Department of Computer Science
Colorado School of Mines

CSM# 2

A Method for Approximating Ground-state Wave Functions of Schrödinger's Equation

Client:
Scott Strong

Authors:
Jonathan Hendricks,
Thomas Cullison

Adviser:
Dr. Roman Tankelevich

June 18, 2008

Contents

1 Introduction **2**
1.1 Objective 2
1.2 Background 2

2 Requirements and Specifications **3**
2.1 Non-Functional Requirements 3
2.2 Functional Requirements 4

3 Design and Solution Approach **5**
3.1 Theoretical Solution Approach 5
3.2 Computational Solution Approach 7
3.3 Software Design 9
3.3.1 Computational Software Design 10
3.3.2 GUI Software Design 11

4 Implementation **11**
4.1 Computational Implementation 12
4.2 GUI Implementation 13

5 Results **13**

6 Conclusion **15**

7 Future Work **15**

References **17**

Appendices **18**

A Hermite Integral Equations **18**

B Figures **20**

Abstract

The goal of this project was to develop a MATLAB software package that when given an eigenfunction basis (i.e. Hermite and Legendre polynomials, or sines and cosines), can approximate a strict upper-bound on the ground state energy of the linear and nonlinear time independent Schrödinger equations for arbitrary potential functions; furthermore, to develop an algorithm to minimize the ground state energy, and to numerically and graphically display the resulting probability density functions. The evaluating algorithm incorporates the variational principle and Lagrange multipliers to determine the coefficients of a bounding eigenbasis. The results are reported numerically and graphically via a graphical user interface (GUI), and multiple physically motivated cases (different potential wells) were analyzed.

Key words. Bose-Einstein condensate, variational principle, nonlinear Schrödinger equation, Hermite polynomials, eigenbasis functions, Hamiltonian.

1 Introduction

1.1 Objective

The objective of this project is to develop a MATLAB program that can approximate a strict upper-bound on the ground state energy of the linear and nonlinear time independent Schrödinger equations for arbitrary potential functions.

1.2 Background

The variational principle of quantum mechanics allows us to place an upper-bound on the ground-state energy E_g for a system described by the Hamiltonian $\hat{H}$, for any normalized function $\psi(x)$ [1].

$$E_g \leq \int_{-\infty}^{\infty} \psi(x)\hat{H}\psi(x)dx \equiv \langle H \rangle \tag{1.1}$$

To do this we will use eigenfunctions of $\hat{H}$ that form an orthonormal basis of solutions to Schrödinger's equation.

$$\psi = \sum_{n=0}^{\infty} a_n \psi_n, \qquad \text{with} \qquad H\psi_n = E_n \psi_n \tag{1.2}$$

Specifically, we will be examining the Hermite Polynomial eigenbasis for infinite domains. We seek to minimize the constants a_n with respect to Equation 1.1 in order to obtain a strict upper-bound of the ground state energy.

Example: Define the linear Hamiltonian operator

$$H_L = \frac{\hbar^2 b}{2m}\frac{d^2}{dx^2} + \frac{1}{2}m\omega^2 x^2 \tag{1.3}$$

assuming a Gaussian wave function

$$\psi(x) = Ae^{-bx^2}, \tag{1.4}$$

where b is a constant which we seek to minimize and $A = (\frac{2b}{\pi})^{1/4}$, as determined by normalization. Using Equation 1.2 we find that

$$\int_{-\infty}^{\infty} \psi(x)H_L(x)\psi(x)dx = \frac{\hbar^2 b}{2m} + \frac{m\omega^2}{8b} = F(b). \tag{1.5}$$

According to the variational principle, $F(b)$ exceeds E_g for any b. To obtain the tightest bounds we will find the minimum of $F(b)$ by using the second derivative test

$$\frac{d}{db}F(b) = \frac{\hbar^2}{2m} - \frac{m\omega^2}{8b^2} = 0 \Rightarrow b = \frac{m\omega}{2\hbar}. \tag{1.6}$$

Placing b back into Equation 1.6 we find that $\langle H\rangle = \frac{1}{2}\hbar\omega$, which happens to be the ground-state energy for the given potential [1]. In most cases, however, we will not choose the trial function with precisely the form of the actual ground state. Also note that when using Hermite Polynomials, we will have a system of n coefficients to minimize, which will require multivariate calculus to solve.

2 Requirements and Specifications

2.1 Non-Functional Requirements

1. Analytically solve as much of the problem as possible
   (a) Research and understand methods for analytically solving partial differential equations (e.g. power series solutions and separation of variables)
   (b) Research and understand the linear and nonlinear Schrödinger equations, the variational principle, Hermite polynomials, and function orthogonality principles and theory
2. Provide a user interface for the program that:
   (a) accepts a parameter for truncation of the $\psi(x)$ expansion approximation (Equation 1.4)
   (b) accepts an arbitrary potential function as a parameter
   (c) can be updated to include various eigenfunction bases that can be used for the $\psi(x)$ expansion approximations
3. a graphical user interface (GUI) with the similar functionality described above

2.2 Functional Requirements

1. Submit a working MATLAB program that computes and plots an approximated ground-state wave equation based on the following user provided parameters:
    (a) an arbitrary potential function
    (b) the desired expansion approximation order
    (c) the ability to approximate $\psi(x)$ using different eigenfunction basis expansions
2. Program must accept a user defined potential function that is defined using the MATLAB function format
    (a) The program derives the Hamiltonian operator from the provided or selected potential function
3. Program must accept a user provided approximation truncation order value (i.e. $a_0 + a_1x + a_2x^2$ is 2nd order)
4. Time permitting, the program must allow $\psi(x)$ to be expanded using different eigenfunction bases (i.e. Hermite polynomials vs. Legendre polynomials)
    (a) The user selects an expansion approximation eigenfunction basis
    (b) The program derives/uses the expectation function that is appropriate for the selected eigenfunction basis

    (a) The user must provide/select an expansion approximation order that is a positive integer
    (b) The program derives/uses the appropriate expectation approximation function of the provided/selected order
5. Program must minimize the coefficients of the expectation function
    (a) The program must solve for the set of coefficients that correspond to the Lagrange multipliers minimization constraint (Equation 3.8)
6. Program must display the approximated ground-state wave equation
    (a) The program must display at the coefficients of the approximated ground-state wave equation
7. Program must plot the probability distribution functions.
    (a) The program must plot the probability distribution functions that are calculated
    (b) The program must plot the probability distribution functions that correspond to the minimum and maximum energy states
8. Program (GUI) can plot general shape of the user provided/selected potential function

3 Design and Solution Approach

To meet the goals of this project, we had two solution approaches. The first approach was a theoretical approach, and the second approach was a computational approach which required that we develop software to perform computations and then display the results. The purpose of the theoretical approach was to analytically simplify, verify, and get into programmable form the mathematical theory and equations involved in approximating the ground-state energy of the linear and nonlinear time-independent Schrödinger equations.

Once the mathematics were simplified and verified, numerical computation was needed to evaluate integrals and to solve for eigenvectors; furthermore, the final results needed to be displayed as well. Therefore, the purpose of the second approach was to develop software to perform the needed numerical computations and to display the final results including plots of the calculated probability distribution functions and the coefficients of the corresponding wave equations. Per the clients request, the software was developed using MATLAB.

3.1 Theoretical Solution Approach

Methods for approximating the upper-bounds of the ground-state energy using the nonlinear time independent Schrödinger equation were explored; however, an adequate method for finding the upper-bounds was elusive, and due to time constraints, the search for a nonlinear method was abandoned. Therefore, the focus of our project has been mostly on finding upper-bounds using the linear time independent Schrödinger equation.

Starting from the Schrödinger's equation we have,

$$\hat{H}\psi = E\psi, \tag{3.1}$$

where

$$\hat{H} = -\frac{\hbar}{2m}\frac{\partial^2}{\partial x^2} + V(x). \tag{3.2}$$

Because the eigenfunctions of $\hat{H}$ form a complete set, $\psi(x)$ can be expressed as a linear combinations of these functions

$$\psi(x) = \sum_{n=0}^{\infty} a_n \psi_n, \tag{3.3}$$

where $\psi(x)$ is any normalized function such that $\langle\psi|\psi\rangle = 1$ with respect to whatever inner product the space is endowed with. From the variational principle we have

$$E_g \leq \langle H\rangle = \langle\psi|H|\psi\rangle\,, \tag{3.4}$$

where E_g is the ground state energy. Under the Hermite basis

$$\psi(x) = \sum_{n=0}^{\infty} a_n \frac{1}{\sqrt{n!2^n\sqrt{\pi}}} e^{-x^2/2} H_n(x) = \sum_{n=0}^{\infty} a_n e_n(x), \tag{3.5}$$

where $H_n(x)$ is a Hermite Polynomial of degree n. By substituting the Hermite Basis into the variational principle, we can bound the ground-state energy with

$$\begin{aligned}\langle H\rangle = \langle\psi|\, H\, |\psi\rangle &= \int_{-\infty}^{\infty} \left[\sum_{n=0}^{\infty} a_n e_n(x)\right] \hat{H} \left[\sum_{m=0}^{\infty} a_m e_n(x)\right] e^{-x^2} dx \\ &= \sum_{n=0}^{\infty} a_n \sum_{m=0}^{\infty} a_m \int_{-\infty}^{\infty} e_n(x)\hat{H} e_m(x) e^{-x^2} dx \\ &= \sum_{n=0}^{\infty} a_n \sum_{m=0}^{\infty} a_m C_{nm},\end{aligned} \tag{3.6}$$

where

$$C_{mn} = \int_{-\infty}^{\infty} e_n(x)\hat{H}_i e_m(x) e^{-x^2} dx. \tag{3.7}$$

Now we seek to minimize the coefficients of $\langle H\rangle$ in order to place strict bounds to the ground state energy. To do this, we will use Lagrange multipliers

$$\nabla F = \lambda \nabla G \tag{3.8}$$

with $F(a_0, a_1, ..., a_N) = \langle H\rangle$, and a constraint function, G, which uses the fact that $\psi(x)$ is a normalized function (i.e. $\langle\psi|\psi\rangle = 1$). Exploiting the orthogonality of the normalized Hermite Basis $e_n(x)$, we see that

$$\begin{aligned}\int_{-\infty}^{\infty} \psi(x)\psi(x) &= \int_{-\infty}^{\infty} \sum_{n=0}^{\infty} a_n e_n(x) \sum_{m=0}^{\infty} a_m e_m(x) dx \\ &= \sum_{n=0}^{\infty} a_n \sum_{m=0}^{\infty} a_m \int_{-\infty}^{\infty} e_n(x) e_m(x) dx \\ &= \sum_{n=0}^{\infty} a_n^2 = 1.\end{aligned} \tag{3.9}$$

This gives us our constraint function

$$G(a_0, a_1, ..., a_N) = \sum_{n=0}^{\infty} a_n^2 = 1. \tag{3.10}$$

Taking the gradient of $F = \langle H\rangle$ with respect to a_i gives

$$\begin{aligned}[\nabla F(a_0, a_1, ..., a_N)]_i &= \frac{\partial}{\partial a_i}\left[\sum_{n=0}^{\infty} a_n\right]\sum_{m=0}^{\infty} a_m C_{mn} + \frac{\partial}{\partial a_i}\left[\sum_{n=0}^{\infty} a_m\right]\sum_{n=0}^{\infty} a_n C_{mn} \\ &= \sum_{m=0}^{\infty} a_m C_{im} + \sum_{n=0}^{\infty} a_n C_{ni}.\end{aligned} \tag{3.11}$$

Then, by reindexing such that $j = m, n$, Equation 3.11 can be rewritten as:

$$\sum_{j=0}^{\infty} a_j \left\{C_{ij} + C_{ji}\right\} = [\nabla F(a_0, a_1, ..., a_N)]_i\,. \tag{3.12}$$

This defines a linear system:

$$\nabla F(a_0, a_1, ..., a_N) = D\vec{a}, \qquad \vec{a} = [a_1 \cdots a_N]^T \tag{3.13}$$

where

$$D = \begin{bmatrix} D_{00} & D_{01} & \cdots & D_{0N} \\ D_{10} & D_{11} & \cdots & D_{1N} \\ \vdots & \vdots & \ddots & \vdots \\ D_{N0} & D_{N1} & \cdots & D_{NN} \end{bmatrix}. \tag{3.14}$$

and

$$D_{ij} = C_{ij} + C_{ji} \tag{3.15}$$

Now taking the gradient of $G(a_0, a_1, ..., a_N)$ with respect to a_i gives

$$[\nabla G]_i = \frac{\partial}{\partial a_i}\left(\sum_{n=0}^{\infty} a_n^2\right) = 2a_i \tag{3.16}$$

Therefore,

$$\nabla F = \lambda \nabla G \qquad \Leftrightarrow \qquad D\vec{a} = \lambda 2\vec{a} \tag{3.17}$$

which is an eigenvalue problem, we can solve for the unknown eigenvectors $\vec{a}$. To find out which of these vectors is the global minimum, we will plug the constants into Equation 3.6 and see which vector gives the tightest bound on the ground state energy.

3.2 Computational Solution Approach

When calculating the D matrix (Equation 3.14), the critical points of $\langle H \rangle$ need to be determined. To determine these, we consider the equation

$$D_{mn} = \int_{-\infty}^{\infty} e_n(x)\hat{H}e_m(x)e^{-x^2}dx. \tag{3.18}$$

This calculation can be costly when using only numerical integration techniques. To reduce the number of calculations, we start by separating the integral into several parts. First, we will consider the dimensionless form of the linear Hamiltonian $\hat{H}$ which can be written as

$$\hat{H} = -\frac{\partial^2}{\partial x^2} + V(x). \tag{3.19}$$

Pulling out the constants and inserting $\hat{H}$ into Equation 3.18 yields

$$\begin{aligned} D_{mn} &= \int_{-\infty}^{\infty} e_n(x)\hat{H}e_m(x)e^{-x^2}dx \\ &= \int_{-\infty}^{\infty} \frac{H_n(x)}{\sqrt{n!2^n\sqrt{\pi}}}\hat{H}\frac{H_m(x)}{\sqrt{m!2^m\sqrt{\pi}}}e^{-x^2}dx \\ &= K(n,m)\int_{-\infty}^{\infty} H_n(x)\left(-\frac{\partial^2}{\partial x^2} + V(x)\right)H_m(x)e^{-x^2}dx \end{aligned} \tag{3.20}$$

where

$$K(n,m) = \frac{1}{\sqrt{\pi n!m!2^{n+m}}}. \tag{3.21}$$

When distributing the $H_m(x)e^{-x^2}$ we must calculate its double derivative which can be shown to equal:

$$\begin{aligned} \frac{\partial^2}{\partial x^2}\left[H_m(x)e^{-x^2}\right] &= \left\{4(m-1)mH_{m-2}(x) - 4mxH_{m-1}(x) + \left(x^2-1\right)H_m(x)\right\}e^{-x^2} \\ &= \gamma. \end{aligned} \tag{3.22}$$

Now plugging back into Equation 3.20 yields

$$\begin{aligned} D_{nm} &= K(n,m)\int_{-\infty}^{\infty} H_n(x)\left(-\gamma + V(x)H_m(x)e^{-x^2}\right)dx \\ &= K(n,m)\left[I_1 + I_2 + I_3 + I_4 + I_V\right] \end{aligned} \tag{3.23}$$

where

$$\begin{aligned}
I_1 &= -4m(m-1)\int_{-\infty}^{\infty} H_n(x)H_{m-2}(x)e^{-x^2}dx \\
I_2 &= 4m\int_{-\infty}^{\infty} xH_n(x)H_{m-1}(x)e^{-x^2}dx \\
I_3 &= -\int_{-\infty}^{\infty} x^2H_n(x)H_m(x)e^{-x^2}dx \\
I_4 &= \int_{-\infty}^{\infty} H_n(x)H_m(x)e^{-x^2}dx \\
I_V &= \int_{-\infty}^{\infty} H_n(x)V(x)H_m(x)e^{-x^2}dx.
\end{aligned} \tag{3.24}$$

In order to find closed form solutions to the above integrals, we must consider integrals of the form

$$\int_{\infty}^{\infty} x^r H_n(x)H_m(x)e^{-x^2}dx, \tag{3.25}$$

which can be solved analytically using special properties of Hermite Polynomials [2](Solutions to $r = 0, 1, 2, 3, 4$ can be found in the appendix). When considering the integral I_V, we would like to exploit the closed form solution of Equation 3.25 in order to avoid numerical integration. This is possible if $V(x)$ can be written as a power series

$$\sum_{n=0}^{N} c_n x^n, \tag{3.26}$$

however, if this is not possible, we must evaluate I_V numerically using MATLAB's trapezoidal rule, which will give us approximate values for the D matrix.

Once the D matrix is calculated we then calculate its eigenvectors, in accordance with the Lagrange multipliers, to find the critical points. This is done using built in MATLAB function that return the desired vectors. Once calculated, we can then use the corresponding coefficients in the eigenbasis expansion to create the respective wave function $\psi(x)$. With a set of $\psi(x)$'s, we will determine which has the tightest bound on the ground state energy by plugging in the respective $\psi(x)$ back into $\langle H \rangle$.

3.3 Software Design

The software package can be broken into two software suites, a computational suite and a GUI suite. The computational suite is a standalone suite of functions that can be used without the GUI which allows for quick results, and it also makes testing new computational functions easier to do. The GUI component of the software is mostly a means for interactively viewing the computational results such as viewing and comparing the probability density functions for all the expectation energies, the associated expectation energies, and the coefficients of each computed $\psi(x)$. The GUI suite also contains a suite of functions that interface with the computation suite in order to retrieve and store the computational results for the GUI.

3.3.1 Computational Software Design

The computational component of the software was designed using a modular decomposition approach. This design approach was chosen for several reasons. The first is that separate functions can be written that parallel the major steps in the theoretical approach; therefore, this design approach follows neatly with the theoretical approach. The second is that this design approach makes the code easy to read, understand, and maintain. The third is that this approach makes adding optional eigenfunction bases expansions of the wave function $\psi(x)$ easy to implement. The final reason is that this design approach can be easily integrated with a GUI.

The diagram in Figure 1 shows the MATLAB functions created or used for the computational suite. If a function calls another function, then the caller function points to the callee function. The beige colored functions were written by us. The light blue colored functions were included in MATLAB, and the gray colored function was written by David Terr [3].

The pseudo code for each of the computational functions that follows well with the theoretical approach is below.

Pseudo Code for Computational operations

```
Call gseComputationalFunction()

for $m = 1 : N$
    for $n = 1 : N$
        Create $\psi(x)$ using eigenfunction basis expansion
        Create $D_{nm}$ matrix by numerically integrating Equation (1.7)
    end for loop
end for loop

Solve for eigenvectors using Lagrangian multiplier (Equation 1.12)

for $i = 1$ :(number of eigenvectors)
    Solve for expectation value (Equation 1.6) using the $i^{th}$ eigenvector coefficients
end for loop

Find eigenvector coefficient set that is associated with minimum expectation value
Display the wave function expansion of $\psi(x)$ using the minimum eigenvector coefficients
Return $|\psi(x)|^2$ for each expectation energy
OPTION: Plot the probability density function $|\psi(x)|^2$ associated with the minimum
```

expectation energy

end gseComputationalFunction()

3.3.2 GUI Software Design

The design approach for the GUI is a hybrid between an object oriented and modular decomposition approach. The choice to use a hybrid design approach was based on necessity due to the nature of MATLAB's GUI design for components/objects such as popdown menus, axes, text fields, etc . . .. These objects are somewhat object-oriented, yet they are also somewhat "black boxed." Therefore, helper functions were created to assist with changing the attributes of some of these GUI objects and for maintaining the computational data that is shared between these GUI objects.

The diagram in Figure 2 shows the MATLAB functions created and used for the GUI suite. If a function calls another function, then the caller function points to the callee function. The beige colored functions were written by us. The light green (or mint-green) colored functions were provided by MATLAB via the *guide* function; however, the bodies of these functions were written by us. The light green functions are callback functions which are called when a GUI object action is initiated by a user such as selecting an option from the *Truncation Value* popdown menu.

The Get Computational Data suite of functions (see Figure 2) were created as an interface between the GUI and the computational suite of functions. These interface functions are called by the GUI, and they in turn call the computational functions and pass the resulting computational data to the GUI.

The GUI was also designed to allow the user to pass a truncation value as argument when launching the GUI ($\psi(x)$ is computed for every order less than the truncation value). A default truncation value of 10 is used otherwise.

The layout for the GUI is shown in Figure 3. The layout of the GUI was created to be simple and easy to use, yet it still needed to display all the necessary data. Therefore, only basic GUI objects/components were used such as popdown menus, text fields and axes (plots).

4 Implementation

The computational suite was written in MATLAB per the client's request. The GUI suite was written in MATLAB because it made it easier to interface the GUI with the computational suite. Whenever possible, existing MATLAB functions were used instead of witting new ones.

4.1 Computational Implementation

When populating the D matrix (Equation 3.14), we must carry out the integrals I_1, I_2, I_3, I_4, and I_V. To do this, we created several closed form functions that allowed us to avoid numerical integration, namely

$$\begin{aligned}
\texttt{closedFormIx0(n,m)} &= \int_{\infty}^{\infty} H_n(x)H_m(x)e^{-x^2}dx \\
\texttt{closedFormIx1(n,m)} &= \int_{\infty}^{\infty} xH_n(x)H_m(x)e^{-x^2}dx \\
\texttt{closedFormIx2(n,m)} &= \int_{\infty}^{\infty} x^2H_n(x)H_m(x)e^{-x^2}dx \\
\texttt{closedFormIx3(n,m)} &= \int_{\infty}^{\infty} x^3H_n(x)H_m(x)e^{-x^2}dx \\
\texttt{closedFormIx4(n,m)} &= \int_{\infty}^{\infty} x^4H_n(x)H_m(x)e^{-x^2}dx.
\end{aligned} \tag{4.1}$$

The closed form solutions to the following equations all contain the Kronecker delta function. In order to carry out these computations, we had to created our own Kronecker Delta function, `Kronic(n,m)`, because MATLAB has no such function.

Using the closed form solution allows us to calculate I_1, I_2, I_3, and I_4 exactly which reduces Equation 3.24 to

$$\begin{aligned}
I_1 &= -4m(m-1)\texttt{closedFormIx0(n,m-2)} \\
I_2 &= 4m\texttt{closedFormIx1(n,m-1)} \\
I_3 &= -\texttt{closedFormIx2(n,m)} \\
I_4 &= \texttt{closedFormIx0(n,m)}.
\end{aligned} \tag{4.2}$$

Ideally, we would like to also reduce the potential $V(x)$ to a sum of closed form solutions in order to obtain exact results for the D matrix. For example, if we have a double well potential $V(x) = \alpha(x^4 - 2\beta^2x^2 + \beta^4)$, we can avoid numerical integration by witting I_V as

$$I_V = \alpha\left[\texttt{closedFormIx4(n,m)} + 2\beta\texttt{closedFormIx2(n,m)} + \beta\texttt{closedFormIx0(n,m)}\right] \tag{4.3}$$

which is the exact solution. If there is no power series representation of $V(x)$, we choose to use MATLAB's built in trapezoidal rule, as it is relatively accurate and performs as fast as if we were to build our own numerical integration function. Also when calculating the D matrix, we pulled out all the constants that did not depend on x and labeled them as $K(n,m)$, as found in Equation 3.21. This is the normalizing function and it is multiplied against I_1, I_2, I_3, I_4, and I_V in order to have a probability density function that sums to 1.

When finding the coefficients for the eigenbasis expansion, we must calculate the eigenvectors of D. To do this, we choose to use MATLAB's `eig()` which automatically exploits any symmetry and diagonalization of the matrix. These exploits can result in more accurate solutions and faster convergence.

With the newly calculated eigenvectors, we can then insert the coefficients into the eigenbasis expansion

$$\psi(x) = \sum_{n=0}^{\infty} a_n \frac{1}{\sqrt{n! 2^n \sqrt{\pi}}} e^{-x^2/2} H_n(x). \tag{4.4}$$

In order do this calculation, a function $H(n, x)$ was created that included everything except for the exponential since it is invariant for any wave function, $\psi_n(x)$

$$H(n, x) = a_n \frac{1}{\sqrt{n! 2^n \sqrt{\pi}}} H_n(x). \tag{4.5}$$

For calculating the Hermite Polynomial $H_n(x)$, we used code that was written by David Terr [3], which returns the coefficients of the n^{th} order Hermite polynomial.

4.2 GUI Implementation

To implement the GUI layout and add GUI components/objects (i.e. popdown menus, fields, and axes), we used the MATLAB *guide* function. This function allowed us to visually construct and layout the GUI, and it created necessary MATLAB .fig and .m GUI files. The .fig file is the GUI object, and the .m file contains functions including the callback functions that interact with the GUI.

The main body of each callback function was written by us. For more information about these functions please see the Programmer's Guide.

The data collected from the computational suite of functions is stored and shared between the GUI objects using the *guidata* data structure that is included in MATLAB. This data structure was used because it was easy to implement.

Per the clients request, the ability to calculate the ground-state energies for several different potentials was built into the GUI. These potentials include a single well (harmonic potential), a double well, a couple lopsided double wells, and a sech(x) function. A second single well potential, which is solved using numerical integration, was also added to the GUI, so that the numerical integrated results can be compared with the closed form integrated results.

5 Results

For most of the built in potentials, we found closed forms of the integrals that were associated with their corresponding Hamiltonian operators. The results in these cases were very good. However, if a user wanted to quickly test and arbitrary potential, or if closed from integrals for a particular Hamiltonian cannot be found, then some numerical integration is needed. When testing these cases, which we did with the single well and sech(x) potentials, the results were mixed.

We found closed form integrals for the single well, the double well, and the double lopsided wells. Figure 4 shows the results for a single well potential when $\psi(x)$ has been approximated to the 29^{th} order and a closed form of the Hamiltonian is used. The probability density functions associated with the minimum and the excited energies look like the analytical solutions. Figure 6 shows the results for a double well potential when $\psi(x)$ has been approximated to the 29^{th} order and a closed form of the Hamiltonian is used. The the probability density functions associated with the minimum and the excited energies in this case were what the client had expected. The same was true for the lopsided double well case shown in Figure 7 (this well is just barely asymmetric). However, the probability density functions associated with a more lopsided double well were unexpected (see Figure 8 and compare the min energy probability density function from this figure with that in Figure 7).

For the sech(x) potential, we did not find the closed from integrals for the corresponding Hamiltonian, so we had to use some numerical integration when calculating the D matrix (Equation 3.14). Figure 9 shows the results when $\psi(x)$ was expanded to the 25^{th} order. The results in this case were what the client had expected; however, the probability distribution function "washes-out" at higher orders which can be seen by examining figures 10 and 11 for which $\psi(x)$ has been expanded to the 42^{nd} and 43^{rd} orders, respectfully. When numerical integration is needed in the Hamiltonian operator, there are also problems associated with the probability density functions of the excited energies (the client is not sure if the excited states have any real physical meaning, nevertheless the numerical and closed form results of these states can still be compared). This behavior can be seen by comparing Figure 4 with Figure 5. Figure 4 shows the results for the single well potential when $\psi(x)$ has been approximated to the 29^{th} order and only closed from integrals have been used in the corresponding Hamiltonian operator. Figure 5 shows the results for the single well potential when $\psi(x)$ has been approximated to the 29^{th} order and numerical integrals have been used in the Hamiltonian operator.

The results in the latter case were unexpected. When using numerical integrations for the single well potential case, the results at lower expansion orders of $\psi(x)$ are good; however, at higher orders the results become increasing bad or unexpected. Unfortunately, for some potential well cases, higher order expansions are required to accurately approximate $\psi(x)$. Therefore, when numerical integration is needed, the error at higher orders may be significant.

We believe the cause of the expected numerical integration behavior is inherit to the numerical integration method that *trapz* uses. Higher order expansions of $\psi(x)$ will likely have many mall peaks and troughs near their edges, which will induce more error when numerically integrating using *trapz*. Due to time constraints, other numerical integration methods such as "Simpson's Rule" were not investigated.

6 Conclusion

Although the results from the linear cases that used Hermite polynomials as an eigenfunction basis were mostly good, we were, initially, mostly interested in the nonlinear Hamiltonian which best describes the Bose-Einstein condensate; furthermore, we were also interested in approximating the bounding wave equations by using several eigenfunction bases. Unfortunately, however, the nonlinear part of Schrödinger's equation made it impossible to use the same theory for the nonlinear case that was being used for the linear case, and due to time constraints, we were only able to implement Hermite polynomials as an approximating eigenfunction basis.

Throughout the progression of the project, we made several attempts to implement a theory that could include the nonlinear case. The first attempt involved an iterative scheme that calculated the coefficients of the eigenbasis expansion one at a time. This approach failed due to there being cubic solutions for the minimized coefficients. This theory did not give us a way to eliminate two of the solutions, which would have resulted in the unique solution that was necessary. Once the iterative scheme failed, we made an attempt to use perturbation theory in order to calculate approximate wave functions. This is a common technique used when the wave function can not be calculated exactly. This theory also failed as it requires a complete eigenbasis expansion for all energy levels of the linear case; however, we only had the expansion for the ground state energy level, which made it impossible to calculate further perturbation. Due to the absence of a bridge from the linear to the nonlinear Hamiltonian, and due to time constraints, we were unable to implement code which bounded the ground state energy of the nonlinear Hamiltonian.

7 Future Work

Time constraints and code debugging limited much of our software development time. As such, there are several features and refinements for both the computational suite and the GUI suite that we would have liked to have added, but we were unable to do. The following is a list of features or refinements that we would like to see included or applied to the software in future releases.

- Add computational functions that expand $\psi(x)$ in other eigenfunction bases (i.e. Legendre polynomials and sines and cosines).
- Add a method for finding the ground-state energy using the nonlinear Schrödinger equation.
- Make it easier for the user to provide an arbitrary potential function via the GUI – currently, new potential functions have to be manually coded into the GUI.
- Further investigate the numerical integration problem and if possible, provide a fix.

- Refactor the computational suite of functions.
  - Minimize the number of computational functions that the user must interface with, possibly into just one function.
  - Reduce the number of computations (e.g. multiplications and additions) that are used to populate the D matrix.
  - Some of the functions return data that were used in earlier releases of the software, but they are not used in the current release; therefore, these data can be removed from the software.

References

[1] Dacid J. Griffiths, *Introduction to Quantum Mechanics* Pearson Prentice Hall, 2006.

[2] George B. Arfken, Hans-Jurgen Weber, *Mathematical Methods for Physicists, 6th Edition* Elsevier, 2005.

[3] David Terr, *HermitePoly.m* , MATLAB Central (2004), available at `http://www.mathworks.com/matlabcentral/fileexchange/loadFile.do?objectId=4911\&objectType=FILE`.

Appendices

A Hermite Integral Equations

Consider the integral of the form

$$I_{x^r}(n, n+p) = \int_{\infty}^{\infty} x^r H_n(x) H_{n+p}(x) e^{-x^2} dx \tag{A.1}$$

where $r, n,$and p are positive integers. In order to compute this integral, we must use two important identities associated with Hermite Polynomials. The first is the recurrence relation

$$H_{n+1}(x) = 2xH_n(x) - 2nH_{n-1} \tag{A.2}$$

which can be re-written as

$$H_n(x) = \frac{1}{2x}\left(H_{n+1}(x) + 2nH_{n-1}\right), \tag{A.3}$$

and the second is the known orthogonal relationship [2]

$$I_{x^2}(n, n+p) = \begin{cases} 2^n\sqrt{\pi}(n+2)! & \text{for } p = 2 \\ 2^{n-1}\sqrt{\pi}(2n+1)! & \text{for } p = 0 \\ 0 & \text{for } p > 2, p = 1. \end{cases} \tag{A.4}$$

Now, lets examine the case when $r = 3$. By replacing H_n with the recurrence relation of Equation A.3, we can rewrite Equation A.1 as

$$\begin{aligned} I_{x^3}(n, n+p) &= \int_{\infty}^{\infty} x^3 H_n(x) H_{n+p}(x) e^{-x^2} dx \\ &= \int_{\infty}^{\infty} x^3 \frac{1}{2x}\left[H_{n+1}(x) + 2nH_{n-1}\right] H_{n+p} e^{-x^2} dx \\ &= \int_{\infty}^{\infty} \frac{x^2}{2}\left[H_{n+1}Hn + p + 2nHn - 1H_{n+p}\right] e^{-x^2} dx \\ &= \frac{1}{2}\left[I_{x^2}(n+1, n+p) + 2nI_{x^2}(n-1, n+p)\right], \end{aligned} \tag{A.5}$$

which can be simplified and rewritten in piecewise form as

$$I_{x^3}(n, n+p) = \begin{cases} 2^n\sqrt{\pi}(n+3)! & \text{for } p = 3 \\ 2^{n-1}\sqrt{\pi}3(n+1)(n+1)! & \text{for } p = 1 \\ 0 & \text{for } p > 3, p = 0, p = 2. \end{cases} \tag{A.6}$$

Similarly, for $r = 4$, we use the recurrence relation of Equation A.3 and simplify

$$\begin{aligned} I_{x^4}(n, n+p) &= \int_{\infty}^{\infty} x^4 H_n(x) H_{n+p}(x) e^{-x^2} dx \\ &= \int_{\infty}^{\infty} x^4 \frac{1}{2x} \left[H_{n+1}(x) + 2nH_{n-1}\right] H_{n+p} e^{-x^2} dx \\ &= \int_{\infty}^{\infty} \frac{x^3}{2} \left[H_{n+1}Hn + p + 2nHn - 1H_{n+p}\right] e^{-x^2} dx \\ &= \frac{1}{2} \left[I_{x^3}(n+1, n+p) + 2nI_{x^3}(n-1, n+p)\right], \end{aligned} \tag{A.7}$$

which can also be simplified and rewritten in piecewise form as

$$I_{x^4}(n, n+p) = \begin{cases} 2^n\sqrt{\pi}(n+4)! & \text{for } p = 4 \\ 2^n\sqrt{\pi}\left[(3+2n)(n+2)!\right] & \text{for } p = 2 \\ 2^{n-2}\sqrt{\pi}\left[3(2n^2+2n+1)n!\right] & \text{for } p = 0 \\ 0 & \text{for } p > 4, p = 1, p = 3. \end{cases} \tag{A.8}$$

In general, we can find the closed form solution of $I_{x^r}(n, n+p)$ using the recurrence

$$I_{x^r}(n, n+p) = \frac{1}{2} \left[I_{x^{r-1}}(n+1, n+p) + 2nI_{x^{r-1}}(n-1, n+p)\right]. \tag{A.9}$$

B Figures

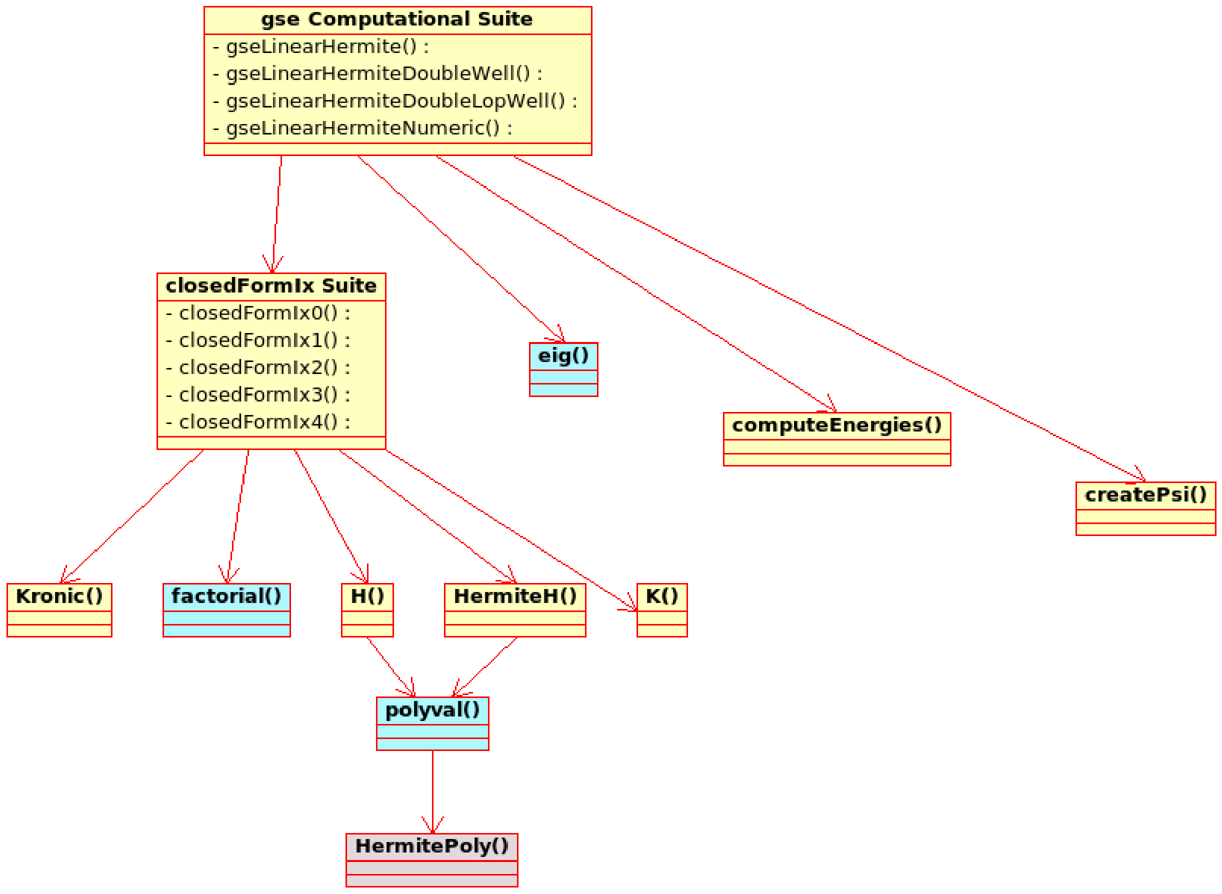


Figure 1: Computational Suite Functional Diagram

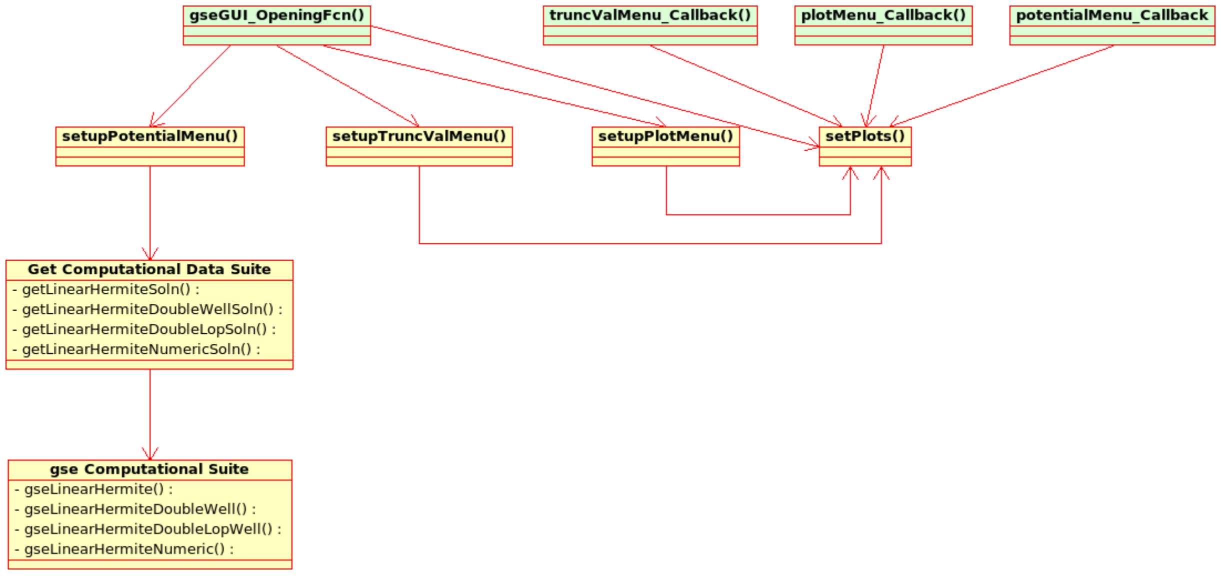
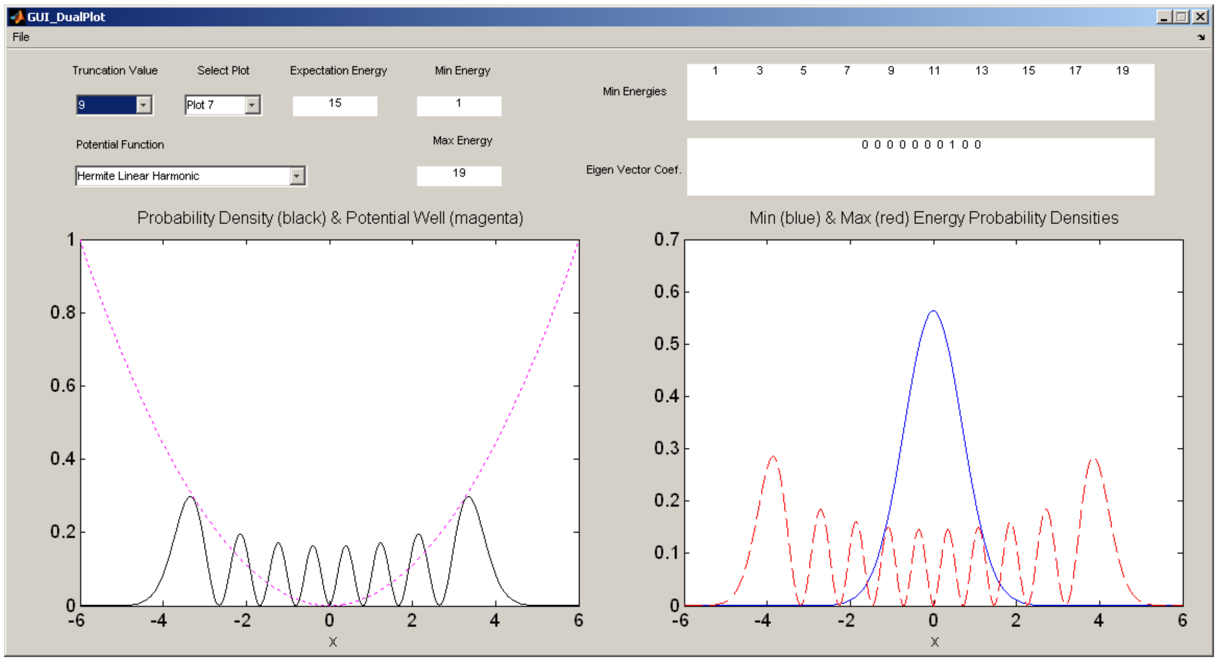
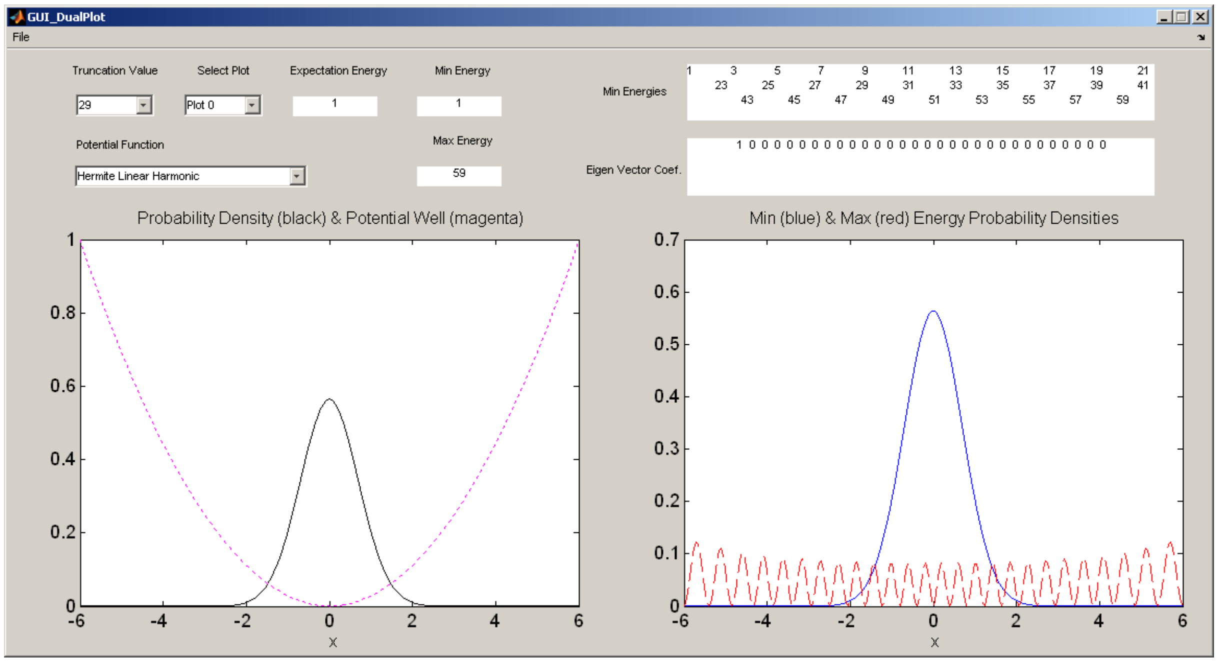

gseGUI_OpeningFcn()
truncValMenu_Callback()
plotMenu_Callback()
potentialMenu_Callback
setupPotentialMenu()
setupTruncValMenu()
setupPlotMenu()
setPlots()
Get Computational Data Suite
- getLinearHermiteSoln() :
- getLinearHermiteDoubleWellSoln() :
- getLinearHermiteDoubleLopSoln() :
- getLinearHermiteNumericSoln() :
gse Computational Suite
- gseLinearHermite() :
- gseLinearHermiteDoubleWell() :
- gseLinearHermiteDoubleLopWell() :
- gseLinearHermiteNumeric() :

Figure 2: GUI Suite Functional Diagram


GUI_DualPlot
File
Probability Density (black) & Potential Well (magenta)
Min (blue) & Max (red) Energy Probability Densities
x
x

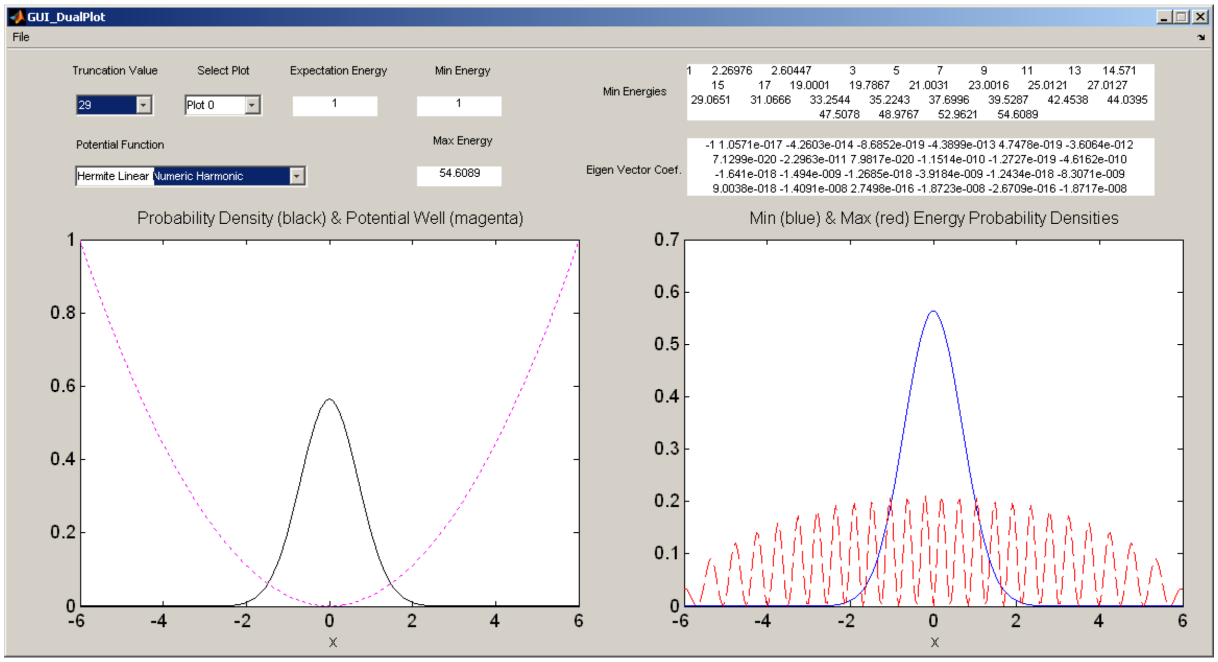
Truncation Value: 9
Select Plot: Plot 7
Expectation Energy: 15
Min Energy: 1
Potential Function: Hermite Linear Harmonic
Max Energy: 19
Min Energies: 1 3 5 7 9 11 13 15 17 19
Eigen Vector Coef.: 0 0 0 0 0 0 0 1 0 0

Figure 3: GUI Interface Layout


GUI_DualPlot
File
Probability Density (black) & Potential Well (magenta)
Min (blue) & Max (red) Energy Probability Densities
x
x

Truncation Value: 29
Select Plot: Plot 0
Expectation Energy: 1
Min Energy: 1
Potential Function: Hermite Linear Harmonic
Max Energy: 59
Min Energies: 1 3 5 7 9 11 13 15 17 19 21 23 25 27 29 31 33 35 37 39 41 43 45 47 49 51 53 55 57 59
Eigen Vector Coef.: 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

Figure 4: Single Well Potential Case Using Closed Form Integrals Only


GUI_DualPlot
File
Probability Density (black) & Potential Well (magenta)
Min (blue) & Max (red) Energy Probability Densities
x
x

Truncation Value: 29

Select Plot: Plot 0

Expectation Energy: 1

Min Energy: 1

Potential Function: Hermite Linear Numeric Harmonic

Max Energy: 54.6089

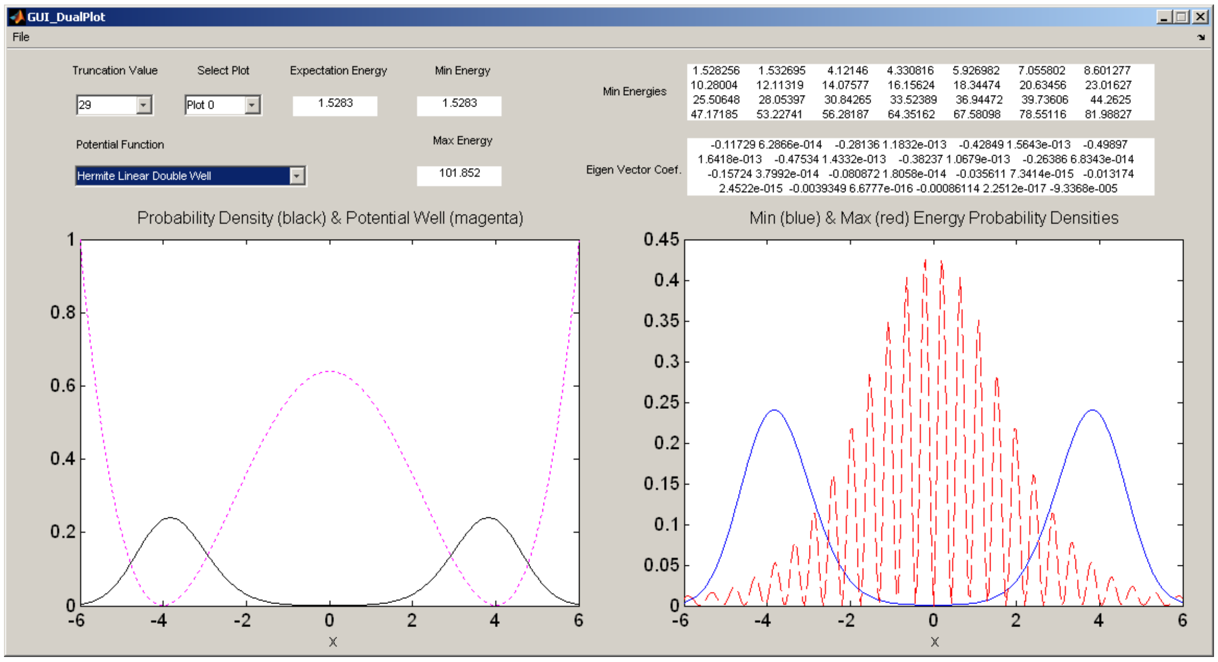
Min Energies:

```
1   2.26976   2.60447   3   5   7   9   11   13   14.571
15   17   19.0001   19.7867   21.0031   23.0016   25.0121   27.0127
29.0651   31.0666   33.2544   35.2243   37.6996   39.5287   42.4538   44.0395
47.5078   48.9767   52.9621   54.6089
```

Eigen Vector Coef.:

```
-1 1.0571e-017 -4.2603e-014 -8.6852e-019 -4.3899e-013 4.7478e-019 -3.6064e-012
7.1299e-020 -2.2963e-011 7.9817e-020 -1.1514e-010 -1.2727e-019 -4.6162e-010
-1.641e-018 -1.494e-009 -1.2685e-018 -3.9184e-009 -1.2434e-018 -8.3071e-009
9.0038e-018 -1.4091e-008 2.7498e-016 -1.8723e-008 -2.6709e-016 -1.8717e-008
```

Figure 5: Single Well Potential Case Using Numeric Integrals


GUI_DualPlot
File
Probability Density (black) & Potential Well (magenta)
Min (blue) & Max (red) Energy Probability Densities
x
x

Truncation Value: 29

Select Plot: Plot 0

Expectation Energy: 1.5283

Min Energy: 1.5283

Potential Function: Hermite Linear Double Well

Max Energy: 101.852

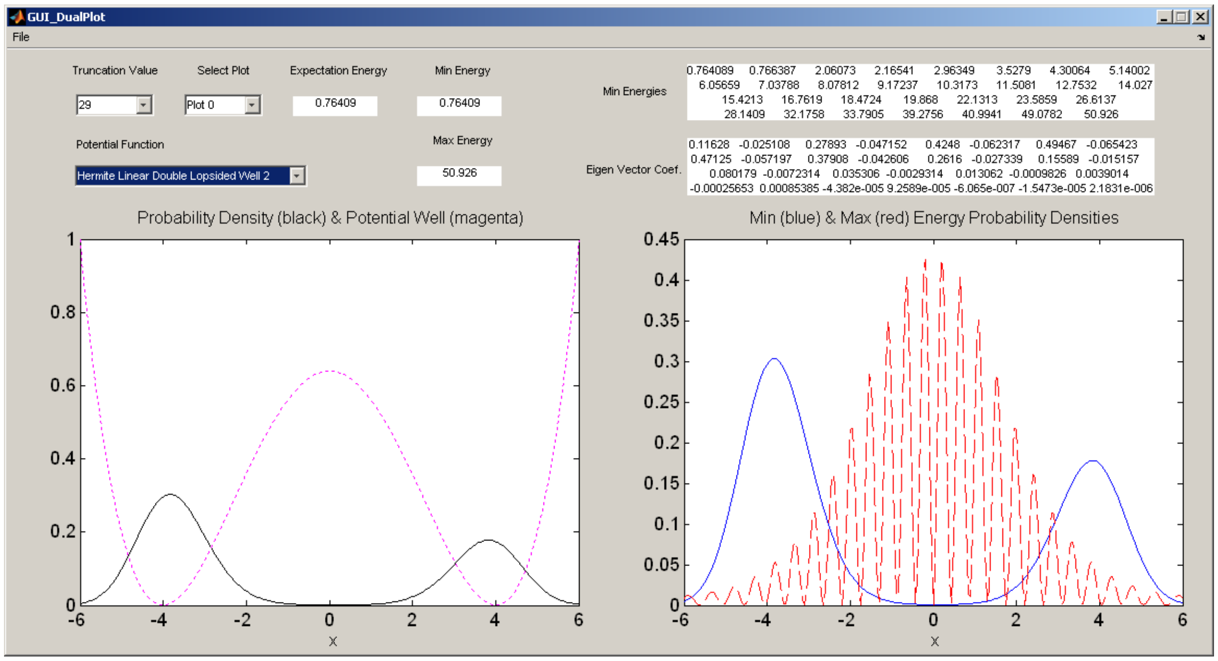
Min Energies:

```
1.528256   1.532695   4.12146   4.330816   5.926982   7.055802   8.601277
10.28004   12.11319   14.07577   16.15624   18.34474   20.63456   23.01627
25.50648   28.05397   30.84265   33.52389   36.94472   39.73606   44.2625
47.17185   53.22741   56.28187   64.35162   67.58098   78.55116   81.98827
```

Eigen Vector Coef.:

```
-0.11729 6.2866e-014   -0.28136 1.1832e-013   -0.42849 1.5643e-013   -0.49897
1.6418e-013   -0.47534 1.4332e-013   -0.38237 1.0679e-013   -0.26386 6.8343e-014
-0.15724 3.7992e-014   -0.080872 1.8058e-014   -0.035611 7.3414e-015   -0.013174
2.4522e-015  -0.0039349 6.6777e-016 -0.00086114 2.2512e-017 -9.3368e-005
```

Figure 6: Double (Symmetric) Well Potential Case Using Closed Form Integrals Only


GUI_DualPlot
File
Probability Density (black) & Potential Well (magenta)
Min (blue) & Max (red) Energy Probability Densities
x
x

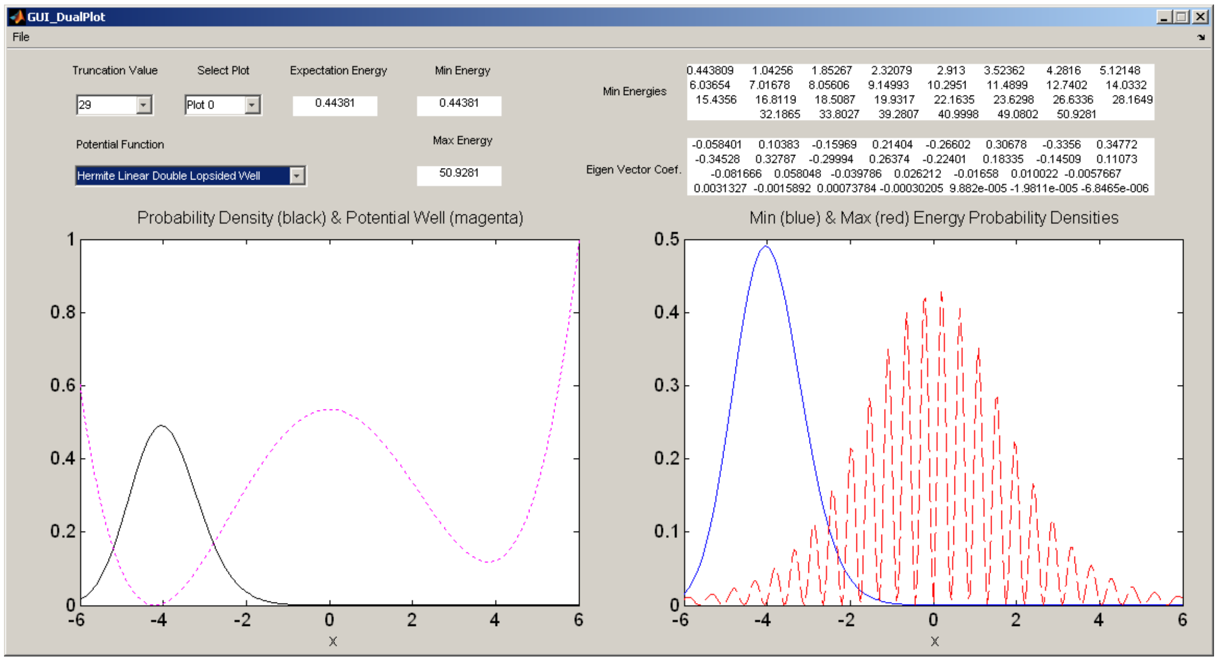
```
Truncation Value   Select Plot   Expectation Energy   Min Energy
29                 Plot 0        0.76409              0.76409

Potential Function                                     Max Energy
Hermite Linear Double Lopsided Well 2                  50.926

Min Energies
0.764089   0.766387   2.06073   2.16541   2.96349   3.5279   4.30064   5.14002
  6.05659   7.03788   8.07812   9.17237   10.3173   11.5081   12.7532   14.027
    15.4213   16.7619   18.4724   19.868   22.1313   23.5859   26.6137
    28.1409   32.1758   33.7905   39.2756   40.9941   49.0782   50.926

Eigen Vector Coef.
0.11628  -0.025108   0.27893  -0.047152   0.4248  -0.062317   0.49467  -0.065423
0.47125  -0.057197   0.37908  -0.042606   0.2616  -0.027339   0.15589  -0.015157
  0.080179 -0.0072314  0.035306 -0.0029314  0.013062 -0.0009826  0.0039014
-0.00025653 0.00085385 -4.382e-005 9.2589e-005 -6.065e-007 -1.5473e-005 2.1831e-006
```

Figure 7: Double Lopsided Well Potential Case Using Closed Form Integrals Only


GUI_DualPlot
File
Probability Density (black) & Potential Well (magenta)
Min (blue) & Max (red) Energy Probability Densities
x
x

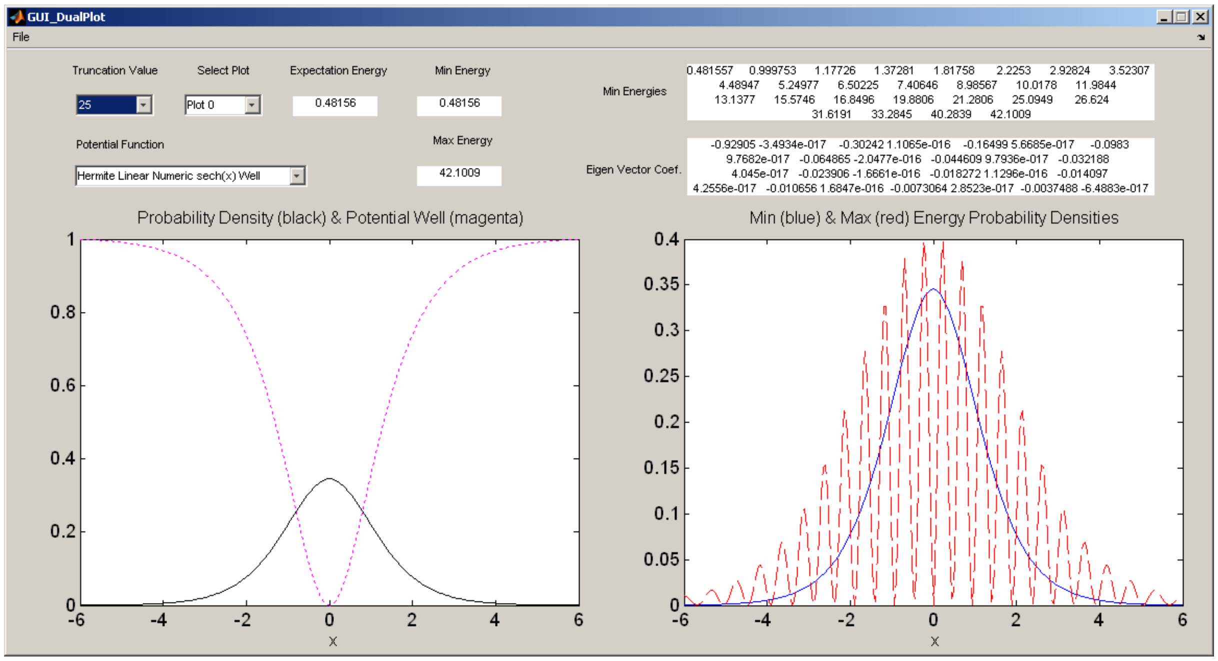
```
Truncation Value   Select Plot   Expectation Energy   Min Energy
29                 Plot 0        0.44381              0.44381

Potential Function                                     Max Energy
Hermite Linear Double Lopsided Well                    50.9281

Min Energies
0.443809   1.04256   1.85267   2.32079   2.913   3.52362   4.2816   5.12148
6.03654   7.01678   8.05606   9.14993   10.2951   11.4899   12.7402   14.0332
 15.4356   16.8119   18.5087   19.9317   22.1635   23.6298   26.6336   28.1649
        32.1865   33.8027   39.2807   40.9998   49.0802   50.9281

Eigen Vector Coef.
-0.058401   0.10383  -0.15969   0.21404  -0.26602   0.30678  -0.3356   0.34772
-0.34528   0.32787  -0.29994   0.26374  -0.22401   0.18335  -0.14509   0.11073
  -0.081666  0.058048  -0.039786  0.026212  -0.01658  0.010022 -0.0057667
0.0031327 -0.0015892 0.00073784 -0.00030205 9.882e-005 -1.9811e-005 -6.8465e-006
```

Figure 8: Double Lopsided (More Asymmetric) Well Potential Case Using Closed Form Integrals Only

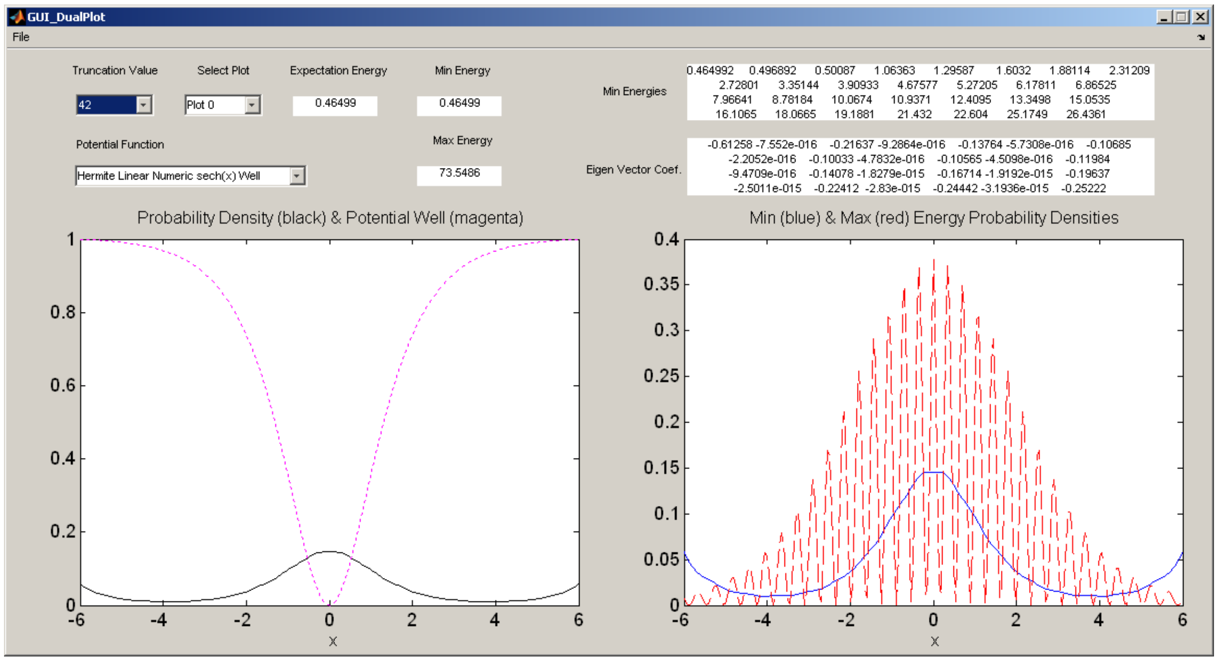

GUI_DualPlot
File
Truncation Value
25
Select Plot
Plot 0
Potential Function
Hermite Linear Numeric sech(x) Well
Probability Density (black) & Potential Well (magenta)
Min (blue) & Max (red) Energy Probability Densities
x
x

Expectation Energy: 0.48156
Min Energy: 0.48156
Max Energy: 42.1009

```
Min Energies
0.481557   0.999753   1.17726   1.37281   1.81758   2.2253   2.92824   3.52307
   4.48947   5.24977   6.50225   7.40646   8.98567   10.0178   11.9844
   13.1377   15.5746   16.8496   19.8806   21.2806   25.0949   26.624
   31.6191   33.2845   40.2839   42.1009

Eigen Vector Coef.
-0.92905 -3.4934e-017  -0.30242 1.1065e-016  -0.16499 5.6685e-017  -0.0983
 9.7682e-017  -0.064865 -2.0477e-016  -0.044609 9.7936e-017  -0.032188
 4.045e-017  -0.023906 -1.6661e-016  -0.018272 1.1296e-016  -0.014097
4.2556e-017  -0.010656 1.6847e-016  -0.0073064 2.8523e-017  -0.0037488 -6.4883e-017
```

Figure 9: sech(x) Well Potential Case Using Numeric Integrals and $\psi(x)$ expansion of order 25

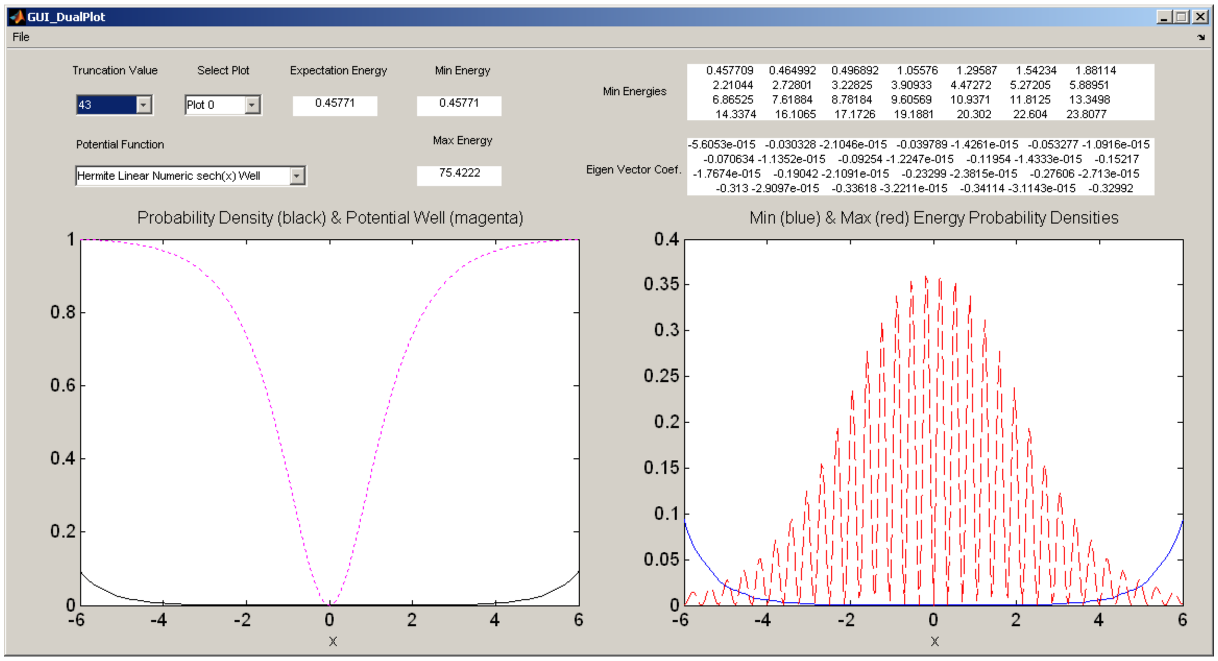

GUI_DualPlot
File
Truncation Value
42
Select Plot
Plot 0
Potential Function
Hermite Linear Numeric sech(x) Well
Probability Density (black) & Potential Well (magenta)
Min (blue) & Max (red) Energy Probability Densities
x
x

Expectation Energy: 0.46499
Min Energy: 0.46499
Max Energy: 73.5486

```
Min Energies
0.464992   0.496892   0.50087   1.06363   1.29587   1.6032   1.88114   2.31209
   2.72801   3.35144   3.90933   4.67577   5.27205   6.17811   6.86525
   7.96641   8.78184   10.0674   10.9371   12.4095   13.3498   15.0535
   16.1065   18.0665   19.1881   21.432   22.604   25.1749   26.4361

Eigen Vector Coef.
-0.61258 -7.552e-016  -0.21637 -9.2864e-016  -0.13764 -5.7308e-016  -0.10685
 -2.2052e-016  -0.10033 -4.7832e-016  -0.10565 -4.5098e-016  -0.11984
 -9.4709e-016  -0.14078 -1.8279e-015  -0.16714 -1.9192e-015  -0.19637
 -2.5011e-015  -0.22412  -2.83e-015  -0.24442 -3.1936e-015  -0.25222
```

Figure 10: sech(x) Well Potential Case Using Numeric Integrals and $\psi(x)$ expansion of order 42


GUI_DualPlot
File
Probability Density (black) & Potential Well (magenta)
Min (blue) & Max (red) Energy Probability Densities
x
x

```
Truncation Value   43
Select Plot        Plot 0
Expectation Energy 0.45771
Min Energy         0.45771
Max Energy         75.4222
Potential Function Hermite Linear Numeric sech(x) Well

Min Energies
   0.457709   0.464992   0.496892    1.05576    1.29587    1.54234    1.88114
    2.21044    2.72801    3.22825    3.90933    4.47272    5.27205    5.88951
    6.86525    7.61884    8.78184    9.60569    10.9371    11.8125    13.3498
    14.3374    16.1065    17.1726    19.1881     20.302     22.604    23.8077

Eigen Vector Coef.
-5.6053e-015  -0.030328 -2.1046e-015  -0.039789 -1.4261e-015  -0.053277 -1.0916e-015
  -0.070634 -1.1352e-015   -0.09254 -1.2247e-015   -0.11954 -1.4333e-015   -0.15217
 -1.7674e-015   -0.19042 -2.1091e-015   -0.23299 -2.3815e-015   -0.27606 -2.713e-015
     -0.313 -2.9097e-015   -0.33618 -3.2211e-015   -0.34114 -3.1143e-015   -0.32992
```

Figure 11: sech(x) Well Potential Case Using Numeric Integrals and $\psi(x)$ expansion of order 43