Numerical Time Development Of One Dimensional Quantum Systems

Steven Barnett

May 19, 2011

In this paper, a set of algorithms is developed for numerically calculating the time evolution of a one dimensional wave packet in a potential field with high efficiency and precision. We use a Chebyshev polynomial expansion to approximate the time evolution operator because of the rapid convergence of the coefficients of this particular expansion. This property of the Chebyshev method makes it a computationally superior method for numerically approximating this computation. The resulting algorithms are given in pseudo-code, and an implementation is used to generate a graphical demonstration of a typical scattering problem.

1 Introduction

Often quantum systems arise where it is difficult or impossible to analytically calculate a time evolved state. It is useful then to have a numerical approximation of the evolution when exact results are not necessary. This paper discusses the logical development of a set of numerical algorithms that can be used to efficiently and with good accuracy calculate the time development of a wave packet in a potential field in one dimension.

It is possible to achieve a very good approximation of the time evolution operation using a Chebyshev expansion. The Chebyshev polynomials converge exponentially, and it is possible to calculate an optimal number of terms to keep in the expansion. The first section of this paper discusses the expansion of the time development operator.

One of the most computationally expensive parts of the time evolution is the calculation of the second derivative in the Hamiltonian. There is a very efficient approach to the numeric estimation of derivatives using Fast Fourier Transforms, which will be developed in the second part of this paper.

In Section (2) the time evolution operator is expanded in terms of the Chebyshev polynomials. We solve for the coefficients of the expansion and find a complex recursion relation that can be used to find all of the needed polynomials. It will be shown that since the Chebyshev polynomials converge exponentially, there is an optimal number of terms to keep for numeric computation. In Section (3) we find an expression for this value.

We will find that we must "normalize" the Hamiltonian for the expansion. In Section (4) we derive the normalized Hamiltonian. In Section (5) an algorithm for the calculation of the second derivative of a discrete set of data is found using the Fast Fourier Transform. We use this method because it exhibits a comparatively high level of efficiency. However, this method also imposes limits on the possible values for momentum. Use of the normalized Hamiltonian requires knowledge of the minimum and maximum possible values for the energy of the system, which will be estimated in Section (6) by considering the natural limits on momentum.

Finally, in Section (7) the results from the preceding sections are used to construct a numeric algorithm for computing the time evolution of a discretely represented wave packet in a potential field. The algorithm is given in the form of pseudo-code. An implementation of the algorithm that was developed with this paper is demonstrated graphically in a relevant scattering problem.

2 Chebyshev Expansion Of A Complex Exponential

In this section we expand the time evolution operator in terms of the Chebyshev Polynomials of the first kind. The time evolution operator is

$$\hat{U} = e^{-\frac{it\hat{H}}{\hbar}}. \tag{1}$$

If there is some initial wave packet $\psi(x, 0)$, then the time-evolved wave packet is

$$\psi(x, t) = \hat{U}\psi(x, 0). \tag{2}$$

In general, the exponential of an operator is defined in terms of a polynomial expansion. Because of its exponential convergence, we choose to apply a Chebyshev polynomial expansion. We will perform the expansion on a function of the general complex exponential form

$$T(u) = e^{-i\alpha u} \tag{3}$$

for simplicity of notation, and then map u to the Hamiltonian and α to the remaining exponential constants.

The complete expansion of $T(u)$ is

$$T(u) = \sum_{m=0}^{\infty} A_m T_m(u), \tag{4}$$

where

$$-1 \leq u \leq 1. \tag{5}$$

A_m are the coefficients of the expansion and $T_m(u)$ are the Chebyshev polynomials of the first kind. The quantity u must be limited to the range $[-1, 1]$ since that is the range over which the Chebyshev polynomials are defined. The Chebyshev polynomials are defined by

$$\begin{aligned} T_0(u) &= 1 \\ T_1(u) &= u \\ T_m(u) &= 2uT_{m-1}(u) - T_{m-2}(u), \end{aligned} \tag{6}$$

and are orthogonal over $[-1, 1]$ by the orthogonality relation

$$\int_{-1}^{1} T_n(u)T_m(u)\frac{1}{\sqrt{1-u^2}}du = \begin{cases} 0 & n \neq m \\ \pi & n = m = 0 \\ \frac{\pi}{2} & n = m \neq 0 \end{cases}. \tag{7}$$

So in order to solve for the coefficients A_m, the sum Eq.(4) can be integrated and multiplied by $1/\sqrt{1-u^2}$ so that it takes the form of the orthogonality relation Eq.(7). This will cause the cancellation of all terms in the expansion except for one so that the summation may be discarded. Multiplying Eq.(4) by $T_n(u)(1-u^2)^{-1/2}$ and integrating, where n is a nonnegative integer, we have

$$\begin{aligned} T(u) &= \sum_{m=0}^{\infty} A_m T_m(u) \\ \int_{-1}^{1} T(u)T_n\frac{1}{\sqrt{1-u^2}}du &= \int_{-1}^{1} \sum_{m=0}^{\infty} A_m T_m T_n \frac{1}{\sqrt{1-u^2}}du \\ &= \sum_{m=0}^{\infty} A_m \int_{-1}^{1} T_m T_n \frac{1}{\sqrt{1-u^2}}du \\ &= \frac{\pi}{2}\sum_{m=0}^{\infty} A_m \delta_{n,m} \\ &= \frac{\pi}{2}A_n \\ &\therefore \\ A_m &= \frac{2}{\pi}\int_{-1}^{1} \frac{T(u)T_m(u)}{\sqrt{1-u^2}}du. \end{aligned} \tag{8}$$

At this point we have A_m defined in a completely general way. If we specify $T(u)$ with Eq.(3) then there is a convenient simplification that can be made for A_m in terms of the Bessel functions of the first kind J_m.

[Add justification]

$$A_m = 2(-i)^m J_m(\alpha) \tag{9}$$

So the expansion of Eq.(3) is

$$\begin{aligned} e^{-i\alpha u} &= \sum_{m=0}^{\infty} A_m T_m(u) \\ &= 2\sum_{m=0}^{\infty} (-i)^m J_m(\alpha) T_m(u). \end{aligned} \tag{10}$$

It will be convenient to "absorb" the $(-i)^m$ term into the polynomial. Then we define a complex version of the Chebyshev polynomials,

$$\Phi_m(u) = (-i)^m T_m(u). \tag{11}$$

Using the recursion relation for $T_m(u)$ Eq.(6), we can find the recursion relation for $\Phi_m(u)$,

$$\begin{aligned} \Phi_0(u) &= 1 \\ \Phi_1(u) &= -iu \\ \Phi_m(u) &= -2iu\Phi_{m-1}(u) + \Phi_{m-2}(u). \end{aligned} \tag{12}$$

Now we can express the expansion Eq.(10) in terms of the complex Chebyshev polynomials.

$$\begin{aligned} e^{-i\alpha u} &= 2\sum_{m=0}^{\infty} J_m(\alpha)\Phi_m(u) \\ &= \sum_{m=0}^{\infty} a_m(\alpha)\Phi_m(u), \end{aligned} \tag{13}$$

where we have defined

$$a_m(u) = \begin{cases} J_0(\alpha) & m = 0 \\ 2J_m(\alpha) & m \neq 0 \end{cases}. \tag{14}$$

At this point we have an exact series representation of a general complex exponential, Eq.(13). However, since we plan to perform the computation numerically, we must truncate the series.

3 Calculation Of The Optimal Number Of Terms In The Chebyshev Expansion

The coefficients in Eq.(14) are multiples of Bessel functions of the first kind. All but the zeroth of the Bessel functions $J_m(\alpha)$ are asymptotically zero around the origin, and begin oscillating with respect to α after some distance from the origin that is positively dependent on m (see Fig.(1)). So for a particular value of $\alpha > 0$, there is some n_{crit} where $J_m(\alpha)|_{m>m_{crit}} \to 0$. This can also be seen in Fig.(2), where the Bessel functions are plotted for a constant α as discrete functions of m.

For $\alpha \gtrsim 20$, it can be shown that relationship between m_{crit} and α is approximated by

$$m_{crit} \approx \alpha + 11.38\alpha^{0.32}. \tag{15}$$

For smaller values of α, m_{crit} diverges from this expression somewhat, so in order to maintain some good level of accuracy, we impose a minimum expansion of 20 terms. So the maximum index of the expansion is given by

$$M = \max(20, \alpha + 11.38\alpha^{0.32}). \tag{16}$$

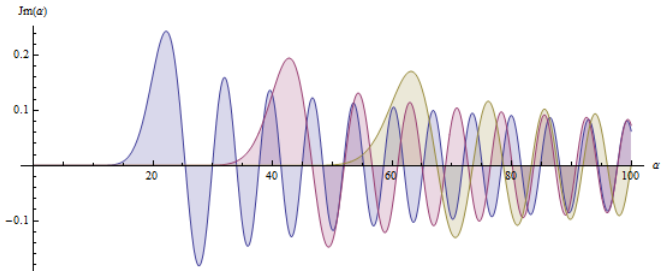

Figure 1: Bessel functions $J_{20}(\alpha), J_{40}(\alpha), J_{60}(\alpha)$

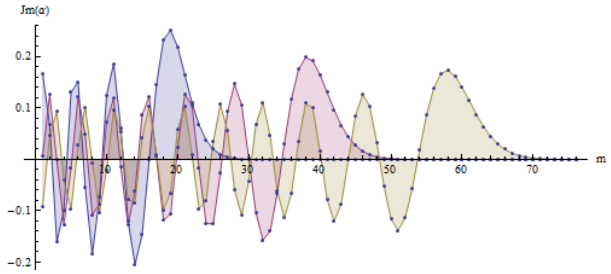

Figure 2: Bessel functions $J_m(20), J_m(40), J_m(60)$

4 Normalization Of The Hamiltonian

Because the Chebyshev polynomials only converge over $[-1, 1]$, the expansion of $e^{-i\alpha u}$ is only defined for u in this range. In Fig.(3), the first ten Chebyshev polynomials have been plotted to illustrate their bounds of convergence. Since we plan to substitute the Hamiltonian operator into this equation in place of u, we must normalize the Hamiltonian to fall within this range. For now we will assume that the minimum and maximum energy eigenvalues that can be found in the system, E_{min} and E_{max} are known. Then we will define the range of possible energies

$$\Delta E = E_{max} - E_{min}. \tag{17}$$

We will define the normalized Hamiltonian $\hat{H}_n$ to be a linear mapping of the Hamiltonian $\hat{H}$, scaled and shifted by the arbitrary constants a and b.

$$\hat{H}_n = a\hat{H} + b\hat{I} \tag{18}$$

Since we want $\hat{H}_n$ to fall in the range $[-1, 1]$, we will map E_{min} and E_{max} onto -1 and 1 respectively. The resulting equations will be used to solve for the constants a and b.

$$\begin{aligned} -1 &= aE_{min} + b \\ 1 &= aE_{max} + b \end{aligned} \tag{19}$$

The solution to the system is

$$\begin{aligned} a &= \frac{2}{\Delta E} \\ b &= -\left(1 + 2\frac{E_{min}}{\Delta E}\right), \end{aligned} \tag{20}$$

so the normalized Hamiltonian is

$$\begin{aligned} \hat{H}_n &= \frac{2}{\Delta E}\hat{H} - \left(1 + 2\frac{E_{min}}{\Delta E}\right) \\ &= \frac{2}{\Delta E}\left(\hat{H} - E_{min}\hat{I}\right) - \hat{I}. \end{aligned} \tag{21}$$

Solving for $\hat{H}$ results in

$$\begin{aligned} \hat{H} &= \left(\frac{\Delta E}{2}\hat{H}_n + \left(2\frac{E_{min}}{\Delta E}\right)\hat{I}\right) \\ &= \frac{\Delta E}{2}\left(\hat{H}_n + \hat{I}\right) + E_{min}\hat{I}. \end{aligned} \tag{22}$$

In order to apply the Chebyshev expansion from the previous section to the time evolution operator $\hat{U}$, we put Eq.(1) in terms of $\hat{H}_n$ via Eq.(22).

$$\begin{aligned} \hat{U} &= e^{-\frac{it}{\hbar}\left(\frac{\Delta E}{2}(\hat{H}_n + \hat{I}) + E_{min}\hat{I}\right)} \\ &= e^{-\frac{it}{\hbar}\frac{\Delta E}{2}\hat{H}_n} e^{-\frac{it}{\hbar}\left(\frac{\Delta E}{2} + E_{min}\right)} \\ &= \lambda(t)\hat{u}, \end{aligned} \tag{23}$$

where

$$\lambda(t) = e^{-\frac{it}{\hbar}\left(\frac{\Delta E}{2} + E_{min}\right)} \tag{24}$$

$$\hat{u} = e^{-\frac{it}{\hbar}\frac{\Delta E}{2}\hat{H}_n}. \tag{25}$$

The form of $\hat{u}$ matches that of the general Chebyshev expansion of a complex exponential Eq.(13). So

$$\hat{u}\psi(x,0) = \sum_{m=0}^{\infty} a_m(\alpha)\phi_m, \tag{26}$$

where

$$\alpha = \frac{t\Delta E}{2\hbar} \tag{27}$$

and the functions ϕ_m are expressed in terms of the normalized Hamiltonian in the following way.

$$\begin{aligned}
\phi_0 &= \Phi_0(\hat{H}_n)\psi(x,0) \\
&= \psi(x,0) \\
\phi_1 &= \Phi_1(\hat{H}_n)\psi(x,0) \\
&= -i\hat{H}_n\phi_0 \\
\phi_2 &= \Phi_2(\hat{H}_n)\psi(x,0) \\
&= -2i\hat{H}_n\phi_{m-1} + \phi_{m-2}
\end{aligned} \tag{28}$$

At this point we have expanded the time evolution operator in terms of the normalized Hamiltonian, and found the relationship between the regular and normalized Hamiltonian operators. We now need a method for computing the Hamiltonian that appears in the expansion.

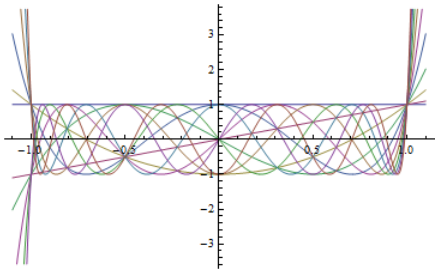

Figure 3: The first 10 Chebyshev polynomials

5 FFT Estimation Of The Second Derivative In The Hamiltonian

In this section, we derive the method that we will use for calculating the second derivative found in the Hamiltonian operator. The Hamiltonian operator is

$$\hat{H} = \hat{K} + \hat{I}V(x) \tag{29}$$

where $\hat{K}$, the kinetic energy operator, is

$$\begin{aligned}
\hat{K} &= \frac{\hat{p}^2}{2m} \\
&= \frac{\hbar^2\hat{k}^2}{2m} \\
&= -\frac{\hbar^2}{2m}\frac{\partial^2}{\partial x^2},
\end{aligned} \tag{30}$$

and

$$\hat{k} = -i\frac{\partial}{\partial x}. \tag{31}$$

Numerically, the derivatives could be estimated by taking every two adjacent points, finding the slope between them, and then repeating the process. However, this can be slow and lacks precision for course grids. A better solution is to employ the Fast Fourier Transform, under which the derivative becomes a simple multiplication in momentum space.

The discrete forward and inverse FFTs of the set of N points f_n are respectively

$$\begin{aligned}
g_m &= \mathcal{F}_D[f_n] \\
&= \sum_{n=0}^{N-1} f_n e^{-2\pi imnN^{-1}}
\end{aligned} \tag{32}$$

$$\begin{aligned}
f_n &= \mathcal{F}_D^{-1}[g_m] \\
&= \frac{1}{N}\sum_{m=0}^{N-1} g_m e^{2\pi imnN^{-1}}.
\end{aligned} \tag{33}$$

At this point we define the lattice spacing to be

$$\Delta x = \frac{L}{N}, \tag{34}$$

where L is the length of the space, for which we keep N points. The position on the x axis of a point with index

n is x_n. So we define

$$\begin{aligned} x_0 &= 0 \\ x_n &= \Delta x n. \end{aligned} \tag{35}$$

If f_n is the discrete representation of the continuous function $f(x)$, we say

$$f_n = f(x_n). \tag{36}$$

The discrete representation of the derivative of $f(x)$ is

$$\begin{aligned} \frac{df_n}{dx_n} &= \frac{d}{dx_n}\left[\mathcal{F}_D^{-1}[g_m]\right] \\ &= \frac{d}{dx_n}\left[\frac{1}{N}\sum_{m=0}^{N-1} g_m e^{2\pi i m n N^{-1}}\right] \\ &= \frac{1}{N}\sum_{m=0}^{N-1} g_m \frac{d}{dx_n}\left[e^{2\pi i m \frac{x_n}{\Delta x} N^{-1}}\right] \\ &= \frac{1}{N}\sum_{m=0}^{N-1}\left(2\pi i m \frac{1}{\Delta x N}\right) g_m e^{2\pi i m N^{-1}} \\ &= \frac{1}{N}\sum_{m=0}^{N-1}(2\pi i m L^{-1}) g_m e^{2\pi i m n N^{-1}} \\ &= \mathcal{F}_D^{-1}[2\pi i m L^{-1} g_m] \\ &= \mathcal{F}_D^{-1}[(2\pi i m L^{-1})\mathcal{F}_D[f_n]]. \end{aligned} \tag{37}$$

And the second derivative can be found in similar fashion.

$$\begin{aligned} \frac{d^2 f_n}{dx_n^2} &= \mathcal{F}_D^{-1}[(2\pi i m L^{-1})^2 \mathcal{F}_D[f_n]] \\ &= \mathcal{F}_D^{-1}[-(2\pi m L^{-1})^2 \mathcal{F}_D[f_n]] \end{aligned} \tag{38}$$

Procedurally speaking, in order to take the second derivative, our algorithm must perform an FFT on the original data $f_n \to g_m$, multiply every point of g_m by the expression $-(2\pi m L^{-1})^2$, and transform it back to "n-space". Then the process is

$$\begin{aligned} f_n &\to g_m \\ -(2\pi m L^{-1})^2 g_m &\to f_n''. \end{aligned} \tag{39}$$

The transforms sum only over positive values for n and m. Effectively this means that the algorithm cannot (at this point) support negative values for position or momentum. For position this is not an issue since we can adjust our initial wave packets and potential functions to show up on the positive x axis. But for momentum this is a problem because we must be able to support movement of wave packets in both directions.

By using a finite Fourier Transform, we are essentially treating a section of a non-periodic function as a periodic function. If it were a continuous transform, the "period" would be infinite in both directions in both spaces. However, we can only sum over values in position space in the range $[0, L]$. In momentum space the range of k is $[0, 2\pi(N-1)L^{-1}]$. However, since the discrete representations of the functions behave periodically under the transform, in order to support negative momenta we can treat the data as values from the range $[0, \pi N L^{-1}]$, concatenated with the data from $[-\pi(N-1)L^{-1}, 0]$.

In the intermediate step, after the forward transform of the wave packet, we can multiply the first $N/2$ points by $-(2\pi m L^{-1})^2$. In order to treat the second half of the points as if they were in the range $[-\pi(N-1)L^{-1}, 0]$, we multiply them instead by $-(2\pi(m-N)L^{-1})^2$.

The only piece of information we have yet to find in order to construct a complete set of algorithms is an estimated number for the minimum and maximum energy eigenvalues. Finding these will allow us to properly normalize the Hamiltonian. In order to obtain these values, we will use the limits on the possible values for momentum that we have just imposed by use of the FFT method for taking the derivative inside the Hamiltonian.

6 Calculation Of The Minimum And Maximum Energy Eigenvalues

We've already noted that k is now bound to the range $[-\pi(N-1)L^{-1}, \pi N L^{-1}]$. From this we can determine the maximum kinetic energy value. By substituting the maximum (or minimum) value for k into the expression for kinetic energy Eq.(30) we find

$$\begin{aligned} K_{max} &= \frac{\hbar^2 k_{max}^2}{2m} \\ &= \frac{\hbar^2}{2m}\left(\pi \frac{N}{L}\right)^2 \\ &= \frac{(\hbar \pi N)^2}{2mL^2}. \end{aligned} \tag{40}$$

So we can estimate the maximum and minimum total energies to be

$$E_{max} = K_{max} + V_{max} \tag{41}$$

$$E_{min} = \begin{cases} V_{min} & V_{min} < 0 \\ 0 & V_{min} \geq 0 \end{cases}.$$

Then we define

$$\Delta E = E_{max} - E_{min}. \tag{42}$$

At this point we have everything needed to construct the numerical computation of the normalized Hamiltonian and the time evolution operator.

7 The Time Evolution Algorithm; Pseudo-Code And Implementation

In this section we will outline the pseudo-code for the algorithms involved in the numeric time evolution. The algorithms will be implemented and demonstrated graphically.

The algorithm is split into three embedded functions. We will call the time evolution function $U(psi[N], V[N], t)$, where $psi[N]$ and $V[N]$ are arrays of size N containing the discrete data representing the initial wave function and the potential function respectively. The time evolution function calls $HNorm(f[N], V[N])$, a function for the normalized Hamiltonian. Finally, $D2(f[N])$ takes the second derivative of the discrete set of N points, $f[N]$.

The algorithms are written in pseudo-code in Listings (1), (2), and (3). The pseudo-code is written in a c-like style. The functions do not return values, but modify the data that is passed in through pointers, for reasons of memory conservation and speed considerations.

In the graphical demonstration shown in Fig.(4), a Gaussian wave packet is given an initial momentum that sends it into a square potential barrier, causing partial reflection and partial transmission. The algorithm was applied five times, each with a time step of 10 nanoseconds. A lattice of size $N = 1024$ is used, and the physical length represented is 100 nanometers. Throughout the process, normalization was maintained to 12 digits.

Listing 1: Time Evolution

```
U(psi[N], V[N], t) {

    // Find alpha(t)
    alpha = (t * DeltaE) / (2 * hBar);

    // Find lambda(t), the factor that resulted from scaling Hnorm
    lambda = exp(-(i * t / hBar) * (DeltaE / 2 + Emin));

    // Save the first two phi_m(x)'s as copies of psi(x, 0).
    // Also zero the psi function - after this point we will only use it to
    // store the total summation of polynomial terms.
    for (n = 0; n < N; n ++) {
        phi_0[n] = phi_1[n]
                 = phi_m[n]
                 = psi[n];
        psi[n]   = 0;
    }

    // Perform the normalized Hamiltonian operation on the second phi term & multiply it
        by -i.
    HNorm(phi_1, V);
    for (n = 0; n < N; n ++) phi_1[n] *= -i;

    // Calculate the number of terms to keep in the summation
    M = max(20, alpha(t) + 11.38 * pow(alpha(t), .32));

    // Begin summation loop
    for (m = 0; m <= M; m ++) {

        // Get phi_m
        if (m > 1) { // This is the nth time through, m > 2

            // Find the new phi_m from the previous two.
            // NOTE: phi_1 -> phi_m_minus_1
            //       phi_0 -> phi_m_minus_2
            for (n = 0; n < N; n ++) phi_m[n] = phi_1[n];
            HNorm(phi_m, V);
            for (n = 0; n < N; n ++) phi_m[n] = (-2*i) * phi_m[n] + phi_0[n];

```

```
39              // Update the containers for the previous two phi_m's
40              for (n = 0; n < N; n ++) {
41                  phi_0[n] = phi_1[n];
42                  phi_1[n] = phi_m[n];
43              }
44          } else if (m > 0) { // This is the 2nd time through
45              for (n = 0; n < 2*N; n ++) phi_m[n] = phi_1[n];
46          }
47
48          // Find the current expansion coefficient, a_m(t)
49          if (m == 0) {
50              a_m = J(0, alpha);
51          } else {
52              a_m = 2 * J(m, alpha);
53          }
54
55          // Add this component to the total wave packet
56          for (n = 0; n < 2*N; n ++) psi[n] += a_m * phi_m[n];
57      }
58
59      // Multiply the entire sum by the Hnorm scaling factor, lambda(t)
60      for (n = 0; n < 2*N; n ++) psi[n] *= lambda;
61  }
```

Listing 2: Normalized Hamiltonian

```
1   Hnorm(f[N], V[N]) {
2
3       // Make a copy of the f array and take its derivative
4       for (n = 0; n < N; n ++) Df[n] = f[n];
5       D2(Df);
6
7       // Perform the algebra on each point in the f array
8       for (n = 0; n < N; n ++) {
9           f[n] = (2 / DeltaE)
10                  * f[n] * (-Emin + DeltaE) / 2
11                  + Df[n] * (-hBar^2 / (2 * m))
12                  + f[n] * V[n]
13      }
14  }
```

Listing 3: Second Derivative

```
1   D2(f[n]) {
2
3       // Do a forward Fast Fourier Transform into momentum space
4       FFT(f);
5
6       // Save the multiplicative coefficient that effectively "takes the derivative"
7       k_over_m_squared = -(4 * pi / L)^2;
8
9       // Multiply the first half of the elements by -(2*pi*m/L)^2 and the second half by
            -(2*pi*(m-N)/L)^2
10      for (s = 0; s <= N / 2; s ++) f[n] *= k_over_m_squared * m^2;
11      for (      ; s < N     ; s ++) f[n] *= k_over_m_squared * (m - N)^2;
12
13      // Do the inverse Fast Fourier Transform back into coordinate space
14      FFT_inv(f);
15  }
```

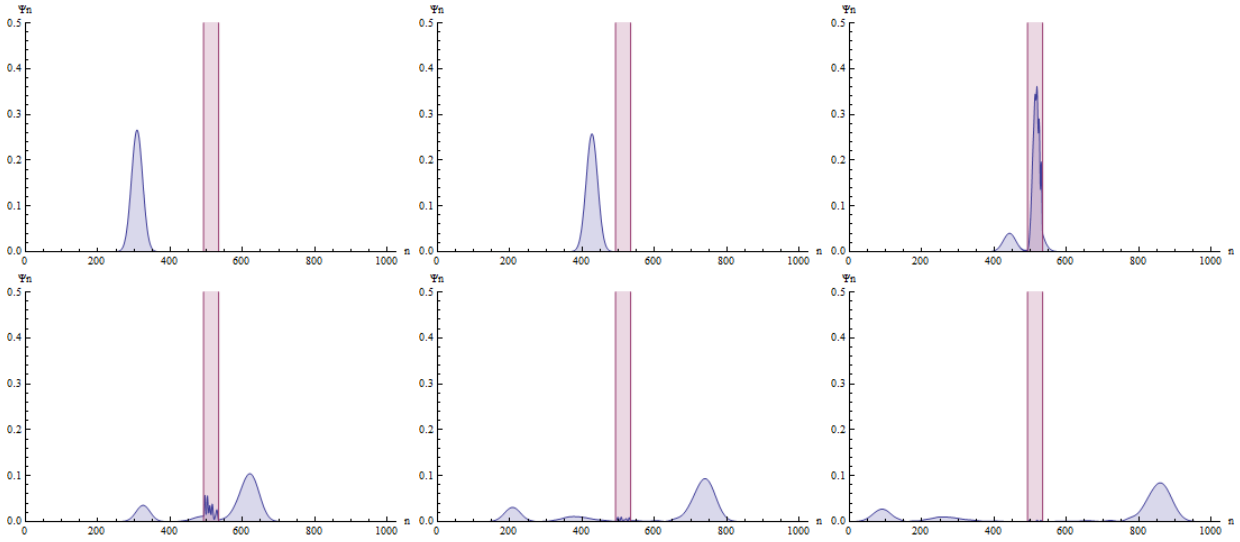

Figure 4: ψ_n with finite square barrier potential, time evolved to $t = 0, 10, 20, 30, 40, 50$ ns

8 Conclusion

We have shown that the methods discussed in this paper for the numerical estimation of the time evolution of one dimensional quantum systems are numerically precise and computationally efficient. In particular, we took advantage of the usefulness of the Chebyshev method for approximating exponentials, which grows from the rapid convergence of the coefficients of a Chebyshev polynomial expansion.

This paper was developed with the intention of eventually expanding these methods to two dimensional systems. Such an expansion, though not trivial, should be very possible given this basis on which to build.