Physics 410: Project 2

Drew Spencer

November 2019

1 Introduction

In this project, we will be creating simulations and analyzing the 1-D and 2-D Schrödinger wave equations. First I will be working with the 1-D wave equation using the Crank-Nicolson Scheme to discretize our equation. After the discretization of the 1-D equation, and the set up of a tri-diagonal system, an update system can be established that will allow of wave to propagate. In the 2-D scenario, I use the Alternating Direction Implicit (ADI) Method as opposed to the Crank-Nicolson Scheme for the 1-D equation. In this case, there is two tri-diagoinal systems set up. This is to allow the update across every x component value for each y value then after this updating each y component value across all x. All equations will be non-dimensionalized and during the calculation process we did not need to normalize our wave functions until later in the plotting sequences.

2 The 1-D Schrödinger Wave Equation

For the 1-D equation, we were given a set of test inputs that allowed us to begin . To start, we initialized many parameters through the inputs given, the main parameters were:

- $\lambda = \frac{\Delta t}{\Delta x}$
- $n_x = 2^l + 1$
- $\Delta x = 2^{-l}$
- $\Delta t = \lambda \Delta x$
- $n_t = \text{round}(t_{max}/\Delta t) + 1$
- $\psi = \text{zeros(nt, nx)}$

Here, tmax, and l (level input) are inputs that can be easily changed by the user, nt and nx were used to set up the matrices for the ψ to make a n_t x n_x matrix. We were then given the equation for the "family" of exact solution to the non-dimentionalized continuum equation. The family of solutions equation is:

$$\psi(x,t) = \sin(m\pi x) \tag{1}$$

Here m is a positive integer.

2.1 Crank-Nicolson Scheme

Once this was set up, the Crank-Nicolson scheme was set up. The equation given is shown below:

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

This equation was solved by isolating all the ψ^{n+1} terms (all indexes of j, j+1, j-1) onto the LHS of the equation then the right hand side was defined as f. The resulting solution took this form:

$$\left(\frac{i}{\Delta t}-\frac{1}{\Delta x^2}-\frac{1}{2}V_j^{n+\frac{1}{2}}\right)\psi_j^{n+1}+\left(\frac{i}{\Delta t}+\frac{1}{2\Delta x^2}\right)\left(\psi_{j+1}^{n+1}+\psi_{j-1}^{n+1}\right)$$
$$=\left(\frac{i}{\Delta t}+\frac{1}{\Delta x^2}+\frac{1}{2}V_j^{n+\frac{1}{2}}\right)\psi_j^n-\left(\frac{1}{2\Delta x^2}\right)\left(\psi_{j+1}^n+\psi_{j-1}^n\right) \tag{3}$$

On the LHS there is three different ψ terms, ψ_j^{n+1}, ψ_{j-1}^{n+1}, and ψ_{j+1}^{n+1}. The coefficients paired with these terms allowed for the set up of the tri-diagonal matrix $\mathbf{A}$ by using Matlab's spdiags function. A quick summary below shows the breakdown:

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

The three coefficients found here are the diagonals of the tri-diagonal matrix $\mathbf{A}$. C_0 is the main diagonal, C_+ is the diagonal above the main, and C_- is the diagonal below the main. In this system, C_+ and C_- happen to be the same term. Each term in each respective diagonal takes the value of of these coefficients.
Moving to the RHS of the equation defined as f.

$$f=\left(\frac{i}{\Delta t}+\frac{1}{\Delta x^2}+\frac{1}{2}V_j^{n+\frac{1}{2}}\right)\psi_j^n-\left(\frac{1}{2\Delta x^2}\right)\left(\psi_{j+1}^n+\psi_{j-1}^n\right) \tag{4}$$

This is the portion that iterates inside the loop to update each position. The first and last elements of f are held at zero, that is $f(0)=0$ and $f(nx)=0$. Thus the elements $f(2:nx-1)$ are all updated with over each iteration in time. After each iteration, the tri-diagonal matrix $\mathbf{A}$ is left divided by f which updates the next position of ψ. To summarize what happens:

$$\psi_j^{n+1}=\mathbf{A}\backslash f \tag{5}$$

Once all of this was set up, there was the matter of taking into account the idtype and vtype for the convergence testing.

2.2 Convergence Testing

Moving forward, the next parts involved doing some convergence testing. Before getting to that, there are two input parameters that are defined as idtype, and vtype. What these paramters do is define the type of wave and whether there is a potential or not. Idtype has two condition, idtype = 0 where the wave is just the exact solution or idtype = 1, a boosted gaussian.

What idtype = 0 does is give a wave function with no momentum, that is, its just a standing wave form of the exact solution. and Idtype 2 multiplies an exponential term with a potential term that gives the wave a velocity. This is sumamrized below

- idtype = 0 gives: $\psi(x,0) = sin(m\pi x)$
- idtype = 1 gives: $\psi(x,0)) = e^{-ipx}e^{(\frac{x-x_0}{\delta})^2}$

Moving on to vtype. Vtype was similar to idtpe in that there was two cases, vtype = 0 corresponded to no potential and vtype = 1 gives a potential, either a barrier (positive value) or a well (negative value). The potential would be defined between 2 values of x which, when the wave reached these positions, would cause some transmission. The extent of the transmission depended on the magnitude of the potential, the larger the value the more noticeable the interaction. Another summary listed below:

- vtype = 0 gives: V = 0
- vtype = 1 gives: $\mathrm{V} = \begin{cases} 0 & x \leq x_{min} \\ V_c & x_{min} \leq x \leq x_{max} \\ 0 & x \geq x_{max} \end{cases}$

Where V_c is another user input value.

Moving on to the convergence tests. We were asked to complete three sets of convergence testing. The first set was to have idtype = 0, just the exact solution, and vtype = 0 giving no pentential. What we would expect to see here is just an oscillating sine wave. which is exactly what happened. According to the math as well we would expect that with each increasing level, we would see convergence on the order $O(\Delta h^2)$. That is for each level increase we would expect that after scaling, level 7 would be multiplied by 2^2, level 8 by 4^2 and level 9 by 8^2 to line up with the level 6 graph.

Convergence Test for Psi

- idtype = 0
- vtype = 0
- idpar = [3]
- $t_{max} = 0.25$
- lambda = 0.1
- levels from 6 to 9

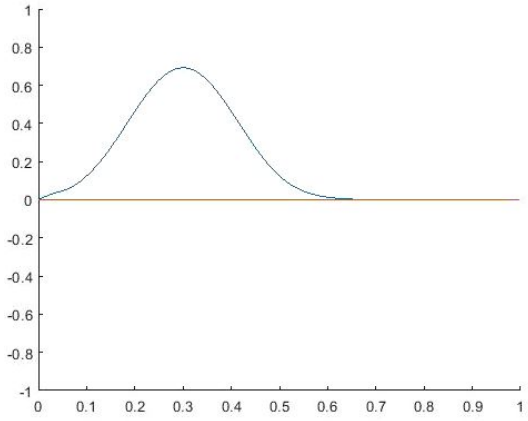
Using these inputs with the function I was able to produce the following graphs which are shown below. Before doing this, as each vector length increases with increasing level, I had to ensure that the lengths of each vector were consistent. for level 7, every second element was taken, for level 8 every 4th element was taken, and for level 9 every 8th element was taken.

2.2.1 Animation Tests

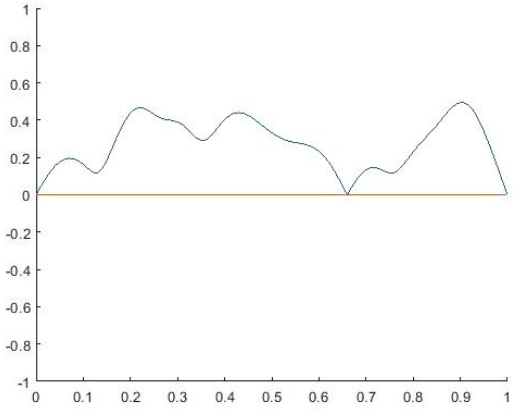
Before running the test, I created animations to try and see if the waves were doing what was expected.

The main reason for doing this was to get a physical interpretation as opposed to just seeing different arrays/vectors of numbers. It allowed for a better understanding if our code was producing what was expected. The following images are taken from our animations to show that what was happening was correct.

1-D With No Potential



Initial frame with no potential

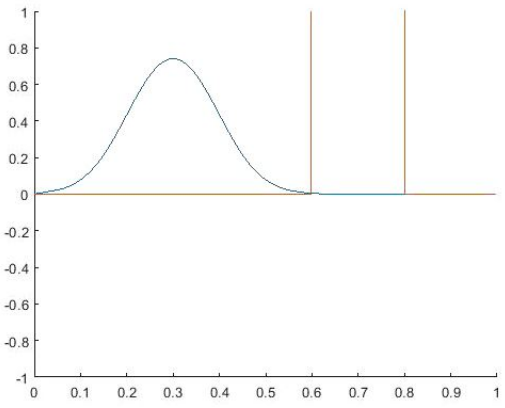


Approximately Halfway through the Animation

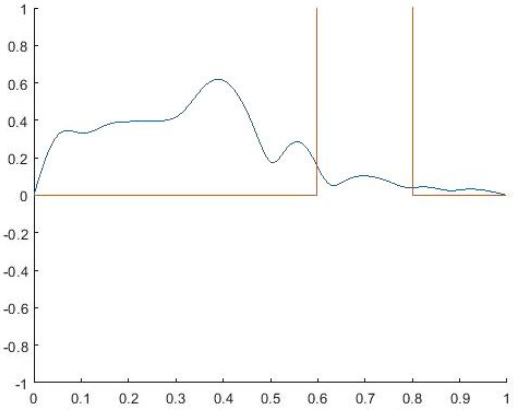
The following images were obtained using the 1-d function with the input listed below. The wave has no potential and has fairly consistent motion throughout.

[x t psi psire psiim psimod prob v] = sch_1d_cn(0.04, 8, 0.11, 1, [0.3, 0.075, 0], 0, []);

1-D With a Potential Barrier



Initial frame with Potential Barrier

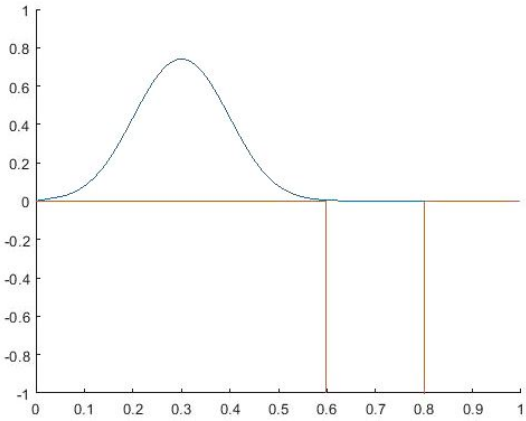


Approximately Halfway through the Animation

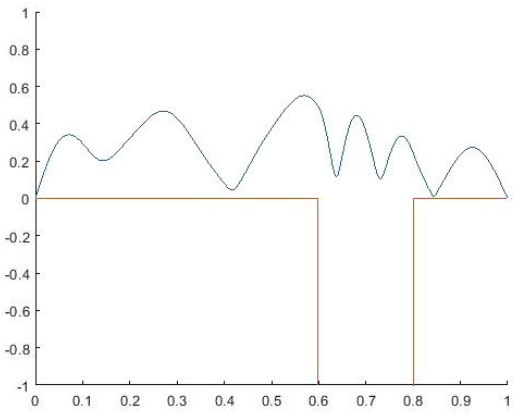
The following images were obtained using the 1-D function with the inputs listed below. As can be seen, with the potential at about 1000, much of the wave is reflected back, but there is some transmission of the wave through the well and over to the other side.

[x t psi psire psiim psimod prob v] = sch_1d_cn(0.04, 8, 0.11, 1, [0.3, 0.075, 0], 1, [0.6, 0.8, 1000]);

1-D With a Potential Well



Initial frame with Potential Well



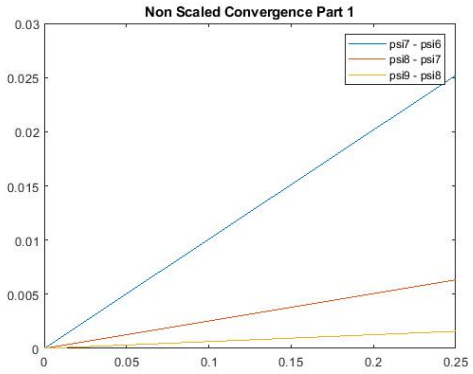
Approximately Halfway through the Animation

The following images were obtained using the 1-D function with the input listed below. Here we can see that some of the wave reflect from the well, some gets trapped inside the well, and some passes over.

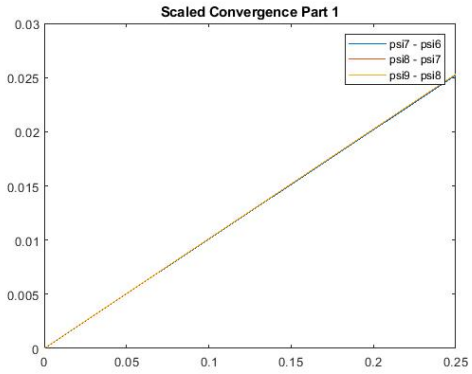
[x t psi psire psiim psimod prob v] = sch_1d_cn(0.04, 8, 0.11, 1, [0.3, 0.075, 0], 1, [0.6, 0.8, -1000]);

2.2.2 Convergence Tests

Testing Returned ψ



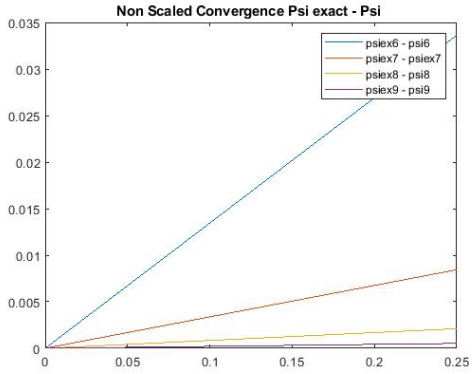
Error Difference between levels



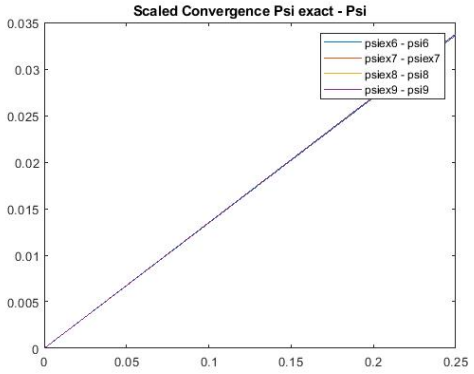
Scaled difference

These first two graphs are showing the scaled convergence test for ψ at the four different levels. This produces 3 tests overall, level7 - level6, level8 - level7, and level9 - level8. As it is shown on the left graph, each graphs error gets smaller with an increase in the level by a factor of $O(h^2)$. The scaled convergence test on the right shows that multiplying by the correct factor, the convergence test in satisfied. After multiplying $(\psi_8 - \psi_7) * 4$, and $(\psi_9 - \psi_8) * 16$ they both line up directly with $\psi_7 - \psi_6$

Testing ψ_{exact}



Error Difference between levels



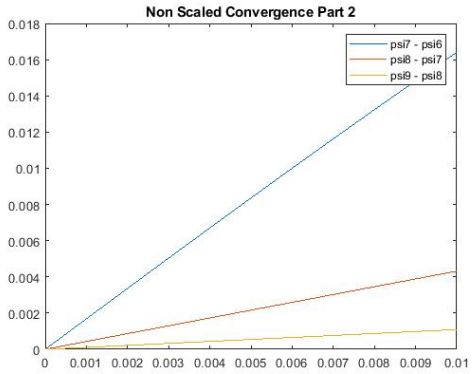
Scaled difference

Here we have the same type of testing being done here except for now we are testing our returned ψ at level 6, 7, 8, and 9 to the exact solution ψ_{exact} at the respective levels. Again the non scaled version is on the left and the scaled version is on the right. The graphs line up and the convergence test is satisfied. The following breakdown shows the scaling. $(\psi_{exact6} - \psi_6)$, $(\psi_{exact7} - \psi_7)$*4, $(\psi_{exact8} - \psi_8)$*16, $(\psi_{exact9} - \psi_9)$*64.

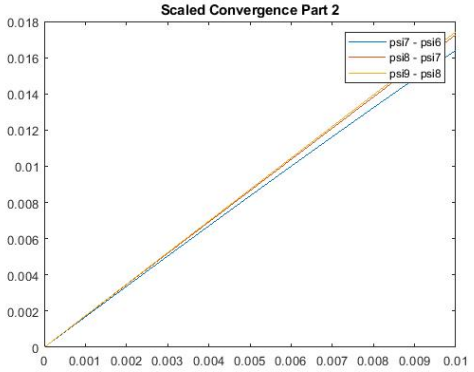
Testing Boosted Gaussian Wave

The boosted Gaussian wave has some momentum provided by an exponential term, the boosted Gaussian waveform is the following equation.

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



Error Difference between levels



Scaled difference

Again, using the exact same method as with the returned ψ method. The left graphs shows the difference in errors for different levels and the right graphs shows the convergence test being satisfied.

2.3 Numerical Testing

For this portion, we were asked to look at the temporal average that a particle will spend in a given space interval. The main difference with this part, is that, while we have a boosted Gaussian again, we now have a potential. In the first case, the potential is positive giving a positive potential which gives a potential "barrier" and the second case has a negative potential which gives a potential "well". In other words, vtype is now equal to 1.

To find the temporal average, we were given an equation to try and implement in our code, the equation was the following:

$$\bar{P}_j = \frac{\sum_{n=1}^{n_t} P_j^n}{n_t} \tag{6}$$

To do this, the summation over P_j^n could be done quickly using Matlab's mean function (mean(P_j^n) will give the value for the $\bar{P}$. The next portion was to ensure that this temporal average is normalized this was done by taking $\bar{P}$ and dividing by the temporal average by the last element in the vector. The normalized values now steadily increase from 0 at the first element to 1 at the last element. This shows normalization as when we have the full wave, there is a probability of 100% in finding the particle somewhere in that region of space. The following equation shows what is happening

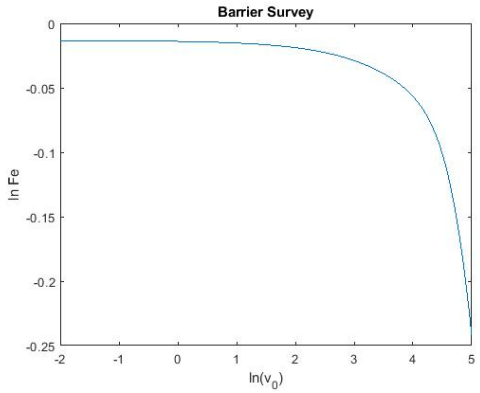
$$\bar{P}_j := \frac{\bar{P}}{\bar{P}_{n_x}}, \quad j = 1, 2, ..., n_x \tag{7}$$

Once this was done, we could then find the fractional probability that the particle was in a given section of space. The fractional probability value is the the amount of time the particle spends in that interval of space. The fractional probability is shown below.

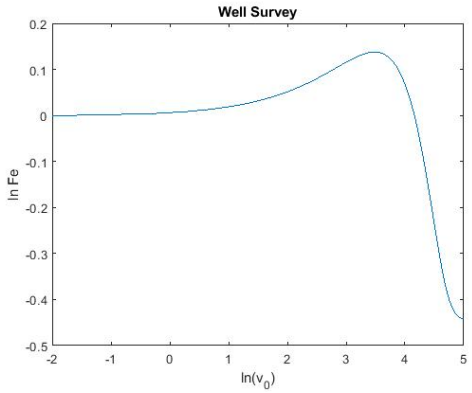
$$\ln \bar{F}_e(x_1, x_2) = \ln \frac{\bar{P}(x_2) - \bar{P}(x_1)}{x_2 - x_1} \tag{8}$$

For case one, we were to find the fractional probability between the space interval of $0.8 \leq x \leq 1$ while there was a potential barrier.

The second test implemented the same method but now the interval is $0.6 \leq x \leq 0.8$ with a potential well. The graphs displaying the findings are shown below:



Barrier Survey



Well Survey

The two graphs here both show the fractional probability (time) vs the potential value. As can be noted on both graphs, the higher the potential becomes, the less chance the particle will be found on the "other side" of the barrier or well. For the barrier (left graph) as the potential increases the chance for

the particle to transmit through the barrier goes to negative infinity, as $V_0 \to \infty$, $F_e \to 0$

For the Well survey, we can see that there is a critical point on the graph at about $ln(V_0) = 3.5$. Initially, if the well is weak it will hardly affect the particles/waves motion at all. As the potential gets larger (negatively, making the well "deeper") portions of the wave will get "trapped" inside the well, however once the potential becomes too large and the wave will reflect off the boundary.

3 The 2-D Schrödinger Wave Equation

To start this problem, we were to use the Alternating Direction Implicit Method (ADI) to discretize our system. The initial parameters at the start were essentially the same as the 1-D case, but now we have an extra spacial dimension, y, to be implemented. This method also involved two for loops (spacial updates) nested inside a for loop (temporal updates).

- $\lambda = \frac{\Delta t}{\Delta x}$
- $n_x = 2^l + 1$
- $n_y = 2^l + 1$
- $\Delta x = 2^{-l}$
- $\Delta t = \lambda \Delta x$
- $n_t = \text{round}(t_{max}/\Delta t) + 1$
- $\psi = \text{zeros}(ny, nx, nt)$
- $V = \text{zeros}(ny, nx)$

3.1 Alternating Direction Implicit Method (ADI)

The first steps to this problem involved discretizing the ADI method, the equations given to be solved were:

$$\left(1 - i\frac{\Delta t}{2}\partial_{xx}^h\right)\psi_{i,j}^{n+\frac{1}{2}} = \left(1 + i\frac{\Delta t}{2}\partial_{xx}^h\right)\left(1 + i\frac{\Delta t}{2}\partial_{yy}^h - i\frac{\Delta t}{2}V_{i,j}\right)\psi_{i,j}^n \quad i,j \tag{9}$$

$$\left(1 - i\frac{\Delta t}{2}\partial_{yy}^h + i\frac{\Delta t}{2}V_{i,j}\right)\psi_{i,j}^{n+1} = \psi_{i,j}^{n+\frac{1}{2}} \tag{10}$$

In both the above equations the indexes follow :

- i = 2, 3,..., n_x - 1
- j = 2, 3,..., n_y - 1
- n = 1, 2,..., n_t - 1

Through the breakdown of each equation, we were given FDA definitions for the operators on ψ to substitute. The following

To find the solution, we were shown in class that there a commutation rule that can be applied to the above equations to allow for a more "symmetric form". This commutation rule ultimately allowed for a much quicker solution to the scheme. The solution of the ADI scheme is shown below:

3.1.1 Solution of Part 1 of ADI

$$\left(1 + i\frac{\Delta t}{\Delta x^2}\right)\psi_{i,j}^{n+1/2} - \left(i\frac{\Delta t}{2\Delta x^2}\right)\left(\psi_{i+1,j}^{n+1/2} + \psi_{i-1,j}^{n+1/2}\right) = \left(1 - i\frac{\Delta t}{\Delta y^2} - i\frac{\Delta t}{2}V_{i,j}\right)\psi_{i,j}^n + \left(i\frac{\Delta t}{2\Delta y^2}\right)\left(\psi_{i,j+1}^n + \psi_{i,j-1}^n\right) \tag{11}$$

The solution above allowed for the set up of the first tri-diagonal system which followed the exact same steps as the 1-D method. The following coefficients were found:

- $C_0 = \left(1 + i\frac{\Delta t}{\Delta x^2}\right)$
- $C_- = -\left(i\frac{\Delta t}{2\Delta x^2}\right)$
- $C_+ = -\left(i\frac{\Delta t}{2\Delta x^2}\right)$

Again these were to set up the tri-diagonal matrix **A** with the right hand sides of the equation being set equal to f. This portion solves for $\psi^{n+1/2}$ which is a sub step before updating the actual position of ψ. Here I defined a "false" ψ called ψ_{fake} that took all these intermittent positions, which held j constant and summed over each i. These values were then used in the right hand side of part 2, which, for the sake of convenience, I will call f'.

3.1.2 Solution of Part 2 of ADI

$$\left(1 + i\frac{\Delta t}{\Delta y^2} + i\frac{\Delta t}{2}V_{i,j}\right)\psi_{i,j}^{n+1} - \left(i\frac{\Delta t}{2\Delta y^2}\right)\left(\psi_{i,j+1}^{n+1} + \psi_{i,j-1}^{n+1}\right) = \left(1 - i\frac{\Delta t}{\Delta x^2}\right)\psi_{i,j}^{n+1/2} + \left(i\frac{\Delta t}{2\Delta x^2}\right)\left(\psi_{i+1,j}^{n+1/2} + \psi_{i-1,j}^{n+1/2}\right) \quad (12)$$

The same method was applied to this part of the scheme, the coefficients will be labeled as B_0, B_+, and B_- to not confuse them with the C coefficients earlier. From the above equation the following coefficients were found to set up the second tri-diagonal matrix which I will call **B**

- $B_0 = \left(1 + i\frac{\Delta t}{\Delta y^2} + i\frac{\Delta t}{2}V_{i,j}\right)$
- $B_- = -\left(i\frac{\Delta t}{2\Delta y^2}\right)$
- $B_+ = -\left(i\frac{\Delta t}{2\Delta y^2}\right)$

The tri-diagonal matrix for **B** has a dependence with the potential so this system was all inside the second loop to keep the iterations consistent if there was a potential term that was defined. The RHS of the solution to this part, takes the updates of all x positions from ψ_{fake} and then updated all the y positions to finally give a matrix with the fully updated position. ψ^{n+1} was now able to be updated and the next time iteration can be started.

3.1.3 Convergence Test

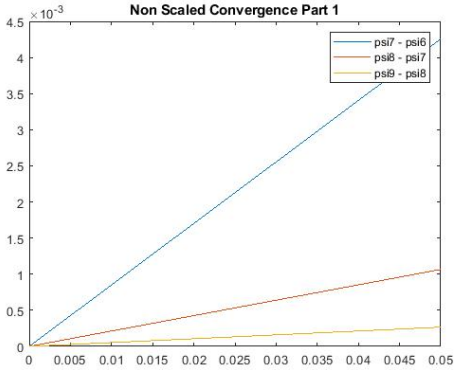
Before going over the convergence tests here, there is the explanation again of idtype and vtype. The idtype here is the same as the idtype for the 1-D equation, except with an extra spacial dimension. idtype = 0 gives the exact solution, idtype = 1 gives the boosted gaussian. A summary is shown below:

- idtype = 0 gives $\psi(x,0) = sin(m_x\pi x)sin(m_y\pi y)$
- idtype = 1 gives $\psi(x,0) = e^{-ip_x x}e^{-ip_y y}e^{(\frac{x-x_0}{\delta_x})^2 + \frac{y-y_0}{\delta_y})^2}$

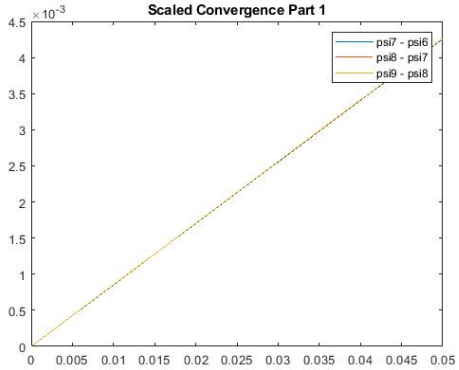
Moving to vtype, there is 3 different scenarios, vtype = 0 give no potential, vtype = 1 gives a square well (negative) or barrier (positive), and vtype = 2 give a double slit potential. again, this is summarized below:

- vtype = 0 gives: V = 0
- vtype = 1 gives: V = $\begin{cases} V_c & \text{x}_{min} \leq x \leq x_{max} \text{ and } y_{min} \leq y \leq y_{max} \\ 0 & otherwise \end{cases}$
- vtype = 2 gives a double slit potential

Moving on to the convergence tests. These tests followed the same method as in the 1-D case. idtype = 0 and vtype = 0. This gives the family of exact solutions with no potential affecting the wave function. The returned psi value from our function is returned at the 4 different levels, level 6, level 7, level 8, and level 9. The first test is the exact same, taking our returned ψ and checking if they go like $O(h^2)$. The graphs are shown below



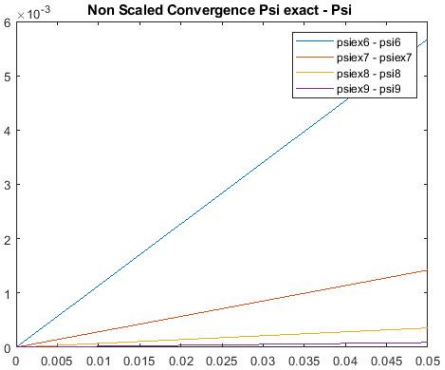
Error difference between levels



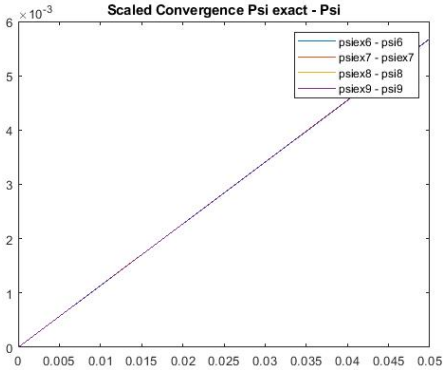
Converged errors

After multiplying $(\psi_8 - \psi_7) * 4$, and $(\psi_9 - \psi_8) * 16$ it can be seen that the right graph shows they both line up directly with $(\psi_7 - \psi_6)$ showing $O(h^2)$ convergence which satisfies the questions.

For the second convergence test, we also used idtype = 0 and vtype = 0. Again like the 1-D case, we are now comparing the exact solution ψ_{exact} minus our functions returned ψ and comparing the difference. These should also scale by $O(h^2)$.



Error difference between levels



Converged errors

Again, after multiplying $(\psi_{exact7} - \psi_7) * 4$, $(\psi_{exact8} - \psi 8) * 16$, and $(\psi_{exact9} - \psi_9) * 64$ it can be seen that the scaled graph shows they all line up directly with $(\psi_{exact6} - \psi_6)$ showing $O(h^2)$ convergence which satisfies the questions.

4 Animation and Contour Plots

Once the code was running everything properly and all questions were completed, then came making some animations! Here all the different v types were tested for the 2-D animations, vtype = 1 allowed for making either a potential barrier or a potential well, and vtype = 2 allowed for the animation of a double slit potential wall. All the surface plots were run at level 6 to allow for easier viewing on images. Higher level cause grid spacing to be too small, thus everything looked quite dark due to the lines. The plots were made with a simple loop like we have used in many previous assignments and tutorials.

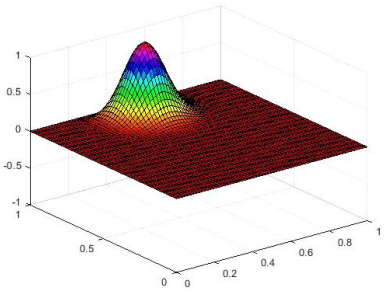
4.1 Rectangular Potential Barrier

The first animations done were with vtype = 1 and a positive potential thus creating a potential barrier. The following inputs for the function were used to created the following sets of images.

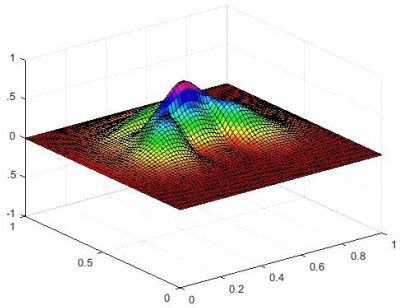
[xt yt tt psit psiret psiimt psimodt vt] = sch_2d_adi(0.012, 6, 0.03, 1, [0.5, 0.9, 0.12, 0.12, 0, -20], 1, [0.4, 0.6, 0.4, 0.6, 900]);

The following surface plots were obtained through this.

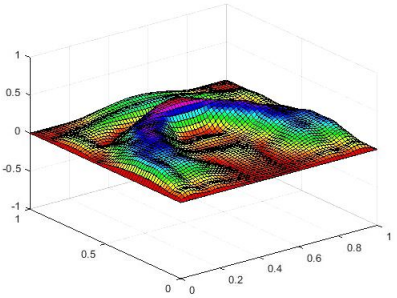
2-D With a Rectangular Potential Barrier



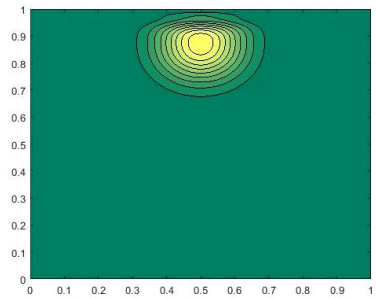
Initial frame with Potential Barrier



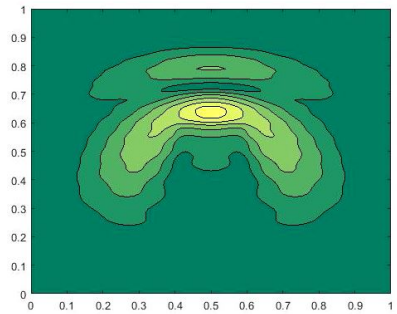
Approximately Halfway



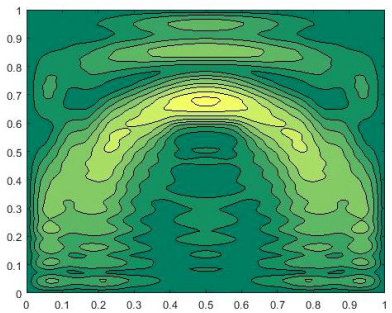
End of the Animation



Initial frame with Potential Barrier



Approximately Halfway



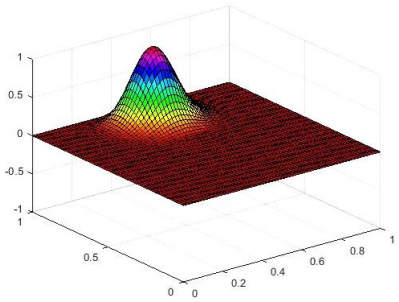
End of the Animation

In the images above it is clear that there is an interaction between the wave and the potential barrier. Most of the wave is reflected back from the well when it comes in contact but as can be seen in the second image, some of the wave transmits through the barrier to the other side. The last image on the right shows the completed animation with much of the wave spread along the plot.

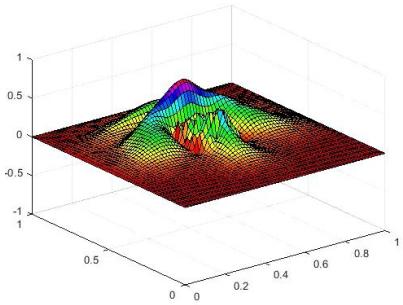
2-D With a Rectangular Potential Well

These animation were also obtained using vtype = 1. However these plots now show the interaction of the wave with a potential well. The following inputs were used to create these images.

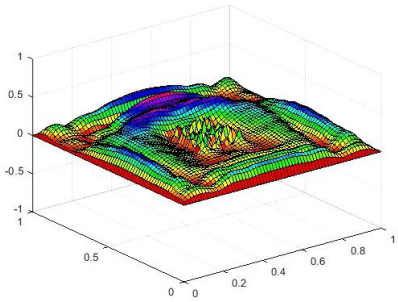
[xt yt tt psit psiret psiimt psimodt vt] = sch_2d_adi(0.015, 6, 0.03, 1, [0.5, 0.9, 0.12, 0.12, 0, -20], 1, [0.4, 0.6, 0.4, 0.6, -10000]);



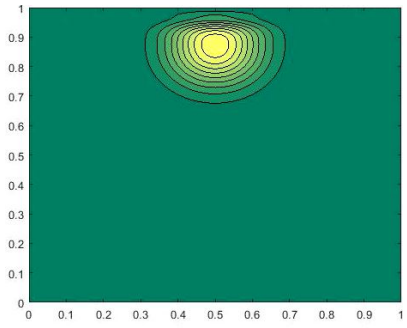
Initial frame with Potential Well



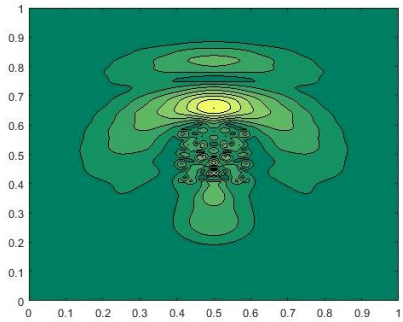
Approximately Halfway



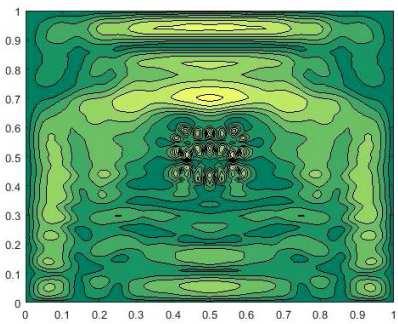
End of the Animation



Initial frame with Potential Well



Approximately Halfway



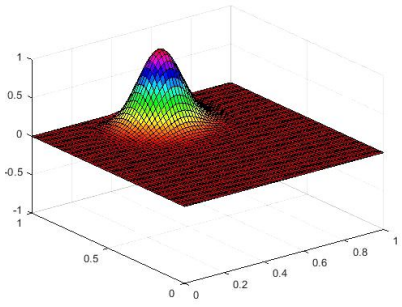
End of the Animation

The potential is in the same spot as the previous animation for the potential barrier but now the potential is negative which creates a well. As the wave approaches and hit the well area, some of the wave is reflected, some is transmitted into the well and is "trapped" and some passe through/over the well to the other side. Its clear to see the trapped portion in both the contour plots and the surface plots.

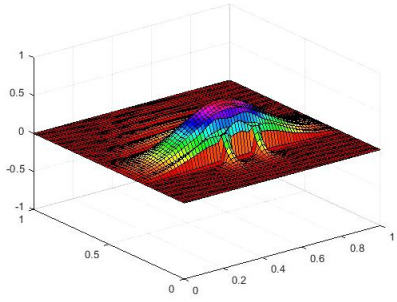
4.2 2-D Double Slit Potential

For this part of the animations, which is my opinion was very cool too see, vtype is equal to 2 giving a double slit potential wall. The following inputs gave the animations for the graphs below:

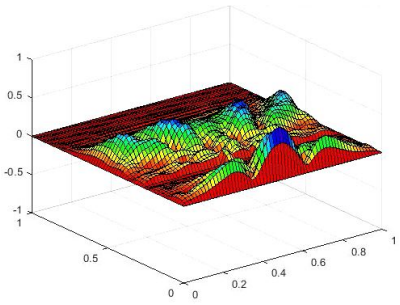
[xt yt tt psit psiret psiimt psimodt vt] = sch_2d_adi(0.0135, 6, 0.03, 1, [0.5, 0.9, 0.12, 0.12, 0, -50], 2, [0.4,0.45,0.55,0.6, 25000]);



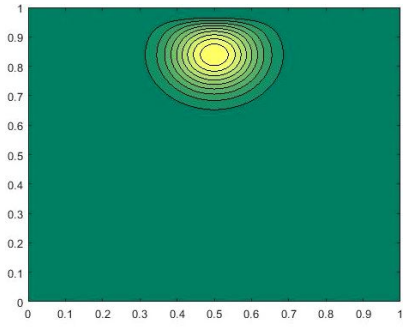
Initial frame with Double Slit



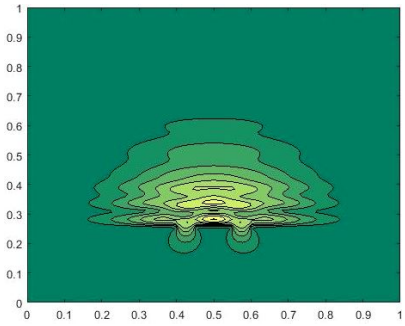
Approximately Halfway



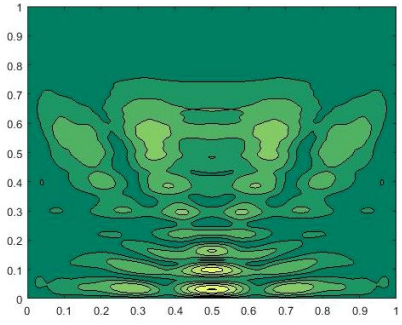
End of the Animation



Initial frame with Double Slit



Approximately Halfway



End of the Animation

The interference pattern is quite clear to see and from all the times throughout our physics classes, the interference pattern looks very familiar. On the other side of the potential barrier, between the two slits has a high peak that tapers off quickly, then has another peak on each side. This is very characteristic of the double slit experiment that has been shown multiple times throughout courses. It was pretty cool to crate a system that can simulate this.

5 Functions and Inputs

5.1 Main Functions

The main two functions that everything worked off of were:

The 1-D Function

[x t psi psire psiim psimod prob v, psiex] = sch_1d_cn(tmax, level, lambda, idtype, idpar, vtype, vpar)

The 2-D Function

[x y t psi psire psiim psimod prob v psiex] = sch_1d_cn(tmax, level, lambda, idtype, idpar, vtype, vpar)

Where

- tmax = max time for integration
- level = discretization level
- lambda = dt/dx
- idtype selects initial data type
- idpar = vector of the initial parameters
- vtype selects potential type
- vpar = vector of initial parameters for potential

For the outputs I added psiex which gives ψ_{exact}. I ran this in my main function which was then returned to be used in the convergence tests.

6 Scripts

The following files are the scripts I used and created to complete all the tasks.

- **sch_1d_cn.m** is the main function written for the 1-D Schrödinger
- **sch_2d_adi.m** is the main function written for the 2-D Schrödinger
- **convergence_1d.m** has all the code for question one convergence testing
- **convergence_2d.m** has all the code for question two convergence testing
- **numerical.m** has the code for the numerical test for question one
- **animations_1d.m** has the code to create the 1-d animations
- **animations_2d** has all the code for the surface and contour plot animations for question 2

The code in the convergence testing files to run the main functions has been commented out. This was done because once the variables were saved in the work space, there was no need to run the main functions anymore. while the 1-D scripts ran quickly regardless of the level, the 2-D script took quite a while to run through level 9, hence the main function code being commented out to save time while running convergence tests.