Time–Splitting Spectral Methods for Schrödinger Equations in the Semiclassical Regime

Eva Lott - BSc Project
University of Dundee
Supervised by Dr Agis Athanassoulis

April 12, 2021

Abstract

In quantum mechanics, objects are described by a probability of observing them in a specific state. The wavefunction is an important tool that returns the probabilities of observations in a system when acted on by operators, and the semiclassical Schrödinger equation is used to describe how this function evolves in time. Quantum theory in general is a fundamental part of physics with applications in many different areas of industry and research. The aim of this project is to examine the use of splitting methods and spectral theory on partial differential equations to approximate wavefunction solutions to the Schrödinger equation in the semiclassical regime. Many numerical examples are used to illustrate properties of the approximation schemes.

Contents

1 Introduction

1.1 Topics Covered

The aim of this project is to research splitting methods and apply them to the semiclassical Schrödinger equation, solving split parts with spectral methods, so wavefunctions solutions can be better approximated.

Splitting Methods
Splitting theory describes ways to split given differential equations into mutliple other differential equations, which this paper shall examine in the form of systems of ordinary differential equations (ODEs) in matrix form, and using those principles we will further examine splitting of partial differential equations (PDEs), namely the semiclassical Schrödinger equation. With differential equations being so applicable to many disciplines, more efficient ways to solve them have wide reaching benefits in a large number of areas.

Quantum Mechanics and the Schrödinger Equation
This project focusses on Schrödinger equations in the semiclassical regime, which are equations of the form [7]

$$i\varepsilon\frac{\partial u}{\partial t} + \frac{\varepsilon^2}{2}\nabla^2 u - V(\vec{x})u = 0 \tag{1}$$

This equation will be explained along with neccessary principles in quantum mechanics. This Schrödinger equation descibes the time evolution of a wavefunction, which can be used to provide observation probabilities of its corresponding object. Quantum mechanics is an entirely unavoidable description of nature on a small enough scale, and so methods to solve relevant equations have wide reaching applications in physics. This project will not examine the natural properties the Schrödinger equation or the wavefunction embody in great detail, of which a comprehensive description would detract from the mathematics examined.

Spectral Methods
We will examine spectal methods and how they can be applied to solving PDEs, another important and area of research wth widespread application.

1.2 A Brief History of Topics Covered

Splitting Methods
Splitting methods were developed by Richard Varga in 1960 [1], and they have used in the solving of many iterative methods. It is common to express systems of differential equations in matrix form and many iterative methods have been developed to provide approximate solutions to these equations. They are a useful method for solving of PDEs.

Quantum Mechanics and the Schrödinger Equation
Note: this section features information on the history of quantum mechanics, making use of Brian C Hall's - 'Quantum Theory for Mathematicians' [2] heavily

Quantum theory can best be summerised by the notion that at the smallest level, energy is quantised, that is to say, composed of discrete quanta. In 1900, in an attempt to explain peculiar habits of black body radiation, Max Planck submitted a paper postulating that on the fundamental level energy in an electromagnetic field at a frequency ω is described by packets of quanta which we now call photons, a particle with energy proportional to the frequency. This led to a fresh debate on the nature of light - a wave as had been previously thought, or a particle. In 1905, one of Albert Einstein's seminal papers described the photoelectric effect, a breakthrough for which he would later go on to win the Nobel prize in 1921. Put simply, when electromagnetic radiation (light) strikes a metal, the metal is ionized, meaning electrons are discarded. Einstein found that when the intesity of light is increased, it corresponds to a greater increase in the number of electrons emitted and not an increase in the energy each electron is emitted with. This provided further evidence that light was in fact a particle instead of a wave. The particle behaviour directly contradicted James Clerk Maxwell's equations of electromagnetism, which had reliably described light as a wave.
In actuality, light behaves as both a wave and particle, a wave-particle duality. In 1907 the double slit experiment was revisited - this had provided evidence of the wave like nature of light in the previous century. It was postulated by Paul Dirac that the interference seen when light was passed through two small holes was individual photons interfering with themselves causing a single photon to pass through both slits simultaneously and not, as had been suspected over one hundred years prior, interference from the light coming through two holes.
It was paradoxical experiments such as these that lead to Heisenberg suggesting in 1925 that a particle can and should be described entirely by its position and momentum in order to explain the odd behaviour in quantum mechanics. The neccessity of a wavefunction for descibing matter came the following year by Schrödinger in a series of papers, he also described how it evolves through time with the Schrödinger equation, which this project aims to approximate the solutions of using splitting methods.

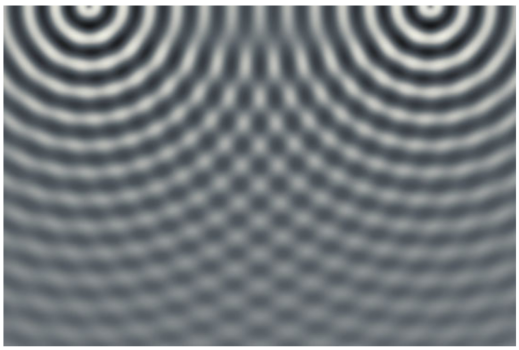


Figure 1: Interference patterns in the double slit experiment [2]

Spectral Methods
Spectral methods for solving PDEs were largerly developed by Steve Orszag and David Gottlieb in the 1950s and 1960s, though many mathematicians contributed [4]. This paper will focus on Fourier series and Fourier transform methods, discovered by Joseph Fourier in the Early 1800s [5].

2 Ordinary Differential Equation Splitting

In order to understand splitting methods of PDEs we will first explore splitting methods in ODEs.

2.1 Splitting Methods

We seek solutions to

$$\frac{du}{dt} = f(u), \qquad t \in [T_0, T_1] \tag{2}$$

where the following criteria is true [3]

- We can make a 'split' of $f(u)$ into k different summed functions $f(u) = f_1(u) + f_2(u) + \cdots + f_k(u)$
- We can solve for u for all of the functions $f_1, f_2, \cdots, f_k$ analytically when they equal $\frac{du}{dt}$
- We have an Dirichlet initial condition, such that the value of $u(T_0)$ is given

Theorem 2.1 (Domain discretization [6])**.**
Suppose we have a function

$$u(t), \quad t \in [T_0, T_1] \tag{3}$$

We can discretize the domain into N parts, with a timestep increment $\delta = \frac{T_1 - T_0}{N}$ such that the for the nth timestep $t_n = T_0 + n\delta$. We then describe the function in terms of a difference equation $u(t_n)$, such that there are a finite number of points which can be solved for to approximate the continuous function. Many approximation methods in numerical analyis rely on discretising the domain.

.
We can develop a splitting approximation for f by discretising the domain $t \in [T_0, T_1]$ into N timesteps

$$\delta = \frac{T_1 - T_0}{N}, \text{ such that } t_n = T_0 + n\delta, \quad t_0 = T_0, \quad t_N = T_1 \tag{4}$$

and repeatedly iterating through and solving each $f_1, f_2, \ldots, f_k$ [3]. For each timestep $t \in [t_n, t_{n+1}]$ we solve an initial value problem (IVP) with an initial condition given at the beginning of the timestep $u(t_n)$ [3].

For each timestep, we need to solve all the smaller subproblems to find solutions of u.

$$\frac{du_1}{dt} = f_1(u_1), \qquad u_1(t_n) = u(t_n), \qquad t \in [t_n, t_{n+1}] \tag{5}$$

For each timestep we have the approximate solution of the previous timestep $u(t_n)$ as an initial condition, with the exception of the first timestep, where we use the Dirichlet condition given by the problem $u(T_0)$. The idea is then to use our solution to equation 5 at t_{n+1} as an initial condition for the f_2 system, developing the IVP

$$\frac{du_2}{dt} = f_2(u_2) \qquad u_2(t_n) = u_1(t_{n+1}), \qquad t \in [t_n, t_{n+1}] \tag{6}$$

then $u_3(t_n) = u_2(t_{n+1})$ and we can continue through as many splitted

$$f_3, f_4, \cdots, f_k \tag{7}$$

as we choose, calculating solutions for

$$u_3(t_{n+1}), u_4(t_{n+1}), \ldots, u_k(t_{n+1}) \tag{8}$$

using the initial conditions

$$u_2(t_{n+1}), u_3(t_n{+}1), \ldots, u_{k-1}(t_{n+1}) \tag{9}$$

then an approximation of our original function u in equation 2, at constant time t_{n+1} is given by $u(t_{n+1}) \approx u_k(t_{n+1})$. We can go to the next iteration to find $u(t_{n+2})$ then we use our $u(t_{n+1})$ as an initial condition for the first split problem u_1 [3].

We can repeat this process on all the below subdomains, starting left and going right, using our Dirichlet initial condition. We use the given value of $u(T_0)$ to solve $u_1(t_1)$, and using our approximation of a timestep $u(t_n)$ as an initial condition to u_1 on $[t_n, t_{n+1}]$ [3].

$$[T_0, t_1], [t_1, t_2], [t_2, t_3], \cdots, [t_n, t_{n+1}], \cdots, [T_1 - \delta, T_1] \tag{10}$$

2.2 The Matrix Exponential

The variable u need not be scalar field, if we have a system of ODEs we can use a splitting method on them in matrix and vector form, which requires the matrix exponential [1].

Consider the differential equation

$$\frac{dx}{dt} = ax \tag{11}$$

for scalar $x(t)$ and a. The solution to this differential equation is given by $x(t) = \exp(at)c$, where c is an arbitrary constant which we can eliminate and find a particular solution given an initial condition. We can use a matrix exponential to solve systems of equations (matrix a and vector x) in a similar fashion [3].

The matrix exponential is used to solve systems of ODEs in matrix from We define a vector whose elements are dependent on t, $\vec{x}(t) \in \mathbb{F}^n$, and matrix $\boldsymbol{A} \in M_{n\times n}(\mathbb{F})$ (an element of the set of all $n \times n$ matrices whose elements are $\mathbb{F}$). Usually $\mathbb{F} = \mathbb{C}$ or $\mathbb{R}$.

We can define the system of differential equations

$$\frac{d\vec{x}}{dt} = \boldsymbol{A}\vec{x} \tag{12}$$

This matrix can be solved for a general solution using the matrix exponential [3]

$$\vec{x}(t) = \exp(\boldsymbol{A}t)\vec{c} \tag{13}$$

where $\vec{c}$ is a vector of arbitrary constants and a particular solution can be imposed by considering initial conditions.

Definition 2.1 (Matrix exponential [3])**.**

$$\exp(\boldsymbol{A}) = \boldsymbol{I} + \boldsymbol{A} + \frac{\boldsymbol{A}^2}{2!} + \frac{\boldsymbol{A}^3}{3!} + \cdots = \sum_{n=0}^{\infty} \frac{\boldsymbol{A}^n}{n!} \tag{14}$$

Theorem 2.2 (Differential of matrix exponential)**.**
The matrix exponential provides the same eigenfunction property as the scalar exponential. Substituting $\vec{x}(t) = \exp(\boldsymbol{A}t)\vec{c}$

$$\begin{aligned}
\frac{d}{dt}\exp(\boldsymbol{A}t)\vec{c} &= \frac{d}{dt}(\boldsymbol{I} + \boldsymbol{A}t + \frac{\boldsymbol{A}^2t^2}{2!} + \frac{\boldsymbol{A}^3t^3}{3!} + \frac{\boldsymbol{A}^4t^4}{4!} + \cdots)\vec{c} \\
&= (\boldsymbol{A} + 2\frac{\boldsymbol{A}^2t}{2!} + 3\frac{\boldsymbol{A}^3t^2}{3!} + 4\frac{\boldsymbol{A}^4t^3}{4!} + \cdots)\vec{c} \\
&= \boldsymbol{A}(I + \boldsymbol{A}t + \frac{\boldsymbol{A}^2t^2}{2!} + \frac{\boldsymbol{A}^3t^3}{3!} + \cdots)\vec{c} \\
&= \boldsymbol{A}\exp(\boldsymbol{A}t)\vec{c}
\end{aligned} \tag{15}$$

So, we can use the matrix exponential to solve systems of ODEs.

2.3 Splitting Matrices

It may be more convenient to split an ODE matrix system $\boldsymbol{A}$ into multiple other matrices, maybe its ODE system is non-analytic, or perhaps it has been noticed that diagonalised matrices generated from it are easier to solve. The splitting method centres around describing $\boldsymbol{A}$ as the sum of other split matrices

$$\boldsymbol{A} = \boldsymbol{A}_1 + \boldsymbol{A}_2 + \cdots + \boldsymbol{A}_k \tag{16}$$

and solving [3]

$$\frac{d\vec{x}}{dt} = \boldsymbol{A}_1\vec{x}, \quad \frac{d\vec{x}}{dt} = \boldsymbol{A}_2\vec{x}, \quad \cdots, \quad \frac{d\vec{x}}{dt} = \boldsymbol{A}_k\vec{x} \tag{17}$$

Theorem 2.3 (Commutation of exponential [3]).
For a scalar exponential, with $a, b \in \mathbb{F}$, and under scalar multiplication $\cdot$

$$\exp(a\delta) \cdot \exp(b\delta) = \exp((a+b)\delta) \tag{18}$$

For a matrix exponential, with $\boldsymbol{A}, \boldsymbol{B} \in M_{n\times n}(\mathbb{F})$, and under matrix multiplication $\times$, in general

$$\exp(\boldsymbol{A}\delta) \times \exp(\boldsymbol{B}\delta) \neq \exp((\boldsymbol{A}+\boldsymbol{B})\delta) \tag{19}$$

2.4 Local Error

We shall examine how the non-commutative nature of this exponential gives rise to error.

Definition 2.2 (Local error [3]).
The local error E_l of a matrix exponential split $\boldsymbol{A} = \boldsymbol{A}_1 + \boldsymbol{A}_2$ is the discrepency between $\exp(\boldsymbol{A}\delta)$ and $\exp(\boldsymbol{A}_1\delta) \times \exp(\boldsymbol{A}_2\delta)$

$$E_l = \exp(\boldsymbol{A}\delta) - \exp(\boldsymbol{A}_1\delta)\exp(\boldsymbol{A}_2\delta) \tag{20}$$

Theorem 2.4 (Deriving error).
Consider a matrix $\boldsymbol{A}$ which we split into $\boldsymbol{A}_1 + \boldsymbol{A}_2$

$$\begin{aligned}
&E_l(\vec{x},\delta) = \exp(\boldsymbol{A}\delta) - \exp(\boldsymbol{A}_1\delta)\exp(\boldsymbol{A}_2\delta) = \exp((\boldsymbol{A}_1+\boldsymbol{A}_2)\delta) - \exp(\boldsymbol{A}_1\delta)\exp(\boldsymbol{A}_2\delta)\\
&= (\boldsymbol{I} + (\boldsymbol{A}_1+\boldsymbol{A}_2)\delta + \frac{(\boldsymbol{A}_1+\boldsymbol{A}_2)^2\delta^2}{2!} + \frac{(\boldsymbol{A}_1+\boldsymbol{A}_2)^3\delta^3}{3!} + \frac{(\boldsymbol{A}_1+\boldsymbol{A}_2)^4\delta^4}{4!} + \mathcal{O}(\delta^5))\\
&\quad - (\boldsymbol{I} + \boldsymbol{A}_1\delta + \frac{(\boldsymbol{A}_1\delta)^2}{2!} + \frac{(\boldsymbol{A}_1\delta)^3}{3!} + \frac{(\boldsymbol{A}_1\delta)^4}{4!} + \mathcal{O}(\delta^5))(\boldsymbol{I} + \boldsymbol{A}_2\delta + \frac{(\boldsymbol{A}_2\delta)^2}{2!} + \frac{(\boldsymbol{A}_2\delta)^3}{3!} + \frac{(\boldsymbol{A}_2\delta)^4}{4!}) + \mathcal{O}(\delta^5))\\
&= \delta^2\frac{(\boldsymbol{A}_1+\boldsymbol{A}_2)^2}{2!} + \delta^3\frac{(\boldsymbol{A}_1+\boldsymbol{A}_2)^3}{3!} + \delta^4\frac{(\boldsymbol{A}_1+\boldsymbol{A}_2)^4}{4!}\\
&\quad - \delta^2(\frac{\boldsymbol{A}_1^2+\boldsymbol{A}_2^2}{2!} + \boldsymbol{A_1}\boldsymbol{A}_2) - \delta^3(\frac{\boldsymbol{A}_1^3+\boldsymbol{A}_2^3}{3!} + \frac{\boldsymbol{A}_1^2\boldsymbol{A}_2+\boldsymbol{A}_1\boldsymbol{A}_2^2}{2!}) - \delta^4(\frac{\boldsymbol{A}_1^4+\boldsymbol{A}_2^4}{4!} + \frac{\boldsymbol{A}_1^3\boldsymbol{A}_2+\boldsymbol{A}_1\boldsymbol{A}_2^3}{3!} + \frac{\boldsymbol{A}_1^2\boldsymbol{A}_2^2}{2!2!}) + \mathcal{O}(\delta^5)\\
&= \frac{\delta^2}{2}[\boldsymbol{A}_1,\boldsymbol{A}_2] + \frac{\delta^3}{12}([\boldsymbol{A}_1,[\boldsymbol{A}_1,\boldsymbol{A}_2]] - [\boldsymbol{A}_2,[\boldsymbol{A}_1,\boldsymbol{A}_2]]) + \frac{\delta^4}{24}[\boldsymbol{A}_2,[\boldsymbol{A}_1,[\boldsymbol{A}_1,\boldsymbol{A}_2]]] + \mathcal{O}(\delta^5)
\end{aligned} \tag{21}$$

where $\boldsymbol{A}_1\boldsymbol{A}_2 - \boldsymbol{A}_2\boldsymbol{A}_1$ is known as the "commutator" which we denote $[\boldsymbol{A}_1, \boldsymbol{A}_2]$ [3].

It has been shown that because $\boldsymbol{B} \times \boldsymbol{C}$ is not generally commutative, our matrix exponential splitting method will be in the form of

$$\exp(\boldsymbol{B}\delta + \boldsymbol{C}\delta) = \exp(\boldsymbol{B}\delta)\exp(\boldsymbol{C}\delta) + E_l(\vec{x},\delta) \tag{22}$$

Conversly if the matrices $\boldsymbol{B}$ and $\boldsymbol{C}$ are commutitive then $E_l(\vec{x},\delta) = 0$ however this is not the general case. Error can also be calculated more generally with Baker-Campbell-Hausdorff formula.

Theorem 2.5 (Baker-Campbell-Hausdorff formula [8]).
The Baker-Campbell-Hausdorff formula provides a solution for some $\boldsymbol{A}(\boldsymbol{B},\boldsymbol{C}) = \ln(\exp(\boldsymbol{B})\exp(\boldsymbol{C}))$ for non-commutative quantities $\boldsymbol{B}$ and $\boldsymbol{C}$.

$$\begin{aligned}
\ln(\exp(\boldsymbol{B})\exp(\boldsymbol{C})) &= \boldsymbol{B} + \boldsymbol{C} + \int_0^1 \sum_{n=1}^{\infty} \frac{(\boldsymbol{I} - \exp(ad_{\boldsymbol{B}})\exp(t\cdot ad_{\boldsymbol{C}}))^{n-1}}{n(n+1)} \frac{\exp(ad_{\boldsymbol{B}}-\boldsymbol{I})}{ad_{\boldsymbol{B}}}[\boldsymbol{B},\boldsymbol{C}]dt\\
&= \boldsymbol{B} + \boldsymbol{C} + \frac{1}{2}[\boldsymbol{B},\boldsymbol{C}] + \frac{1}{12}ad_{\boldsymbol{B}-\boldsymbol{C}}[\boldsymbol{B},\boldsymbol{C}] - \frac{1}{24}ad_{\boldsymbol{C}}ad_{\boldsymbol{B}}[\boldsymbol{B},\boldsymbol{C}] + \cdots
\end{aligned} \tag{23}$$

using the adjoint operator (in Lie algebra) $ad_{\boldsymbol{B}}\boldsymbol{C} = [\boldsymbol{B},\boldsymbol{C}]$, such that $ad_{\boldsymbol{B}}[\boldsymbol{B},\boldsymbol{C}] = [\boldsymbol{B},[\boldsymbol{B},\boldsymbol{C}]]$ etc.

For matrix splitting in MATLAB we will use

```
1  expm(A*delt) - expm(B*delt)*expm(C*delt)
```

to calculate local error.

2.5 Popular Types of Splitting

We will introduce a piece of notation, let U^n denote approximate solutions to equation u at time t_n

$$U^n \approx u(t_n) \tag{24}$$

Many different splitting methods have been used to solve ODEs.

Theorem 2.6 (Lie-Trotter splitting [9])**.**
Lie-Trotter seeks solutions by splitting

$$\frac{du}{dt} = \boldsymbol{A}u(t), \quad t \in [T_0, T_1], \quad u(T_0) = U^0 \rightarrow \begin{cases} \dfrac{du_1(t)}{dt} = \boldsymbol{A}_1 u_1(t), & t \in [T_0, T_1] \\ \dfrac{du_2(t)}{dt} = \boldsymbol{A}_2 u_2(t), & t \in [T_0, T_1] \end{cases} \tag{25}$$

where we solve the split equations for each given timestep $[t_n, t_{n+1}]$ according to

1. $U_1^{n+1} = \exp(\delta \boldsymbol{A}_1) U_*^n, \quad U_*^n = \begin{cases} U^0 \text{ the given initial condition in the problem, for } n = 0 \\ U_2^n \text{ the solution for } U_2 \text{ at the previous timestep, for } n > 0 \end{cases}$
2. $U_2^{n+1} = \exp(\delta \boldsymbol{A}_2) U_1^{n+1}$
3. $U^{n+1} \approx U_2^{n+1}$

where $U_1^n = u_1(t_n)$ and $U_2^n = u_2(t_n)$ each of which we solve on a timestep subject to the initial condition of the solution of the other.

In terms of MATLAB [10], with split matrices `B` and `C`:

```
function ret = LieTrotter (delt, B, C, ini, T0, T1)
   n = 0;
   x_ini(:,1) = ini;

   domain = (T1 - T0)/delt;
   while domain >= T0 + n*delt
      x(:,n+1) = x_ini(:,n+1);
      x(:,n+2) = expm(B*delt)*x(:,n+1);
      y(:,n+1) = x(:,n+2);
      y(:,n+2) = expm(C*delt)*y(:,n+1);
      x_ini(:,n+2) = y(:,n+2);
      n = n+1;
   end
   ret = x;
end
```

Theorem 2.7 (Strang splitting [9])**.**
Strang splitting seeks solutions by splitting

$$\frac{du}{dt} = \boldsymbol{A}u(t), \quad t \in [T_0, T_1], \quad u(T_0) = U^0 \rightarrow \begin{cases} \dfrac{du_1(t)}{dt} = \dfrac{1}{2}\boldsymbol{A}_1 u_1(t), & t \in [T_0, T_1] \\ \dfrac{du_2(t)}{dt} = \boldsymbol{A}_2 u_2(t), & t \in [T_0, T_1] \\ \dfrac{du_3(t)}{dt} = \dfrac{1}{2}\boldsymbol{A}_1 u_1(t), & t \in [T_0, T_1] \end{cases} \tag{26}$$

This is a similar method to Lie-Trotter but rather than solving each splitted equation for a full timestep, we solve the first for a half timestep, then the second for a full timestep, and lastly the first again for another half timestep.

We solve the split equations for each given timestep $[t_n, t_{n+1}]$ according to

1. $U_1^{n+1} = \exp(\frac{\delta}{2}\boldsymbol{A}_1)U_*^n, \quad U_*^n = \begin{cases} U^0 \text{ the given initial condition in the problem, for } n = 0 \\ U_3^n \text{ the solution for } U_3 \text{ at the previous timestep, for } n > 0 \end{cases}$
2. $U_2^{n+1} = \exp(\delta\boldsymbol{A}_2)U_1^{n+1}$
3. $U_3^{n+1} = \exp(\frac{\delta}{2}\boldsymbol{A}_1)U_2^{n+1}$
4. $U^{n+1} \approx U_3^{n+1}$

This is equivalent to saying we solve the equations over a half timestep, meaning our δ in steps 1 and 3 is halved

1. $U_1^{n+\frac{1}{2}} = \exp(\delta_{\frac{1}{2}}\boldsymbol{A}_1)U_*^n, \quad U_*^n = \begin{cases} U^0 \text{ the given initial condition in the problem, for } n = 0 \\ U_3^n \text{ the solution for } U_3 \text{ at the previous timestep, for } n > 0 \end{cases}$
2. $U_2^{n+1} = \exp(\delta\boldsymbol{A}_2)U_1^{n+1}$
3. $U_3^{n+\frac{1}{2}} = \exp(\delta_{\frac{1}{2}}\boldsymbol{A}_1)U_2^{n+1}$
4. $U^{n+1} \approx U_3^{n+\frac{1}{2}}$

where $\delta_{\frac{1}{2}}$ is a half timstep $\frac{\delta}{2}$.

In terms of MATLAB [10], with split matrices `B` and `C`:

```
function ret = Strang (delt, B, C, ini, T0, T1)
   n = 0;
   x_ini(:,1) = ini;

   domain = (T1 - T0)/delt;
   while domain >= T0 + n*delt
      x(:,n+1) = x_ini(:,n+1);
      x(:,n+2) = expm(0.5*B*delt)*x(:,n+1);

      y(:,n+1) = x(:,n+2);
      y(:,n+2) = expm(C*delt)*y(:,n+1);

      x(:,n+2) = expm(0.5*B*delt)*y(:,n+2);

      x_ini(:,n+2) = x(:,n+2);
      n = n+1;
   end
   ret = x;
end
```

2.6 Numerical Examples of Matrix Splitting

Example 2.1 (Analytic solution)**.**
Suppose we want to solve the system of differential equations [11]

$$\frac{dx_1(t)}{dt} = x_1 + 2x_2, \qquad \frac{dx_2(t)}{dt} = -x_1 + x_2, \qquad x_1(0) = 2, \qquad x_2(0) = -1 \tag{27}$$

with the corresponding matrix form

$$\frac{d\vec{x}}{dt} = \begin{pmatrix} x_1'(t) \\ x_2'(t) \end{pmatrix} = \begin{pmatrix} 1 & 2 \\ -1 & 1 \end{pmatrix} \begin{pmatrix} x_1(t) \\ x_2(t) \end{pmatrix} = \boldsymbol{A}\vec{x}, \qquad \vec{x}(0) = \begin{pmatrix} 2 \\ -1 \end{pmatrix} \tag{28}$$

We can use the standard method of solving systems of ODEs in MATLAB [12][11], or just a matrix exponential, to get the analytic solution

$$\vec{x} = \begin{pmatrix} 2\exp(t)\cos(2^{\frac{1}{2}}t) - 2^{\frac{1}{2}}\exp(t)\sin(2^{\frac{1}{2}}t) \\ -\exp(t)\cos(2^{\frac{1}{2}}t) - 2^{\frac{1}{2}}\exp(t)\sin(2^{\frac{1}{2}}t) \end{pmatrix} \tag{29}$$

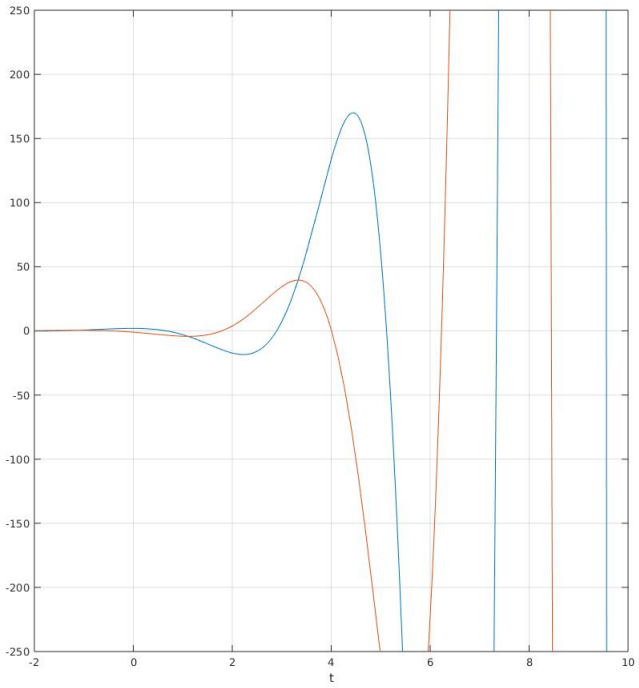


Figure 2: Solution curve of the function $\vec{x}$, x_1 in red, x_2 in blue over $t \in [-2, 10]$ [12]

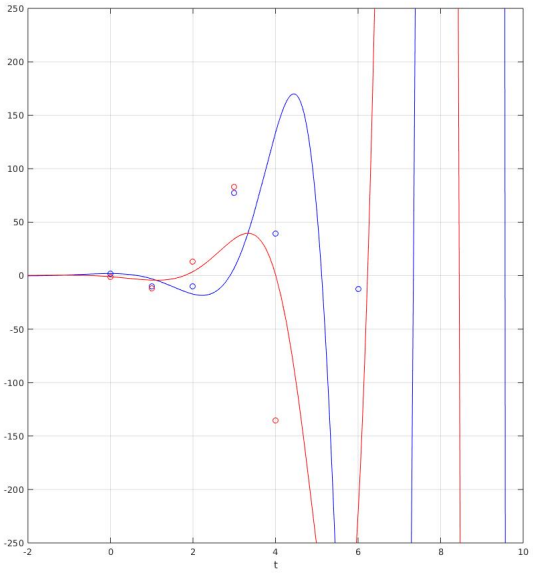
Which we can generate with MATLAB code [12]:

```
syms x(t) y(t)
A = [1 2; -1 1];
X = [x; y];
odeSystem = diff(X) == A*X;
ini = X(0) == [2; -1];
sols = dsolve(odeSystem,ini);
```

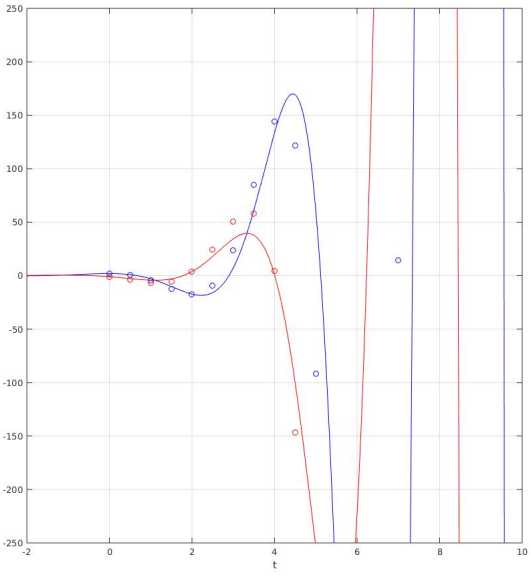
Example 2.2 (Lie-Trotter splitting)**.**
We will approximate solutions to example 2.1 with MATLAB using a Lie-Trotter splitting approximation [10]. Making the split

$$\begin{pmatrix} 1 & 2 \\ -1 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ -1 & 0 \end{pmatrix} + \begin{pmatrix} 0 & 2 \\ 0 & 1 \end{pmatrix} \tag{30}$$

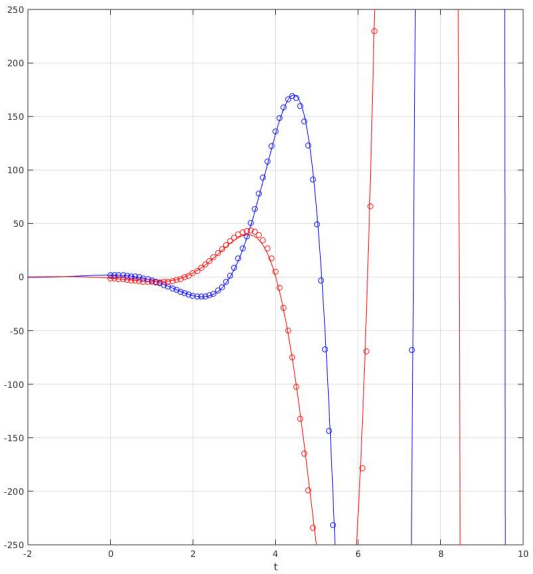
We then discretize time and iteratively calculate as we did in theorem 2.6 over $T_0 = 0$, $T_1 = 10$. Plotting solutions for different timesteps can help understand error behavior.



(a) $\delta = 1,\ N = 10$



(b) $\delta = 0.5,\ N = 20$



(c) $\delta = 0.1,\ N = 100$

Commutative error for each timestep value

$$\delta = 1,\ N = 10 \qquad E_l = \begin{pmatrix} -2.2944 & -5.5443 \\ -0.1803 & 3.6106 \end{pmatrix}$$

$$\delta = 0.5,\ N = 20 \qquad E_l = \begin{pmatrix} -0.3953 & -0.6244 \\ -0.1086 & 0.4464 \end{pmatrix}$$

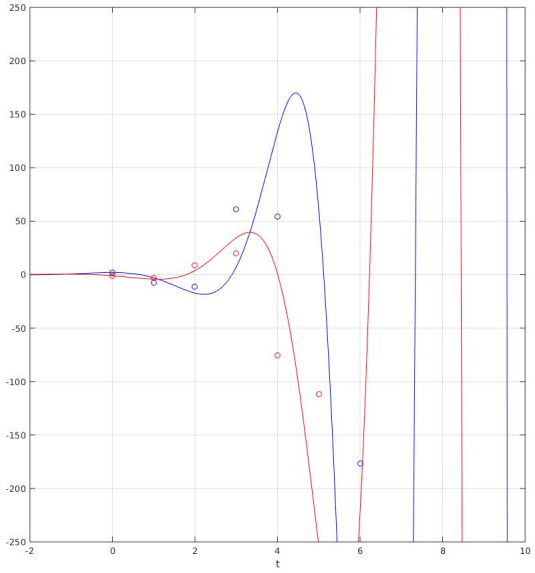
$$\delta = 0.1,\ N = 100 \qquad E_l = \begin{pmatrix} -0.0110 & -0.0122 \\ -0.0050 & 0.0111 \end{pmatrix}$$

Figure 3: Solution curve of the function $\vec{x}$, x_1 in red, x_2 in blue over $t \in [-2, 10]$ [10]

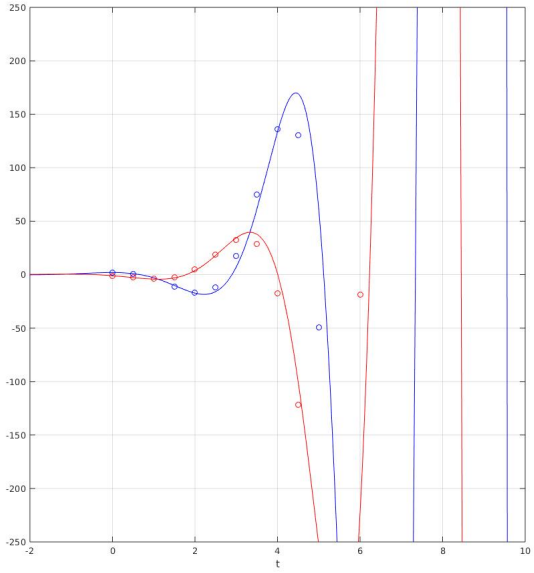
Example 2.3 (Strang splitting)**.**
Just like example 2.2 We will approximate solutions to example 2.1 with MATLAB, instead using a Strang

splitting approximation [10]. We split

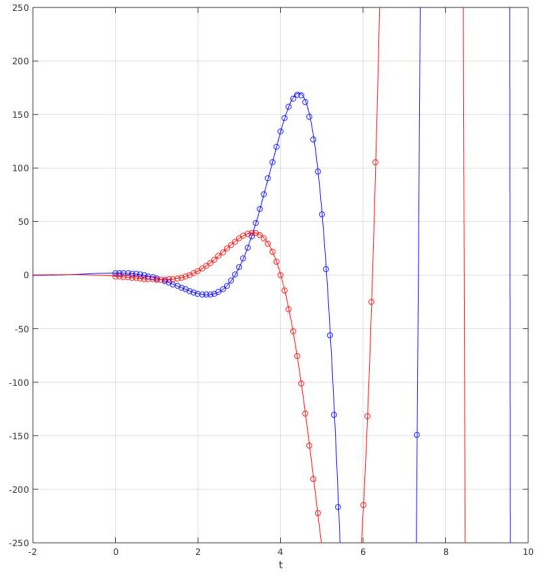
$$\begin{pmatrix} 1 & 2 \\ -1 & 1 \end{pmatrix} = \frac{1}{2}\begin{pmatrix} 1 & 0 \\ -1 & 0 \end{pmatrix} + \begin{pmatrix} 0 & 2 \\ 0 & 1 \end{pmatrix} + \frac{1}{2}\begin{pmatrix} 1 & 0 \\ -1 & 0 \end{pmatrix} \tag{31}$$



(a) $\delta = 1,\ N = 10$



(b) $\delta = 0.5,\ N = 20$



(c) $\delta = 0.1,\ N = 100$

Commutative error for each timestep value, this does not change with increase to the order of the method from Lie-Trotter, this is the local error.

$$\delta = 1,\ N = 10 \qquad E_l = \begin{pmatrix} -2.2944 & -5.5443 \\ -0.1803 & 3.6106 \end{pmatrix}$$

$$\delta = 0.5,\ N = 20 \qquad E_l = \begin{pmatrix} -0.3953 & -0.6244 \\ -0.1086 & 0.4464 \end{pmatrix}$$

$$\delta = 0.1,\ N = 100 \qquad E_l = \begin{pmatrix} -0.0110 & -0.0122 \\ -0.0050 & 0.0111 \end{pmatrix}$$

Figure 4: Solution curve of the function $\vec{x}$, x_1 in red, x_2 in blue over $t \in [-2, 10]$ [10]

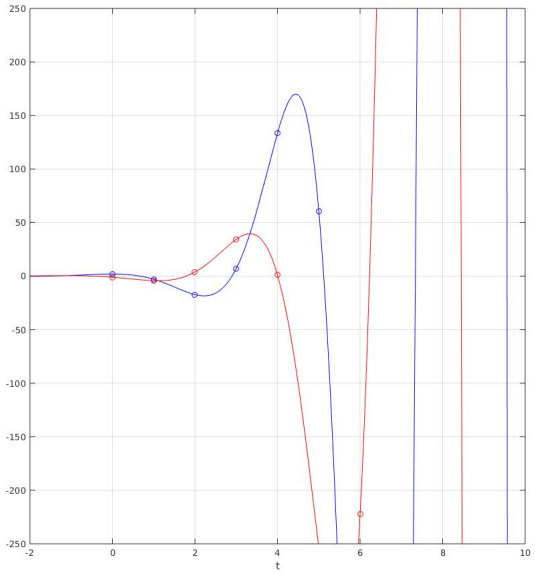
Example 2.4 (Commutative case)**.**
Consider the split

$$\begin{pmatrix} 1 & 2 \\ -1 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} + \begin{pmatrix} 0 & 2 \\ -1 & 0 \end{pmatrix} \tag{32}$$

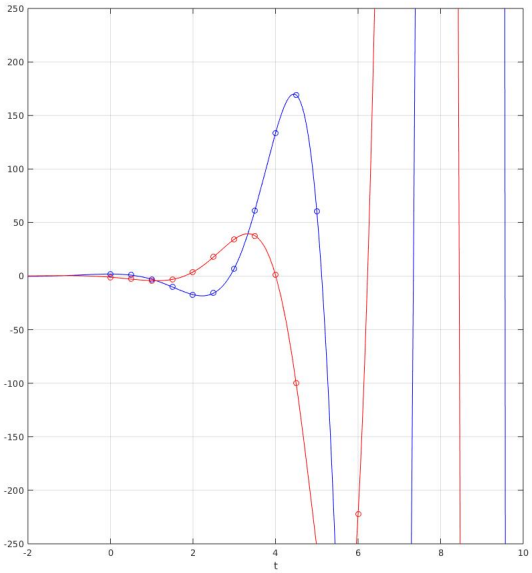
This the split matrices are clearly commutitive since

$$\boldsymbol{I} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \iff \boldsymbol{IA} = \boldsymbol{AI} = \boldsymbol{A} \quad \forall \boldsymbol{A} \in M_{2\times 2}(\mathbb{R}) \tag{33}$$

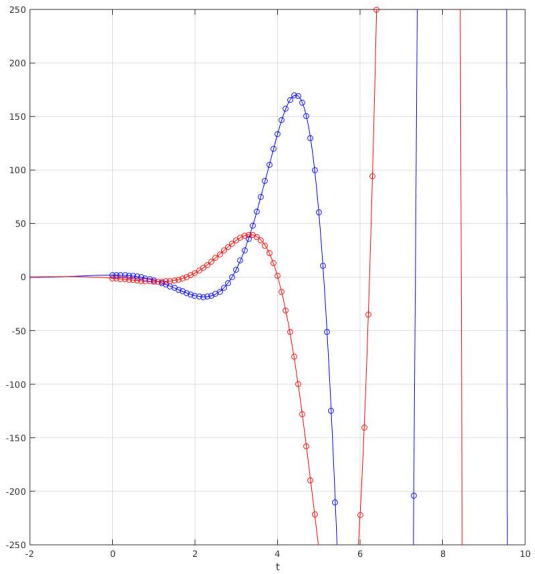
So we have no local error for any timestep.



(a) $\delta = 1,\ N = 10$



(b) $\delta = 0.5,\ N = 20$



(c) $\delta = 0.1,\ N = 100$

Commutative error is 0 for any timestep

$$\delta = 1,\ N = 10 \qquad E_l = \begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix}$$

$$\delta = 0.5,\ N = 20 \qquad E_l = \begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix}$$

$$\delta = 0.1,\ N = 100 \qquad E_l = \begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix}$$

Figure 5: Solution curve of the function $\vec{x}$, x_1 in red, x_2 in blue over $t \in [-2, 10]$, notice there is no error for any timestep [10]

3 Partial Differential Equation Splitting

We now transition into describing splitting in PDEs, which we will need to tackle the Schrödinger equation since it is descibed by partial derivatives.

3.1 Partial Differential Equations

Partial differential equations are the multivariable general version of ODEs. Where ODEs are in the form of sums of differentials of single variable functions representing rate of change of functions with only one dependent variable, partial differentiation represents the rate of change of a function with respect to one of its variables, even though it may be dependent on multiple variables. The Schrödinger equation is a famous example though PDEs exist in many different areas of science and engineering. The heat equation (as discussed below), the Navier-Stokes equation in fluid dynamics, Maxwell's equations in electrodynamics.

Theorem 3.1 (Multivariable domain discretization [6])**.**
consider a multivariable function

$$u(x,t), \quad x \in [X_0, X_1], \quad t \in [T_0, T_1] \tag{34}$$

We now have to discretize over both variables t and x, we select N_x x points in our discretized domain and N_t t points, corresponding to spacestep and timestep

$$\begin{cases} \delta_x = \dfrac{X_1 - X_0}{N_x}, \quad \delta_t = \dfrac{T_1 - T_0}{N_t} \\ x_j = X_0 + \delta_x j, \quad t_n = T_0 + \delta_t n \end{cases} \tag{35}$$

3.2 Heat Equation and PDE Splitting

We introduce PDE splitting (and later Fourier spectral methods) in terms of the heat equation. This equation is a common example since it is so widely found in nature and was among the first PDEs to be solved with Fourier analysis methods.

Definition 3.1 (Time-dependent heat equation)**.**
The time-dependent heat equation is of the form [15]

$$\frac{\partial u}{\partial t} = \alpha \nabla^2 u \tag{36}$$

where u is dependent on $\vec{x}$ and t, traditionally the termperature of a system at time t and position $\vec{x}, = (x_1, x_2, \ldots, x_d)$, ∇^2 is the Laplacian $\nabla^2 = \frac{\partial^2}{\partial {x_1}^2} + \frac{\partial^2}{\partial {x_2}^2} + \cdots + \frac{\partial^2}{\partial {x_d}^2}$ for d spatial dimensions, and $\alpha \in \mathbb{R} \backslash \{0\}$ is a diffusion constant.

This form of heat equation is time dependent as $\frac{\partial u}{\partial t} \neq 0$. When $\frac{\partial u}{\partial t} = 0$ we have the steady-state heat equation. Often $d = 3$ for physical models since cartesian space in nature consists of three orthonormal bases $\vec{x} = (x, y, z)$.

Definition 3.2 (Semilinear heat equation)**.**
The semilinear heat equation is the heat equation with an extra u dependent function f [6]

$$\frac{\partial u}{\partial t} = \nabla^2 u + f(u) \tag{37}$$

If we were using this equation to model heat then $f(u)$ could be some heat source dependent on temperature u, in general it is a source/sink term.

Example 3.1 (Approximating a PDE without splitting)**.**
Consider the semilinear heat equation

$$\frac{\partial u}{\partial t} = \nabla^2 u + f(u), \qquad u(\vec{x}, t), \qquad t \in [T_0, T_1],\ \vec{x} \in \mathbb{R}^d \tag{38}$$

with a given initial condition for $u(\vec{x}, T_0)$.

A traditional method of approximating solutions to this PDE is to discretize time [6].

We discretize $t \in [T_0, T_1]$ into N_t time elements and consider a specific finite spatial domain which we can discretize, with timestep increment δ_t and spacestep increment δ_x. Then we approximate the time derivative using Crank-Nicolson [6]

$$\frac{\partial u}{\partial t}\Big|_{(x_j, t_n)} \approx \frac{U_j^{n+1} - U_j^n}{\delta_t} \tag{39}$$

and then we approximate the PDE at t_n as [6]

$$\frac{U_j^{n+1} - U_j^n}{\delta_t} - \nabla_{\delta_x}^2 U_j^{n+1} = P_{\delta_x} f(U_j^{n+1}) \tag{40}$$

with $U_j^0 = P_{\delta_x} U_j^0$ and P_{δ_x} is some mapping to the finite element space, an approximation of $\nabla^2 u$ [6].

Example 3.2 (Approximating a PDE with splitting)**.**
We can instead use splitting to solve the semilinear heat equation [6]

$$\frac{\partial u}{\partial t} = \nabla^2 u + f(u) \rightarrow \begin{cases} \dfrac{\partial u}{\partial t} = \nabla^2 u \\ \dfrac{\partial u}{\partial t} = f(u) \end{cases} \tag{41}$$

and solve each problem on the same timestep, once again using initial conditions from the previous to solve the next. This would be a Lie-Trotter splitting scheme for this PDE.

Splitting can provide far superior performance to this Crank-Nicolson finite element method, since we can solve both the split equations analytically, as we will examine in section 5.

4 Quantum Mechanics

In this section, neccessary details of quantum mechanics are described so the importance and purpose of the Schrödinger equation can be understood. Beyond the prior brief history we aren't going to explore the physical neccessity of quantum mechanics to describe nature. We will treat the explored concepts as purely mathematical and not physical, since a true physical understanding of quantum mechanics would take considerably more description and would detract from the aim of this project.

4.1 Time Independent Wavefunction Operators

The idea behind the wavefunction is to encapsulate everything about the physcial nature of a quantum system with a function, which can be acted on using operators in order to derive information about the system.

Definition 4.1 (Time independent wavefunction)**.**
Consider a time independent wavefunction ψ acting on one spatial dimension $x \in \mathbb{R}$ such that

$$\psi : \mathbb{R} \to \mathbb{C} \tag{42}$$

the $\mathbb{C}$ the wavefunction maps to is the probability amplitude of the system, operators are defined to derive to information about the system given by the wavefunction [2].

Definition 4.2 (Position density operator)**.**
The position density of wavefunction ψ is given by $|\psi|^2$, where $|\cdot|$ is the magnitude of complex ψ [2]. This gives a probability density function from the wavefunction. It is standard to have the constraint that the position density is normalised

$$\int_{-\infty}^{\infty} |\psi(x)|^2 dx = 1 \tag{43}$$

Definition 4.3 (Momentum operator)**.**
The momentum operator (of a time independent wavefunction with only one spatial dependent variable) is

$$P = -i\hbar \frac{d}{dx} \tag{44}$$

where $\hbar = \frac{h}{2\pi}$, for Planck's constant h [2].

4.2 Schrödinger Equation

Our wavefunction probability amplitude can also be time dependent, and also depend on more than one spatial dimension.

Definition 4.4 (Time dependent wavefuntion)**.**
The general time dependent wavefuntion is of the form

$$\psi : \mathbb{R}^d \times \mathbb{R} \to \mathbb{C} \tag{45}$$

where d is the number of spatial dimensions, the dimension of the vector space of $\vec{x}$ [2].

Definition 4.5 (Hamiltonian operator)**.**
The kinetic energy and potential energy operators, $\hat{T}$ and $\hat{V}$ respectively are defined as

$$\hat{T} = -\frac{\hbar^2}{2m}\nabla^2 \tag{46}$$

$$\hat{V} = V(\vec{x}, t) \tag{47}$$

where m is the mass of the particle [2].

The Hamiltonian operator $\hat{H}$ is the sum of the kinetic and potential energies [2], giving total energy of the system

$$\hat{H} = \hat{T} + \hat{V} = -\frac{\hbar^2}{2m}\nabla^2 + V(\vec{x}, t) \tag{48}$$

We've introduced the notion of time dependence in the wavefunction, the Schrödinger equation is an axiom that time dependent wavefunctions must to adhere to.

Definition 4.6 (Schrödinger equation)**.**
The Schrödinger equation, describing the time evolution of a wavefunction, is given by [2]

$$i\hbar\frac{\partial}{\partial t}\psi(\vec{x},t) = \hat{H}\psi \tag{49}$$

$$= -\frac{\hbar^2}{2m}\nabla^2\psi(\vec{x},t) + V(\vec{x},t)\psi(\vec{x},t) \tag{50}$$

Note that equation 50 is the non-relativistic Schrödinger equation.

4.3 Semiclassical Reigime

Definition 4.7 (Semiclassical Schrödinger equation)**.**
In the semiclassical reigime, the Schrödinger equation is

$$i\varepsilon\frac{\partial u}{\partial t} = -\frac{\varepsilon^2}{2}\nabla^2 u + V(\vec{x})u \tag{51}$$

$u(\vec{x},t)$ is the wavefunction with the domain $\mathbb{R}^d \times \mathbb{R}$ (d spatial dimensions and time), where $0 < \varepsilon \ll 1$ is a small scaled Planck constant. All physical quantities have been non-dimensionalised such that the only dimensionless parameter is ε. We only consider an energy potential which is space dependent and independent of time [13].

5 Spectral Methods

In this section, we will first examine Fourier transforms, then cover what is meant by spectral methods in general.

5.1 Continuous Fourier Transforms

Fourier transforms were amoung the first spectral methods, and we will use them to solve a certain term when we split the semiclassical Schrödinger equation in the section 6. In this section, we introduce Fourier transforms and how they can be applied to solve PDEs.

Definition 5.1 (Continuous Fourier transform [14])**.**
The Fourier transform for a function $f : \mathbb{R} \to \mathbb{C}$ is defined as

$$\hat{f}(k) = \mathcal{F}[f(x)] = \int_{-\infty}^{\infty} \exp(-i2\pi kx) f(x)\, dx \quad \forall k \in \mathbb{R} \tag{52}$$

Note that $\hat{f}(k) = \mathcal{F}[f(x)]$ are two different notations for the same Fourier transformation.

Definition 5.2 (Inverse Fourier transform [14])**.**
To return our original function $f(x)$ from $\hat{f}(x)$ we use an inverse Fourier transform

$$f(x) = \mathcal{F}^{-1}[\hat{f}(k)] = \int_{-\infty}^{\infty} \exp(i2\pi kx) \hat{f}(x)\, dk \tag{53}$$

Definition 5.3 (Fourier transform on a finite interval [14])**.**
The Fourier coefficients for a function $f : [X_0, X_1] \subset \mathbb{R} \to \mathbb{C}$, with preimage length $L = X_1 - X_0$, are defined as

$$\hat{f}_k = \frac{1}{L} \int_{X_0}^{X_1} \exp(-i2\pi kx/L) f(x)\, dx \quad \forall k \in \mathbb{Z} \tag{54}$$

with $\hat{f}_k$ being the kth Fourier coefficient.

Definition 5.4 (Fourier series [14])**.**
We define the inverse Fourier series of the above Fourier transform as

$$f(x) \sim \sum_{k=-\infty}^{\infty} \hat{f}_k \exp(i2\pi kx/L) \tag{55}$$

5.2 Discrete Fourier Transforms

Definition 5.5 (Discrete Fourier transform [15])**.**
Suppose we discretize our domain into $x \in [X_0, X_1]$ into N points, such that

$$x_j = X_0 + \delta j = X_0 + \frac{X_1 - X_0}{N} j, \quad j = 0, 1, \ldots, N-1 \tag{56}$$

then we can express our Fourier transform as the sum

$$\hat{f}_k = \sum_{j=0}^{N-1} f(x_j) \exp(-i2\pi kj/N) \quad k = 0, 1, \ldots, N-1 \tag{57}$$

This is a useful approximation for computer computation when, for efficiency, we want to restrict our domain and compute discretely.

Definition 5.6 (Inverse Discrete Fourier transform [15])**.**
We define the inverse discrete Fourier transform (IDFT) to be

$$f(x_j) = \frac{1}{N} \sum_{k=0}^{N-1} \hat{f}_k \exp(i2\pi kj/N) \tag{58}$$

5.3 Fast Fourier Transforms

A fast Fourier transfom (FFT) is a computational method for calculating Fourier coefficients in a discrete Fourier transform faster. We look to minimse the number of actual computations a computer is required to perform when transforming and calculating the Fourier coefficients.

For the discrete function $f(x_j)$ we wish to transform, we have a fixed N then the kth Fourier coefficient is

$$\hat{f}_k = \sum_{j=0}^{N-1} f(x_j) \exp(-i2\pi kj/N) \tag{59}$$

This computation is of order $O(N^2)$ [16], meaning the upper bound on how many computations is that for each N elements in the domain, we have to perform N computations.

Theorem 5.1 (Fast Fourier transform [16])**.**
if $N = 2^n$ for some $n \in \mathbb{N}$, with $\exp(-i2\pi kj/N) = \exp(-i2\pi kj/2^n)$ then a time complexity for calculating Fourier coefficients can be made $O(Nlog(N))$ using an FFT.

Consider one DFT transform for an even N, then we can seperate one Fourier sum into two transforms of length $N/2$

$$\hat{f}_k = \sum_{j=0}^{N-1} f(x_j) \exp(-i2\pi kj/N) \tag{60}$$

$$= \sum_{j=0}^{N/2-1} f(x_{2j}) \exp(-i2\pi k(2j)/N) \quad + \quad \sum_{j=0}^{N/2-1} f(x_{2j+1}) \exp(-i2\pi k(2j+1)/N) \tag{61}$$

$$= \sum_{j=0}^{N/2-1} f(x_{2j}) \exp(-i2\pi kj/(N/2)) \quad + \quad \exp(-i2\pi k/N) \sum_{j=0}^{N/2-1} f(x_{2j+1}) \exp(-i2\pi k/(N/2)) \tag{62}$$

Since

$$\exp(-i2\pi k(2j+1)/N) = \exp(-i2\pi k(2j)/N) \exp(-i2\pi k/N) \tag{63}$$

and $\exp(-i2\pi k/N)$ does not change with the summation index j.

This is true for mesh with increment $\delta = \frac{X_1 - X_0}{N}$ for some even number $N \in \mathbb{N}$. If $\frac{N}{2}$ is also divisible by 2 then we can repeat the process, seperating each sum in two once more. This motivates us to use $N = 2^n,\ n \in \mathbb{N}$, then we can seperate the sums further times to get the dramatically faster logerithmic time complexity.

5.4 Solving PDEs with Spectral Methods

We have examined how to apply a Fourier transform to decompose a function into a sum of trigonometric polynomials. We will now examine what spectral methods actually are.

Example 5.1 (Heat equation solutions)**.**
In the section 3 we splitted the semilinear Heat equation [6]

$$\frac{\partial u}{\partial t} = \nabla^2 u + f(u) \rightarrow \begin{cases} \dfrac{\partial u}{\partial t} = \nabla^2 u \\ \dfrac{\partial u}{\partial t} = f(u) \end{cases} \tag{64}$$

with initial condition $u(\vec{x}, T_0)$ given.

We need a solution to the PDE with the Laplacian term, a heat equation with diffusion constant $\alpha = 1$. We can solve the one spatial dimesional case $\vec{x} = x$, $\nabla^2 = \frac{\partial^2}{\partial x^2}$, using a Fourier transform across x on both

sides of the equation, the $\frac{\partial^2}{\partial x^2}$ operator on u becomes $(ik)^2$ on $\hat{u}$ [17]

$$\frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2} \tag{65}$$

$$\frac{d\hat{u}(k,t)}{dt} = -k^2 \hat{u}(k,t) \tag{66}$$

Equation 66 is just a linear ODE, which we can solve with the exponential [17]

$$\hat{u}(k,t) = \hat{u}(k,T_0)\exp(-k^2 t) \tag{67}$$

where $\hat{u}(k,T_0)$ is our Fourier coefficients from the given initial conditions, which we can put through an inverse Fourier transform to calculate the solution of $u(x,t)$ from.

Example 5.2 (Solving the heat equation in MATLAB)**.**
Consider the 1D heat equation

$$\frac{\partial u}{\partial t} = \alpha \frac{\partial^2 u}{\partial x^2} \tag{68}$$

We can transform this and achieve solution

$$\hat{u}(k,t) = \hat{u}(k,T_0)\exp(-\alpha k^2 t) \tag{69}$$

Lets look at a specific case where $\alpha = 0.01$, $N_x = 256$ (choosing a power of 2 for FFT), and $u(x,T_0) = \text{sech}(10x)^2$.

Then we can achieve solutions at $t = 5$ using MATLAB `fft` and `ifft` function [17] [18]

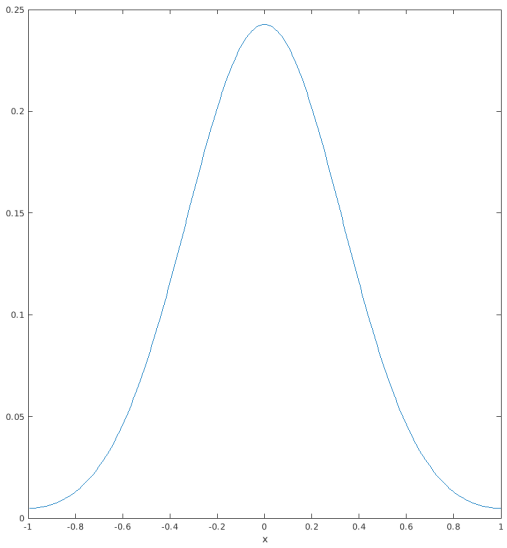


Figure 6: $(x, u(x,5))$

We can scale up this approximation of spatial derivative by considering k in a case with more than one spatial dimension.

Example 5.3 (Schrödinger equation solutions with Fourier transform)**.**
Let's look at a non-semiclassical Schrödinger equation, which we can solve convieniently without splitting or discretising. Consider the Schrödinger equation with no energy potential and two spatial dimensions. We have a wavefunction $\psi(\vec{x},t)$ and $\vec{x} = (x,y)$

$$i\hbar \frac{\partial \psi}{\partial t} + \frac{\hbar^2}{2m}\nabla^2 \psi = 0 \tag{70}$$

or without Laplacian notation

$$i\hbar\frac{\partial\psi}{\partial t}+\frac{\hbar^2}{2m}(\frac{\partial^2\psi}{\partial x^2}+\frac{\partial^2\psi}{\partial y^2})=0 \tag{71}$$

We can take a Fourier transform with respect to space $\vec{x}$, giving Fourier variable $\vec{k}=(k_x,k_y)$ [19]. The nabla operator ∇ becomes a factor of $i\vec{k}$.

$$\nabla\psi\to i\vec{k}\psi \iff \nabla^2\psi=i^2\vec{k}\cdot\vec{k}\psi=-k^2\psi \tag{72}$$

where $k^2=(k_x^2+k_y^2)$.

So after Fourier transforming, $\psi(\vec{x},t)\to\hat{\psi}(\vec{k},t)$, equation 71 becomes [19]

$$i\hbar\frac{\partial\hat{\psi}(\vec{k},t)}{\partial t}+\frac{\hbar^2}{2m}\nabla^2\hat{\psi}(\vec{k},t) \;=\; i\hbar\frac{\partial\hat{\psi}(\vec{k},t)}{\partial t}-\frac{\hbar^2}{2m}k^2\hat{\psi}(\vec{k},t) \;=\; 0 \tag{73}$$

We can solve this by using an integrating factor as we did in example 5.1 and 5.2.

$$\hat{\psi}(\vec{k},t)=\hat{\psi}(\vec{k},0)\exp(-i\frac{\hbar k^2}{2m}t) \tag{74}$$

As with our problem in the semiclassical reigime, a Schrödinger equation will often have an initial condition provided. For a particle in a square well with infinite potentials on the walls $x=0,X_1$ and $y=0,Y_1$, our wavefunction will be zero on those boundaries. It will have wavelength for the x direction $\lambda_x=\frac{2}{1}X_1,\frac{2}{2}X_1,\frac{2}{3}X_1,\ldots$, the same wavelength for the y direction $\lambda_y=\frac{2}{1}Y_1,\frac{2}{2}Y_1,\frac{2}{3}Y_1,\ldots$ Then for our wavenumbers [19]

$$k_x=\frac{2\pi}{\lambda_x}=\frac{a\pi}{X_1} \tag{75}$$

$$k_y=\frac{2\pi}{\lambda_y}=\frac{b\pi}{Y_1} \tag{76}$$

for $a,b=1,2,\ldots$

This gives wavefuction solutions of the form [19]

$$\psi(\vec{x},t)=A\sin(\frac{a\pi x}{X_1})\sin(\frac{b\pi y}{Y_1})\exp(-i\frac{\hbar k^2}{2m}t) \tag{77}$$

So these diffusion PDEs are efficiently solved by breaking the functions up into a series of Fourier basis functions.

Once a function has been Fourier transformed we can apply the operations, before taking the inverse to get our original function back. This is the core idea behind spectral methods, approximating a solution to a differential equation by a sum of smooth (C^∞) basis functions [17]. In our handling of the Schrödinger equation, these basis funcions will also be the Fourier coefficients.

6 Applying Splitting to the Schrödinger Equation

6.1 Splitting Methods on the Schrödinger equation

In a semiclassical reigime, the Schrödinger equation is [7]

$$\varepsilon\frac{\partial u}{\partial t} - i\frac{\varepsilon^2}{2}\nabla^2 u + iV(\vec{x})u = 0 \tag{78}$$

where $0 < \varepsilon \ll 1$ is constant, $u(\vec{x}, t)$ is the wavefunction with the domain $\mathbb{R}^d \times \mathbb{R}$. A problem in this form also comes with an initial value $u(\vec{x}, T_0)$ with T_0 being the earliest time value considered, meaning we have a function in terms of $\vec{x}$ given for the time value $t = t_0 = T_0$.

Example 6.1 (Lie-Trotter Splitting the Schrödinger equation [7])**.**
Consider the case with one spatial dimension $\vec{x} = x$, $u(x, t)$

$$\varepsilon\frac{\partial u}{\partial t} - i\frac{\varepsilon^2}{2}\frac{\partial^2 u}{\partial x^2} + iV(x)u = 0 \tag{79}$$

We split the full equation into a part resembling a heat equation with an imaginary constant coefficient $i\frac{\varepsilon}{2}$, and a term with only one partial derivative.

$$\varepsilon\frac{\partial u}{\partial t} - i\frac{\varepsilon^2}{2}\frac{\partial^2 u}{\partial x^2} + iV(x)u = 0 \quad \rightarrow \quad \begin{cases} \dfrac{\partial u}{\partial t} = i\dfrac{\varepsilon}{2}\dfrac{\partial^2 u}{\partial x^2} \\ \dfrac{\partial u}{\partial t} = -\dfrac{i}{\varepsilon}V(x) \end{cases} \tag{80}$$

For a timestep $t \in [t_n, t_{n+1}]$ we solve for the discretized wavefunction $U_j^{n+1} \approx u(x_j, t_{n+1})$ for all j values, subject to U_j^n as and initial condition, in other words we already have a value for the wavefunction at timestep t_n and want to solve for the t_{n+1} timestep.

1. We solve the first split part of equation 80 with a Fourier transform.

$$\frac{\partial u}{\partial t} = i\frac{\varepsilon}{2}\frac{\partial^2 u}{\partial x^2} \tag{81}$$

   We do this by calculating N_x (the number of Fourier coefficients, each with N_x different l values)

$$\hat{U}_l^n = \sum_{j=0}^{N_x} U_j^n \exp(-i\frac{2\pi l}{X_1 - X_0}j\delta_t) \quad l = -\frac{N_x}{2}, \ldots, \frac{N_x}{2} - 1 \tag{82}$$

   Then we solve equation 81 by summing over the Fourier coefficients, denoting the solution as U_j^*

$$U_j^* = \frac{1}{N_x}\sum_{l=-\frac{N_x}{2}}^{\frac{N_x}{2}-1} \exp(i\varepsilon\delta_t\frac{2\pi^2 l^2}{X_1 - X_0})\hat{U}_l^n \exp(i\frac{2\pi l}{X_1 - X_0}j\delta_t) \tag{83}$$

   which is our solution to equation 81 at time t_{n+1}.

2. Now we have our initial condition U_*^j we use it as it for

$$\frac{\partial u}{\partial t} = -\frac{i}{\varepsilon}V(x) \tag{84}$$

   at time step t_n. The solutions are just given using an exponential

$$U_j^{n+1} = \exp(i\frac{\delta_t}{\varepsilon}V(x_j))U_j^* \tag{85}$$

   Solutions to the wavefunction at the given timestep are then approximated as the solution to this step's ODE, equation 85.

We're able to solve across a timestep $[t_n, t_{n+1}]$ given U_j^n as an initial condition. We know solutions at timestep t_0 as a Dirichlet condition on the problem itself and from this we are able to approximate across an entire domain, by using the given solution U_j^0 as an initial condition to solve U_j^1, and that approximated U_j^1 solution to solve for U_j^2 and so on.

The example above is first order Lie-Trotter splitting of the Schrödinger equation. We solve the PDE section with Fourier transforms, and then solve an ODE section with the exponential, using the PDE solution as an intial condition. We could instead use a second order Strang splitting scheme, solving an ODE for half of a timestep first, and using the solutions for that as an initial condition to solve the PDE, then solving the ODE again for another half timestep, using initial conditions from the PDE solutions.

Example 6.2 (Strang splitting the Schrödinger equation [7])**.**
For a timestep $t \in [t_n, t_{n+1}]$ we solve for the discretized wavefunction $U_j^{n+1} \approx u(x_n, t_{n+1})$ subject to U_j^n as and initial condition, this is the same goal as the first order Lie-Trotter example, but we will use a different approach to reach it.

$$\varepsilon\frac{\partial u}{\partial t} - i\frac{\varepsilon^2}{2}\frac{\partial^2 u}{\partial x^2} + iV(x)u = \varepsilon\frac{\partial u}{\partial t} - i\frac{\varepsilon^2}{2}\frac{\partial^2 u}{\partial x^2} + \frac{i}{2}V(x)u + \frac{i}{2}V(x)u = 0 \quad \rightarrow \quad \begin{cases} \dfrac{\partial u}{\partial t} = -\dfrac{i}{2\varepsilon}V(x) \\ \dfrac{\partial u}{\partial t} = i\dfrac{\varepsilon}{2}\dfrac{\partial^2 u}{\partial x^2} \\ \dfrac{\partial u}{\partial t} = -\dfrac{i}{2\varepsilon}V(x) \end{cases} \tag{86}$$

1. We first solve the ODE
$$\frac{\partial u}{\partial t} = -\frac{i}{\varepsilon}V(x) \tag{87}$$
for half a timestep, which corresponds to solving
$$\frac{\partial u}{\partial t} = -\frac{i}{2\varepsilon}V(x) \tag{88}$$
Using the exponential and initial condition U_j^n, we denote solutions equation as U_j^*
$$U_j^* = \exp(-i\frac{\delta_t}{2\varepsilon}V(x_j))U_j^n \tag{89}$$

2. We solve the second part of the split equation 86 with a Fourier transform, using initial conditions given by U_j^*, the solution to equation 88 at t_{n+1}
$$\frac{\partial u}{\partial t} = i\frac{\varepsilon}{2}\frac{\partial^2 u}{\partial x^2} \tag{90}$$
Summing over N_x different space points, for each different l value
$$\hat{U}_l^n = \sum_{j=0}^{N_x} U_j^* \exp(-i\frac{2\pi l}{X_1 - X_0}j\delta_t) \quad l = -\frac{N_x}{2}, \ldots, \frac{N_x}{2} - 1 \tag{91}$$
Then we solve equation 90 by summing over the Fourier coefficients, denoting the solution as U_j^{**}
$$U_j^{**} = \frac{1}{N_x}\sum_{l=-\frac{N_x}{2}}^{\frac{N_x}{2}-1} \exp(i\varepsilon\delta_t\frac{2\pi^2 l^2}{X_1 - X_0})\hat{U}_l^n \exp(i\frac{2\pi l}{X_1 - X_0}j\delta_t) \tag{92}$$
which is our solution at time t_{n+1}.

3. Lastly, we solve equation 87 for another half timestep, which is again equivalent to solving equation 88, but with initial conditions given by the solutions to the Fourier step U_j^{**}
$$U_j^{n+1} = \exp(-i\frac{\delta_t}{2\varepsilon}V(x_j))U_j^{**} \tag{93}$$
which we denote as U_j^{n+1} which is our approximation $U_j^{n+1} \approx u(x_j, t_{n+1})$

Theorem 6.1 (Strang splitting with constant potential)**.**
When the energy potential is constant, $V(x) = V = \text{constant}$, Strang splitting has a faster one step form [7]

$$U_j^n = \frac{1}{N_x} \sum_{l=-\frac{N_x}{2}}^{\frac{N_x}{2}-1} \exp(-i(\varepsilon 2\pi^2 l^2 + \frac{V}{\varepsilon} t_n)\hat{U}_l^0 \exp(i2\pi l j \delta_x), \quad \hat{U}_l^0 = \sum_{j=0}^{N_x-1} U_j^n \exp(-i2\pi l j \delta_x) \tag{94}$$

since the exponential terms in

$$U_j^* = \exp(-i\frac{\delta_t}{2\varepsilon}V)U_j^n, \quad U_j^{n+1} = \exp(-i\frac{\delta_t}{2\varepsilon}V)U_j^{**} \tag{95}$$

remain constant with respect to change of j.

Note that in further examples of Strang splitting which use a constant potential, for the purposes of explanation we do not refine the splitting into this first order form.

Example 6.3 (Schrödinger Strang splitting numerical example [7])**.**
We choose the intial condition

$$u(x, T_0) = \exp(-25(x - \frac{1}{2})^2) \exp(-\frac{i}{\varepsilon}\frac{1}{5} \ln(\exp(5(x - \frac{1}{2})) + \exp(-5(x - \frac{1}{2})))) \tag{96}$$

With a spatial domain $x \in [X_0 = 0, X_1 = 1]$, a time domain $t \in [T_0 = 0, T_1 = 1]$. We have a constant energy potential $V = 10$. We can select the number of points in the space and time mesh N_x and N_t respectively. This adjusts the size of the timestep increment $\delta_t = \frac{T_1 - T_0}{N_t} = \frac{1}{N_t}$ and spacestep increment $\delta_x = \frac{X_1 - X_0}{N_x} = \frac{1}{N_x}$. Selecting $N_x = N_t = 100 \iff \delta_t = \delta_x = 0.01$ we can calculate across the entire domain in MATLAB [20].

1. We set dimensions of the discretized domain, we also choose ε.

```
tM = 100;
t0 = 0;
t1 = 1;
tH = (t1 - t0)/tM;
xM = 100;
x0 = 0;
x1 = 1;
xH = (x1 - x0)/xM;
vEps = 10^-3;
```

2. We set our constant energy potential.

```
syms V(x)
V(x) = 10;
```

3. Since we've discretized the x domain it is useful to evaluate the potential at each discrete spacestep $j = 0, 1, \ldots, N_x - 1$ to speed up computation, since passing symbolics into a function in MATLAB is computationally expensive.

```
Vx = zeros(xM);
for j = 1 : xM
   Vx(j) = V(x0 + (j-1)*xH);
end
```

4. We set the initial condition function, and again generate a vector for the evaluation of the function at each spacestep.

```
syms u0(x)
u0(x) = exp(-25*((x-0.5)^2))*exp(1i*(-1/5)*log(exp(5*(x - 0.5))+exp(-5*(x - 0.5)))/
   vEps);
U0 = zeros(xM);
for j = 1 : xM
   U0(j) = u0(x0 + (j-1)*xH);
end
```

5. We will have a Strang splitting function in terms of the parameters so far, which we call. It will perform a second order Strang split and return a matrix `u` with N_x rows representing each element of the discretized spatial mesh and N_t columns representing each element of the time mesh.

```
u = Strang(vEps, tM, t0, t1, tH, xM, x0, x1, xH, Vx, U0);
```

6. We set the first column of the `u` matrix to the passed in initial condition, since this vector is spatial values of the wavefunction at time T_0 given as an initial condition to the problem.

```
function ret = Strang (vEps, tM, t0, t1, tH, xM, x0, x1, xH, Vx, U0)
   u = zeros(xM,tM);
   x = zeros(xM);

   for j = 1 : xM
      u(j,1) = U0(j);
   end
```

7. We loop, solving for each time element of the matrix.

```
   for n = 1 : tM-1
      uSt1 = zeros(xM);
      uSt2 = zeros(xM);
```

8. We solve the first split ODE using the solution at the last timestep `u(j,n)` as an initial condition.

```
      for j = 1 : xM
         uSt1(j) = exp(-1i*Vx(j)*tH/(2*vEps))*u(j,n);
      end
```

9. We solve the PDE with a Fourier transform, using the solution of the first ODE `USt1` as an initial condition.

```
      for j = 1 : xM
         sumval = 0;
         for l = -xM/2 : (xM/2 - 1)
            mul = (2*pi*l)/(x1 - x0);

            uStFourier = 0;
            for j2 = 1 : xM
               uStFourier = uStFourier + uSt1(j2)*exp(-1i*mul*((j2-1)*xH));
            end
            sumval = sumval + exp(-1i*vEps*tH*(mul^2)/2) * uStFourier * exp(1i*mul*((j
-1)*xH));
         end
         uSt2(j) = (1/xM)*sumval;
      end
```

10. Solve the ODE for a second half timestep using the solution to the PDE `uSt2` as an initial condition. The solution to the wavefunction at this timestep `u(j,n+1)` is the solution to this ODE.

```
      for j = 1 : xM
         u(j, n+1) = exp(-1i*Vx(j)*tH/(2*vEps))*uSt2(j);
      end
```

11. After we have looped through all timesteps we return the solution matrix from the function.

```
   end
   ret = u;
end
```

6.2 Numerical Examples of Observables

Theorem 6.2 (Position density of the wavefunction)**.**
Given a wavefunction ψ, the position density of the wavefuntion is $|\psi|^2$ [2]. Given a discretized wavefunction `u` the MATLAB code to calculate position density is [20]

```
[rows, columns] = size(u);
expVal = zeros(rows, columns);
for i = 1 :  rows
   for j = 1 : columns
      expVal(i,j) = norm(u(i,j),2)^2;
   end
end
```

here `expVal` is the position density of the wavefunction matrix with the same row, column relationship as `u`.

Example 6.4 (Position density of a wavefunction approximated with Lie-Trotter splitting [20])**.**
We determine the position density of a discretized wavefunction, solved using the Lie-Trotter splitting method [7] with initial conditions given from equation 96, $\varepsilon = 10^{-3}$, and constant potential $V = 10$. We solve for different values of N_x and N_t, plotting with axes of time, position, and position denisty, as well as a cross section of position and position density at specific time $t = 0.54$.

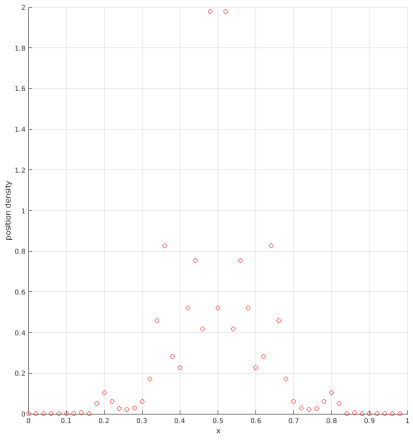


Figure 7: $(x, |u(x, 0.54)|^2)$
$\delta_t, \delta_x = 0.02,\ N_t, N_x = 50$

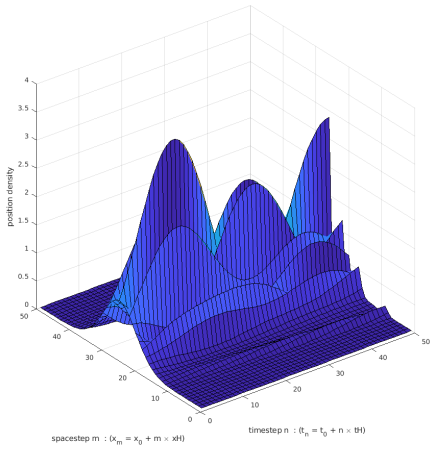


Figure 8: $(t, x, |u(x, t)|^2)$
$\delta_t, \delta_x = 0.02,\ N_t, N_x = 50$

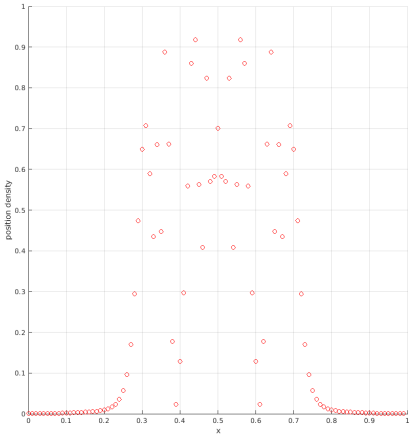


Figure 9: $(x, |u(x, 0.54)|^2)$
$\delta_t, \delta_x = 0.01,\ N_t, N_x = 100$

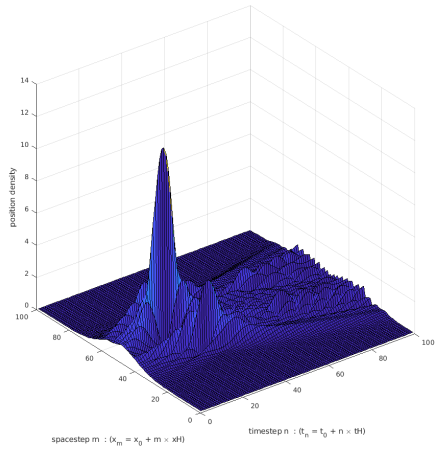


Figure 10: $(t, x, |u(x, t)|^2)$
$\delta_t, \delta_x = 0.01,\ N_t, N_x = 100$

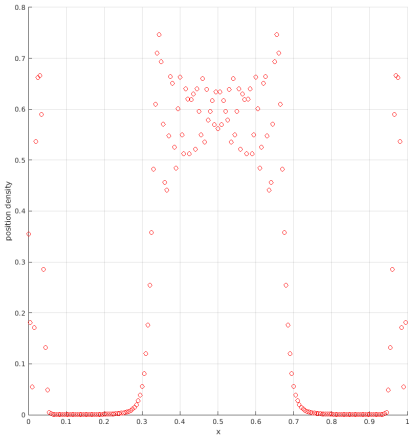


Figure 11: $(x, |u(x, 0.54)|^2)$
$\delta_t, \delta_x = 0.005,\ N_t, N_x = 200$

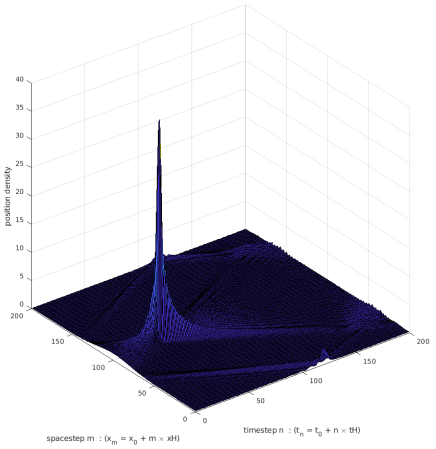


Figure 12: $(t, x, |u(x, t)|^2)$
$\delta_t, \delta_x = 0.005,\ N_t, N_x = 200$

Example 6.5 (Position density of a wavefunction approximated with Strang splitting [20])**.**
We determine the position density of a discretized wavefunction, solved using the Strang splitting method [7] with initial conditions given from equation 96, $\varepsilon = 10^{-3}$, and constant potential $V = 10$. We solve for different values of N_x and N_t, plotting with axes of time, position, and position denisty, as well as a cross section of position and position density at specific time $t = 0.54$.

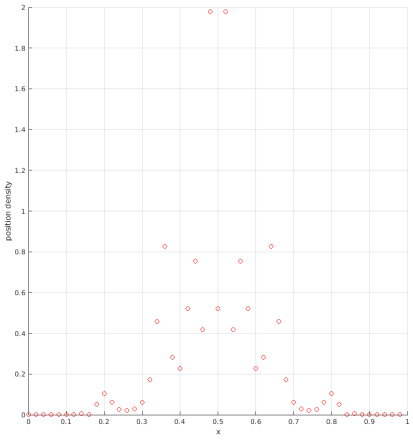


Figure 13: $(x, |u(x, 0.54)|^2)$
$\delta_t, \delta_x = 0.02,\ N_t, N_x = 50$

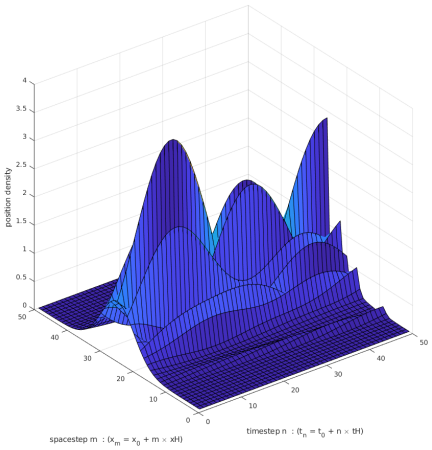


Figure 14: $(t, x, |u(x, t)|^2)$
$\delta_t, \delta_x = 0.02,\ N_t, N_x = 50$

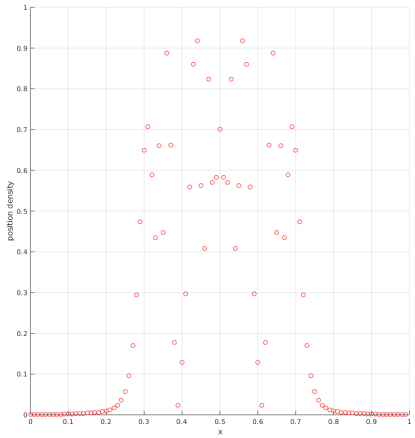


Figure 15: $(x, |u(x, 0.54)|^2)$
$\delta_t, \delta_x = 0.01,\ N_t, N_x = 100$

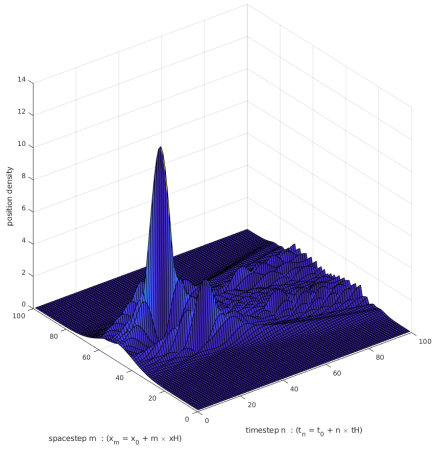


Figure 16: $(t, x, |u(x, t)|^2)$
$\delta_t, \delta_x = 0.01,\ N_t, N_x = 100$

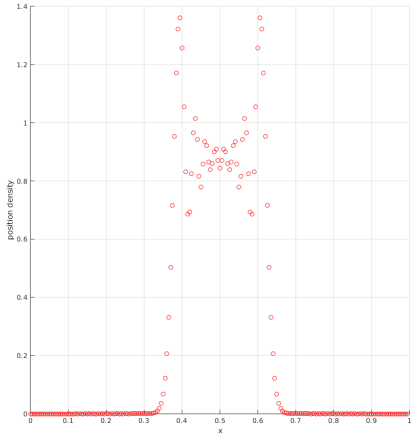


Figure 17: $(x, |u(x, 0.54)|^2)$
$\delta_t, \delta_x = 0.005,\ N_t, N_x = 200$

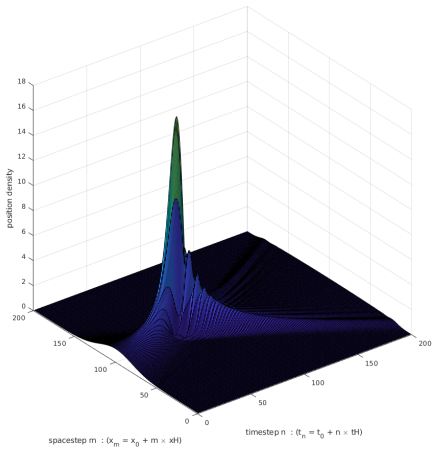


Figure 18: $(t, x, |u(x, t)|^2)$
$\delta_t, \delta_x = 0.005,\ N_t, N_x = 200$

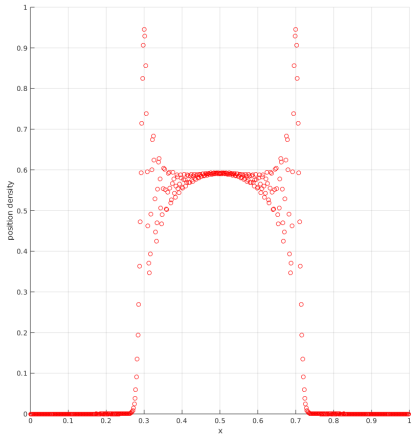


Figure 19: $(x, |u(x, 0.54)|^2)$
$\delta_t, \delta_x = 0.002,\ N_t, N_x = 500$

As we can see in figures 17 and 19, as we increase the number of points, we converge on a similar shaped curve.

Theorem 6.3 (Current density of the wavefunction)**.**
The current density from the wavefunction $u(\vec{x}, t)$ is defined as [7]

$$\vec{J}(\vec{x}, t) = \varepsilon \operatorname{Im}(\bar{u}\nabla u) \tag{97}$$

where $\bar{u}$ is the complex conjugate of u, and Im just takes the imaginary part of its contents. In the case with one spatial dimension $u(x, t)$, the current density is

$$J(x, t) = \varepsilon \operatorname{Im}(\bar{u}\frac{\partial u}{\partial x}) \tag{98}$$

Theorem 6.4 (Differentiating discretized wavefunction)**.**
Since we want to apply the current density operator to a discrete approximation, we must find a way to take the x partial derivative in a non-tranditional way, since the traditional derivative evaluates the instantaneous rate of change about a continuous neighborhood of the function at a point.

1. We can use the Crank-Nicolson method [21]

$$\frac{\partial u}{\partial x}|_{(x_j, t_n)} \approx \frac{U^n_{j+1} - U^n_{j-1}}{2\delta_x} \quad \forall j \in (0, N_x - 1) \tag{99}$$

   We loop through every space and time point of the matrix (row and column) and calculate the spatial derivative at that point, then fill a new matrix holding the corresponding current densities.

2. We can also use a discrete Fourier transform over the discretized elements of U, and differentiate in terms of j. This is known as spectral differentiation [22].

   We Fourier transform U^n, a function at time n of N_x different space points

$$\hat{U}_k = \sum_{j=0}^{N_x - 1} U^n_j \exp(-i2\pi kj/N_x) \tag{100}$$

   The inverse Fourier transform is then

$$U^n_j = \frac{1}{N_x} \sum_{k=0}^{N_x - 1} \hat{U}_k \exp(i2\pi kj/N_x) \tag{101}$$

   so we can express an approximation for $\frac{\partial u}{\partial x}|_{(x_j, t_n)}$ by differentiating this term in terms of j

$$\frac{\partial u}{\partial x}|_{(x_j, t_n)} \approx \frac{\partial}{\partial j}(\frac{1}{N_x} \sum_{k=0}^{N_x - 1} \hat{U}_k \exp(i2\pi kj/N_x)) = \frac{1}{N_x} \sum_{k=0}^{N_x - 1} (i2\pi k/N_x)\hat{U}_k \exp(i2\pi kj/N_x) \tag{102}$$

   where $\hat{U}_k$ are the Fourier coefficients of the spatial components at t_n.

It is a useful property of the wavefunction that once we have our approximation itself, we do not need to adjust it to examine the different qualities of the physical state it represents. We just have to act on it with the different operators, though the accuracy of the given approximation may have different reliability.

Example 6.6 (Current density of a wavefunction approximated with Strang splitting [20])**.**
We solve the current density of a discretized wavefunction, using the same solved wavefunction matrices from the Strang splitting example 6.5. We use Crank-Nicolson to approximate the derivative of the wavefunction so that we can derive current density.

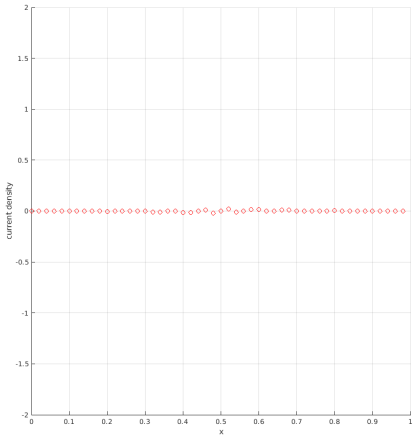


Figure 20: $(x, J(x, 0.54))$
$\delta_t, \delta_x = 0.02,\ N_t, N_x = 50$

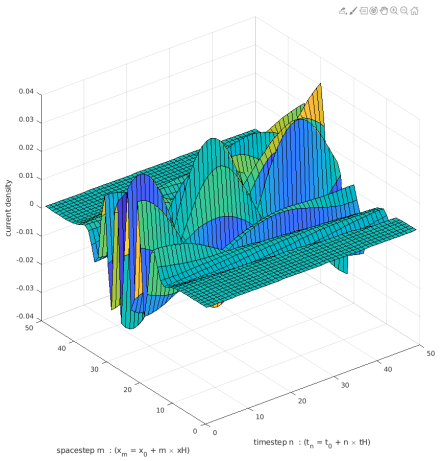


Figure 21: $(t, x, J(x, t))$
$\delta_t, \delta_x = 0.02,\ N_t, N_x = 50$

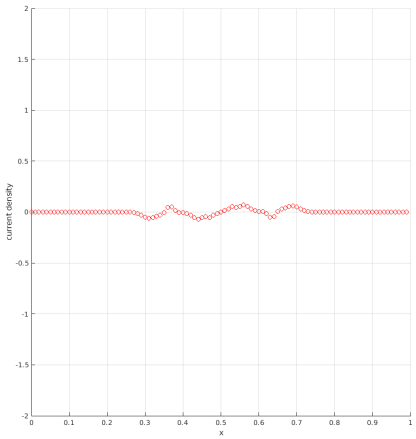


Figure 22: $(x, J(x, 0.54))$
$\delta_t, \delta_x = 0.01,\ N_t, N_x = 100$

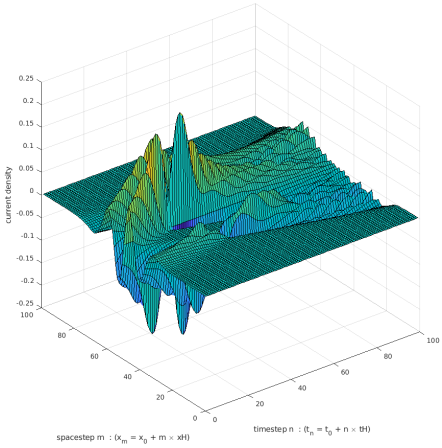


Figure 23: $(t, x, J(x, t))$
$\delta_t, \delta_x = 0.01,\ N_t, N_x = 100$

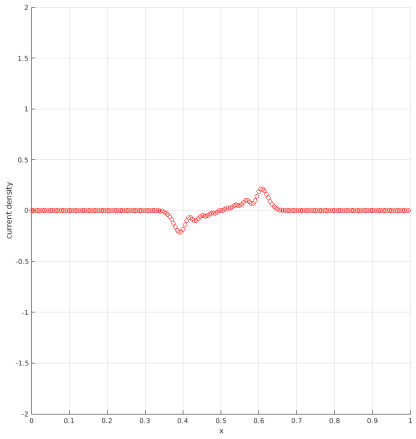


Figure 24: $(x, J(x, 0.54))$
$\delta_t, \delta_x = 0.005,\ N_t, N_x = 200$

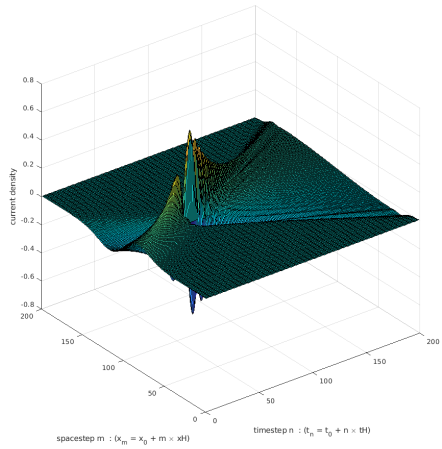


Figure 25: $(t, x, J(x, t))$
$\delta_t, \delta_x = 0.005,\ N_t, N_x = 200$

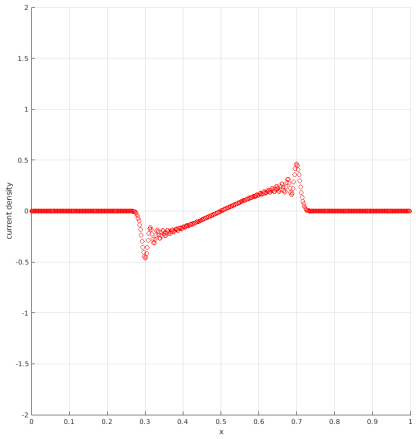


Figure 26: $(x, J(x, 0.54))$
$\delta_t, \delta_x = 0.002,\ N_t, N_x = 500$

The selection of our number of points N_x and N_t can be tricky. In example 6.5, plotting position density from Strang splitting, figures 17 and 19 shared a far similar qualitative structure than this example with figures 24 and 26, despite having the same differences in N_x, N_t. In section 6.3 we will examine the properties of error in Schrödinger splitting.

Example 6.7 (Approximating derivative)**.**
In example 6.6 we used Crank-Nicolson over spectral differentiation. It is important to note there is a quantitave difference in the error between the two methods, figures 27, 28 plot current density using the same wavefunction derived in example 6.5, the only difference is how the derivative is approximated [20].

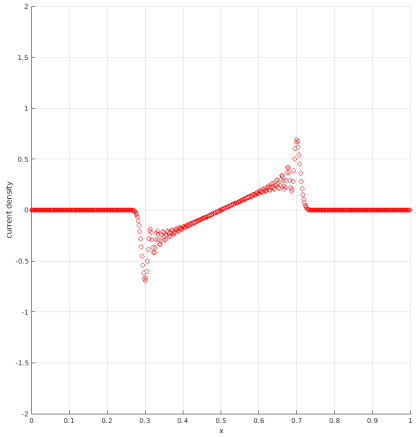


Figure 27: $(x, J(x, 0.54))$
$\delta_t, \delta_x = 0.002,\ N_t, N_x = 500$, derived with spectral differentiation

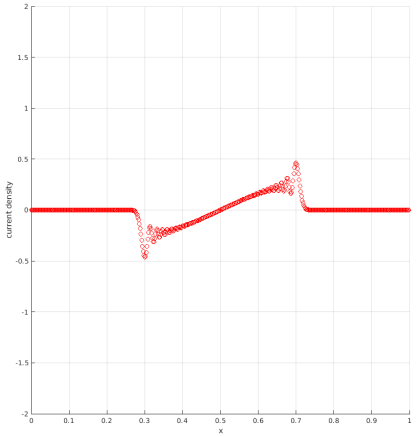


Figure 28: $(x, J(x, 0.54))$
$\delta_t, \delta_x = 0.002,\ N_t, N_x = 500$, derived with Crank-Nicolson differentiation

6.3 Error in Splitting

For any wavefunction approximation, there are three parameters we can alter: ε, N_t, and N_x. An important result is that in both Strang and Lie-Trotter Splitting of the Schrödinger equation, discretization error is spatial [7].

Example 6.8 (Error in N_t against N_x in position density)**.**
We solve the Strang splitted Schrödinger equation [13]

$$\varepsilon\frac{\partial u}{\partial t}-i\frac{\varepsilon^2}{2}\frac{\partial^2 u}{\partial x^2}+iV(x)u=\varepsilon\frac{\partial u}{\partial t}-i\frac{\varepsilon^2}{2}\frac{\partial^2 u}{\partial x^2}+\frac{i}{2}V(x)u+\frac{i}{2}V(x)u=0 \quad \rightarrow \quad \begin{cases}\dfrac{\partial u}{\partial t}=-\dfrac{i}{2\varepsilon}V(x)\\ \dfrac{\partial u}{\partial t}=i\dfrac{\varepsilon}{2}\dfrac{\partial^2 u}{\partial x^2}\\ \dfrac{\partial u}{\partial t}=-\dfrac{i}{2\varepsilon}V(x)\end{cases} \tag{103}$$

with $t\in[0,1]$ $x\in[0,1]$

We choose potential $V(x)=\frac{x^2}{2}$ and an initial condition

$$u(x,T_0)=\exp(-25(x-\frac{1}{2})^2)\exp(\frac{i}{\varepsilon}(1+x)) \tag{104}$$

We allow $\varepsilon=10^{-3}$ and let N_x and N_t vary, plotting the position density [20].

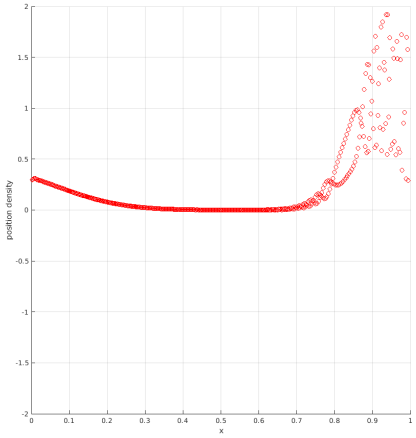


Figure 29: $(x,|u(x,0.54)|^2)$
$\delta_t,\delta_x=0.002,\ N_t,N_x=500$

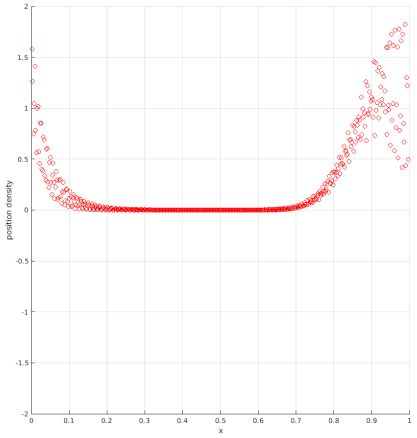


Figure 30: $(t,x,|u(x,0.54)|^2)$
$\delta_t=0.01,\delta_x=0.002,\ N_t=100,N_x=500$

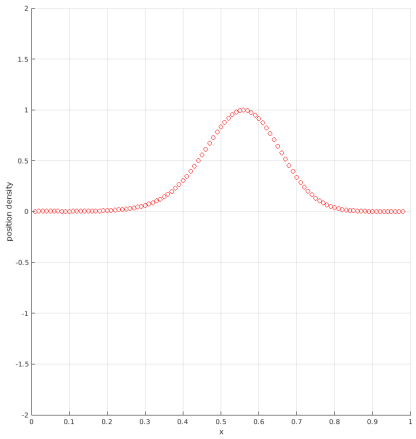


Figure 31: $(x,|u(x,0.54)|^2)$
$\delta_t=0.002,\delta_x=0.01,\ N_t=500,N_x=100$

As we can see, a lowering of N_x causes a huge loss of precision which isn't seen in the lowering of N_t. When selecting a new timestep to compare results, it is important to ensure there is a timestep t_n corresponding to the time t which is being compared. For example, in the above systems, $t=t_n=0.54$ at $n=270$ for $N_t=500$, and $n=54$ for $N_t=100$. If we change the timestep to any arbitraty value this isn't guaranteed. For $N_t=64$ the closest we'd be able to evaluate for $t=0.54$ is $n=34$, 35 which correspond to $t=t_n=0.53125,\ 0.546875$ respectively. The approximated points are finite and so N_t and N_x should be selected with consideration of which space and time points of the wavefunction we need to compare.

Example 6.9 (Error in N_t against N_x in current density)**.**
We use use the same three wavefunctions as derived in example 6.8.
We allow $\varepsilon = 10^{-3}$ and let N_x and N_t vary, plotting the current density derived from Crank-Nicolson differentiation [20].

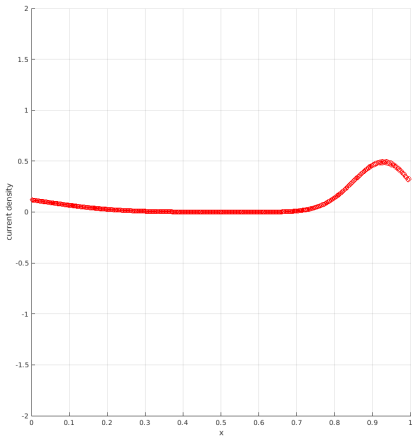


Figure 32: $(x, J(x, 0.54))$
$\delta_t, \delta_x = 0.002,\ N_t, N_x = 500$

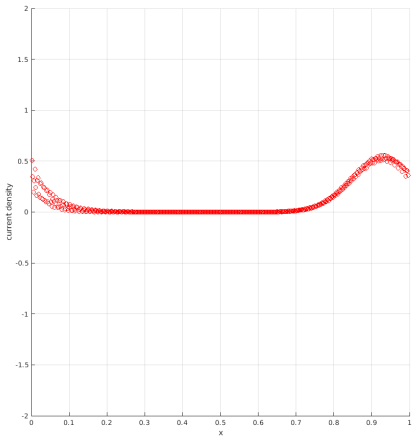


Figure 33: $(x, J(x, 0.54))$
$\delta_t = 0.01, \delta_x = 0.002,\ N_t = 100, N_x = 500$

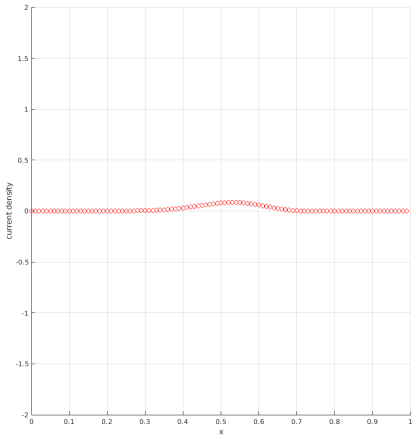


Figure 34: $(x, J(x, 0.54))$
$\delta_t = 0.002, \delta_x = 0.01,\ N_t = 500, N_x = 100$

Or using spectral differentiaition instead of Crank-Nicolson [20]

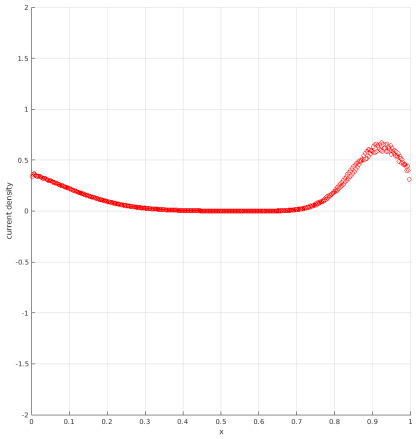


Figure 35: $(x, J(x, 0.54))$
$\delta_t, \delta_x = 0.002,\ N_t, N_x = 500$

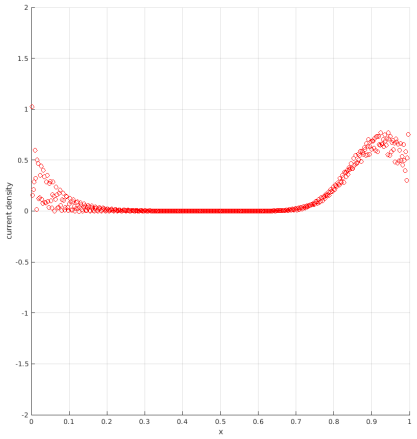


Figure 36: $(x, J(x, 0.54))$
$\delta_t = 0.01, \delta_x = 0.002,\ N_t = 100, N_x = 500$

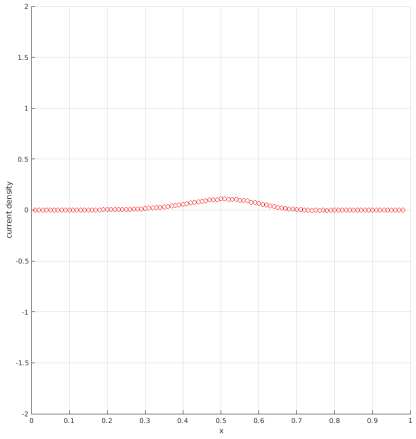


Figure 37: $(x, J(x, 0.54))$
$\delta_t = 0.002, \delta_x = 0.01,\ N_t = 500, N_x = 100$

For both cases we can again see that the lowering of N_x causes a huge loss of precision which is not observed in the lowering of N_t.

We examine stablility using L^2 and l^2 norms.

Definition 6.1 (L^2 and l^2 norms)**.**
For $u(x)$ a continuous wavefunction at timestep t_n with $x \in [X_0, X_1]$, and $\vec{U}^n = (U_0^n, U_1^n, \ldots, U_{N_x-1}^n)$ discrete space approximations at t_n, we define the L^2 [7] norm as

$$\|u(x)\|_{L^2} = \left(\int_{X_0}^{X_1} |u(x)|^2\, dx\right)^{\frac{1}{2}} \tag{105}$$

and the l^2 norm [7] as

$$\|\vec{U}^n\|_{l^2} = \left(\frac{X_1 - X_0}{N_x} \sum_{j=0}^{N_x-1} |U_j^n|^2\right)^{\frac{1}{2}} = \left(\delta_x \sum_{j=0}^{N_x-1} |U_j^n|^2\right)^{\frac{1}{2}} \tag{106}$$

Definition 6.2 (Unconditional stability)**.**
A method is said to be unconditionally stable if for any choice of step sizes δ_x and δ_t, the solutions remain bounded [23].

Theorem 6.5 (Unconditional stability of Lie-Trotter and Strang)**.**
For time splitting Lie-Trotter and Strang, approximations are unconditionally stable [7] for any δ_x and δ_t [7], meaning

$$\|\vec{U}^n\|_{l^2} = \|\vec{U}^0\|_{l^2} \quad \forall n = 1, 2, \ldots \tag{107}$$

where $\vec{U}^0$ is the discrete space points at t_0, given in the problem.

Example 6.10 (Error from ε numerical example)**.**
We choose $u(x, T_0) = \exp(-25(x - \frac{1}{2})^2)\exp(\frac{i}{\varepsilon}\frac{1}{5}(x^2 - x))$ with $x, t \in [0, 1]$, $V = 100$, and $N_x, N_t = 200$ [7]. We solve the semiclassical Schrödinger equation using Lie-Trotter splitting for different values of ε, and plot the position density at $t = 0.54$ [20].

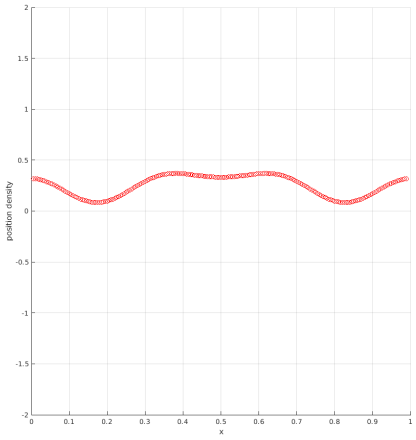


Figure 38: $(x, |u(x, 0.54)|^2)$
$\varepsilon = 10^{-1}$

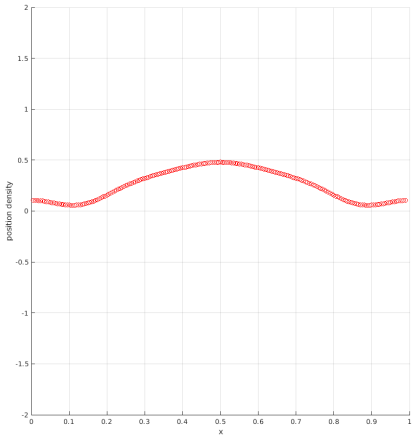


Figure 39: $(x, |u(x, 0.54)|^2)$
$\varepsilon = 10^{-1.2}$

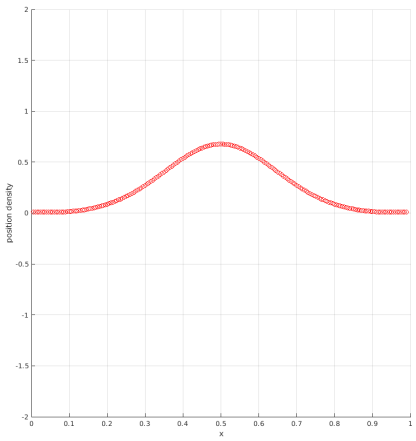


Figure 40: $(x, |u(x, 0.54)|^2)$
$\varepsilon = 10^{-1.5}$

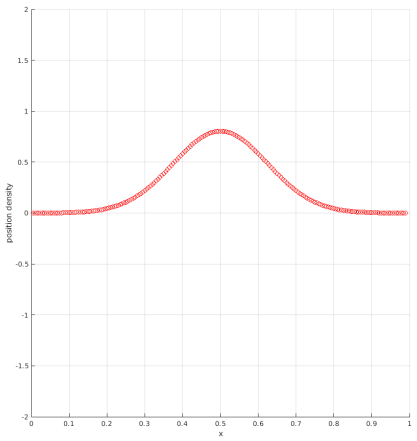


Figure 41: $(x, |u(x, 0.54)|^2)$
$\varepsilon = 10^{-2}$

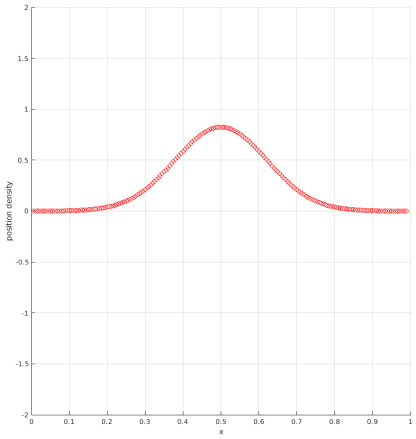


Figure 42: $(x, |u(x, 0.54)|^2)$
$\varepsilon = 10^{-3}$

7 Conclusion

7.1 Summary

The aim of this project was to research splitting methods and apply them to the semiclassical Schrödinger equation. The project examined splitting methods on both ODEs and PDEs, and analysed error in both. We have examined how to use Fourier spectral methods to solve PDEs. Quantum mechanics has been explored enough to understand the relevancy the results of our approximations have to physics. Splitting of the time derivative of the semiclassical Schrödinger equation has been carried out, followed by spectral methods, and approximate solutions to the wavefunctions have been developed. Several numerical examples have been carried out to illustrate properties of splitting schemes and spectral methods, the results of which have been analysed.

The results show that time splitting spectral methods are a successful method in approximating values to semiclassical Schrödinger equation problems. Some margin of error can be introduced by the method used to approximate solutions to the subproblems derived from splitting, or from the splitting method itself.

7.2 Future Development

There are several components of the project which present opportunity for future development.

- Further analyis of the error terms could be explored.
- A fast Fourier transform could be implemented to the solving of the Schrödinger split, improving time efficiency. The time splitting spectral method could also be implemented in C which has the possibility of considerably improving computation time.
- Efforts could be made to expand the method into more spatial dimensions, solving a Laplacian term of more than one spatial derivative in the semiclassical Schrödinger split.
- Adjustments to the method could be explored to solve the semiclassical Schrödinger equation with a time dependent energy potential $V(\vec{x}, t)$.

8 References

[1] Zbigniew I. Woznicki.
Matrix Splitting Methods. (International Journal of Mathematics and Mathematical Sciences, 15/03/2001)

[2] Brian C. Hall.
Quantum Theory for Mathematicians. (Springer, Graduate Texts in Mathematics, 2013)

[3] Jason Frank, Ben Leimkuhler.
Computational Modelling and Dynamical Systems. (11/09/2012)

[4] Nick Trefenthen.
Who invented the great numerical algorithims?.
https://people.maths.ox.ac.uk/trefethen/inventorstalk.pdf

[5] J J O'Connor, E F Robertson.
History of Joseph Fourier. (1997)
http://mathshistory.st-andrews.ac.uk/Biographies/Fourier.html

[6] Axel Målqvist.
Splitting techniques for solving PDEs. (25/11/2016)
http://www.math.chalmers.se/∼axel/pres/lund16.pdf

[7] Weizhu Bao, Shi Jin, Peter A. Markowich.
On Time-Splitting Spectral Approximations for the Schrödinger Equation in the Semiclassical Regime. (Journal of Computational Physics 175, 2001)

[8] Alexander Van–Brunt, Matt Visser.
Explicit Baker–Campbell–Hausdorff formulae for some specific Lie algebras. (arXiv, 18/05/15)

[9] Malek Diab.
Operator splitting methods. (STIMULATE European Joint Doctorates, 25/25/2019)
http://www.stimulate-ejd.eu/content/operator-splitting-methods

[10] Eva Lott (Author).
MATLAB: Lie-Trotter and Strang splitting solver for ODEs.
https://github.com/evalott100/DissertationProject/blob/master/Matlab/StrangLietrotterSplitting.m

[11] *MATLAB system of differential equations solution.*
https://uk.mathworks.com/help/symbolic/solve-a-system-of-differential-equations.html#buxuujb

[12] Eva Lott (Author).
Simple ODE solver.
https://github.com/evalott100/DissertationProject/blob/master/Matlab/BasicODE.m

[13] Irene Kyza, Charalambos Makridakis, Michael Plexousakis.
Error control for time-spliting spectral approximations of the semiclassical Schrödinger equation. (15/03/2010)

[14] Elias M. Stein, Rami Shakarchi.
Fourier Analysis. (Princeton University Press, 20/10/2007)

[15] Dexter Chua.
Part IB: Methods.
https://dec41.user.srcf.net/notes/IB_M/methods_trim.pdf

[16] Weisstein, Eric W.
Wolfram MathWorld.
https://mathworld.wolfram.com/FastFourierTransform.html

[17] Denys Dutykh.
A brief introduction to pseudo-spectral methods: application to diffusion problems. (2016)
https://cel.archives-ouvertes.fr/cel-01256472v2/document

[18] Eva Lott (Author).
Heat Equation Solver.
https://github.com/evalott100/DissertationProject/blob/master/Matlab/HeatEquation.m

[19] John Peacock.
Fourier lectures. (2015)
https://www.roe.ac.uk/japwww/teaching/fourier/fourier1415.pdf

[20] Eva Lott (Author).
Schrödinger splitting examples.
https://github.com/evalott100/DissertationProject/Schrodinger.m

[21] Daniel J. Duffy.
A Critique of the Crank Nicolson Scheme Strengths and Weaknesses for Financial Instrument Pricing.
(Datasim Component Technology BV, 2004)
https://www.researchgate.net/publication/
260548662_Crank-Nicolson_finite_difference_method_for_solving_time-fractional_diffusion_equation

[22] Steven G. Johnson.
MIT Applied Mathematics: fft spectral differentiation notes. (4/05/2011)
https://math.mit.edu/ stevenj/fft-deriv.pdf

[23] Parviz Moin.
Fundamentals of Engineering Numerical Analysis. (Cambridge University Press, 2010)

[24] Eva Lott (Author).
All MATLAB examles used are available on github
https://github.com/evalott100/DissertationProject