Numerical Approximation of the 1D Time-Independent Schrodinger Equation

Feynman Liang

January 19, 2013

1 Introduction

Solutions to the Schrodinger equation can oftentimes be approximated using numerical methods. This is especially convenient for dealing with potentials where analytical solutions may very difficult to find or not exist. Here, we will investigate numerical solutions to a 1D finite quantum well with potential:

$$\widetilde{V}(u) = \begin{cases} 0 & |u| \leq 0.5 \\ 1 & |u| > 0.5 \end{cases} \tag{1}$$

Rewriting the Schrodinger equation in a dimensionless form:

$$\frac{d^2\psi(u)}{du^2} = -\beta\left[\epsilon - \widetilde{V}(u)\right]\psi(u) \tag{2}$$

where:

$$u = \frac{x}{a}, \;\; \epsilon = \frac{E}{V_0}, \;\; \beta = \frac{2ma^2V_0}{\hbar^2} \tag{3}$$

Since we expect energies below V_0 to be bound, there is a boundary condition $\psi(u) = 0(x \to \infty)$. This boundary condition problem can be made suitable for numerical computation by employing the shooting method, which reduces the problem into an initial value problem (Note that unlike traditional IVPs, here we are choosing values for ϵ rather than $\psi(0)$ and dealing with asymptotic boundary conditions). Using the shooting method, we will guess and successively adjust the energy term ϵ until $\psi(u)$ satisfies the desired asymptotic behavior. Values of ϵ which satisfy the boundary conditions are allowed energy eigenvalues for the $\widetilde{V}(u)$ potential in (1).

2 Numerical Approximation

The accompanying code was run on Sage 5.5 and Python 3.3.0. The SageTEX package is used to show excecution results in-line within LaTeX.

Two global variables we will use are β (=64, see ER Appendix G) and the potential function $\widetilde{V}(u)$. These are specified at the top of the program:

```
V_potential = lambda u: 1 if (abs(u) > 1/2) else 0
beta = 64
```

In order to numerically approximate the limit behavior of $\psi(u)$, we define a function `shooting_solver_1d_finite` to numerically integrate the dimensionless Schrodinger equation (2). Our implementation uses the Forward Euler method, which computes the quadrature using iterative first order Taylor approximations:

$$\phi(u_{i+1}) = \phi(u_i) - \Delta u \cdot \beta \left[\epsilon - \widetilde{V}(u_i)\right] \psi(u_i) \tag{4}$$

$$\psi(u_{i+1}) = \psi(u_i) + \Delta u \cdot \phi(u_i) \tag{5}$$

The function `shooting_solver_1d_finite` takes initial values `psi0` and `dpsi0` (ψ and $\frac{d\psi}{du}$ at $x_0 = 0$), a guess for `epsilon`, an iteration step size `delta_u`. A list of tuples $(u, \psi(u))$ is returned for the range [0, `uf`].

```
def shooting_solver_1d_finite(psi0, dpsi0, uf, epsilon, delta_u):
    num_steps = (uf - 0) / delta_u
    data = [(0, psi0, dpsi0)] # initialize data array
    for i in range(num_steps): # perform forward euler
        u_old = data[i][0]
        psi_old = data[i][1]
        dpsi_old = data[i][2]
        u = u_old + delta_u
        # Taylor approximations given by (3) and (4)
        dpsi = dpsi_old - delta_u * beta \
               * (epsilon - V_potential(u_old)) * psi_old
        psi = psi_old + delta_u * dpsi_old
        data.append((u, psi, dpsi))
    return map(lambda x: (x[0], x[1]), data) # return list of (u, psi) tuples
```

Because the potential (1) is symmetric ($\widetilde{V}(u) = \widetilde{V}(-u)$), the energy eigenfunctions have a definite parity. Thus, the eigenfunctions exhibit symmetry about the origin and a simulation between $(0, \texttt{uf})$ will also yield the result between $(-(uf), 0)$ by simply mirroring across the y-axis and multiplying $\psi(u)$ by -1 if the eigenfunction is of odd parity.

The `plot_data_finite_well` function takes a list `data` of (u, $\psi(u)$) 2-tuples, the wavenumber `n` to label the y-axis with, a `title` string, and uses the matplotlib library to generates a plot of `data` with dashed lines indicating the boundaries of the finite well potential (1).

***Note**: This plotting routine assumes the data approximates an eigenfunction of definite parity.*

```
import matplotlib.pyplot as plt
def plot_finite_well(data, n, title, chained=False):
    if not chained: plt.clf() # only clear if not part of chained call
    # mirror data across origin, assumes data has a definite parity
    (u, psi0) = data[0]
    if psi0 == 0: # node at 0, odd
        data = map(lambda x: (-x[0], -x[1]), data)[::-1] + data
    else: # definite parity => anti-node at 0, even
        data = map(lambda x: (-x[0], x[1]), data)[::-1] + data

    u, psi = [[x[i] for x in data] for i in (0,1)]
    plt.plot(u, psi)
    plt.title(title)
    plt.xlabel('$u$')
    plt.ylabel("$\\psi_{%s}(u)$" % n)
    plt.axvline(x=0, color='black')
    plt.axvline(x=0.5, linestyle='dashed', color='black')
    plt.axvline(x=-0.5, linestyle='dashed', color='black')
    plt.grid(True)
    plt.legend()
    plt.save = plt.savefig
    return plt
```

3 Simulation and Results

3.1 Ground State of Finite Quantum Well

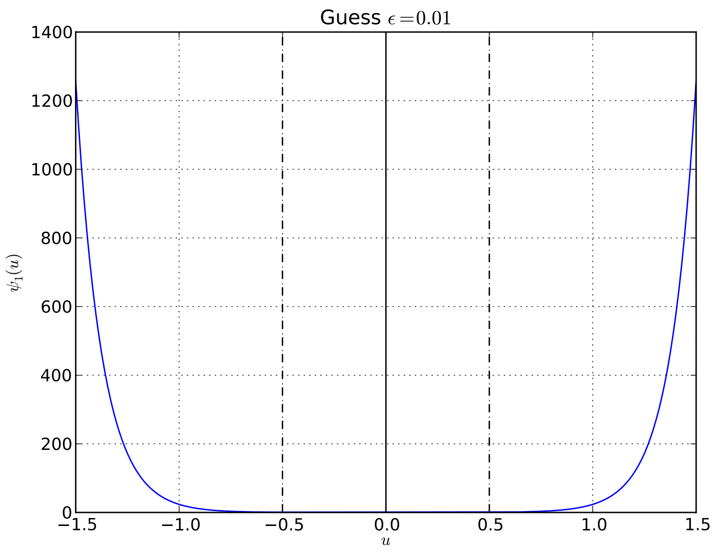
Because the symmetric potential implies energy eigenfunctions of definite parity and the ground state eigenfunction is odd, $\frac{d\psi(u)}{du}$ must necessarily be 0 at $x_0 = 0$.

Rather than worry about the normalization conditions for $\psi(u)$ $\left(\int_{-\infty}^{\infty} \psi^\dagger \psi du = 1\right)$, we recognize that the normalized eigenfunction is simply the unnormalized eigenfunction multiplied by a constant normalization factor. We can compute the normalization factor given the unormalized eigenfunction, so we make the assumption that $\psi = 1.0$ at $x_0 = 0$, recognizing that we are working with the unnormalized eigenfunction. This does not affect our numerical results for ϵ because looking at (2) shows that the normalization constants cancel.

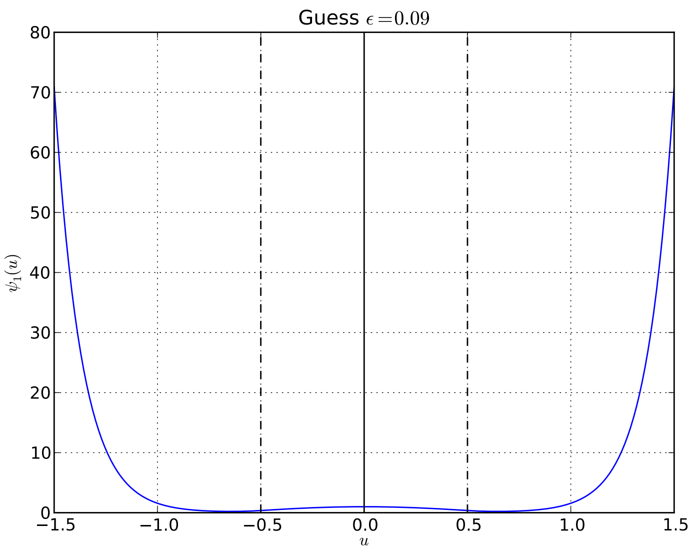
In order to plot our solutions, our simulation will only find eigenfunctions which are real functions. This is not a problem because we have the ability to choose the phase of solutions to the TISE (2) to make $\psi(u) \in \mathbb{R}$ (see P24 - Computation Project Handout).

Using these initial conditions, the "shooting" method to determine ϵ yields:

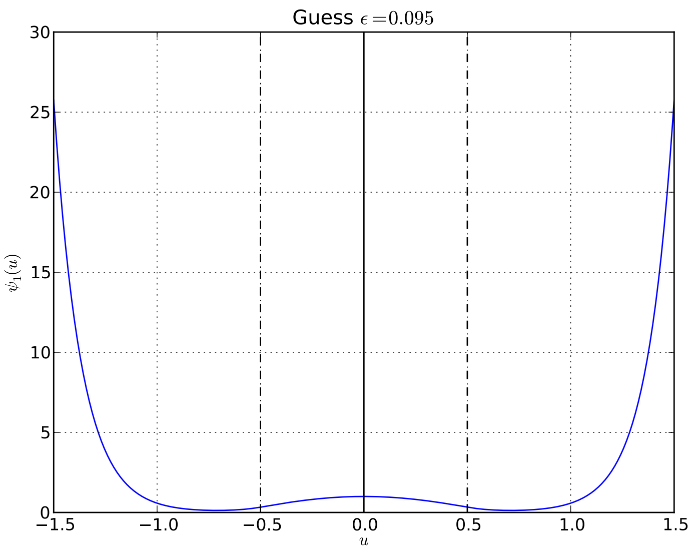
```
plot_finite_well(shooting_solver_1d_finite(1, 0, 1.5, .01, 0.0001), \
                 n=1, title="Guess $\epsilon=0.01$")
```



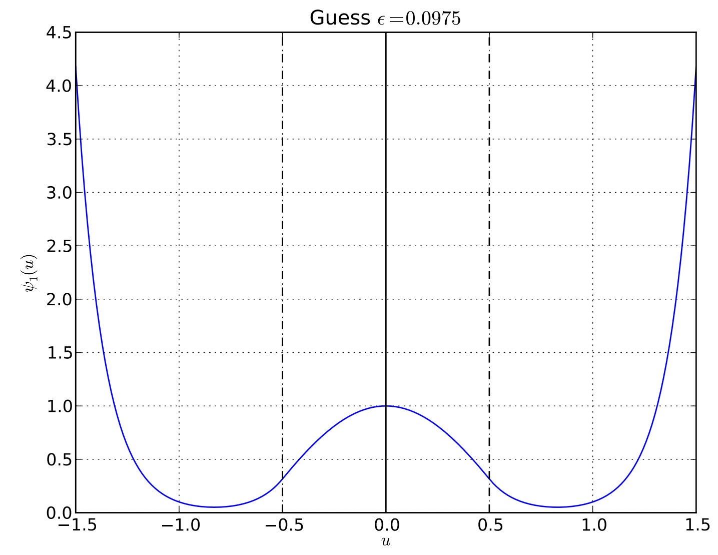
```
plot_finite_well(shooting_solver_1d_finite(1, 0, 1.5, .09, 0.0001), \
                 n=1, title="Guess $\epsilon=0.09$")
```



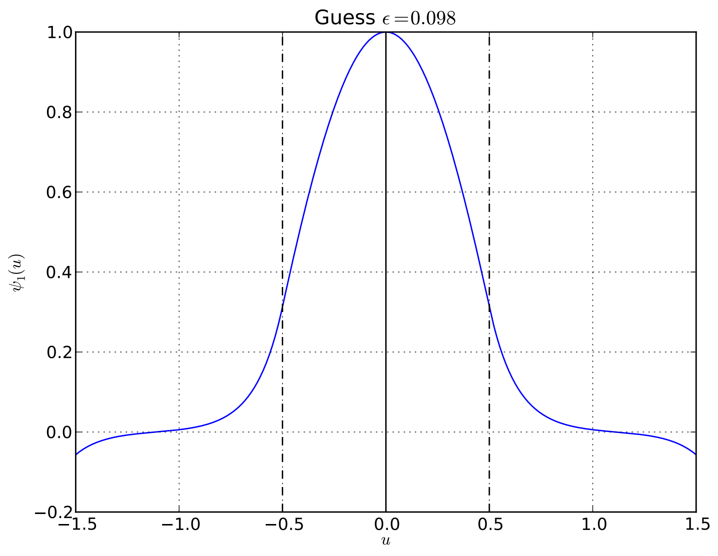
```
plot_finite_well(shooting_solver_1d_finite(1, 0, 1.5, .095, 0.0001), \
                 n=1, title="Guess $\epsilon=0.095$")
```



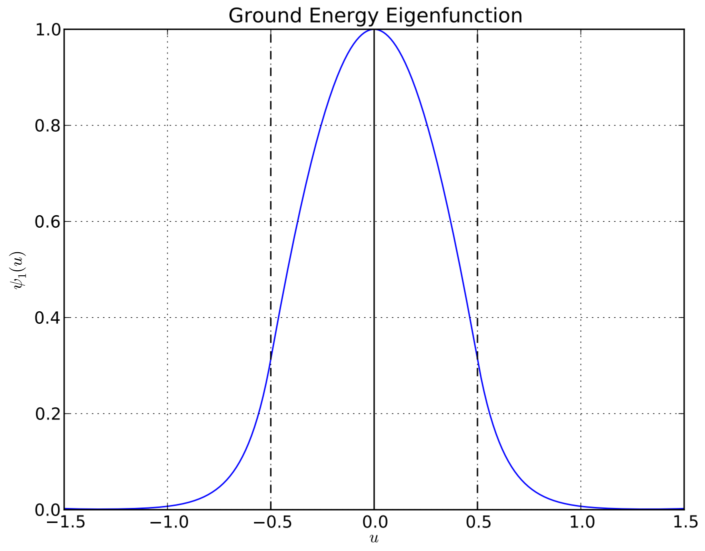
```
plot_finite_well(shooting_solver_1d_finite(1, 0, 1.5, .0975, 0.0001), \
                 n=1, title="Guess $\epsilon=0.0975$")
```



```
plot_finite_well(shooting_solver_1d_finite(1, 0, 1.5, .098, 0.0001), \
                 n=1, title="Guess $\epsilon=0.098$")
```



```
plot_finite_well(shooting_solver_1d_finite(1, 0, 1.5, .097993, 0.0001), \
                                   n=1, title="Ground Energy Eigenfunction")
```



This shows that a valid solution is found at $\epsilon = 0.097993$, implying that an energy eigenstate exists with eigenvalue $0.097993V_0$, where V_0 is the height of the finite well potential. We see that there is a single node in this wavefunction, indicating that this eigenstate is the ground state.

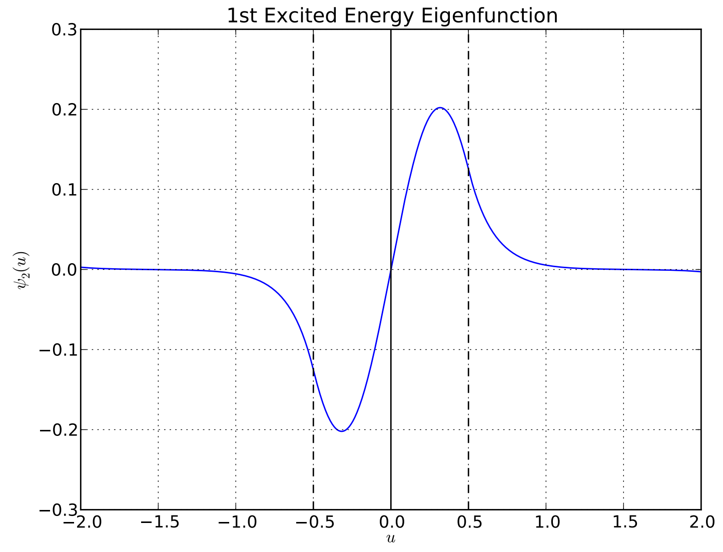
This solution agrees quite well with the analytic solution $\epsilon = 0.0980$ (ER Appendix H).

3.2 Excited States of Finite Quantum Well

Bound states will exist so long as $E < V_0 \equiv \epsilon < 1$ so we can search for additional energy eigenstates by guessing larger ϵ. The first excited state will have a node at the origin ($\psi(0) = 0$). The linearity property of the Schrodinger equation (2) as well as our focus on the unnormalized eigenfunction allows us to choose any arbitrary value for $\frac{d\psi}{du}$ (we will use $\frac{d\psi}{du} = 1$).

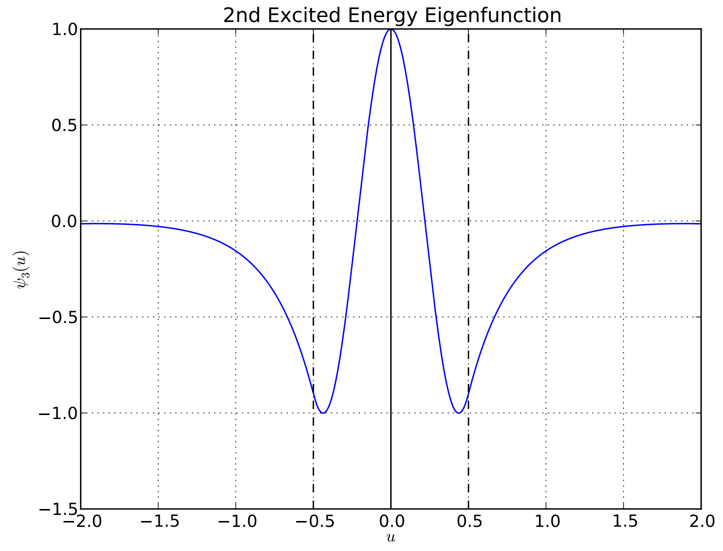
Running the simulation using these initial conditions (multiple trials omitted) an energy eigenvalue to V_0 proportionality constant of $\epsilon = 0.382605$ for ψ_2:

```
plot_finite_well(shooting_solver_1d_finite(0, 1, 2, .382605, 0.0001), \
                            n=2, title="1st Excited Energy Eigenfunction")
```



For ψ_3, $\epsilon = 0.80767$:

```
plot_finite_well(shooting_solver_1d_finite(-1, 0, 2, .80767, 0.0001), \
                            n=3, title="2nd Excited Energy Eigenfunction")
```



3.2.1 Other Numerical Methods

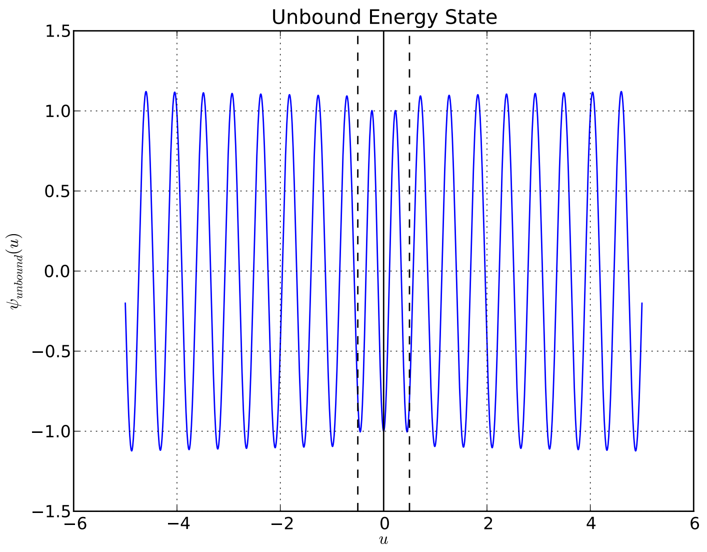
The shooting method provides a convenient way to numerically solve the finite-well problem by converting the boundary value problem to an initial value problem. However, this is not a traditional boundary value problem. Rather than solving for the values of $\psi(u)$ given the boundary constraints, we are solving for the ϵ value in the differential equation which give rise to solutions satisfying the boundary conditions. Additionally, we are dealing with asymptotic boundary conditions.

Nonetheless, we can reformulate techniques such as finite difference methods to accompany the additional variable ϵ. The asymptotic boundary condition can be mitigated by choosing the boundary to be at a large value of u. By treating ϵ as an unknown, the multiplication of ϵ with $\psi(u)$ in equation (2) will result in a system of non-linear equations. This system of non-linear equations can then be solved using Newton's method, yielding both the eigenfunction $\psi(u)$ as well as ϵ.

3.3 "Scattering" States of Finite Quantum Well

States where ($E > V_0 \rightarrow \epsilon > 1$) are where the quantum well is no longer able to bind the particle. For these unbound energy states, the allowable energy levels are no longer discrete. One example of an acceptable unbound solution to equation (2) is:

```
plot_finite_well(shooting_solver_1d_finite(-1, 0, 5, 3, 0.0001), \
                 n="unbound", title="Unbound Energy State")
```

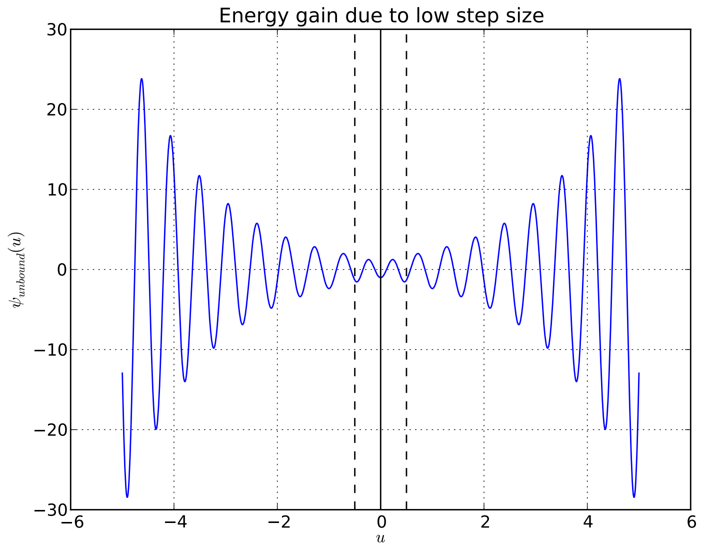


Any $\epsilon > 1$ is a valid energy which will tend towards a sinusoidal steady state in the limit $u \rightarrow \infty$. As energy is increased, so does the frequency of ψ. In the limit $E \rightarrow \infty$, the probability density approaches the clasically expected uniform probability density.

3.3.1 Step size error

This result, however, is quite sensitive to the step size used for numerical integration. Forward Euler is a first order solver which can be numerically unstable and introducing additional energy. For example, the result no longer results in sinusoidal limit behavior when ΔU is changed from 0.0001 to 0.01. Rather, the Forward Euler approximation adds enough energy to result in infinite blowup:

```
plot_finite_well(shooting_solver_1d_finite(-1, 0, 5, 3, 0.01), \
                      n="unbound", title="Energy gain due to low step size")
```



To address the numerical stability issues of Forward-Euler as well as increase rate of convergance of error to 0 as $\Delta U \to \infty$, implicit and symplectic solvers of higher order can be used to carry out numerical integration.

4 Infinite Quantum Well

The simulation code can be easily adjusted for different potentials. For example, the infinite quantum well can be simulated by modifying the definition of the potential:

```
# 999 to approximate infinite potential
V_potential = lambda u: 999 if (abs(u) > 1/2) else 0
```

Next, by interpreting $V_0 = 1$ eV and using the boundary condition $\psi(|u| = \frac{1}{2}) = 0$, the energy eigenstates can be found.

Since we have an analytical solution for an infinite square well (ER Eq. 6-79, Eq. 6-80):

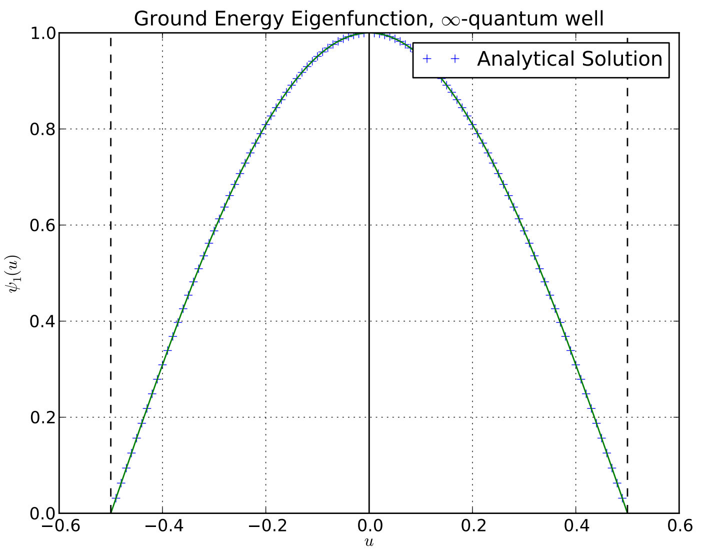
$$\psi_n(x) = \begin{cases} B_n \cos k_n x & \text{where } k_n = \frac{n\pi}{a} n = 1, 3, 5, \ldots \\ A_n \sin k_n x & \text{where } k_n = \frac{n\pi}{a} n = 2, 4, 6, \ldots \end{cases} \tag{6}$$

We can visualize how well our simulation approximates analytical results by modifying our plotting function to also plot the analytical eigenstate. We set the constant factors A_n to 1 maintain consistency with our initial condition $\psi(0) = 1$.

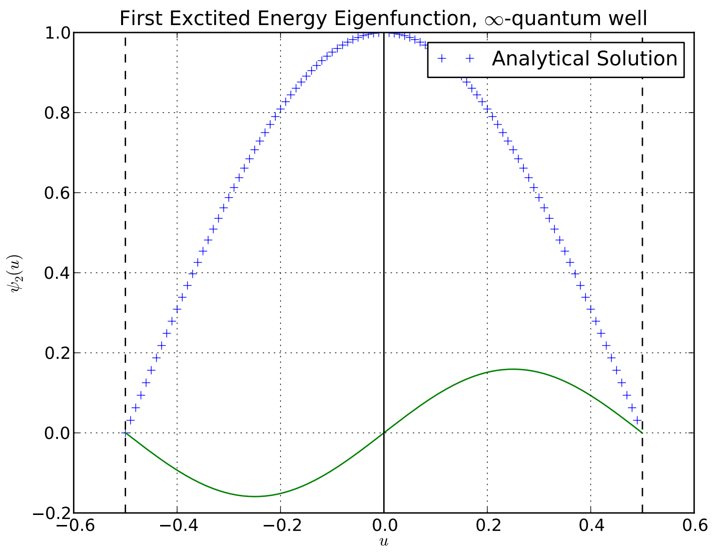
```
from numpy import arange
def plot_inf_sq_eigenstate(n, uf, data, title, chained=False):
    if not chained: plt.clf()
    u = arange(-float(uf),float(uf),2*float(uf)/100)
    if n % 2 != 0: #
        psi = [cos(i * pi) for i in u]
    else:
        psi = [cos(i * pi) for i in u]
    plt.plot(u, psi, label="Analytical Solution", marker='+', linestyle='None')
    plt.save = plt.savefig
    return plot_finite_well(data, n, title, True)
```

This results in (multiple trials omitted):

```
plot_inf_sq_eigenstate(1, 0.5, shooting_solver_1d_finite(1, 0, 0.5, .1542, 0.0001), \
                       title="Ground Energy Eigenfunction, $\infty$-quantum well")
```



```
plot_inf_sq_eigenstate(2, 0.5, shooting_solver_1d_finite(0, 1, 0.5, 0.6179, 0.0001), \
                       title="First Exctited Energy Eigenfunction, $\infty$-quantum well")
```



We see that our approximations of the eigenfunction follows analytical results very closely and the energy eigenstates for the ∞-quantum well has energy eigenvalues of 0.1542eV and 0.6179 eV. When compared to the analytical solutions:

$$m = \frac{\beta\hbar^2}{2a^2V_0} \text{ (From (3))}$$

$$E_n = \frac{\pi^2\hbar^2n^2}{2ma^2} = \frac{\pi^2n^2V_0}{\beta}$$

$$E_1 = \frac{\pi^2}{64} \approx 0.1542 \tag{7}$$

$$E_2 = \frac{4\pi^2}{64} \approx 0.6168 \tag{8}$$

we see that our simulation yields an answer accurate to the hundredth of an eV.

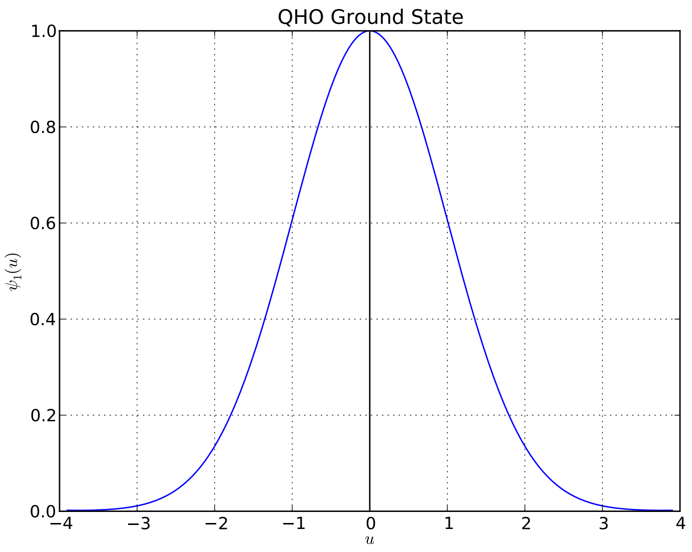
5 Quantum Harmonic Oscillator

The numerical methods developed so far can be extended to other non-trivial potentials.

```
hbar = 6.58211928*10**-16 # eV s
C = 1
m = 1

omega = sqrt(C / m)
x0 = sqrt(hbar / (m * omega))

V_potential = lambda u: C * (u * x0)**2 / 2

def shooting_solver_qho(psi0, dpsi0, uf, epsilon, delta_u):
    num_steps = (uf - 0) / delta_u
    data = [(0, psi0, dpsi0)];
    for i in range(num_steps):
        u_old = data[i][0]
        psi_old = data[i][1]
        dpsi_old = data[i][2]
        u = u_old + delta_u
        dpsi = dpsi_old - delta_u * (epsilon - u**2) * psi_old
        psi = psi_old + delta_u * dpsi_old
        data.append((u, psi, dpsi))
    return map(lambda x: (x[0], x[1]), data)

def plot_data_qho(data, n, title):
    # mirror data based on parity, assumes eigenfunction
    # has a definite parity
    (u, psi0) = data[0]
    if psi0 == 0: # node at 0, odd
        data = map(lambda x: (-x[0], -x[1]), data)[::-1] + data
    else: # definite parity => anti-node at 0, even
        data = map(lambda x: (-x[0], x[1]), data)[::-1] + data

    u, psi = [[x[i] for x in data] for i in (0,1)]
    fig = plt.figure()
    plt.plot(u, psi)
    plt.title(title)
    plt.xlabel('$u$')
    plt.ylabel("$\\psi_{%s}(u)$" % n)
    plt.axvline(x=0, color='black')
    plt.grid(True)

    # plt.savefig
    # uncomment if plotting from sagetex
    plt.save = plt.savefig
    return plt
```

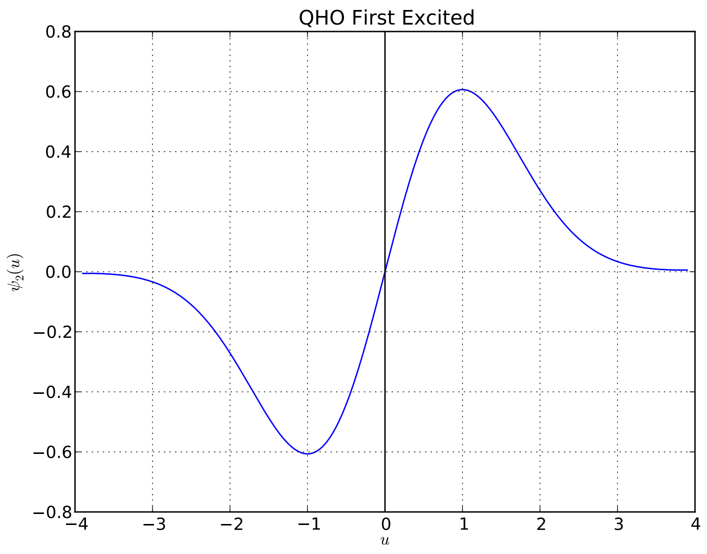
Using the definite parity of the eigenfunctions, we only need to evaluate the behavior for $u > 0$ and mirror across the y-axis. We can find allowed energy values by requiring $\psi(u) = 0$ as $u \to \infty$ (because

```
plot_data_qho(shooting_solver_qho(1, 0, 3.8, 1.00005, 0.0001), \
              n=1, title="QHO Ground State")
```



The first excited state can be characterized by forcing a node at the origin and finding the next eigenvalue which satisfies the boundary condition.

```
plot_data_qho(shooting_solver_qho(0, 1, 3.9, 3, 0.0001), \
              n=2, title="QHO First Excited State")
```


QHO First Excited
$\psi_2(u)$
u
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6
−0.8
−4
−3
−2
−1
0
1
2
3
4