0 & 0 & 0 \\ 0 & 0 & 0 & ... & C_0 & C_+ & 0 & 0 \\ 0 & 0 & ... & 0 & C_- & C_0 & C_+ & 0 \\ 0 & ... & 0 & 0 & 0 & C_- & C_0 & C_+ \\ ... & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{12}$$

I implemented $\underline{\underline{A}}$ and $\underline{f}$ in the code, and solved for the next iteration using left division, `psi(n+1,:)= A\f`. $\underline{\underline{A}}$ was implemented as a tridiagonal matrix using the `spdiags()` function as follows:

```
A = spdiags([dl dm du], -1:1, nx, nx);
```

Here `dl`, `dm` and `du` refer to the lower, main and upper diagonals of the $\underline{\underline{A}}$ matrix as shown in (12).

Initial data types

Our implementation is required to solve (1) with different types of initial conditions (ICs). In order to select which one to solve, the `idtype` and `idpar` variables are inputted. These are used as follows:

- `idtype == 0`: Exact family

  $$\psi(x,0) = \sin(m\pi x)$$

  `idpar(1): m`

- `idtype == 1`: Boosted Gaussian

  $$\psi(x,0) = e^{ipx} e^{-((x-x_0)/\delta)^2}$$

  `idpar(1), idpar(2), idpar(3)`: x_0, δ, p

To implement these, I set the value of `psi(1,:)`, differently depending on the value of `idtype`.

Potential types

We will also be selecting from two types of potential using the `vtype` input variable, and specifying its parameters via `vpar`, as follows:

- `vtype == 0`: No potential

  $$V(x) = 0$$

- `vtype == 1`: Rectangular barrier or well

  $$V(x) = \begin{cases} 0 & \text{for} \quad x < x_{\min} \\ V_c & \text{for} \quad x_{\min} \le x \le x_{\max} \\ 0 & \text{for} \quad x > x_{\max} \end{cases}$$

  `vpar(1),vpar(2),vpar(3)`: $x_{\min}, x_{\max}, V_c$

To implement this, I began with the potential term `v=zeros(nx,1)`. Then, if `vtype==1` was selected, I set the values of the potential between $x_{\min}$ and $x_{\max}$ to V_c using `v(x>xmin & x<xmax)= Vc`.

Additional elements

The above describes the minimum implementation necessary for the Crank-Nichelson method. However, in addition to the main output `psi`, the function must also produce the real and imaginary parts of `psi`, the mod of `psi`, the running intergral from equation (4), and an array of potential values.

- The real part of ψ, `psire`

  Implemented using `psire = real(psi);`

- The imaginary part of ψ, `psiim`

  Implemented using `psiim = imag(psi);`

- The mod of ψ, `psimod`

  Implemented using $|\psi|$=`psimod = abs(psi);`

- The running integral of the probability density, $P(x,t)$

  Computed to $\mathcal{O}(h^2)$ using the trapezoidal formula,

$$\int_{x_1}^{x_n} f(x)\mathrm{d}x \approx \frac{1}{2}\sum_{i=1}^{n-1}\left(f_i + f_{i+1}\right)\left(x_{i+1} - x_i\right).$$

  Rewriting (2) we have

$$P(x,t) = \int_0^x \psi(\tilde{x},t)\psi^*(\tilde{x},t)\mathrm{d}\tilde{x} = \int_0^x |\psi(\tilde{x},t)|^2\mathrm{d}\tilde{x} = \int_0^x \texttt{abs(psi)}^2\mathrm{d}\tilde{x}.$$

  Applying the trapezoidal formula to this we obtain, for some particular time t_j:

$$P(x_n,t) = \int_0^{x_n} |\psi(\tilde{x},t)|^2\mathrm{d}\tilde{x} \approx \frac{1}{2}\sum_{i=1}^{n-1}\left(|\psi_i|^2 + |\psi_{i+1}|^2\right)\left(x_{i+1} - x_i\right)$$

$$= \texttt{1/2 * sum((psimod(tj,1:xi-1).\^{}2 + psimod(tj,2:xi).\^{}2).*(x(2:xi)- x(1:xi-1)))}$$

  The above involves 2 `for` loops. I later implemented an improved version of this which avoids for loops and uses a built-in function in MATLAB, the `cumtrapz()` function. This performs a cumulative trapezoidal calculation. I have left the code for the above implementation as a comment for reference, but have replaced it with

$$P(x_n,t) = \texttt{prob = cumtrapz(psimod.\^{}2, 2)}$$

  which I then normalize using

```
prob = prob ./ prob(:,nx)
```

- The potential, $V(x,t)$ =`v`

  This was computed as described in the Potential types section above.

Analysis

Convergence testing

We will now test the convergence of our implementation. The idea is to check that the error between levels l goes as $\mathcal{O}(h^2)$, where $h = \Delta x = \Delta t/\lambda$. Let ψ^l be a solution obtained from the `psi` output from running `sch_1d_cn()` with level `level`=l. Then we can define the difference $d\psi^l = \psi^{l+1} - \psi^l$. Here we'll need to coarsen both the space and time dimensions of ψ^{l+1} to be able to make the subtraction. The convergence test is then performed by computing

$$||d\psi^l||_2(t^n) \qquad \text{where } ||v||_2 = \sqrt{\frac{\sum_{j=1}^m |v^j|^2}{m}}. \tag{13}$$

In words, this means the computation of $||d\psi^l||_2(t^n)$ is done by finding the spatial norms of the subtraction of grid functions at different levels, $|d\psi^l|$.
As mentioned, the solution should be $\mathcal{O}(h^2)$ accurate, and so of the form

$$\psi^l(x,t) = \psi(x,t) + h_l^2 e_2(x,t) + \mathcal{O}(h_l^4) \tag{14}$$

where $e_2(x,t)$ is an error function. For a test of levels $l = l_{\min}, l_{\min}+1, l_{\min}+2, ..., l_{\max}$, we will plot

$$||d\psi^{l_{\min}}||_2,\ 4||d\psi^{l_{\min}+1}||_2,\ 4^2||d\psi^{l_{\min}+1}||_2,\ 4^3||d\psi^{l_{\min}+1}||_2,\ ...,\ 4^{l_{\max}-l_{\min}-1}||d\psi^{l_{\max}-1}||_2, \tag{15}$$

and check if the curves align.
Additionally, if `idtype=0`, i.e. if the initial condition is $\psi(x,0) = \sin(m\pi x)$, then we have the exact solution (3), and we can plot the l-2 norm of the error too. In other words, we can replace ψ^{l+1} with ψ_{exact} to calculate

$$||E(\psi^l)||_2(t^n) = ||\psi_{\text{exact}} - \psi^l||_2(t^n). \tag{16}$$

The convergence testing was implemented in `ctest_1d.m` on two cases. The results are discussed below. Before we proceed to those, there are a few comments to make on the implementation of the convergence testing:

- `ctest_1d.m` calls on `ctest_1d_func()` to calculate $||d\psi^l||_2(t^n)$ for each level, and on `ctest_1d_func_exact()` to calculate $||E(\psi^l)||_2(t^n)$ using the exact solution (3) when `idtype=0`.
- The calculation of the exact solution by `ctest_1d_func_exact()` is done by calling `sch_1d_exact(x, t, m)`, which does not use any `for` loops. Instead, it uses `repmat` to extend the input vectors `x` and `t` into `[nt x nx]` arrays, and uses these to compute the exact solution via `psi_exact = exp(-1i*m^2*pi*2*t_array).*sin(m*pi*x_array)`.

Now, let's look at the results of convergence tests on two cases.

1. `idtype = 0`, `vtype = 0`

   ```
   idpar  = [3];
   vpar   = 0;
   tmax   = 0.25;
   lambda = 0.1;
   lmin   = 6;
   lmax   = 9;
   ```

   For this one, since `idtype=0`, plots of both $||d\psi^l||_2(t^n)$ and $||E(\psi^l)||_2(t^n)$ could be created. These are shown below.

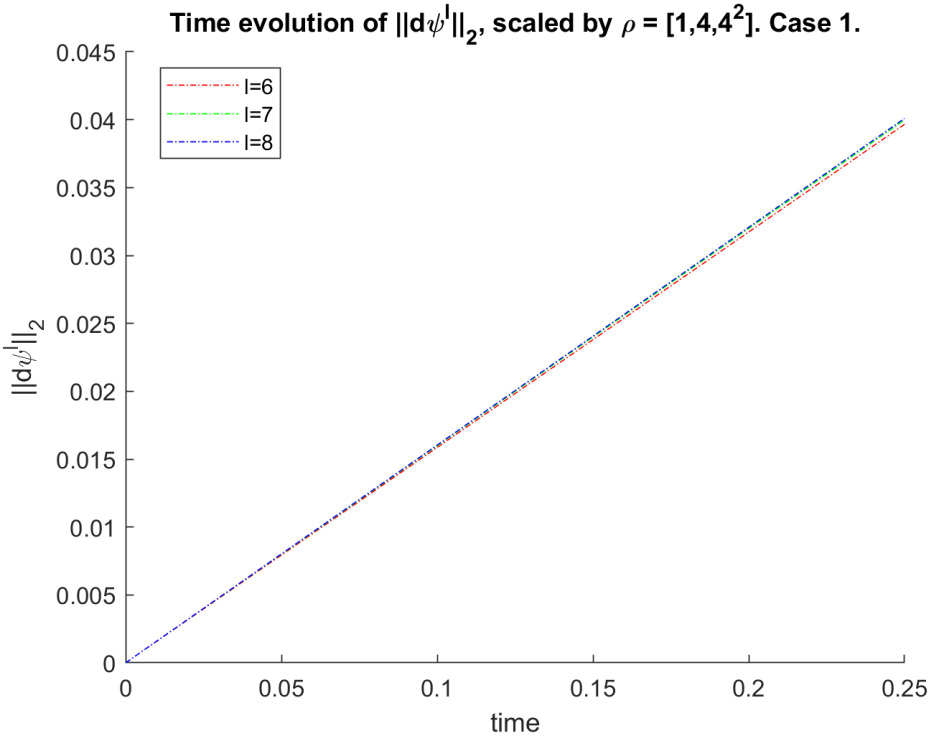

Figure 1: Plot of the evolution of rescaled $||d\psi^l||_2(t^n)$ with time for the first case with `idtype=0`. We can see that $||d\psi^6||_2(t^n)$, $4||d\psi^7||_2(t^n)$, and $4^2||d\psi^8||_2(t^n)$ align, indicating that the solution is indeed $\mathcal{O}(h^2)$ accurate.

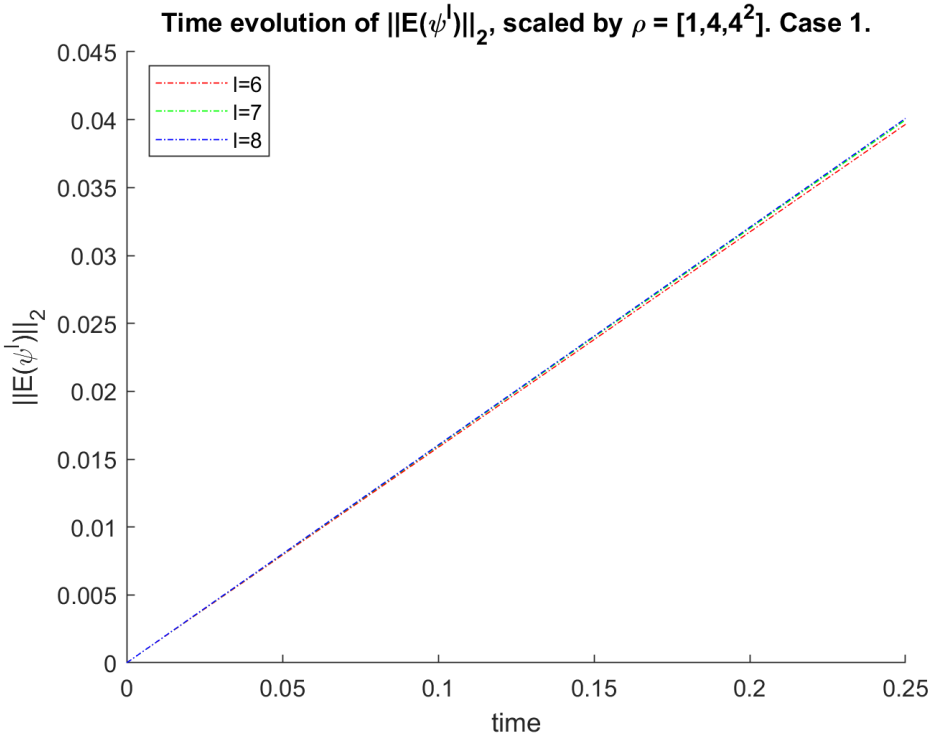

Figure 2: Plot of the evolution of rescaled $||E(\psi^l)||_2(t^n)$ with time for the first case with `idtype=0`. We can see that $||E(\psi^6)||_2(t^n)$, $4||E(\psi^7)||_2(t^n)$, and $4^2||E(\psi^8)||_2(t^n)$ align, indicating that the solution is indeed $\mathcal{O}(h^2)$ accurate.

2. `idtype = 1`, `vtype = 0`

```
idpar  = [0.50 0.075 0.0];
vpar   = 0;
tmax   = 0.01;
lambda = 0.01;
lmin   = 6;
lmax   = 9;
```

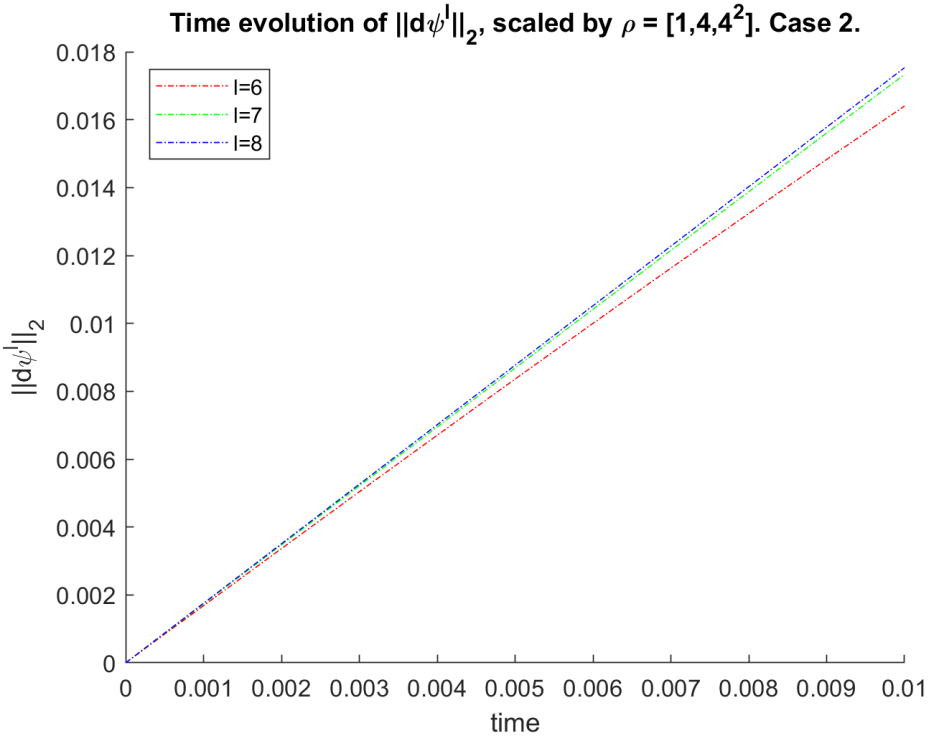

Figure 3: Plot of the evolution of rescaled $||E(\psi^l)||_2(t^n)$ with time for the second case with `idtype=1`. Once again, we can see that $||d\psi^6||_2(t^n)$, $4||d\psi^7||_2(t^n)$, and $4^2||d\psi^8||_2(t^n)$ align, indicating that the solution is indeed $\mathcal{O}(h^2)$ accurate. The lowest level (level 6) solution diverges slightly towards the end of the time evolution.

Numerical Experiments

Note that everywhere here log is the natural logarithm.

In this section, we will investigate the local probabilities of finding a function in a particular region of our domain, $x \in [0, 1]$. For this we will need to introduce a couple of definitions which use the integral of the probability density (2). The discretized version of (2) can be written as

$$P_j^n = P(x_j, t^n), \qquad j = 1, 2, 3, ..., n_x \qquad n = 1, 2, 3, ..., n_t. \tag{17}$$

Using this we can define the **temporal average** to be

$$\bar{P}_j = \frac{\sum_{n=1}^{n_t} P_j^n}{n_t},$$

Which in Matlab we will implement with `mean()`. This temporal average of the probability density will be normalized via $\bar{P}_j \to \bar{P}_j / \bar{P}_{n_x}$. With this new version, the quantity $\bar{P}(x_2) - \bar{P}(x_1)$ represents the fraction of time that the particle spends in the interval $[x_1, x_2]$. If x_1 or x_2 don't fall on a grid point, the nearest grid point is taken. For a free particle, because the width of our domain is $[0, 1]$, we expect that $\bar{P}(x_2) - \bar{P}(x_1) \to x_2 - x_1$. But for non-zero potentials, this will not always be the case. For the general case of $V \neq 0$, we can define the **excess fractional probability** as

$$\bar{F}_e(x_1, x_2) = \frac{\bar{P}(x_2) - \bar{P}(x_1)}{x_2 - x_1}.$$

The excess fractional probability for the case of a potential barrier and a potential well was investigated. Specifically, the excess fractional probability across regions of the domain was plotted against the height/depth of the barrier/well, on a logarithmic scale since $\bar{F}_e(x_1, x_2)$ spans many orders of magnitude. Let's take a look at the results.

Barrier Survey

First, the barrier survey was performed using `barrier_survey.m`. It involved plotting the dependence of $\log\big(\bar{F}_e(x_1, x_2)\big)$ on $\log(V_0)$ for 251 values of V_0. The parameters were as follows:

```
tmax = 0.10;
level = 9;
lambda = 0.01;
idtype = 1;
idpar = [0.40, 0.075, 20.0];
vtype = 1;
xmin = 0.6;
xmax = 0.8;
x1 = 0.8;
x2 = 1.0;
```

To interpret this physically, we have a system where the initial condition is a gaussian, located at $x = 0.40$. There is a barrier of varying height $V_0 = e^n$ where $n =$`linspace(-2,5,251)`. The barrier is located on $x = [0.6, 0.8]$. We are looking at $\log\big(\bar{F}_e(x_1, x_2)\big) = \log\big(\bar{F}_e(0.8, 1)\big)$, which is the area to the right of the barrier. The particle starts to the left of the barrier. We are looking for the excess fractional probability to the right of the barrier, which is a measure of how much of its time the particle spends on the side of the barrier opposite its initial position, relative to the case of a free particle. Thinking of it this way, the interpretation of the resulting graph becomes more intuitive.

Before presenting a graph of the results, a couple of details should be pointed out about the implementation:

- `barrier_survey.m` calls on `excess_frac_prob()` to calculate the excess fractional probabilities. This function takes in the usual parameters required by `sch_1d_cn()` (i.e. tmax, level, lambda, idtype, idpar, vtype, vpar), but also the two bounds for the calculation of $\bar{F}_e$, x_1 and x_2.
- The temporal average is implemented using `Pbar = mean(prob)`. It is then normalized via `Pbar = Pbar / Pbar(nx)`.
- `excess_frac_prob()` uses `abs(x-x1)==min(abs(x-x1))` to produce the location of the nearest neighbour to `x1` in the array of positions `x` (and it does the same for `x2`).

Using `barrier_survey.m`, the follow plot was produced:

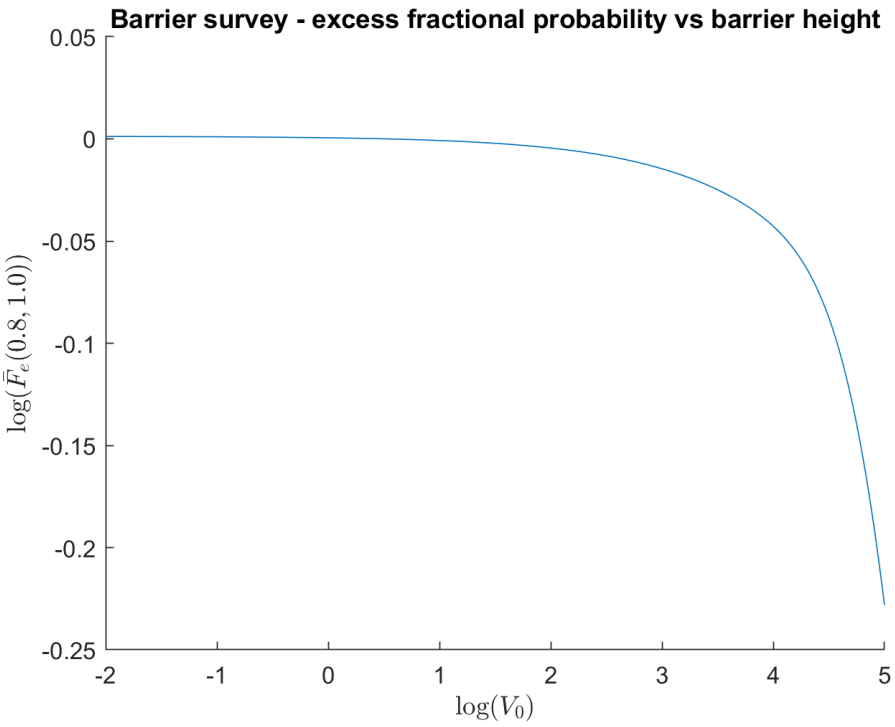

Figure 4: Plot of $\log\big(\bar{F}_e(0.8,1)\big)$ against $\log(V_0)$. A couple of observations can be made from this plot. First of all, for small V_0, $\bar{F}_e(0.8,1) \approx 1$, which means that the particle is spending just as much time to the right of the barrier as a free particle would, i.e. as if there were no barrier at all. This makes sense, since for small V_0 the barrier is only a small "bump" that does not strongly inhibit the particle's motion. For larger V_0, we notice that $\bar{F}_e(0.8,1) \ll 1$, which means that the particle is spending much less time in the interval to the right of the barrier than it would if the barrier were not there. Here the barrier is tall enough to successfully decrease the probability of the particle being found in the region to the right of it. A comment can therefore be made on the trend: we see that as the barrier height V_0 increases, the excess fractional probability of the particle spending time in the region to the right of the barrier decreases exponentially. The particle is further and further inhibited from "tunneling" (being found) to the right of the barrier, the side opposite from its starting point, as V_0 is increased.

Well Survey

Let's now perform an almost identical exercise with the case of a potential well. The associated script is `well_survey.m`. The parameters are:

```
tmax = 0.10;
level = 9;
lambda = 0.01;
idtype = 1;
idpar = [0.40, 0.075, 0.0];
vtype = 1;
xmin = 0.6;
xmax = 0.8;
x1 = 0.6;
x2 = 0.8;
```

This time, we still begin with a Gaussian centered around $x = 0.4$, but we now have a potential well of variable depth $V_0 = -e^n$ where $n =$`linspace(2,10,251)`. The well is located on $x = [0.6, 0.8]$. We are looking at $\log\big(\bar{F}_e(x_1,x_2)\big) = \log\big(\bar{F}_e(0.6,0.8)\big)$, which is the area inside of the well.

The resulting plot is shown below. Note that this time $\log|V_0|$ was used instead of $\log V_0$, since $V_0 < 0$.

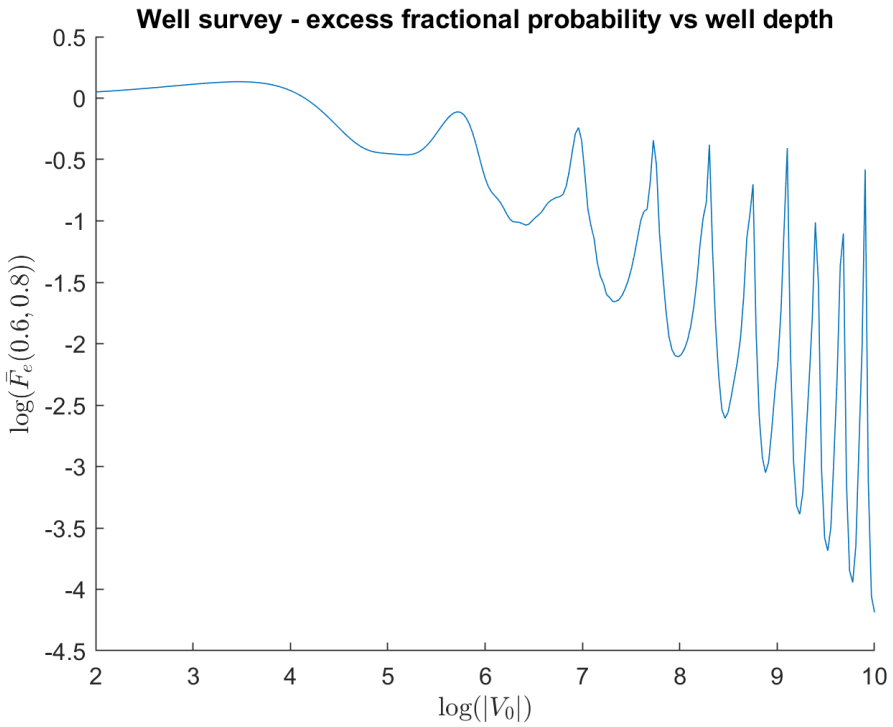

Figure 5: Plot of $\log\big(\bar{F}_e(0.6, 0.8)\big)$ against $\log(|V_0|)$. Again, we comment on the magnitude and trend: We observe that for shallow well depths, i.e. for small V_0, the particle is not trapped in the barrier and spends relatively equal amounts of time outside it as inside it, so $\bar{F}_e(0.6, 0.8) \approx 1$. As $|V_0|$ is increased, we again observe a downward trend in $\log\big(\bar{F}_e(0.6, 0.8)\big)$, but with an interesting repeating pattern. The excess fractional probability seems to intermittently return very close to a magnitude of 1, for some values of V_0. There appears to be some kind of resonant behavior behind this, related to the width and height of the Gaussian in comparison with the width and depth of the well, which causes the particle to find it very easy to break out of and fall into the well for certain values of V_0.

Conclusion

In this problem, we solved the 1D Schrödinger equation using the Crank-Nicolson method. Specifically, we solved it for two cases of initial conditions, a Gaussian and a sinusoid, and for two (technically three) types of potential, a free particle and a well or a barrier. We used `spdiags` to reduce the number of computations we needed to perform, and determined that our solution was $\mathcal{O}(h^2)$ accurate through convergence tests. Finally, we explored the probabilities of a particle tunneling across a potential barrier or remaining within a potential well, by comparing the excess fractional probability within certain regions of the domain to the height of the barrier or the depth of the well.

Problem 2

Background

In this problem we solve the 2D Schrödinger equation using the alternating direction implicit (ADI) method. The equation we are solving is

$$\psi_t = i\left(\psi_{xx} + \psi_{yy}\right) - iV(x,y)\psi, \qquad 0 \le x \le 1, \quad 0 \le y \le 1, \quad 0 \le t \le t_{\max}, \tag{18}$$

subject to the boundary conditions

$$\psi(x,y,0) = \psi_0(x,y) \tag{19}$$

$$\psi(0,y,t) = \psi(1,y,t) = \psi(x,0,t) = \psi(x,1,t) = 0. \tag{20}$$

Again, there is a family of exact solutions,

$$\psi(x,y,t) = e^{-i(m_x^2+m_y^2)\pi^2 t}\sin(m_x\pi x)\sin(m_y\pi y), \tag{21}$$

where m_x, m_y are integers.

Implementation

Our implementation is completed in the function `sch_2d_adi()`, which has the following header:

```
function [x y t psi psire psiim psimod v] = ...
sch_2d_adi(tmax, level, lambda, idtype, idpar, vtype, vpar)
% Inputs
%
% tmax: Maximum integration time
% level: Discretization level
% lambda: dt/dx
% idtype: Selects initial data type
% idpar: Vector of initial data parameters
% vtype: Selects potential type
% vpar: Vector of potential parameters
%
% Outputs
%
% x: Vector of x coordinates [nx]
% y: Vector of y coordinates [ny]
% t: Vector of t coordinates [nt]
% psi: Array of computed psi values [nt x nx x ny]
% psire Array of computed psi_re values [nt x nx x ny]
% psiim Array of computed psi_im values [nt x nx x ny]
% psimod Array of computed sqrt(psi psi*) values [nt x nx x ny]
% v Array of potential values [nx x ny]
```

Discretization

We will discretize the domain to a level l, with the time and spatial spacings defined by

$$\lambda = \frac{\Delta t}{\Delta x} = \frac{\Delta t}{\Delta y}. \tag{22}$$

From this we have the standard discretization,

$$n_x = n_y = 2^l + 1 \tag{23}$$

$$\Delta x = \Delta y = 2^{-l} \tag{24}$$

$$\Delta t = \lambda \Delta x = \lambda \Delta y \tag{25}$$

$$n_t = \text{round}(t_{max}/\Delta t) + 1. \tag{26}$$

We will also need the difference operators ∂^h_{xx} and ∂^h_{yy},

$$\partial^h_{xx}\psi^n_{i,j} \equiv \frac{\psi^n_{i+1,j} - 2\psi^n_{i,j} + \psi^n_{i-1,j}}{\Delta x^2}, \tag{27}$$

$$\partial^h_{yy}\psi^n_{i,j} \equiv \frac{\psi^n_{i,j+1} - 2\psi^n_{i,j} + \psi^n_{i,j-1}}{\Delta y^2}. \tag{28}$$

Alternating direction implicit method discretization

Discretizing (18) we obtain

$$\left(1 - i\frac{\Delta t}{2}\partial^h_{xx}\right)\psi^{n+\frac{1}{2}}_{i,j} = \left(1 + i\frac{\Delta t}{2}\partial^h_{xx}\right)\left(1 + i\frac{\Delta t}{2}\partial^h_{yy} - i\frac{\Delta t}{2}V_{i,j}\right)\psi^n_{i,j} \tag{29}$$

for the first step, and

$$\left(1 - i\frac{\Delta t}{2}\partial^h_{yy} + i\frac{\Delta t}{2}V_{i,j}\right)\psi^{n+1}_{i,j} = \psi^{n+\frac{1}{2}}_{i,j} \tag{30}$$

for the second step, where both (29) and (30) have i, j, n with ranges

$$i = 2, 3, 4, ..., n_x - 1, \qquad j = 2, 3, 4, ..., n_y - 1 \qquad n = 1, 2, 3, ..., n_t - 1.$$

The boundary and initial conditions are discretized as follows:

$$\psi^1_{i,j} = \psi_0(x_i, y_j), \tag{31}$$

$$\psi^n_{1,j} = \psi^n_{n_x,j} = \psi^n_{i,1} = \psi^n_{i,n_y} = 0 \tag{32}$$

Now, let's walk through how we will perform a full ADI method step.

The idea is to first solve for the intermediate step, $\psi^{n+\frac{1}{2}}_{i,j}$, from (29), which actually amounts to solving many 1D CN schemes, one for each j. This is step 1 described below. Once $\psi^{n+\frac{1}{2}}_{i,j}$ is obtained, the next step is to solve (30), again in 1D, this time by looping through values of i.

Step 1

As explained above, our first step is to calculate $\psi^{n+\frac{1}{2}}_{i,j}$ from (29) by looping through values of j. This is called the first ADI step. To do this, for each j we will transform the equation into a similar form as in problem 1, where we will have

$$\underline{\underline{A}}\psi^{n+\frac{1}{2}}_{i,j} = F^n_{i,j}. \tag{33}$$

Let's begin by determining $\underline{\underline{A}}$. Considering the left hand side of (29) we have

$$\begin{aligned}
\left(1 - i\frac{\Delta t}{2}\partial^h_{xx}\right)\psi^{n+\frac{1}{2}}_{i,j} &= \psi^{n+\frac{1}{2}}_{i,j} - i\frac{\Delta t}{2}\partial^h_{xx}\psi^{n+\frac{1}{2}}_{i,j} \\
&= \psi^{n+\frac{1}{2}}_{i,j} - i\frac{\Delta t}{2}\left(\frac{\psi^{n+\frac{1}{2}}_{i+1,j} - 2\psi^{n+\frac{1}{2}}_{i,j} + \psi^{n+\frac{1}{2}}_{i-1,j}}{\Delta x^2}\right) \\
&= \left(-i\frac{\Delta t}{2\Delta x^2}\right)\psi^{n+\frac{1}{2}}_{i+1,j} + \left(1 + i\frac{\Delta t}{\Delta x^2}\right)\psi^{n+\frac{1}{2}}_{i,j} + \left(-i\frac{\Delta t}{2\Delta x^2}\right)\psi^{n+\frac{1}{2}}_{i-1,j} \\
&= C_+\psi^{n+\frac{1}{2}}_{i+1,j} + C_0\psi^{n+\frac{1}{2}}_{i,j} + C_-\psi^{n+\frac{1}{2}}_{i-1,j}
\end{aligned}$$

Where the constants C_+, C_0, C_- are

$$C_+ = -i\frac{\Delta t}{2\Delta x^2} \tag{34}$$

$$C_0 = 1 + i\frac{\Delta t}{\Delta x^2} \tag{35}$$

$$C_- = C_+ \tag{36}$$

This gives us $\underline{\underline{A}}$, with the same form as (12), back in Problem 1, and so $\underline{\underline{A}}$ can once again be turned into a tridiagonal matrix. We've now simplified the left hand side into $\underline{\underline{A}}\psi_{i,j}^{n+\frac{1}{2}}$. For simplifying the right hand side, we will define a temporary matrix $\underline{\underline{G}}_{i,j}^n$ as follows.

$$\begin{aligned}\underline{\underline{A}}\psi_{i,j}^{n+\frac{1}{2}} &= \left(1 + i\frac{\Delta t}{2}\partial_{xx}^h\right)\left(1 + i\frac{\Delta t}{2}\partial_{yy}^h - i\frac{\Delta t}{2}V_{i,j}\right)\psi_{i,j}^n \\ &= \left(1 + i\frac{\Delta t}{2}\partial_{xx}^h\right)\underline{\underline{G}}_{i,j}^n,\end{aligned}$$

where

$$\begin{aligned}\underline{\underline{G}}_{i,j}^n &= \left(1 + i\frac{\Delta t}{2}\partial_{yy}^h - i\frac{\Delta t}{2}V_{i,j}\right)\psi_{i,j}^n \\ &= \psi_{i,j}^n + i\frac{\Delta t}{2}\partial_{yy}^h\psi_{i,j}^n - i\frac{\Delta t}{2}V_{i,j}\psi_{i,j}^n \\ &= \psi_{i,j}^n + i\frac{\Delta t}{2}\left(\frac{\psi_{i,j+1}^n - 2\psi_{i,j}^n + \psi_{i,j-1}^n}{\Delta y^2}\right) - i\frac{\Delta t}{2}V_{i,j}\psi_{i,j}^n \\ &= \left(i\frac{\Delta t}{2\Delta y^2}\right)\psi_{i,j+1}^n + \left(1 - i\frac{\Delta t}{\Delta y^2} - i\frac{\Delta t}{2}V_{i,j}\right)\psi_{i,j}^n + \left(i\frac{\Delta t}{2\Delta y^2}\right)\psi_{i,j-1}^n \\ &= G_-\psi_{i,j+1}^n + G_0\psi_{i,j}^n + G_+\psi_{i,j-1}^n\end{aligned}$$

The coefficients G_-, G_0, G_+ are used to simplify the implementation of $\underline{\underline{G}}_{i,j}^n$ and are

$$G_+ = i\frac{\Delta t}{2\Delta y^2} \tag{37}$$

$$G_0 = 1 - i\frac{\Delta t}{\Delta y^2} - i\frac{\Delta t}{2}V_{i,j} \tag{38}$$

$$G_- = G_+ \tag{39}$$

This $\underline{\underline{G}}$ matrix will need to be computed with each iteration timestep n.

Now, we continue simplifying, absorbing the leftmost bracketed term of the right hand side of (29) as follows:

$$\begin{aligned}\underline{\underline{A}}\psi_{i,j}^{n+\frac{1}{2}} &= \left(1 + i\frac{\Delta t}{2}\partial_{xx}^h\right)\underline{\underline{G}}_{i,j}^n \\ &= F_{i,j}^n\end{aligned}$$

It's important to understand although here we will calculate $\underline{\underline{F}}_{i,j}^n$, an `[nx x ny]` matrix, and will only use 1 row of it at a time in (33). But here we calculate the entire matrix, before entering the ADI step. Note that we drop the double underline notation on $G_{i,j}^n$ to simplify notation.

$$\begin{aligned}\underline{\underline{F}}_{i,j}^n &= \left(1 + i\frac{\Delta t}{2}\partial_{xx}^h\right)G_{i,j}^n \\ &= G_{i,j}^n + i\frac{\Delta t}{2}\partial_{xx}^h G_{i,j}^n \\ &= G_{i,j}^n + i\frac{\Delta t}{2}\left(\frac{G_{i+1,j}^n - 2G_{i,j}^n + G_{i-1,j}^n}{\Delta x^2}\right) \\ &= \left(i\frac{\Delta t}{2\Delta x^2}\right)G_{i+1,j}^n + \left(1 - i\frac{\Delta t}{\Delta x^2}\right)G_{i,j}^n + \left(i\frac{\Delta t}{2\Delta x^2}\right)G_{i-1,j}^n \\ &= F_-G_{i+1,j}^n + F_0G_{i,j}^n + F_+G_{i-1,j}^n\end{aligned}$$

We are once again using coefficients to simplify the implementation. These are defined as

$$F_+ = i\frac{\Delta t}{2\Delta x^2} \tag{40}$$

$$F_0 = 1 - i\frac{\Delta t}{\Delta x^2} \tag{41}$$

$$F_- = F_+ \tag{42}$$

Now are finally done simplifying the second step. The result is (33), and is solved via left division:

$$\psi_{i,j}^{n+\frac{1}{2}} = \underline{\underline{A}} \setminus F_{i,j}^n \tag{43}$$

$\underline{\underline{A}}$ and $\underline{\underline{F}}_{i,j}^n$ can be computed only once, but (43) will need to be solved repeatedly for $j = 2, 3, 4, ..., n_y - 1$ by taking columns of $\underline{\underline{F}}_{i,j}^n$. Once this is done (for one timestep), we move to step 2 for this timestep, which involves solving (30).

Step 2

After completing step 1 we now have the right hand side of (30), $\psi_{i,j}^{n+\frac{1}{2}}$. Let's organize the right hand side into a new matrix $\underline{\underline{B}}$ multiplied by $\psi_{i,j}^{n+1}$, and solve for $\psi_{i,j}^{n+1}$ via left division. Note that in this second ADI step, we will need to iterate through the rows i. Because $\underline{\underline{B}}$ has a dependence on the potential in its main diagonal, a new $\underline{\underline{B}}$ matrix will need to be formed with each iteration i. This is different from $\underline{\underline{A}}$ in step 1 where we only needed to form one matrix for all iterations through j. So, for some iteration i, from the right hand side we have:

$$\begin{aligned}
\left(1 - i\frac{\Delta t}{2}\partial_{yy}^h + i\frac{\Delta t}{2}V_{i,j}\right)\psi_{i,j}^{n+1} &= \psi_{i,j}^{n+1} - i\frac{\Delta t}{2}\partial_{yy}^h\psi_{i,j}^{n+1} + i\frac{\Delta t}{2}V_{i,j}\psi_{i,j}^{n+1} \\
&= \psi_{i,j}^{n+1} - i\frac{\Delta t}{2}\left(\frac{\psi_{i,j+1}^{n+1} - 2\psi_{i,j}^{n+1} + \psi_{i,j-1}^{n+1}}{\Delta y^2}\right) + i\frac{\Delta t}{2}V_{i,j}\psi_{i,j}^{n+1} \\
&= \left(-i\frac{\Delta t}{2\Delta y^2}\right)\psi_{i,j+1}^{n+1} + \left(1 + i\frac{\Delta t}{\Delta y^2} + i\frac{\Delta t}{2}V_{i,j}\right)\psi_{i,j}^{n+1} + \left(-i\frac{\Delta t}{2\Delta y^2}\right)\psi_{i,j-1}^{n+1} \\
&= B_+\psi_{i,j+1}^{n+1} + B_0\psi_{i,j}^{n+1} + B_-\psi_{i,j-1}^{n+1}
\end{aligned}$$

where

$$B_+ = -i\frac{\Delta t}{2\Delta y^2} \tag{44}$$

$$B_0 = 1 + i\frac{\Delta t}{\Delta y^2} + i\frac{\Delta t}{2}V_{i,j} \tag{45}$$

$$B_- = -i\frac{\Delta t}{2\Delta y^2} \tag{46}$$

$$\tag{47}$$

Again, we can write $\underline{\underline{B}}$ in the same tridiagonal form as $\underline{\underline{A}}$ in (12), with the coefficients B_+, B_0, B_- instead of C_+, C_0, C_-. In the end we have

$$\underline{\underline{B}}\psi_{i,j}^{n+1} = \psi_{i,j}^{n+\frac{1}{2}} \tag{48}$$

which we solve via left division for each i to obtain

$$\psi_{i,j}^{n+1} = \underline{\underline{B}} \setminus \psi_{i,j}^{n+\frac{1}{2}} \tag{49}$$

This second step is repeated across all is, i.e. for $i = 2, 3, 4, ..., n_x - 1$.

Now that we've discussed the two ADI steps, let's look at the initial data types and potential types that could be selected.

Initial data types

Again, we select from two different initial condition types using the inputted `idtype` and `idpar` variables. These can be the following:

- `idtype == 0`: Exact family
$$\psi(x, y, 0) = \sin(m_x \pi x) \sin(m_y \pi y)$$
`idpar(1), idpar(2)`: m_x, m_y
- `idtype == 1`: Boosted Gaussian
$$\psi(x, y, 0) = e^{ip_x x} e^{ip_y y} e^{-\left((x-x_0)^2/\delta_x^2 + (y-y_0)^2/\delta_y^2\right)}$$
`idpar(1), idpar(2), idpar(3), idpar(4), idpar(5), idpar(6)`: $x_0, y_0, \delta_x, \delta_y, p_x, p_y$

I implemented these in an identical fashion to Problem 1. One interesting note to make that is specific to Problem 2, is that for `idtype=1`, I compute the Gaussian using `x` and the transpose of `y`, so that the result is a `nx x ny` grid, which gets assigned to `psi(1,:,:)`.

Potential types

In addition to the two types of potential from Problem 1, we also have a slit. Again, the potential is selected using the `vtype` input variable, with its parameters specified via `vpar`, as follows:

- `vtype == 0`: No potential
$$V(x) = 0$$
- `vtype == 1`: Rectangular barrier or well
$$V(x, y) = \begin{cases} V_c & \text{for} \quad (x_{\min} \leq x \leq x_{\max}) \quad \text{and} \quad (y_{\min} \leq y \leq y_{\max}) \\ 0 & \text{otherwise} \end{cases}$$
`vpar(1),vpar(2),vpar(3),vpar(4),vpar(5)`: $x_{\min}, x_{\max}, y_{\min}, y_{\max}, V_c$
- `vtype == 2`: Double slit. Take $j' = \frac{1}{4}(n_y - 1) + 1$.
$$V_{i,j'} = V_{i,j'+1} = 0 \quad \text{for } [(x_1 \leq x_i) \text{ and } (x_i \leq x_2)] \text{ or } [(x_3 \leq x_i) \text{ and } (x_i \leq x_4)]$$
$$V_{i,j'} = V_{i,j'+1} = V_c \quad \text{otherwise}$$
$$V_{i,j} = 0 \text{ for } j \neq (j', j'+1)$$
`vpar(1),vpar(2),vpar(3),vpar(4),vpar(5)`: x_1, x_2, x_3, x_4, V_c

A few comments about the implementation of these:
The potential matrix was initialized via `v = zeros(nx, ny)`, and `if` statements were used to modify it if `vtype` was set to 1 or 2.
The rectangular barrier/wall potential was implemented using `v((x>xmin & x<xmax)& (y_T>ymin & y_T<ymax))= Vc`, where `y_T` is the transpose of the vector of y gridpoints.
The double slit was implemented by first setting the two columns j' and $j'+1$ to have a value of V_c using `v(:,j_slit0:j_slit1)= Vc`, and then cutting slits in them using `v(((x1<=x)& x<=x2)| ((x3<=x)& (x<=x4)),:)= 0`.
And finally, a couple of interesting details regarding the implementation in general:

- The boundary conditions were reinforced between the steps, i.e. on $\psi_{i,j}^{n+\frac{1}{2}}$.
- For $\underline{\underline{B}}$, the two non-main diagonals were initialized as `[1 x ny]` vectors, but the main diagonal was initialized as a `nx x ny` array, and `1 x ny` vectors were taken from it as we iterated over i.

- The norm of the entire `[nx x ny]` solution matrix at each timestep was plotted against time to check for conservation of probability, as a quick way of verifying the implementation. This was done by calculating `probability(n)= norm(squeeze(psi(n,:,:)));` for every `n`. The result is that it is indeed conserved. I included a plot showing this result below:

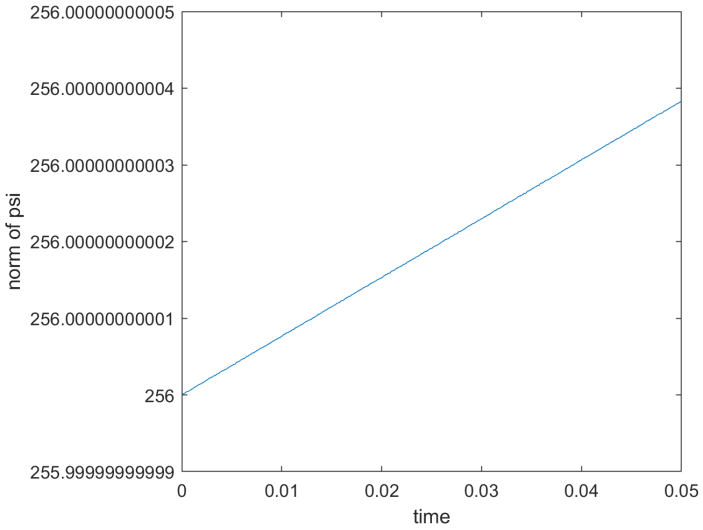

Figure 6: Test of the conservation of probability with time, obtained by computing `probability(n)= norm (squeeze(psi(n,:,:)));` for every `n`. Probability is conserved to ∼`1e-10`.

Analysis

Convergence testing

The system was tested for convergence in a manner almost identical to Problem 1. More specifically, the same two-level spacial deviation norms, $||d\psi^l||_2(t^n) = ||\psi^{l+1} - \psi^l||_2(t^n)$, were computed, though this time in 2D. Accounting for the extra dimensions was done as follows, making use of MATLAB's built-in `mean()` function:

```
rms1 = sqrt(mean(dpsi0mod.^2,2));
dpsi0rms = sqrt(mean(rms1.^2,3));
```

Here we first take the l-2 norm of `dspi0mod`$= |d\psi^l|(t^n)$ in the x-direction, collapsing the 2nd dimension, and then in the y-direction, collapsing the 3rd. Similarly the deviation from the exact solution was computed as $||E(\psi^l)||_2(t^n) = ||\psi_{\text{exact}} - \psi^l||_2(t^n)$, where the exact solution is given in (21).
The convergence test was performed on the following parameters:
`idtype = 0`, `vtype = 0`

```
idpar  = [2, 3];
vpar   = 0;
tmax   = 0.05;
lambda = 0.05;
lmin   = 6;
lmax   = 9;
```

It is implemented in `ctest_2d.m`. This script calls on `ctest_2d_func()` to determine the two-level spacial deviation norms, and on `ctest_2d_func_err()` to compute the errors $||E(\psi^l)||_2(t^n)$. `ctest_2d_func_err()` calls on `sch_2d_exact(x, y, t, mx, my)` which computes the exact solution to the 2D Schrödinger equation as given by (21).

Using this script, plots of rescaled $||d\psi^l||_2(t^n)$ and $||E(\psi^l)||_2(t^n)$ evolutions were made. The plots with levels $6, 7, 8$ were rescaled by $1, 4, 4^2$ respectively, as is expected of an $\mathcal{O}(h^2)$ solution. The resulting plots are shown below:

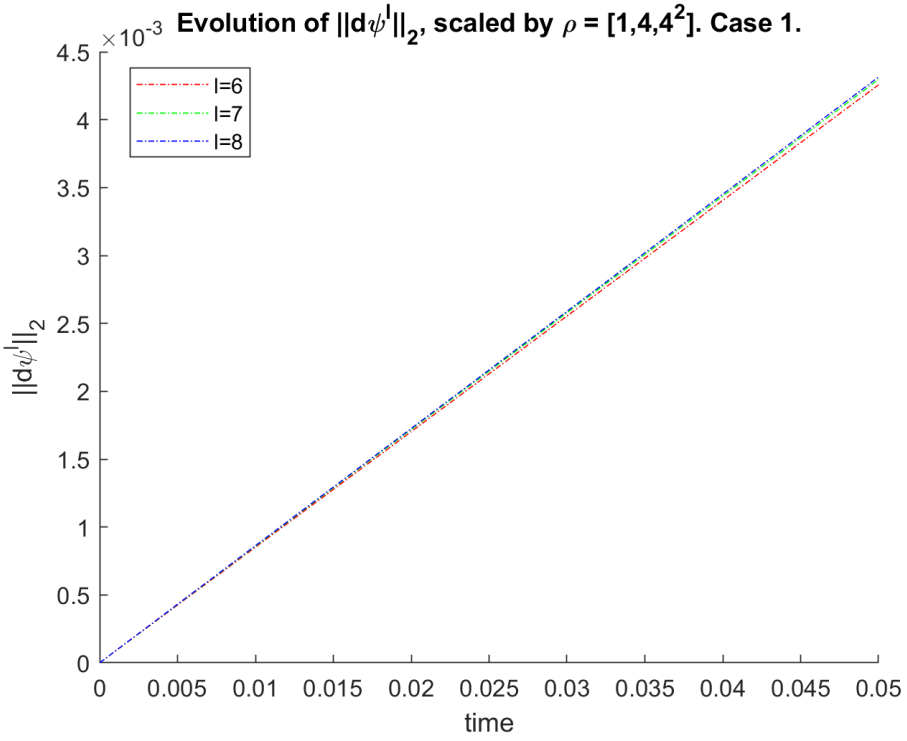

Figure 7: Plot of the evolution of rescaled $||d\psi^l||_2(t^n)$ with time for the 2D Schrödinger equation solved via the ADI scheme, with `idtype=0`. As with the 1D Crank-Nicolson scheme, we can see that $||d\psi^6||_2(t^n)$, $4||d\psi^7||_2(t^n)$, and $4^2||d\psi^8||_2(t^n)$ align, indicating that the solution is indeed $\mathcal{O}(h^2)$ accurate.

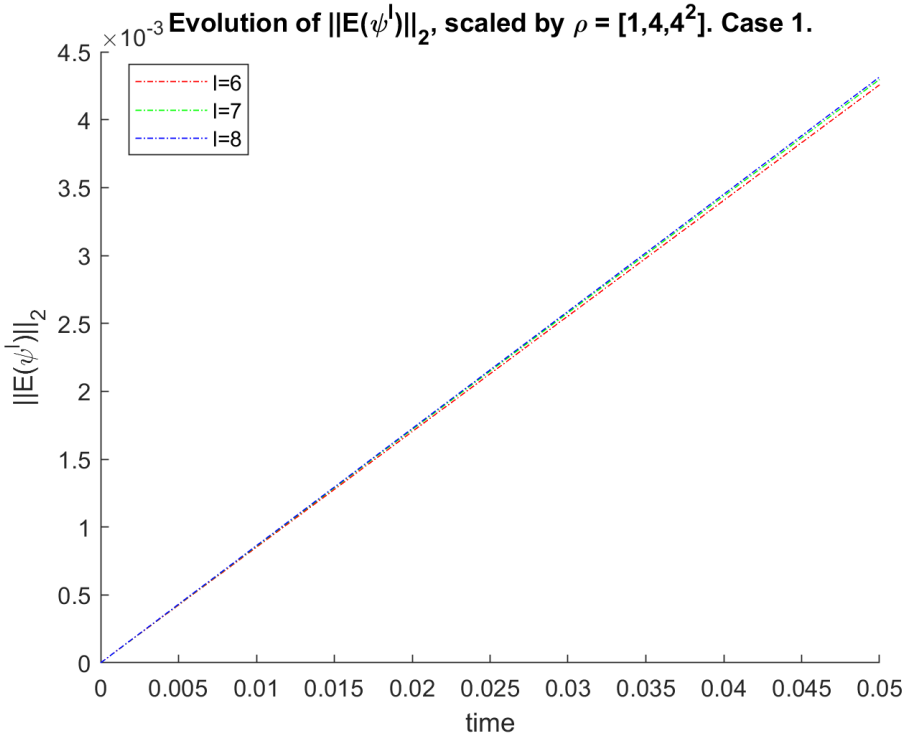

Figure 8: Plot of the evolution of rescaled $||E(\psi^l)||_2(t^n)$ with time for solutions to the 2D Schrödinger equation with `idtype=0`. Again, we can see that $||E(\psi^6)||_2(t^n)$, $4||E(\psi^7)||_2(t^n)$, and $4^2||E(\psi^8)||_2(t^n)$ align, indicating that the solution is indeed $\mathcal{O}(h^2)$ accurate.

Having verified that the solution is indeed $\mathcal{O}h^2$, I could proceed to creating animations of the solution using 2D and 3D plotting techniques.

Numerical Experiments

In this section, we create plots to visualize the evolution of the solutions to the 2D Schrödinger equation for various initial condition and potential types. All of the plots and animations are generated using the `numerical_experiments.m` script.

Potentials

The natural place to start was to plot each of the non-trivial potential cases, `vtype=1,2`. This is done using `plot_2d_potential()`. The two potentials are plotted below, with the parameters shown in the plot, at at level $l = 6$.

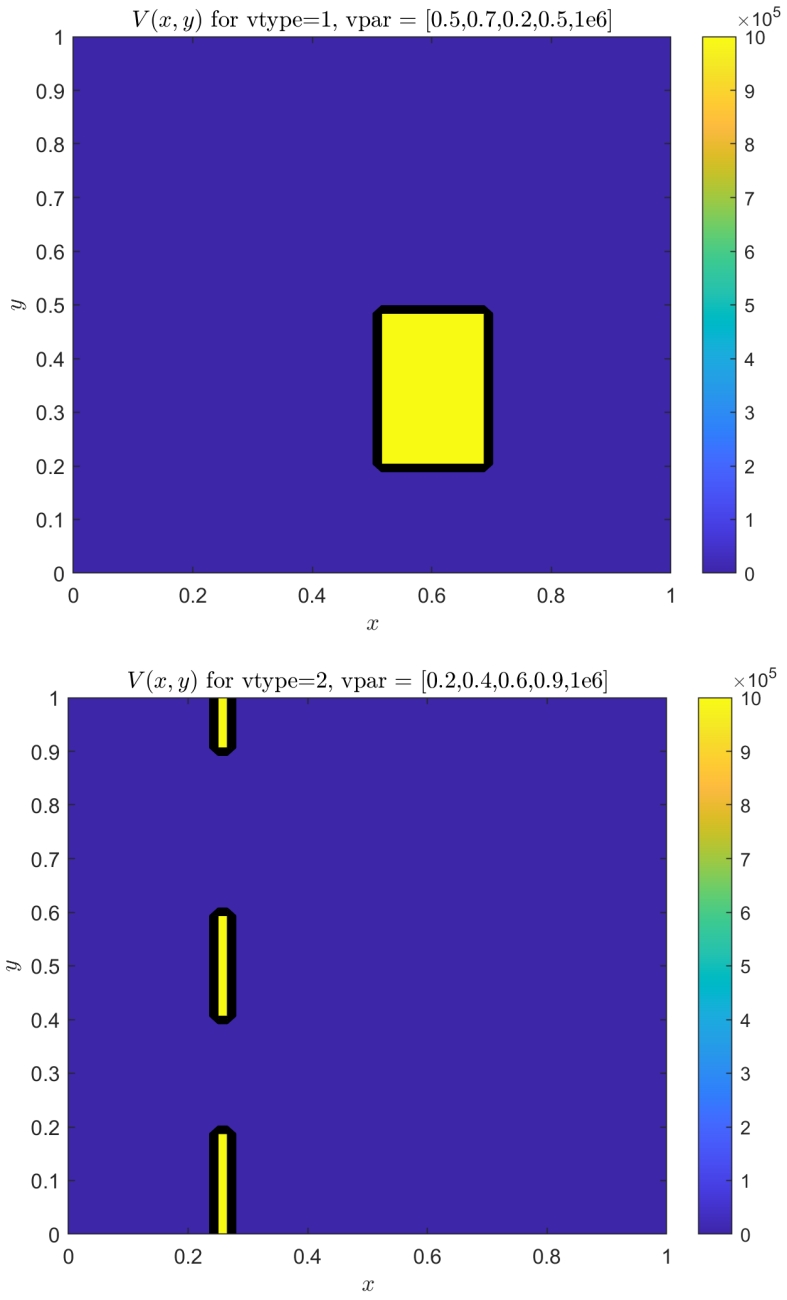

Figure 9: Left: rectangular barrier. Right: double slit.

We can see that the barrier and double slit is placed in the correct desired location. With this, we can proceed to animating the solutions.

Case 1—Free particle sinusoidal

The following parameters were used:

```
idtype = 0;
vtype  = 0;
idpar  = [2, 3];
vpar   = 0;
```

```
tmax   = 0.05;
lambda = 0.05;
level = 7;
```

An animation of the real part of the solution using these parameters was created by calling `plot_2d_psire(x,y,t,psire,case_name)`. This is the file called `Case 1.avi`. The solution oscillates between maxima and minima of the 6 separate sections—2 modes for x and 3 modes for y.

Case 2—Free particle Gaussian

The following parameters were used:

```
idtype = 1;
vtype  = 0;
idpar  = [0.5,0.5,0.075,0.075,0,0];
vpar   = 0;
tmax   = 0.05;
lambda = 0.05;
level = 9;
```

Next, an animation of a Gaussian with no potential was created. This is the file called `Case 2.avi`. We see the Gaussian disperse from its central position, and then create interesting symmetrical shapes due to the symmetry of the initial condition and the cube.

With these simple examples completed, we are now ready to move to more complex simulations with scattering from a barrier/well/double slit. All of the cases that follow will use the Gaussian initial condition.

Case 3—Scattering off a rectangular barrier

The following parameters were used:

```
idtype = 1;
vtype  = 1;
idpar  = [0.8,0.5,0.050,0.050,-25,0];
vpar   = [0.2,0.4,0.4,0.6,1e15];
tmax   = 0.015;
lambda = 0.02;
level = 9;
```

The idea was to generate a Gaussian waveform to the right of a barrier—a 0.2×0.2 pillar. The Gaussian waveform is given a boost in the $-x$ direction, so that it moves left towards the pillar. Another function, `plot_2d_psire_potential(x,y,t,psire,case_name,vpar)`, was used to create a contour of the barrier on top of the real part of the solution. This was used to generate a video called `Case 3.avi`. The waveform is seen to hit the barrier and scatter away from it. A small, low amplitude remnant is also seen to oscillate within the barrier itself.

Case 4—Gaussian particle inside well potential

Next, I wanted to investigate the wall potential. Specifically, I wanted to observe how a Gaussian would behave when constrained inside a 0.40×0.40 box with a potential 1e6 times smaller than the rest of the box. The following parameters were used to implement this:

```
idtype = 1;
vtype  = 1;
idpar  = [0.65,0.5,0.075,0.075,0,0];
vpar   = [0.45,0.85,0.25,0.75,-1e6];
tmax   = 0.0075;
lambda = 0.01;
level = 9;
```

The video of this case is called `Case 4.avi`. The Gaussian starts centered at (0.65,0.5), and breaks into several peaks which travel to and bounce off the edges of the smaller box. Small remnants are observed as ripples outside of the box.

Case 5—Scattering off a rectangular well

This time, I switched to a 3D plotting method instead, by adding using MATLAB's `surf()` function in additon to `contourf()`. I created a new function for 3D plotting, `plot_3d_psire(x,y,t,psire,case_name)`. I used this to demonstrate the scattering of a particle with Gaussian initial conditions against a rectangular well. More specifically, the particle starts off to the left (which is closer to us in the 3D view) with an initial velocity in the positive x direction, which guides it directly into a rectangular well. The following parameters are used to describe this:

```
idtype = 1;
vtype  = 1;
idpar  = [0.2,0.5,0.075,0.075,20,0];
vpar   = [0.45,0.85,0.25,0.75,-1e4];
tmax   = 0.015;
lambda = 0.005;
level = 8;
```

The resulting video, `Case 5.avi`, shows that upon hitting the leftmost edge of the well, the particle's Gaussian breaks into two peaks, as well as many smaller amplitude hills which propagate within the well itself to the other side, reconvening with the two large peaks at the rightmost edge of the box towards $t \sim 0.0145$. We therefore see that the initial rightward momentum given to the particle was enough for it to travel through the well without getting trapped in it, but slow enough to produce some interesting scattering effects.

Case 6—Scattering through a double slit

Finally, the double slit potential `vtype=2` was used. The particle, with a Gaussian waveform, was initialized to the left of a double slit with a positive x momentum so it moves towards it. The double slit had two 0.15 openings positions 0.1 apart at the centre, as shown below:

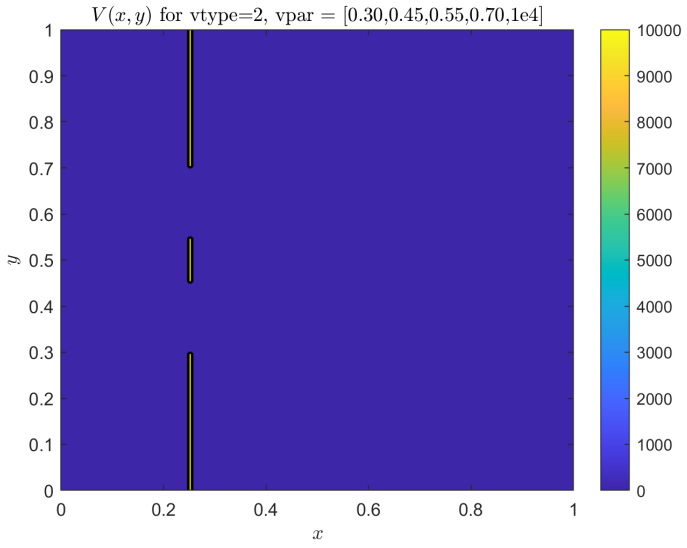

Figure 10: The double slit potential for Case 6. $V =$`1e4` on the slitted boundary. The Gaussian was initialized as centered at $(x, y) = (0.1,0.5)$.

The following parameters were used:

```
idtype = 1;
```

```
vtype  = 2;
idpar  = [0.1,0.5,0.075,0.075,40,0];
vpar   = [0.30,0.45,0.55,0.70,1e6];
tmax   = 0.02;
lambda = 0.005;
level = 8;
```

The video `Case 6.avi` shows the evolution. We see the Gaussian move towards the slit and hit it, at which point some of the wavefunction bounces back to the left and the other part propagates through the slits. A dispersing wave is first produced on the other side, very similar to the uniform ripples produced in water when an object is dropped in. The wavefunction eventually forms interference patterns on the right end of the square, which are clearly visible past $t \sim 0.011$. These interference patterns, or fringes, oscillate from 2 up to 5 peaks lined up along the right edge of the cube at any particular time.

Conclusion

In this problem, we solved the 2D time dependent Schrödinger equation on a square. We did so using the alternating direction implicit (ADI) method, which involved two steps. In the first step, we iterated through the columns and determined an intermediate step solution. We then iterated through the rows to obtain the solution at the next timestep. A convergence test was performed, and the solution was found to be $\mathcal{O}(h^2)$ accurate. Finally, the real part of the solution was used to create visualizations of its evolution over time, in 2D and 3D plots/videos. We explored the simple free sinusoidal particle case, followed by a free particle with a Gaussian waveform. Potentials were then added and scattering effects were explored: A particle with a Gaussian waveform was sent towards a pillar with high potential, off of which it scattered. The same was done with the particle travelling towards a potential square well instead. The evolution of a Gaussian created inside of a small potential well was also observed. Finally, a Gaussian waveform was sent through a double slit, producing an interference pattern on the other end of the square.