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:

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

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. 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

Two global variables we will use are β (set to 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` 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:

$$\phi(u_{i+1}) = \phi(u_i) - \Delta u \cdot \beta \left[\epsilon - \widetilde{V}(u_i)\right] \psi(u_i) \tag{3}$$

$$\psi(u_{i+1}) = \psi(u_i) + \Delta u \cdot \phi(u_i) \tag{4}$$

`shooting_solver` 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_finite_well(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.

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

```
import matplotlib.pyplot as plt # use matplotlib
def plot_data(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.axvline(x=0.5, linestyle='dashed', color='black')
    plt.axvline(x=-0.5, linestyle='dashed', color='black')
    plt.grid(True)

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

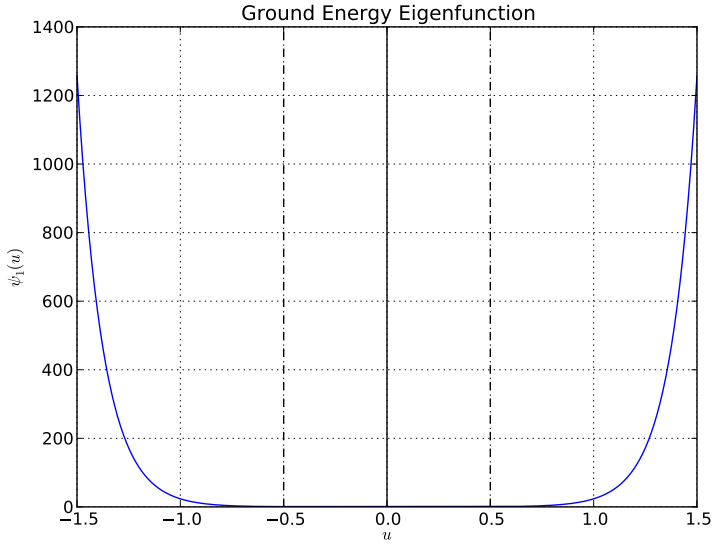
3 Simulation and Results

3.1 Ground State of Finite Quantum Well

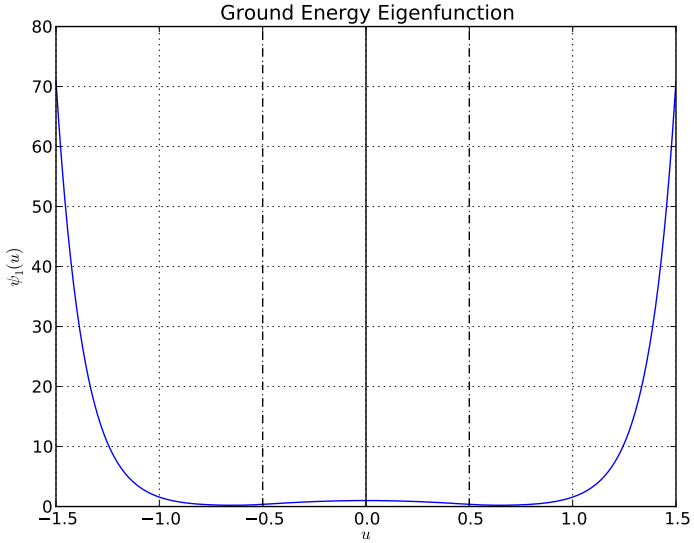
The simulation uses initial conditions of $\psi = 1.0, \frac{d\psi(u)}{du} = 0$ at $x_0 = 0$. This is a safe assumption to make because functions of definite odd parity have anti-node at $x = 0$ and will necessarily have $\frac{d\psi(u)}{du}(0) = 0$. By choosing $\psi(0) = 1.0$, we specify that the $\psi(u)$ will have amplitude 1.0.

Using these initial conditions and adjusting estimates to ϵ, we find:

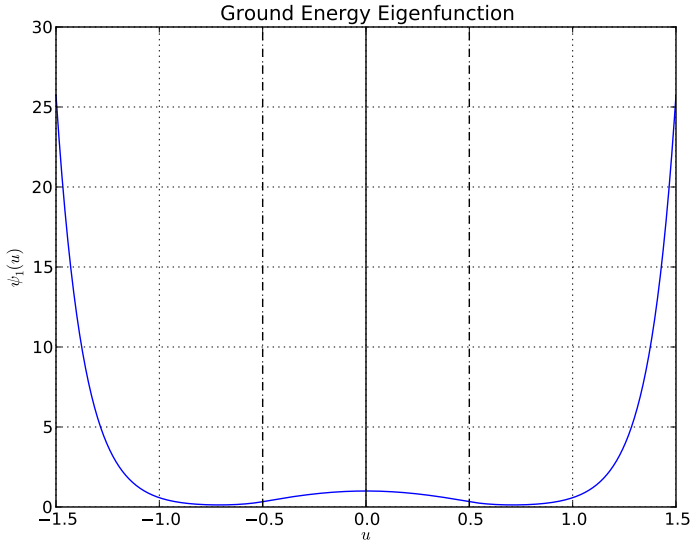
```
plot_data(shooting_solver_finite_well(1, 0, 1.5, .01, 0.0001), \
          n=1, title="Ground Energy Eigenfunction")
```



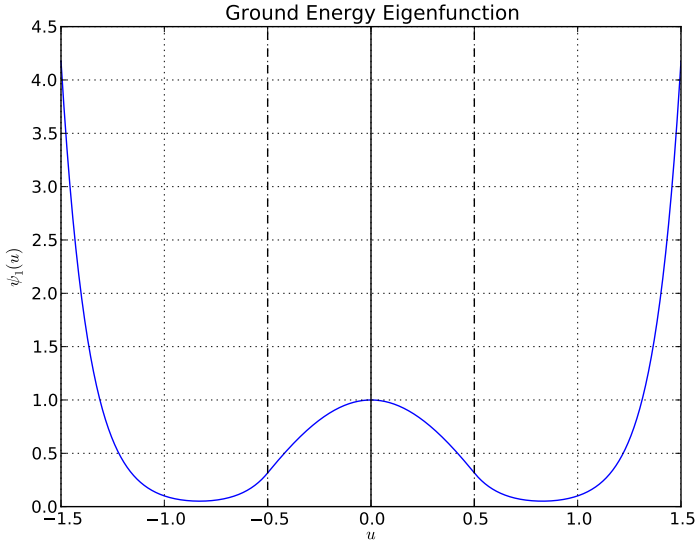

```
plot_data(shooting_solver_finite_well(1, 0, 1.5, .09, 0.0001), \
          n=1, title="Ground Energy Eigenfunction")
```



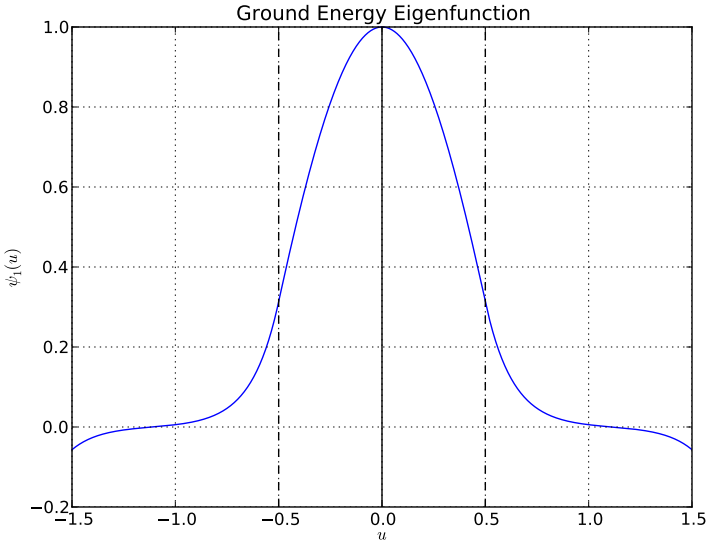

```
plot_data(shooting_solver_finite_well(1, 0, 1.5, .095, 0.0001), \
          n=1, title="Ground Energy Eigenfunction")
```



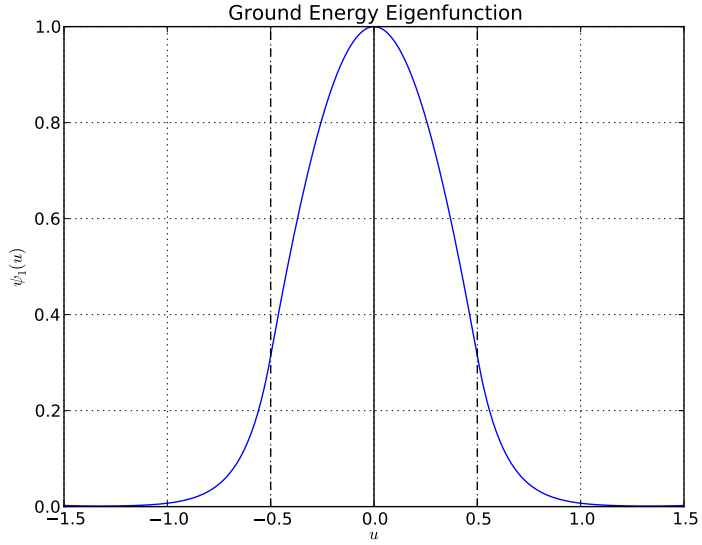

```
plot_data(shooting_solver_finite_well(1, 0, 1.5, .0975, 0.0001), \
          n=1, title="Ground Energy Eigenfunction")
```




```
plot_data(shooting_solver_finite_well(1, 0, 1.5, .098, 0.0001), \
          n=1, title="Ground Energy Eigenfunction")
```




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

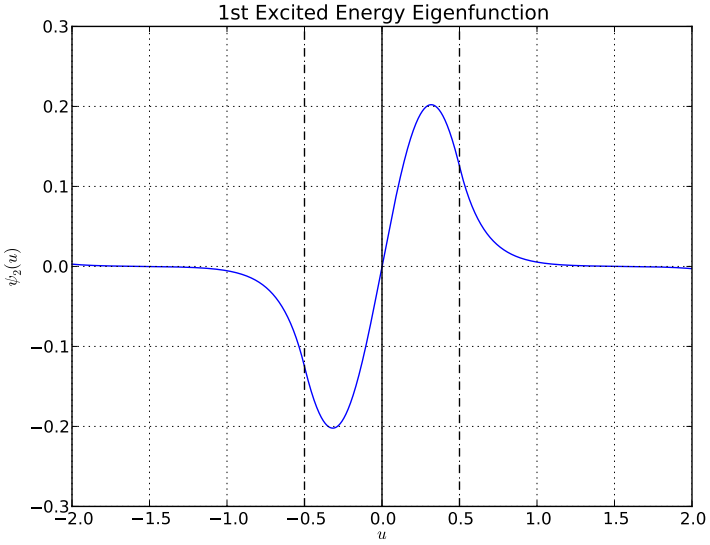


This shows that a valid solution is found at $\epsilon = 0.097993$, implying that an energy eigenvalue exists at $0.097993V_0$. We see that there is a single node in this wavefunction, indicating that this solution is the groud state.

3.2 Excited states of finite quantum well

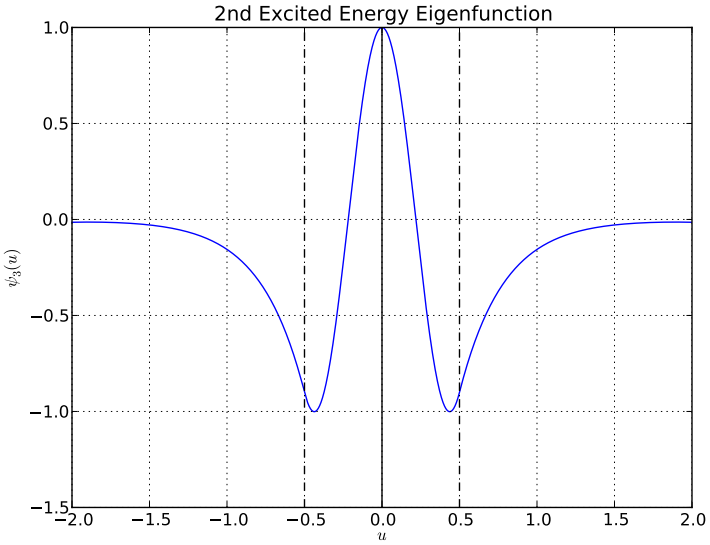
Bound states will exist as long as $\epsilon < 1$ so we can continue increasing ϵ to look for additional bound states. The first excited state will have a node at the origin ($\psi(0) = 0$). Due to the linearity of eigenfunctions, $\frac{d\psi}{du}$ can take on any value (we assume $\frac{d\psi}{du} = 1$. Running the simulation using these initial conditions, the shooting method yields an energy eigenvalue of $\epsilon = 0.382605$ for ψ_2:

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



For ψ_3, $\epsilon = 0.80767$:

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



The shooting method provides a convenient way to numerically solve the finite-well problem by converting a boundary condition problem to an initial value problem.

3.3 Unbound states

Examining solutions which are not bound ($E > V_0 \rightarrow \epsilon > 1$), we find:

```
plot_data(shooting_solver_finite_well(-1, 0, 5, 3, 0.0001), \
          n=3, title="Unbound Energy")
```

In actuality, any value for $\epsilon > 1$ is a valid solution which results in a sinusoidal steady state in the limit $u \to \infty$. As energy is increased, so does the frequency of ψ. In the limit $E \to \infty$, 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 during numerical integration. Forward Euler is a first order solver which violates the conservation of energy by introducing additional energy. As a consequence, the result seen above no longer results in a sinusoidal steady state when δU is changed from 0.0001 to 0.01. Rather, the Forward Euler approximation adds enough energy to result in infinite blowup:

```
plot_data(shooting_solver_finite_well(-1, 0, 5, 3, 0.01), \
          n=3, title="Unbound Energy")
```

4 Infinite Quantum Well

The modularity of the code allows for easy re-use. One can simulate the infinite quantum well by simply modifying the definition of the potential:

```
# 999 to approximate infinite potential
V_potential = lambda u: 999 * int(abs(u) > 1/2)
```

Next, by interpreting $V_0 = 1$ eV and aiming the shooting method to satisfy $\psi(x = a/2) = 0$, the energy eigenstates can be found. This results in:

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

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

These results imply that the particle in the ∞-quantum well has energy eigenvalues of 0.1542eV and 0.6179 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")
```