# PHYS410 - Project 2

Name: Brendan Lai, Student Number: 19241173

Due: December 5, 2022

## Contents

## 1 Project Objective

This project investigates the one dimensional time dependent and two dimensional Schrodinger equations. Particularly it involves the practice of applying FDA and numerical solutions to time dependent partial differential equations (PDEs). Doing such involved implementing tridiagonal systems. To verify the success of the algorithms implemented convergence tests and survey's were executed.

In the first problem we implemented an implicit Crank-Nicolson scheme to solve the problem and in the 2nd problem an alternating direction implicit scheme (ADI). Inclusive to these solutions to verify our algorithms we used convergence tests in both problems as well as surveys in the first problem and videos in the second problem which are all provided in the report.

# 2 Theory

## 2.1 Partial Differential Equations

Briefly a note on partial differential equations to preclude the specific set of equations that this project investigates. In time-dependent PDEs there are a few different issues in their finite difference solutions. Namely, accuracy, stability, convergence, and computational costs are just a few of them. There are two types of PDEs really that we could think about however for our project we primarily focus on just the initial boundary value problems where the spatial domain is finite. In this assignment the spatial domains are restricted from [0,1]. The 1D case is x=[0,1] while in the 2d case by x,y are restricted 0 through 1 inclusive. The boundary conditions we impose are also homogeneous Dirichlet Boundary Conditions as well for both the problems.

## 2.2 Finite Difference Approximation Methods

When it comes to the finite difference methods of PDEs not much is too different from what we have previously seen. Particularly we still implement our normal 2nd order centered approach most often. This is:

$$f''(x) = \frac{f(x+h) - 2f(x) + f(x-h)}{\Delta x^2} = \frac{f_{i+1} - 2f_i + f_{i-1}}{\Delta x^2}$$

The other FDA we implement is the 1st order forward approach which gives us an FDA for the first time derivative. This is:

$$f'(x) = \frac{f(x+h) - f(x)}{\Delta x} = \frac{f_{i+1} - f_i}{\Delta x}$$

Throughout the implementations of both the Crank Nicolson and the ADI we require implementing this derivation to compute our schemes and their associated equations. For example then in our Crank-Nicolson approach when expanding our scheme we use this FDA to define our partial derivative of the continuum equation. This helps us discretize the solution. These FDAs are still included in the scheme derivations as well for completeness. This section simply provides a review of what we had previously derived.

Notation to recall for later on is that when we are dealing with the spatial and time domains that the time domain is kept as superscript while spatial is subscript. And for the 2d case we let the subscripts be defined i,j where i corresponds with the x domain and j with the y domain.

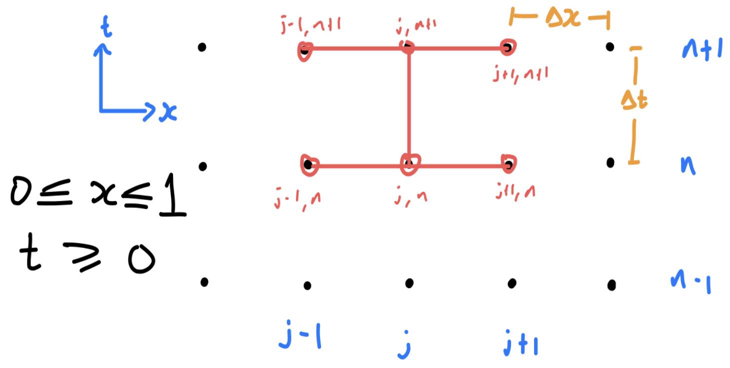
Lastly, it is important to recall that when it comes to FDA schemes to date we had only really had two types, explicit and implicit. Explicit schemes were generally the easier of the two. In explicit schemes the forward and central difference FDAs were used and the grid sizes are restricted. The downside of explicit schemes however is that stability conditions were required. While for implicit methods they are more involved using the backward and central difference FDAs. The benefit being that they are stable regardless of other conditions. Moving forwards, the Crank-Nicolson approach is a combination of these two types of schemes (explicit and implicit)

## 2.3 Crank-Nicolson Approach

Prior to the Crank-Nicolson approach we also need to discuss how the temporal and spatial mesh are generated and the discretization of the domain. As always this is step 1 in developing the solution.

The importance of the Crank-Nicolson Scheme is that it pairs the explicit and implicit schemes which are both only first order accurate to give us 2nd order accuracy for both the time and spatial domains. This happens by implementing the n+1/2 time step. Another useful thing to note about the Crank-Nicolson approach is that it is unconditionally stable. Essentially the Crank-Nicolson scheme is derived from taking the average between the explicit and implicit schemes

The stencil we use in this case is as shown below for the 1D Schrödinger equation.



The motivation for this method can msot easily be shown through an investigation of the 1D diffusion equation.

$$u_t = \sigma u_{xx}$$

Implementing the FDAs shown in section 2.2 we then get the following equation which defines our diffusion equation.

$$\frac{u_j^{n+1} - u_j^n}{\Delta t} = \frac{1}{2}\sigma\left(\frac{u_{j+1}^n - 2u_j^n + u_{j-1}^n}{\Delta x^2} + \frac{u_{j+1}^{n+1} - 2u_j^{n+1} + u_{j-1}^{n+1}}{\Delta x^2}\right)$$

Analyzing this result we then can also compute the truncation error ($\tau$) by computing expansions about $(x, t) = (x, t^{n+1}) = (x, t^n + \Delta t/2)$. This results in:

$$\tau = \frac{1}{24}\Delta t^2 (u_{ttt})_j^{n+1/2} - \frac{1}{8}\sigma\Delta x^2 (u_{ttxx})_j^{n+1/2} - \frac{1}{12}\sigma\Delta x^2 (u_{xxxx})_j^{n+1/2} + O(\Delta t^2) + O(\Delta x^4)$$

Then with the truncation error we see that the result is a solution with second order accuracy in both the time and spatial domain ($O(\Delta t^2, \Delta x^2)$).

Looking at the stability now of the 1D diffusion equation we write the scheme as:

$$(1 - \frac{\alpha}{2}D^2)u_j^{n+1} = (1 + \frac{\alpha}{2}D^2)u_j^n$$

Then we take the Fourier transform to get:

$$(1 + 2\alpha\sin^2(\frac{\xi}{2}))\tilde{u}^{n+1} = (1 - 2\alpha\sin^2(\frac{\xi}{2}))\tilde{u}^n$$

And therefore we get the amplification factor $\tilde{G}(\xi)$. And where the amplification factor is less than 1 then we have stability. And thus we see unconditional stability due to the amplification factor being of form $\frac{1-X}{1+X}$.

$$\tilde{G}(\xi) = \frac{(1 - 2\alpha\sin^2(\frac{\xi}{2}))}{(1 + 2\alpha\sin^2(\frac{\xi}{2}))} = \frac{\tilde{u}^{n+1}}{\tilde{u}^n}$$

This motivates the notion of the Crank Nicolson method for these types of equations as unconditionally stable and it also holds true for the Schrödinger equation.

More on the exact implementation of our Crank-Nicolson scheme is discussed in section 2.5.2 where the discretization of the domain and the specific finite difference approximations are used to derive the scheme. With these schemes the goal is then to collect the similar time step related terms on each side of the equation which either becomes the coefficients used to build our tridiagonal matrix or the terms used to compute the right hand side which is the divisor of the left division which we use in MATLAB due to performance gains when solving these systems.

## 2.4 Convergence Testing (Brief Review)

Since convergence testing has been covered throughout previous homeworks and project the discussion for this will be briefer than previous reports just to serve reminder as to it's purpose. Convergence testing is used in our case to verify the global errors of the solutions we have implemented. The two types of convergence tests executed were four level and exact solution comparisons.

In the four level convergence test the solutions for four consecutive levels are computed (in our case levels 6 thorough 9). Then we would calculate the 'scaled' errors by computing the differences between adjacent levels (ex. $sol_{l7} - sol_{l6}$) ensuring that we had properly adjusted the results of these solutions to match the correct values for the time steps. Since an increase in level doubles the number of both time and spatial steps we would need to take alternating points for example when comparing the side by side levels. After computing these three adjacent differences (9-8, 8-7, 7-6 levels) we lastly then would compute the l2 norms for each of these. Lastly, then plotting these errors with appropriate scaling we should then see near coincidence. In this case for both the Crank-Nicolson and ADI schemes implemented we expected fourth order accuracy so the scaling we needed to verify was with $4^{p-1}$ where p would be the difference from the original level.

For the four level convergence test we would let the level to level; difference as:

$$d\psi^l = \psi^{l+1} - \psi^l$$

The l-2 norm is then computed for these level to level differences and we would plot the following to verify our fourth order accuracy prediction:

$$||d\psi^{l_{min}}||, 4||d\psi^{l_{min}+1}||, 4||d\psi^{l_{min}+1}||, 4^2||d\psi^{l_{min}+2}||$$

The same plotting strategy is employed for the exact solution analysis.
The second type of convergence test executed was comparing the results to the exact solution. A similar process was followed to previously except when calculating our differences we would use the exact solution instead of the different level. Beyond this the results should and were also essentially the same with fourth order accuracy meaning that similar scaling would be necessary. And this was seen in the convergence plots shared in section 4 of the report.

Also, note that while we describe the complete solution to have fourth order accuracy throughout the report our implementations on their own are actually second order accurate as our derivations will describe. However, since these methods span two domains of both second order accuracy our complete solution accuracy will be fourth order. Where the FDAs are $O(h^2)$ we get:

$$\psi^l(x,t) = \psi(x,t) + h_l^2 e_2(x,t) + O(h_l^4)$$

Convergence testing is important because it validates our solution and helps us ensure the implementation is behaving as we expect.

## 2.5 Problem 1

### 2.5.1 The 1D Schrödinger equation

For the first problem of the project the time dependent Schrödiner equation was analyzed. After non-dimensionalization it can be given as follows below:

$$i\psi(x,t)_t = -\psi_{xx} + V(x,t)\psi$$

Here, $\psi(x,t)$ is a complex and for ease of use the equation is strictly solved on the following domain as well as homogeneous boundary conditions and these initial conditions:

$$0 \le x \le 1, \qquad 0 \le t \le t_{max}$$

$$\psi(x,0) = \psi_0(x)$$

$$\psi(0,t) = \psi(1,t) = 0$$

As mentioned, since $\psi$ is complex it takes on the following interpretation. Where $\psi$ is broken into the real and imaginary components

$$\psi = \psi_{Re} + i\psi_{Im}$$

One quantity that we also choose to measure in problem 1 which is specifically used for our numerical experiments i s the probability density, $\rho = |\psi|^2$. This is the "running integral' $P(x,t)$ and is as follows:

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

.

Lastly, the exact solution of this 1d Schrödinger equation is given below and is used to validate the accuracy of the implemented solution in the convergence tests.

$$\psi(x,t) = e^{-im^2\pi^2 t} * \sin(m\pi x)$$

The other avenue to discuss about the 1D Schrödinger equation which should be discussed is the different behaviours for our varying initialization types and potential types. Further, there were two different initialization types; Exact family and Boosted Gaussian.

The exact family initialization type is the simpler of the two and behaves predictably which meant that we also knew the exact solution which would be used in our convergence test. The initialization takes the form of:

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

The boosted Gaussian initialization type has three parameters which were $x_0, \delta, p$. Each represent the initial position, the radius, and the speed of which the potential from this is travelling.

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

The different potential types were: no potential, rectangular barrier, and rectangular well. The no potential type is simple in that there is no additional potential in the spatial domain. In the case with a rectangular barrier and well however there are some things to note about these behaviours.

In the rectangular barrier when the particle hits the barrier it acts as a wall in which the particle is to bounce off this wall. A barrier occurs when the potential is greater than zero. For a rectangular well the potential is less than zero and it is when a particle enters the well it is not able to leave the well and will bounce between its walls given once it has entered.

### 2.5.2 1D Schrödinger Equation Discretization

As always, our approach to computationally solving these problems begins with discretizing both the domain and the solution. Looking at the continuum domain we then can introduce our discretization levels, $l$ as the ratio of the two different meshes, temporal and spatial. This is

$$\lambda = \frac{\Delta t}{\Delta x}$$

Then we have the following other variables which set our domain length and the spacings as well.

$$\begin{aligned} n_x &= 2^l + 1 \\ \Delta x &= 2^{-l} \\ \Delta t &= \lambda \Delta x \\ n_t &= round(t_{max}/\Delta t) + 1 \end{aligned}$$

Then we employ the 2nd order finite difference approximations (FDAs) which has been derived in previous homeworks and projects. The general 2nd order FDA approximation that we use is the central difference method which is given with generality as:

$$f''(x) = \frac{f(x+h) - 2f(x) + f(x-h)}{\Delta x^2} = \frac{f_{i+1} - 2f_i + f_{i-1}}{\Delta x^2}$$

Additionally, we also use a 1st order scheme to approximate the first order time derivative which is given as follows:

$$f'(t) = \frac{f(t+h) - f(t)}{\Delta t} = \frac{f^{n+1} - f^n}{\Delta t}$$

In both of these cases h represents some step size which is given by the level. The second subscript form of the scheme is the notation we generally will opt to use.

Revisiting our continuum equation: $i\psi(x,t)_t = -\psi_{xx} + V(x,t)\psi$ we then now can substitute our FDAs into the equation giving us the Crank-Nicolson discretization.

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

## 2.6 Problem 2

The second problem involved implementing an alternating direction implicit scheme to solve the 2D Schrödinger equation.

### 2.6.1 The 2D Schrödinger equation

The continuum equation for the 2D Schrödinger equation is as shown below and also can be described as the following line:

$$i\psi(x,y,t)_t = -(\psi_{xx} + \psi_{yy}) + V(x,y)\psi$$
$$\psi_t = i(\psi_{xx} + \psi_{yy}) - iV(x,y)\psi$$

Similar to problem 1 we are imposing Dirichlet boundary conditions and we are also restricting our spatial domain to be from zero to one in both the x and y domain. This is that the equations are solved on the following domain as well as subject to the intial and boundary conditions below the domain

$$\begin{aligned}
&0 \leq x \leq 1 \qquad 0 \leq y \leq 1 \qquad 0 \leq t \leq t_{max} \\
&\psi(x, y, 0) = \psi_o(x, y) \\
&\psi(0, y, t) = \psi(1, y, t) = \psi(x, 0, t) = \psi(x, 1, t) = 0
\end{aligned}$$

Moreover, the family of the exact solutions for this continuum equation where $m_x, m_y$ are positive integers then is:

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

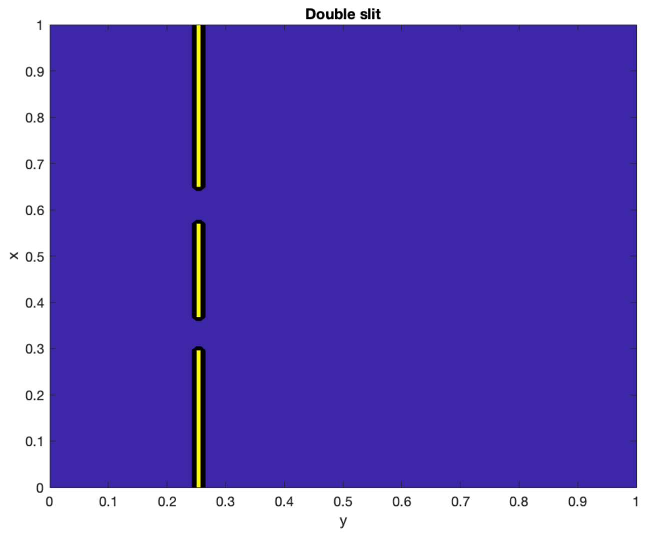
In our implementation we also discuss the implementation of different initialization and potential types. The actual implementation and meaning behind these are discussed in the implementation stage however a couple brief notes on their physics interpretations.

The Boosted Gaussian in the 2d scenario is the same effectively however just with another dimension. We gain another set of the three parameters (initial position, radius, and momentum) for the y domain now giving us a total of 6 parameters for this initialization type. The associated Boosted Gaussian is:

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

For the potential types the no potential, rectangular barrier, and rectangular well are all essentially the same except with the added dimension. These all behave the same as the 1D scenarios. However, we also introduce the double slit potential type.

In the double slit implementation the the particle travels towards the double slit and then separates on hitting the slit. The double slit looks roughly as shown below:



### 2.6.2 Alternating Direction Implicit Methods

Instead of a 2D Crank-Nicolson scheme the Alternating direction Implicit method was employed. The idea behind this method is that in the Crank-Nicolson update we factor the operations into a product of operators of which each acts in 1 dimension. Similar to the original Crank-Nicolson discussion the diffusion equation and in this case 2D diffusion equation will be discussed to motivate the method. Note that in our development of the method we also originally tested with the Diffusion equation as it is less involved to implement (non trivial though). The 2D diffusion equation then is:

$$u_t = u_{xx} + u_{uu}$$

Letting $L^h = \partial^h_{xx} + \partial^h_{yy}$ we then get:

$$(1 - \frac{\Delta t}{2} L^h) u^{n+1}_{i,j} = (1 + \frac{\Delta t}{2} L^h) u^n_{i,j}$$

Now:

$$\left(1 - \frac{\Delta t}{2}(\partial^h_{xx})\right)\left(1 + \frac{\Delta t}{2}(\partial^h_{yy})\right) u^{n+1}_{i,j} = \left(1 + \frac{\Delta t}{2}(\partial^h_{xx})\right)\left(1 + \frac{\Delta t}{2}(\partial^h_{yy})\right) u^n_{i,j} + \frac{\Delta t^2}{4}\partial^h_{xx}\partial^h_{yy}(u^{n+1}_{i,j} - u^n_{i,j})$$

However the final term can be neglected as we can still achieve 2nd order accuracy in the time domain so we opt to remove it. This leaves us with on form of the ADI scheme and actually what we would use for the first stage of our method.

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

Further we redefine $\left(1 + \frac{\Delta t}{2}(\partial^h_{yy})\right) u^{n+1}_{i,j} = \psi^{n+1/2}_{i,j}$ so that we then introduce a definition for our "half time step". Note that this isn't actually the literal half time step however it is just notation signifying part 1 of the 2 step solution. Coincidentally this also provides the scheme for the second step of our solution. This is that the first and second stages of the ADI scheme for the 2D diffusion equation are as below:

$$\begin{aligned}
&\text{Stage 1:} \quad \left(1 - \frac{\Delta t}{2}\partial^h_{xx}\right) u^{n+1/2}_{i,j} = \left(1 + \frac{\Delta t}{2}(\partial^h_{xx})\right)\left(1 + \frac{\Delta t}{2}(\partial^h_{yy})\right) u^n_{i,j} \\
&\text{Stage 2:} \quad \left(1 + \frac{\Delta t}{2}(\partial^h_{yy})\right) u^{n+1}_{i,j} = u^{n+1/2}_{i,j}
\end{aligned}$$

These two stages then provide the base for what is required to do for each time-step. Note that the stages requires the solution of n systems of size n which can solve each in $O(n) time$. This means that our total time for solution is $O(nxn) = O(n^2) = O(N)$ which is optimal when compared to what the 2D Crank Nicolson would have been.

Here we have to solve n tridiagonal systems for each stage and specifically that for the first stage to get $u^{n+1/2}_{i,j}$ we need a tridiagonal solve and then int he second stage to get $u^{n+1}_{i,}$ we also need a tridiagonal solve.

In the case of the 2D Schrödinger equation the additional complexity is with the potential term and the ADI scheme becomes these two stages:

$$\begin{aligned}
&\text{Stage 1:} \quad \left(1 - i\frac{\Delta t}{2}\partial^h_{xx}\right) \psi^{n+1/2}_{i,j} = \left(1 + i\frac{\Delta t}{2}(\partial^h_{xx})\right)\left(1 + i\frac{\Delta t}{2}(\partial^h_{yy}) - i\frac{\Delta t}{2}V_{i,j}\right) \psi^n_{i,j} \\
&\text{Stage 2:} \quad \left(1 + \frac{\Delta t}{2}(\partial^h_{yy}) + i\frac{\Delta t}{2}V_{i,j}\right) \psi^{n+1}_{i,j} = \psi^{n+1/2}_{i,j}
\end{aligned}$$

Lastly a reminder that the difference operators are utilized like how we use our FDAs previously which are shown in section 2.2. This is that the difference operators are:

$$\begin{aligned}
\partial^h_{xx}(u^n_{i,j}) &= \frac{u^n_{i+1,j} - 2u^n_{i,j} + u^n_{i-1,j}}{\Delta x^2} \\
\partial^h_{yy}(u^n_{i,j}) &= \frac{u^n_{i,j+1} - 2u^n_{i,j} + u^n_{i,j-1}}{\Delta x^2}
\end{aligned}$$

And lastly, for both stages 1 and 2 we also need to supplement that both results are adhering to the original initial and boundary conditions. Which are:

$$\psi^1 = \psi_0(x_i, y_j)$$
$$\psi^n_{1,j} = \psi^n_{0,j} = \psi^n_{i,0} = \psi^n_{i,1} = 0$$

This is the ADI used to solve problem 2 with the implementation of this method fully discussed in the implementation section of problem 2.

## 3 Implementation

This section discusses the two different implementations of our solutions for the 1d time dependent and 2d Schrödinger equations. The first problem employed the Crank-Nicolson approach shown in section 2.1 of this report while an ADI scheme was used for problem 2 as discussed in section 2.2

### 3.1 Problem 1: One-Dimensional Time Dependent Schrödinger Equation

For the implementation in this problem we employed the Crank-Nicolson approach. The code along with the function documentation is found in "sch_1d_cn.m"

We begin with determining some key parameters; this being:

- nx: number of x grid points $\longrightarrow nx = 2^{level} + 1$
- x: vector for x (length nx)
- dx: grid point spacing $\longrightarrow dx = 2^{-level}$
- dt: time grid spacing $\longrightarrow dt = lambda * dx$
- nt: number of time steps $\longrightarrow nt = round(tmax/dt) + 1$
- t: vector for t (length nt)

The other part of the implementation was including options for the initialization and potential types. Denoted by idtype and vtype there were 2 different possible options for each of these (idtype = 0,1 and vtype=0,1).

idtype option 1 was the exact family solution and had just one parameter m. The equation of the exact family being:

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

idtype option 2 was the Boosted Gaussian and had three parameters; $x_0$, $\delta$, $p$. These were apart of the following equation for a Boosted Gaussian:

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

Parameters for idtype = 2 would then be passed in as vector with the variables in the corresponding locations [$x_0$, $\delta$, p] Then for the potential types the first option is vtype is zero which meant there is no potential.

For the second option vtype == 1 meant that we would have a rectangular barrier or well. While having the same architecture the barrier versus well is then determined by the sign of the potential we assign ($V_c$). Where if $V_c > 0$ we would have a barrier while $V_c < 0$ it would be a well.

The framework for this was the following:

$$V(x) = \begin{cases} 0 & \text{for } x \leq x_{min} \\ V_c & \text{for } x_{min} \leq x \leq x_{max} \\ 0 & \text{for } x \geq x_{max} \end{cases}$$

With this the passed parameters would be in vpar = $[x_{min},\, x_{max},\, V_c]$

The Crank-Nicolson scheme derived for this is given as below and our implementation used the following scheme to build our tridiagonal matrix.

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

Where the left hand side was represented as below after expanding and solving the Crank-Nicolson description:

$$\text{LHS} = \frac{1}{2\Delta x^2}(\psi_{j-1}^{n+1} + \psi_{j+1}^{n+1}) + (i\frac{1}{\Delta t} - \frac{1}{\Delta x^2} - V_j^{n+1/2}) * \psi_j^{n+1}$$

.
With this we would then use the LHS to build our tridiagonal matrix using spdiags. The lower and upper diagonals would be the same as shown by the j+1 and j-1 coefficients being the same while the main diagonal would correspond to the j-th term coefficient. This is found in the function.

Upon creation of our sparse matrix the other step then was redefining our RHS of the linear system which would require updating over each iteration. This was also derived from the Crank Nicolson scheme of the original equation. It was derived to be:

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

This RHS could then be calculated using the surrounding neighbours in our Crank-Nicolson scheme to compute the next iteration. Computing the complete next time step utilized the left division functionality in MATLAB, which is an optimized calculation method particularly for sparse matrices like our tridiagonal system.

After looping through all our time steps the final step was to calculate the fractional excess probability. This was done by implementing a running integral by using the trapezoidal method. The trapezoidal method is described below:

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

With our running values for each x then we would fill our probability vector (the fractional excess probability) which would be later used in the numerical experiments and specifically surveys.
This concludes the Crank-Nicolson implementation of the 1D - time dependent Schrödinger equation

## 3.2 Problem 2: Two-Dimensional Schrödinger Equation

This section discusses the ADI implementation for the two-dimensional Schrödinger equation. The code for the function of this solution is found in "2d_2d_adi.m" which includes the function documentation for the input and output arguments as well as their interpretations and for the arrays the dimensions of such.

Like the 1d Crank-Nicolson scheme the 2d ADI scheme requires the initial calculations for a few parameters. Precisely:

- nx: number of x grid points $\longrightarrow nx = 2^{level} + 1$
- ny: number of y grid points (equivalent to nx in this case as it is symmetric
- dx: space between spatial grid points (x) $\longrightarrow dx^{-level}$
- dy: space between spatial grid points (y).. equivalent to dx
- dt: space between time points $\longleftarrow lambda * dx$
- nt: number of time steps

For the idtype in this implementation there are also two possible initialization types. The exact family solution and the boosted Gaussian. Similar to the 1d case we use a vector idpar to complete the associated solutions. For idtype = 0 the exact solution and idpar as as given below:

$$\psi(x, y, 0) = \sin(m_x \pi x) * \sin(m_y \pi y)$$

$$\text{idpar} = [m_x,\ m_y]$$

For idtype =1 the corresponding boosted Gaussian and idpar are as shown below:

$$(x, y, 0) = e^{ip_x x} e^{ip_y y} e^{-((x-x_0)^2/\delta_x^2 + (y-y_0)^2/\delta_y^2)}$$

$$\text{idpar} = [x_0, y_0, \delta_x, \delta_y, p_x, p_y]$$

Then for the potential types there were some more notable differences to the 1d section. Here there were three different vtypes; no potential, rectangular barrier or well, and double slit. No potential is straightforward and is exactly as it sounds (potential = zero).

For the rectangular barrier or well it is defined as:

$$V(x, y) = \begin{cases} V_c & \text{for } (x_{min} \leq x \leq x_{max}) \text{ and } (y_{min} \leq x \leq y_{max}) \\ 0 & \text{otherwise} \end{cases}$$

vpar would be given like this $[x_{min}, x_{max}, y_{min}, y_{max}, V_c]$

Then lastly the double slit implementation is associated with vtype = 2. The double slit as it sounds would define a section along the y axis with thickness j' to j'+1 with two slits in the section. Further, we would define j' as

$$j' = \frac{n_y - 1}{4} + 1$$

And then we will say that:

$$V_{i,j} = Vi, j' + 1 = 0 \text{ for } [(x_1 \leq x_i) \text{ and } (x_i \leq x_2)] \text{ or } [(x_3 \leq x_i) \text{ and } (x_i \leq x_4)]$$
$$V_{i,j'} = V_{i,j'+1} = V_c \text{ otherwise}$$
$$V_{i,j} = 0 \text{ for } j \neq (j' \text{ or } j' + 1)$$

These slit openings are then adjustable by vpar and are given by the following:

$$\text{vpar} = [x_1, x_2, x_3, x_4, V_c]$$

The ADI scheme as discussed in the theory section was a two stage process. In the first stage we also implemented a calculation for a temporary psi to simplify our calculations with this intermediate step. Note that the notation used for the first stage indicates that we are computing the half time step but that this is not actually a literal half time step and more so notation indicating that we are in between the stages of calculating a full time step.
Now with the potential and initialization parameters set up we can move onto our ADI scheme. Focusing on the primary sparse matrix which is used in our scheme we found that the LHS to be:

$$\text{LHS} = -(i\frac{\Delta t}{2\Delta x^2}) * (\psi_{i+1,j}^{n+1/2} + \psi_{i-1,j}^{n+1/2}) + (1 + i * \frac{\Delta t}{\Delta x^2})\psi_{i,j}^{n+1/2}$$

These coefficients would yet again help us build the sparse matrix which we would use to calculate the first stage psi ($\psi^{n+1/2}$). The upper and lower diagonals were comprised of the leading term in the above LHS equation as they correlate with the i+1 and i-1 terms while the main diagonal was determined by the i term. Therefore the lower and upper diagonals were, $(i\frac{\Delta t}{2\Delta x^2})$ while the main diagonal was $(1 + i * \frac{\Delta t}{\Delta x^2})$.

The next step was to then begin our iterations for the time steps. In each step the following process was completed. The associated equation for the 1st stage of the ADI was:

$$(1 - i\frac{\Delta t}{2}\partial_{xx}^h)\psi_{i,j}^{n+1/2} = (1 + i\frac{\Delta t}{2}\partial_{xx}^h)(1 + i\frac{\Delta t}{2}\partial_{yy}^h - i\frac{\Delta t}{2}V_{i,j})\psi_{i,j'}^n$$

In the first stage of the ADI we broke the original full term into two steps as well to simplify the number of terms that needed to be computed. This also minimized the chance of error in our derivation as there were less terms to be concerned with. The first value for $\psi$ calculated in this stage was then represented as $\psi_{temp}$ which was given as below:

$$\begin{aligned}\psi_{temp} &= (1 + i\frac{\Delta t}{2}\partial_{yy}^h - i\frac{\Delta t}{2}V_{i,j})\psi_{i,j'}^n \\ \psi_{temp} &= i * \frac{\Delta t}{2\Delta y^2} * (\psi_{i,j+1}^n + \psi_{i,j-1}^n) + (1 - i\frac{\Delta t}{\Delta y^2} - i\frac{\Delta t}{2} * V_{i,j})\psi_{i,j}^n\end{aligned}$$

With $\psi_{temp}$ computed then we still needed to compute our final RHS then which was then derived by taking the 2nd partial derivative with respect to $x$ to complete the first stage. This was then derived from our previous equations shown above:

$$(1 - i\frac{\Delta t}{2}\partial_{xx}^h)\psi_{i,j}^{n+1/2} = (1 + i\frac{\Delta t}{2}\partial_{xx}^h) * \psi_{temp}$$

This then yielded our final RHS equation as follows:

$$RHS = i\frac{\Delta t}{2\Delta x^2}(\psi_{i+1,j}^{temp} + \psi_{i-1,j}^{temp}) + (1 - i\frac{\Delta t}{\Delta x^2}) * \psi_{i,j}^{temp}$$

With the RHS now computed then the first stage could be completed by using left division between our first sparse matrix given by LHS defined earlier in this section and the RHS here (LHS \ RHS). This resulting matrix we would then represent as $\psi^{n+1/2}$ which is to represent the half time step.

$$\psi^{n+1/2} = LHS \backslash RHS$$

Moving onto stage 2, we then use this equation to compute the full time step.

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

From this we then could derive the LHS and RHS which we define in this derivation as LHS2 and RHS2 to avoid confusion with the LHS and RHS computed in the first stage. These would give us a new sparse matrix as well. Firstly LHS2:

$$LHS2 = (i\frac{\Delta t}{2\Delta y^2})(\psi_{i,j-1}^{n+1} + \psi_{i,j+1}^{n+1}) + (1 + i\frac{\Delta t}{\Delta y^2} + i\frac{\Delta t}{2} * V_{i,j})\psi_{i,j}^{n+1}$$

This LHS2 would then give rise to our 2nd sparse matrix which we use in stage 2 to compute the full time step where the lower and upper diagonals are given by the first term coefficients $((i\frac{\Delta t}{2\Delta y^2})$ and the main diagonal is given by the other coefficient $(1 + i\frac{\Delta t}{\Delta y^2} + i\frac{\Delta t}{2} * V_{i,j})$
Next the RHS2 needed to be derived and implemented and it is much simpler as it was already given in the stage 2 equation

$$RHS2 = \psi_{i,j}^{n+1/2}$$

Lastly with LHS2 and RHS2 we could then compute the next time step:

$$\psi^{n+1} = LHS2 \backslash RHS2$$

One thing to note is that in some of the implementation to ensure our computations were able to be executed we required to transpose vectors/matrices and when doing such it was crucial to be sure that the non-conjugating transpose was taken.

This process would then be executed nt - 1 times and upon completion of these iterations the final step would be to extract the real, imaginary, and modulus of $\psi$ so we could output these results as well.

## 4 Numerical Experiments and Results

### 4.1 Problem 1: One-Dimensional Time Dependent Schrödinger Equation

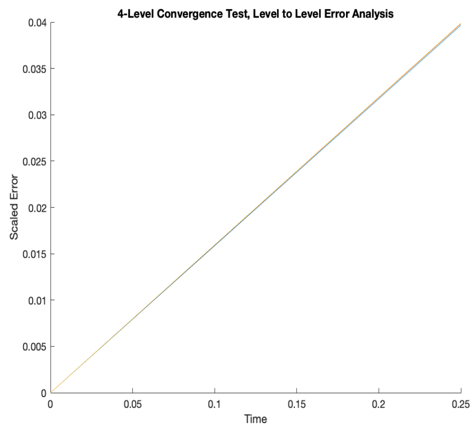
In our results and experiments for the 1D time dependent Schrödinger equation we completed three different convergence tests and then completed surveys for both a well and barrier setting.

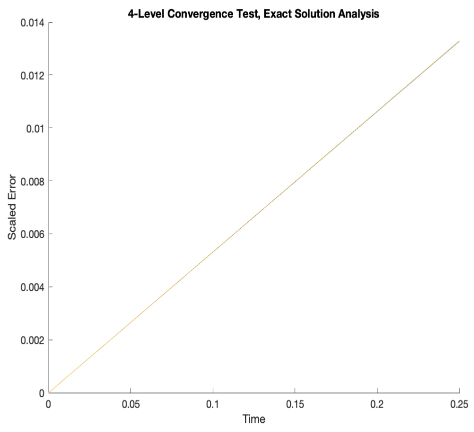
#### 4.1.1 Convergence Tests

All the convergence tests were executed for levels 6 through 9. Meaning that the solutions were calculated for each level 6 through 9 and then we were using the differences to calculate the errors. Through combining these errors then we were able to check and verify that the Crank-Nicolson implementation was indeed a fourth order global accuracy, which is the same as our derivation of the scheme's solution.

Additionally, for this first convergence test parameters we were able to include analysis comparing the Crank-Nicolson approach to the exact solution. Below are the parameters for this convergence test as well as the results of the level to level analysis and the exact solution.

- itype = 0 (Exact Family)
- vtype = 0
- m = 3
- tmax = 0.25
- lambda = 0.1

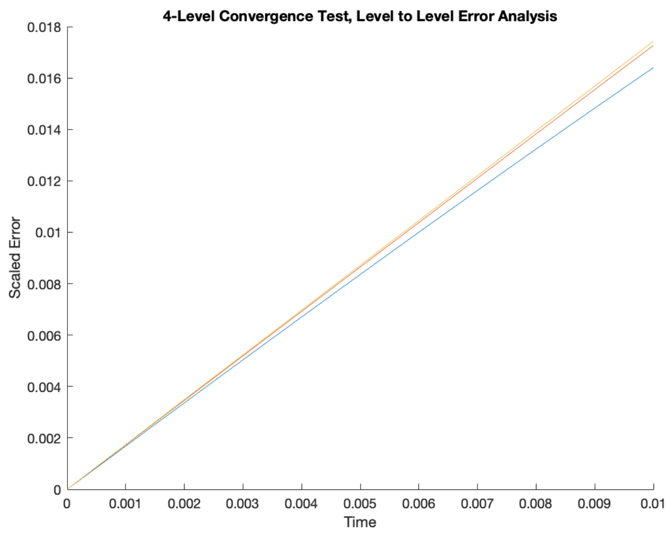




For the second convergence test we the following parameters were used:

- itype = 1 (Boosted Gaussian)
- vtype = 0
- idpar = [m = 0.5, , $x_0 = 0.075$, $\rho = 0.0$]
- tmax = 0.25
- lambda = 0.1

The results of this convergence test were just a level to level solution. as the exact solution for this case could not be computed analytically.



The one thing that is precisely obvious about this solution is that the solutions are not nearly precisely coincidental as the previous convergence tests. In fact on recollection of the convergence tests we have executed throughout previous projects and homework's as well this is the first one where the lines decay in how they coincidental they are. Also, looking at the scale of the error we can see that it is roughly $10^{-3}$ accurate which differs greatly from some of our previous tests where we expected at least $10^{-5}$ accuracy. This also applies to the convergence tests of the first ones shown in this section (Exact family) initialization type.

### 4.1.2 Numerical Experiments

For the second problem we executed two different survey's as our numerical experiments. This included simulating a barrier and well. For the barrier it was meant that the potential was greater than zero whereas for a well the potential was set to be less than zero.

A note on these survey's the calculations and background for how these results can be interpreted is included below:

The probability matrix computed and returned in our 1d CN implementation could be used to calculate the temporal average which we define as

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

This would also be normalized so that we would have:

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

Further calculating the difference between some x values $x_2$ and $x_1$ we could then use the difference between their associated temporal averages to determine a fraction of time that the quantum particle spends within that interval $x_1 \leq x \leq x_2$

For our survey's then in a case where the potential is non-zero the excess fractional probability that the particle spends in the given spatial interval is then defined by:

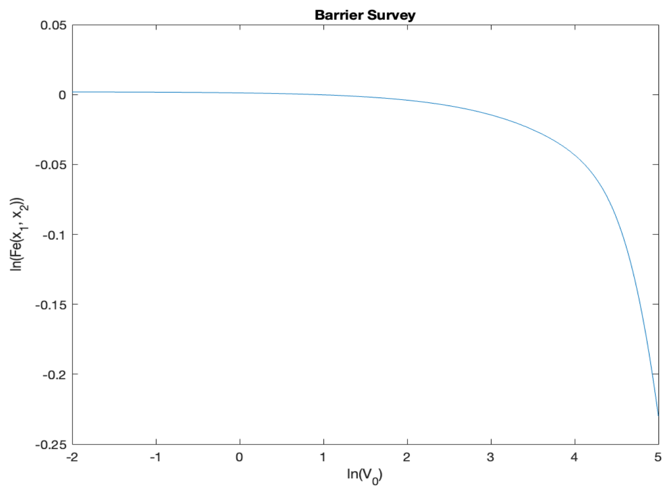
$$\bar{F}e(x_1, x_2) = \frac{\bar{P}(x_2) - \bar{P}(x_1)}{x_2 - x_1}$$

However, our survey's spans many orders of magnitude and thus for plotting purposes we considered the logarithm of this value:

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

The barrier survey used the following parameters and resulted the figure below the parameters:

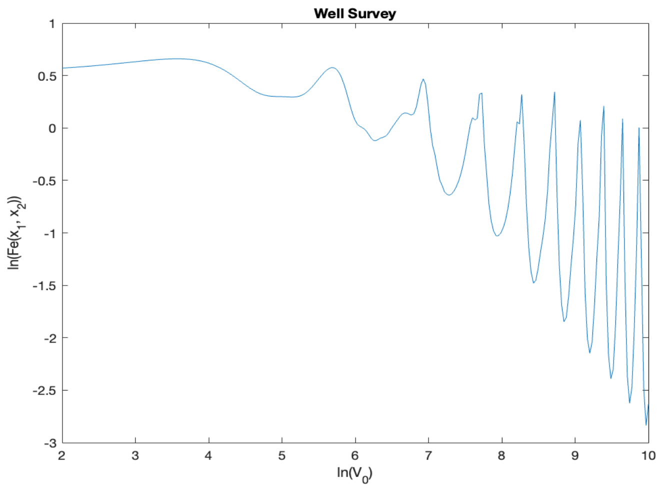
- tmax = 0.10
- level = 9
- lambda = 0.01
- idtype = 1 (boosted Gaussian)
- idpar = [m = 0.5, , $x_0$ = 0.075, $\rho$ = 0.0]
- vtype = 1 (rectangular barrier)
- vpar = [$x_{min}$ = 0.6, $x_{max}$ = 0.8, $V_c$ = 0 ]
- $x_1$ = 0.8
- $x_2$ = 1.0



For this survey the potential constant was uniformly ranging between $e^{-2}$ to $e^5$ which we plotted as $ln(V_c)$ meaning that the range was for $ln(V_c)$ [-2,5] for 251 grid points. Moreover, the survey meets our expectations as we would see that the particle would spend time outside the range. Essentially this is that we see that as the potential grows and becomes greater than the barrier then the probability that the particle goes beyond the barrier decays to zero which is shown in the figure.

The second survey was the well survey which used the following parameters and resulted the figure below the parameters:

- tmax = 0.10
- level = 9
- lambda = 0.01
- idtype = 1 (boosted Gaussian), idpar = [m = 0.4, , $x_0$ = 0.075, $\rho$ = 0.0]
- vtype = 1 (rectangular barrier), vpar = [$x_{min}$ = 0.6, $x_{max}$ = 0.8, $V_c$ = 0 ]
- $x_1$ = 0.6, $x_2$ = 0.8



For this survey we see then that the particle spends it time in the range while bouncing of the walls in the well explaining the fluctuations in the figure. This then agrees with what we would expect as the well prevents the particles from escaping and maintains significant amounts of time in the well.

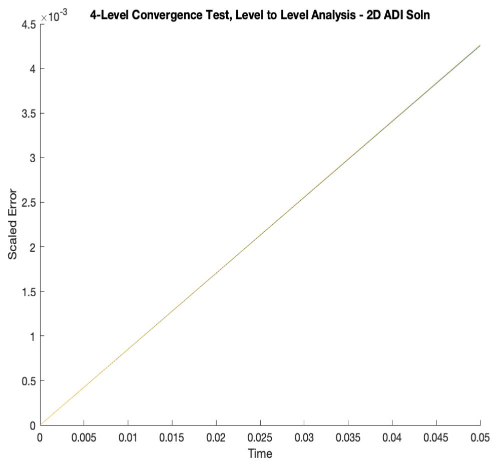
## 4.2 Problem 2: Two-Dimensional Schrödinger Equation

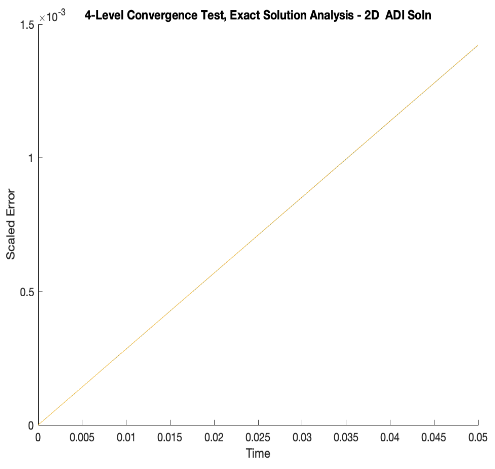
For the 2d Schrödinger equation we completed two convergence tests for one set up of the ADI solution scheme (without any potential so we can compute exact error analysis). And for the numerical experiments videos of consecutive contour plots describing the behaviours of different scenarios and set ups were used to verify the correctness of our ADI implementation. This included cases with barrier, well, and a double slit implementation.

### 4.2.1 Convergence Tests

For the 2d convergence test the one difference which we needed to implement was now calculating the l2 norm of the spatial norm (2 dimensional solution) which was a little bit more involved however the results and process was otherwise identical. The parameters and results of the convergence test is given below:

- tmax = 0.05
- level = 6 to 9
- lambda = 0.05
- idtype = 0
- idpar = $[m_x = 2, m_y = 3]$
- vtype = 0





Similar to the previous convergence tests we see that the errors were all coincidental suggesting that our ADI scheme was also fourth order globally accurate which is what we expected. Likewise the magnitude of these solutions were not particularly small at roughly $10^{-3}$. However, with complex equations like this we would obviously be expecting to be less accurate when compared the solutions which we had analyzed much earlier in the course.

### 4.2.2 Numerical Experiments

The videos have been included in a separate files in the submission in the "videos_and_figures" directory. Below are discussions of what we can see from these videos as well as the parameters that were used to generate these videos. Generally, and specifically for the minimum required videos contour plots were generated for easiest interpretation of what the solution was. This was easier to interpret than a surface plot for example. The scripts used to generate these videos are also included in the submission as "video_..." for the each corresponding video (Ex. "video_barrier.m").

The other thing to note with these contours is that for all the videos the modulus of psi is plotted whereas before we were plotting strictly the real parts of psi in the 1d scenario.

**2D Barrier Video:** The parameters for the barrier were as follows:

- tmax = 0.035
- level = 7
- lambda = 0.05
- idtype = 1, vtype = 1
- idpar = [$x_0 = 0.7$, $y_0 = 0.7$, dx = 0.1, dy = 0.1, px = -50, py = -50]
- vpar = [$x_{min} = 0.3$, $x_{max} = 0.6$, $y_{min} = 0.3$, $y_{max} = 0.6$, $V_c = 10^5$]

The barrier interpretation is clear to see as the initial boosted Gaussian travels down and to the left and then when we see it reach the barrier the values disperse on collision with this wall and continue to spread outwards. Additionally we can see that there is some loss to the total potential which we would expect to see as we have zero value homogeneous boundary conditions so there is not complete conservation of energy in this solution. Another thing to note is that if the potential was selected too large in size ($10^9$) then errors in our scheme arose where the potential would then instantaneously grow from the rectangular barrier as well as the initial Gaussian.

**2D Well Video** The parameters for the well were as follows:

- tmax = 0.035
- level = 8
- lambda = 0.05
- idtype = 1, vtype = 1
- idpar = [$x_0 = 0.5$, $y_0 = 0.5$, dx = 0.1, dy = 0.1, px = 0, py = 0]
- vpar = [$x_{min} = 0.2$, $x_{max} = 0.7$, $y_{min} = 0.2$, $y_{max} = 0.7$, $V_c = -10^3$]

The well interpretation is similar to what we saw in the one dimensional survey however in two dimensions this time. The initial boosted Gaussian lies inside the well here and continues to stay within the well on simulation. In some of the additional videos we also tried to simulate when the potential passes through the well wall on one end but then remains within the wall when reaching the other end of the well. However, this was more difficult to simulate as it required extensive parameter tuning. As a reminder for the well this was also the case where the potential was negative. To note was that if the potential was too large in size (still negative) then the potential would simply bounce off the well when reaching the barrier well as opposed to passing through and then being retained on the other end.

**2D Double Slit Video** The parameters for the double slit were as follows:

- tmax = 0.025
- level = 7
- lambda = 0.05
- idtype = 1, vtype = 2
- idpar = [$x_0 = 0.5$, $y_0 = 0.65$, dx = 0.1, dy = 0.1, px = 0, py = -75]
- vpar = [$x_1 = 0.3$, $x_2 = 0.3075$, $x_3 = 0.5$, $y_{max} = 0.5075$, $V_c = 10^6$]

The double slit varies much more from the previous two videos. In this case we began with our boosted Gaussian travelling leftwards towards the double slit. Upon reaching the double slit which acted similar to the rectangular barrier the sections of the wave which hit the slit would bounce off it while the sections going through the slit would continue until reaching the boundary as well. The behaviour beyond hitting the boundary was relatively chaotic as well like the other videos.

**Additional Videos** Lastly, there were a few additional videos provided in the submission. With regards to the rectangular barrier set up there was just one additional video displaying the results for when we use a more straight wave packet for the boosted Gaussian. This video has no real significant difference to the other video described earlier other than the rectangular barrier being much thinner and we can see the propagation off it in different directions. This video is named rectangularBarrierV2.avi

With regards to the set of well videos there were some other's submitted. Of which there were also the cases where the particle would bounce off the well. This was due to the potential being too large in magnitude (still negative) in which case the particle would not enter the well and it would behave like a barrier. Another video submitted was where the well's potential was not large enough to capture the particle and thus the particle would travel straight through the well with potentially some dissipation.

Lastly, there was one extra video simulated for the double slit implementation. The difference was in the Boosted Gaussian parameters and that the initial size of the particle was elongated to be more of a bar then a dot and still travelled leftwards towards the double slit. The result on hitting the double slit is quite similar although the dispersion post contact is different (as you would expect/hope).

These additional videos were both a part of the process to achieving the selected minimum videos that were included earlier in this report and some were just out of curiosity for what we can simulate. Personally, my favourite was the double slit implementation and the rectangular barrier.