Numerical solution of the time-independent Schrödinger Equation

Feynman Liang

January 27, 2013

1 Introduction

Solutions to the Schrödinger equation can oftentimes be approximated using numerical methods. This is especially convenient for dealing with potentials where analytical solutions may not exist or be very difficult to find. 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 Schrödinger equation in a dimensionless form, we obtain:

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

where:

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

Since we expect energies below V_0 to be bound, there are boundary conditions $\psi(u) = 0$ and $\frac{d\psi(u)}{du} = 0$ for $u \to \pm\infty$. 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 an initial value for ϵ rather than $\psi(0)$ and dealing with asymptotic rather than explicit boundary conditions). Using the shooting method and visual inspection, we will guess and successively adjust the energy term ϵ until $\psi(u)$ satisfies the desired asymptotic behavior within the range examined. The ϵ values resulting from this method are allowed energy eigenvalues for the $\widetilde{V}(u)$ potential in (1).

2 Theory and Methods

The code was run using Sage 5.5 and Python 3.3.0. The SageTEX package is used to include evaluation results in-line.

Two global variables we will use are β (set to 64, see assignment specifications 6.1) and the potential function $\widetilde{V}(u)$. These are specified at the top of the program:

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

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

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

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

The function `shooting_solver_1d_finite` takes initial values `psi0` and `dpsi0` (ψ and $\frac{d\psi}{du}$ at $x_0 = 0 = u$), a guess for `epsilon`, and a step size `delta_u`. A list of 2-tuples $(u, \psi(u))$ is returned for $u \in [0, \texttt{uf}]$.

```
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 about $u = 0$ ($\widetilde{V}(u) = \widetilde{V}(-u)$), the energy eigenfunctions have a definite parity. Thus, they exhibit symmetry about the origin and a simulation result between $(0, \texttt{uf})$ will also reveal 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` used for subscripting the y-axis label, a `title` string, and uses the matplotlib plotting library to generates a plot of `data` with dashed lines illustrating the boundaries of the finite well potential (1). This function takes care of mirroring the data from the simulation across $u = 0$ appropriately. The `chained` argument is an implementation detail with no effect on the solution (it prevents the plotting area from being cleared, allowing us to chain the infinite square well plotting function with `plot_finite_well` and reduce redundant code):

Note*: This plotting routine assumes the eigenfunction is odd if* $\psi(0) = 0$ *and even otherwise.*

```
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, label="Euler Approximation")
    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 # hack for proper SageTeX behavior
    return plt
```

3 Results

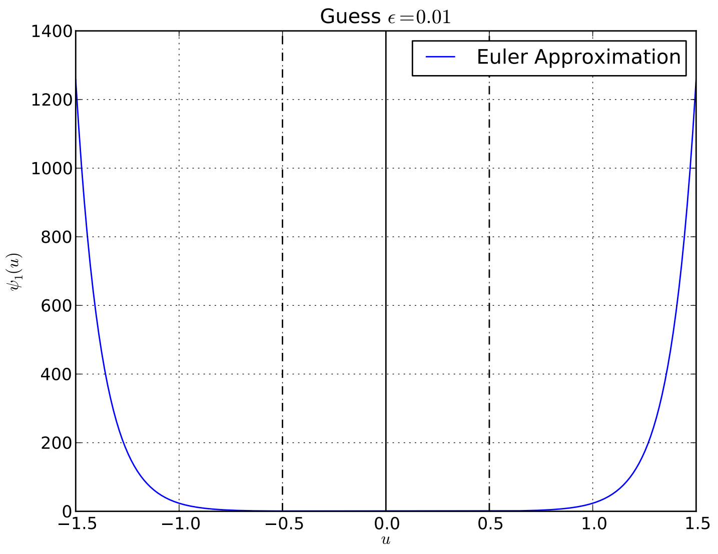
3.1 Ground State of Finite Quantum Well

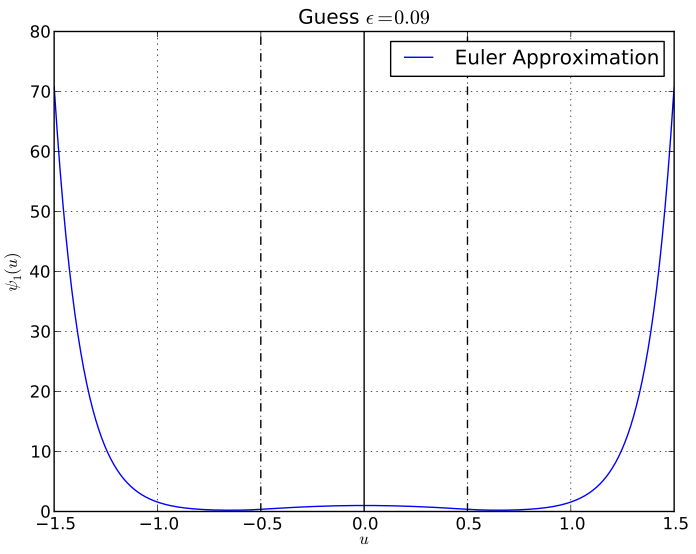
Because the symmetric potential implies energy eigenfunctions of definite parity and the ground state eigenfunction is even, $\frac{d\psi(u)}{du}$ must necessarily be 0 at $x_0 = 0$.

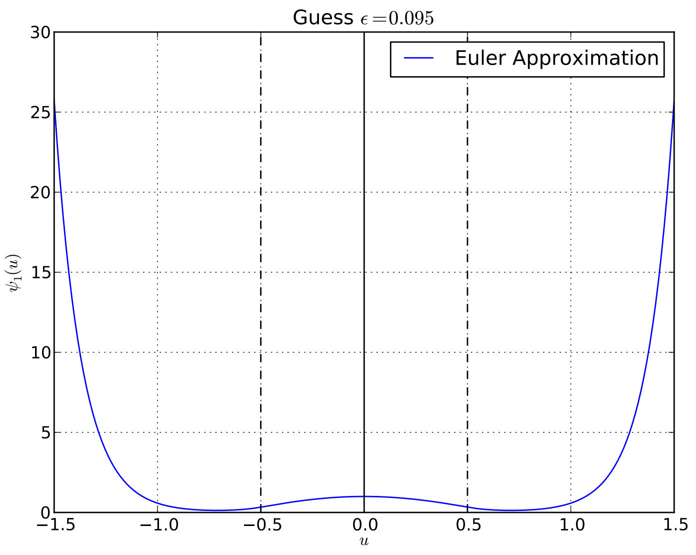
Rather than worry about the normalization conditions for $\psi(u)$ $\left(\int_{-\infty}^{\infty} \psi^{\dagger}\psi du = 1\right)$, we recognize that the normalized eigenfunction is simply the unnormalized eigenfunction multiplied by a constant normalization factor. We can compute the normalization 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 results for ϵ because substitution of $\psi(u)$ times a normalization constant into (2) shows that the normalization constants cancel.

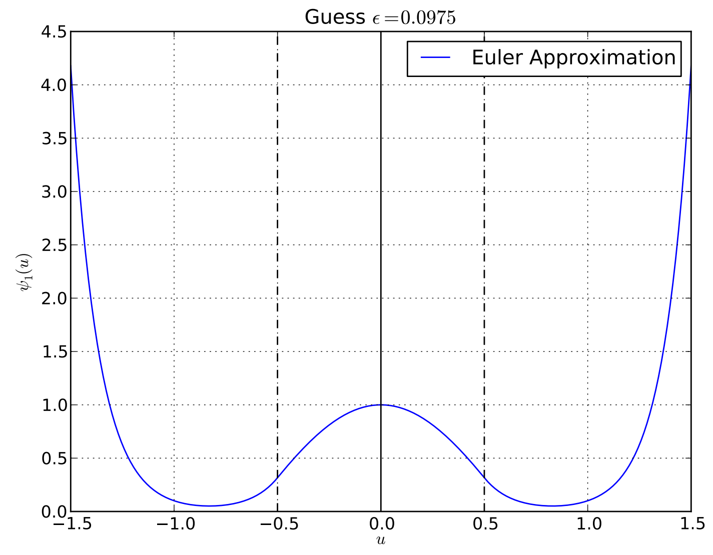
Our simulation will only find eigenfunctions which are real functions. This is not a problem because we can choose the phase of solutions to the TISE (2) to make $\psi(u) \in \mathbb{R}$ (see assignment specifications 3).

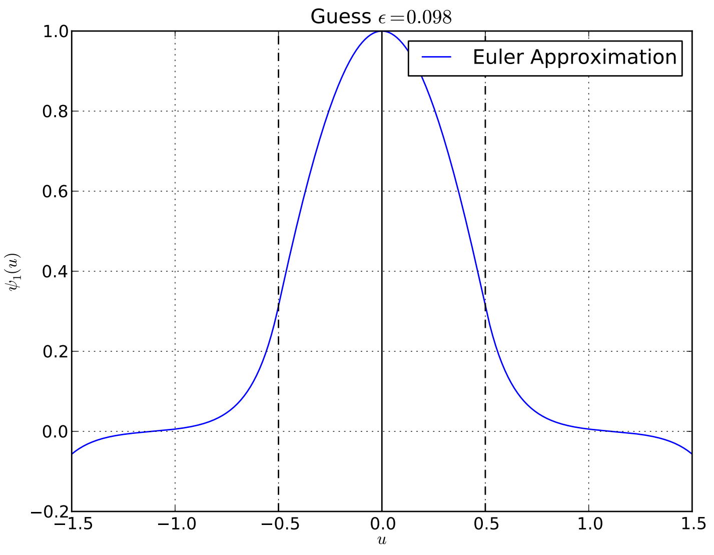
Using these initial conditions, the "shooting" method to determine ϵ yields:

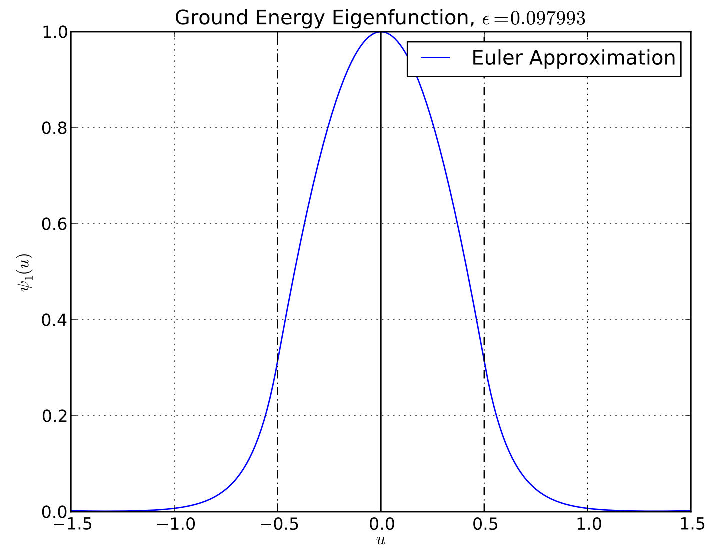



Guess $\epsilon = 0.09$
Euler Approximation
$\psi_1(u)$
u


Guess $\epsilon = 0.095$
Euler Approximation
$\psi_1(u)$
u


Guess $\epsilon = 0.0975$
Euler Approximation
$\psi_1(u)$
u
4.5
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5


Guess $\epsilon = 0.098$
Euler Approximation
$\psi_1(u)$
u
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5



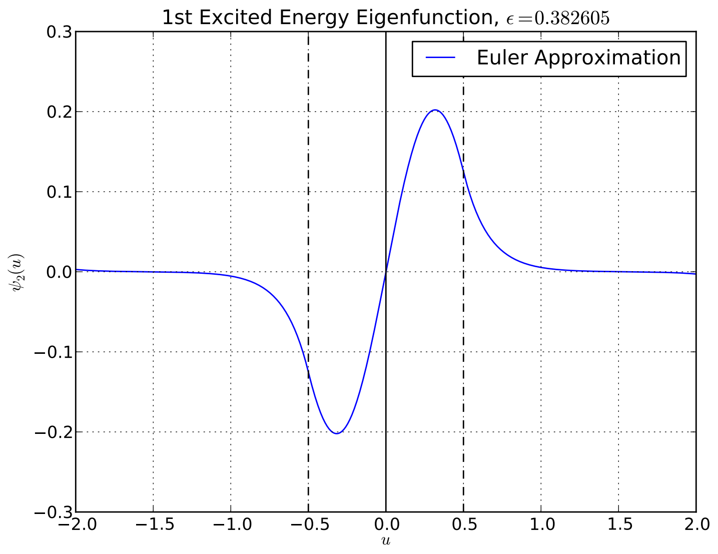
A valid solution is found at $\epsilon = 0.097993$, implying that the ground state energy eigenfunction has an eigenvalue of $0.097993V_0$, where V_0 is the height of the finite well potential. We see that there is a single node in this wavefunction, confirming that this eigenstate is the ground state.

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

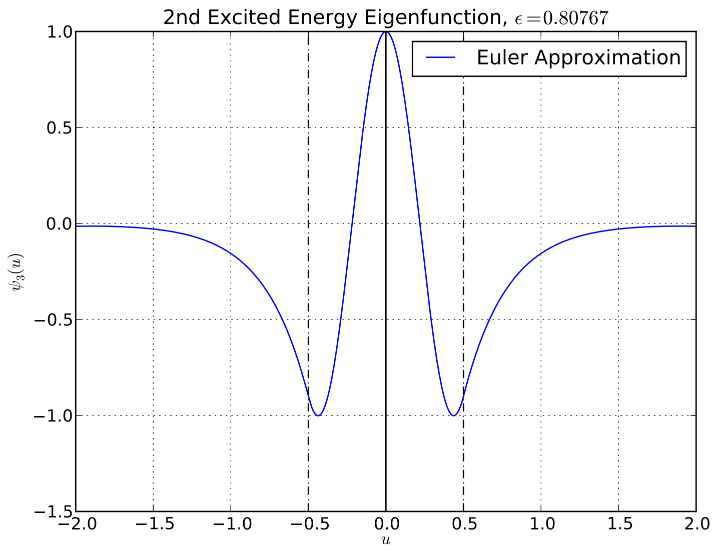
3.2 Excited States of Finite Quantum Well

Bound states will exist so long as $E < V_0 \implies \epsilon < 1$ so we can search for additional energy eigenstates by guessing $\epsilon \in [0.0980, 1]$. The first excited state will have a node at the origin ($\psi(0) = 0$). The linearity property of the Schrödinger equation (2) 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) , we find that the first excited energy eigenfunction ψ_2 has an energy eigenvalue $\epsilon = 0.382605 \implies E = 0.382605V_0$:



For ψ_3, $\epsilon = 0.80767 \implies E = 0.80767V_0$:



3.2.1 Other Numerical Methods

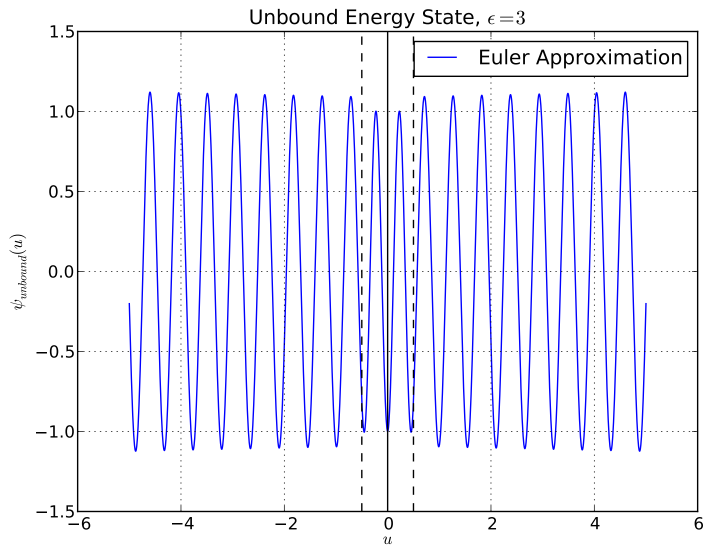
The shooting method provides a convenient way to numerically solve the finite-well problem by converting the boundary value problem to an initial value problem with asymptotic boundary constraints.

Another possible numerical method for solving the finite-well problem could utilize finite difference methods. To accompany asymptotic boundary conditions, we choose the boundary conditions at a large value for u. 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 or any other numerical root-finding method, yielding both ϵ as well as the mappings for the eigenfunction $\psi(u)$ discretized to the same resolution as the step size of the finite difference schema.

In addition, we can also improve on the method for refining our guess for ϵ. Rather than manually update our guess, a binary search procedure could be used to refine subsequent guesses for ϵ until a certain error tolerance is reached.

3.3 "Scattering" States of Finite Quantum Well

States where ($E > V_0 \implies \epsilon > 1$) are where the finite quantum well is no longer able to bind the particle. For these unbound "scattering" states, the allowed energy levels are continuous. One example of an acceptable unbound solution to equation (2) is when $\epsilon = 3$:

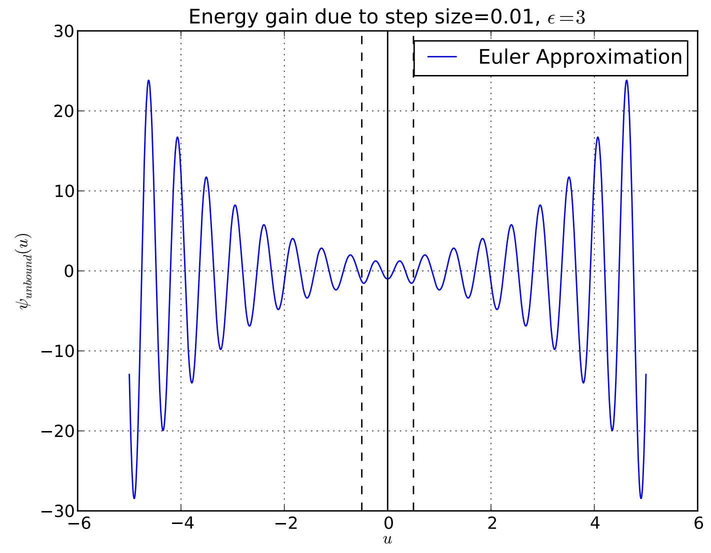


In ”scattering” states, the particle takes on a wavefunction similar to that of a 1D free particle. This makes sense as the particle is not bound by the potential so we expect behavior similar to that of an unbound particle.

In fact, any $\epsilon > 1$ is a valid ”scattering” state where the particle exhibits a behavior similar to that of a free particle. As ϵ is increased, so does the frequency of ψ. In the limit $E \to \infty$, the probability density approaches the clasically expected uniform probability density for a free particle.

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 introduce additional energy. For example, the results from section 3.3 no longer result in sinusoidal asymptotic behavior when ΔU is changed from 0.0001 to 0.01. Rather, the Forward Euler approximation adds enough energy to result in infinite blowup:



To address the numerical stability issues of Forward-Euler as well as increase rate of convergance of error to 0 as $\Delta U \to 0$, implicit and symplectic solvers of higher order (eg. Runge-Katta) 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):

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

We can visualize how well our simulation approximates analytical results by modifying our plotting function to also plot the analytical solutions. The constant factors A_n and B_n are chosen to be consistent with initial conditions:

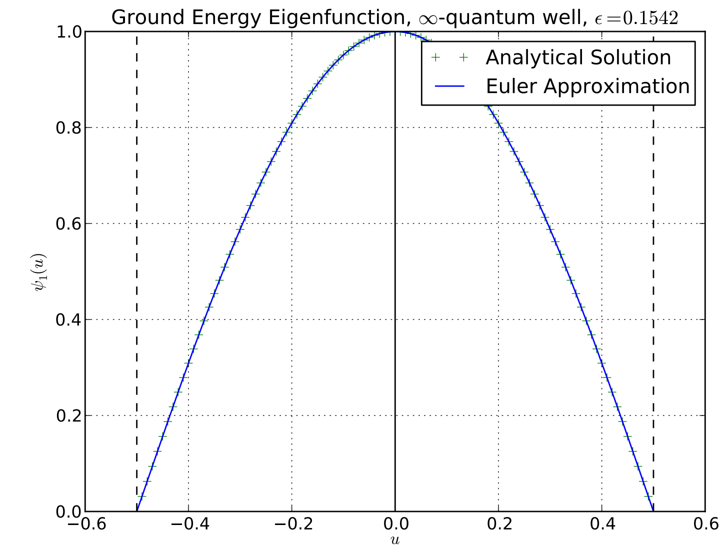
$$\begin{aligned} \psi(0) &= B_n && (6) \\ \left.\frac{d\psi}{du}\right|_{u=0} &= \frac{d}{dx} A_n \sin k_n a \cdot 0 \\ &= A_n k_n \cos k_n a \cdot 0 \\ &= A_n k_n && (7) \end{aligned}$$

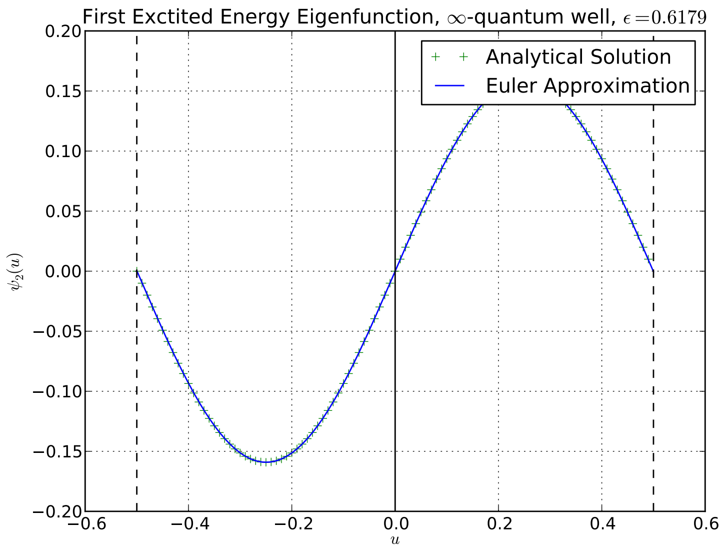
Solving these using the initial conditions $\psi(0) = 1$ and $\frac{d\psi}{du}\big|_{u=0} = 1$ yields $A_n = \frac{1}{k_n}$ and $B_n = 1$,

We use these results to define a `plot_inf_well` method which takes a wave number `n`, a `data` list of simulation results, and a `title` string. This function will first plot the analytical solution for $\psi_n(u)$ in an ∞-square well potential and subsequently make a chained call to `plot_finite_well` to plot the simulation results with dashed well boundaries.

```
from numpy import arange
def plot_inf_well(n, uf, data, title):
    plt.clf() # clear plot
    u = arange(-float(uf),float(uf),2*float(uf)/100) # generate timespan array
    # determine the proper analytic solution to overlay
    if n % 2 != 0: # eigenfunction is even
        psi = [1 * cos(n*i*pi) for i in u]
    else: # eigenfunction is odd
        psi = [(1 / (n * pi)) * sin(n*i*pi) for i in u]
    plt.plot(u, psi, label="Analytical Solution", marker='+', linestyle='None', color='Green')
    plt.save = plt.savefig # hack for proper SageTeX behavior
    return plot_finite_well(data, n, title, chained=True) # chain to plot_finite_well
```

This results in (multiple trials omitted):





We see that our simulated eigenfunctions follow the analytical solutions very closely. The results indicate that the ground and first excited energy eigenstates for this ∞-quantum well have energy eigenvalues of 0.1542 eV and 0.6179 eV. Compared to the energy eigenvalues obtained analytically (ER Eq. 6-81):

$$m = \frac{\beta \hbar^2}{2a^2 V_0} \text{ (from (1))}$$

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

$$E_1 = \frac{\pi^2}{64} \approx 0.1542 \tag{8}$$

$$E_2 = \frac{4\pi^2}{64} \approx 0.6168 \tag{9}$$

we see that our simulation yields an answer accurate to one thousandth of an eV.

5 Quantum Harmonic Oscillator

The numerical methods developed so far can be extended to other non-trivial potentials. For a quantum harmonic oscillator, the potential is defined by $V(x) = \frac{1}{2}Cx^2$ and has a natural frequency of $\omega = \sqrt{\frac{C}{m}}$. We can reformulate the time-independent Schrödinger equation for the quantum harmonic oscillator into a dimensionless form (P24 - PS1 Problem 3):

$$\frac{d^2\psi}{du^2} + (\epsilon - u^2)\psi = 0 \tag{10}$$

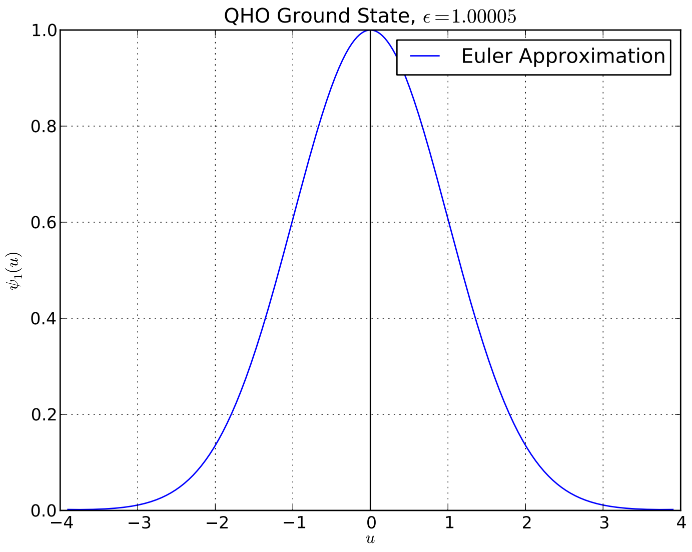
where:

$$\epsilon = \frac{2E}{\hbar\omega}, \ \ u = \frac{x}{x_0}, \ \ x_0 = \sqrt{\frac{\hbar}{m\omega}} \tag{11}$$

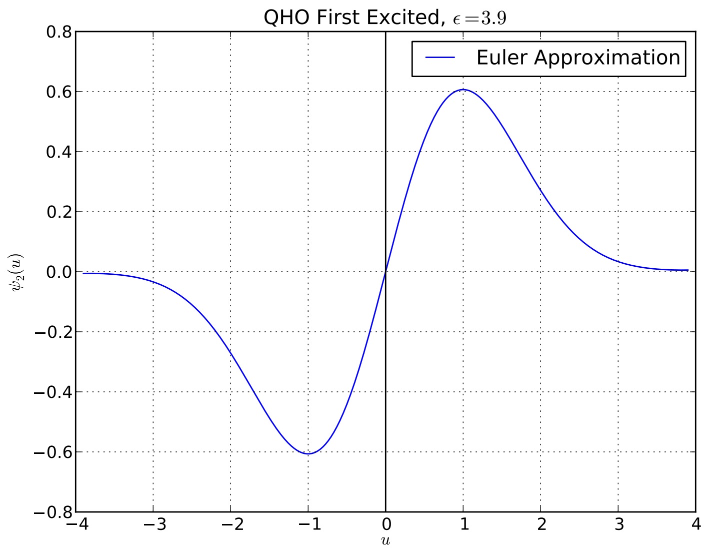
While section 4 illustrated that the potential can be modified by simply changing one line, because we have reparameterized the TISE to a dimensionless quantum harmonic oscillator case, we will need to define a new `shooting_solver_qho` method which implements the reparameterized TISE (10). Additionally, we will define a modified `plot_qho` which does not plot the well boundaries at $u = \pm 0.5$.

```
hbar = 6.58211928*10**-16 # eV s
V_potential = lambda u: C * (u * x0)**2 / 2
C = 1
m = 1
omega = sqrt(C / m)
x0 = sqrt(hbar / (m * omega))

def shooting_solver_qho(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 using (10)
        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) # return list of (u, psi) tuples

def plot_qho(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, label="Euler Approximation")
    plt.title(title)
    plt.xlabel('$u$')
    plt.ylabel("$\\psi_{%s}(u)$" % n)
    plt.axvline(x=0, color='black')
    plt.grid(True)
    plt.legend()
    plt.save = plt.savefig
    return plt
```

Again, we use the symmetry of the potential and only simulate for $u \geq 0$, appropriately mirroring the results in `plot_qho`. Also, the definite parity of eigenfunctions due to the symmetric potential and the linearity of eigenstates again allow us to arbitrarily choose initial conditions of $\psi(0) = 1$ for even eigenfunctions and $\frac{d\psi}{du}\big|_{u=0} = 1$ for odd eigenfunctions. We can now repeatedly guess and refine an estimate for ϵ which causes ψ to satisfy the asymptotic bound $\psi(u) = 0$ as $u \to \infty$ (multiple trials omitted):



The first excited state can be found by requiring a node at $u = 0$, which we do by setting the initial condition $\psi(0) = 0$. Since the first excited state for a QHO is odd, we use the initial condition $\frac{d\psi}{du}\big|_{u=0} = 1$:



Our results estimate the dimensionless energy parameter $\epsilon = \frac{2E}{\hbar\omega}$ to be $\epsilon_1 = 1.00005$ in the ground state and $\epsilon_2 = 3.9$ in the first excited state. This agrees with analytical results (P24 - PS3 - Problem 3c), which require $\epsilon_n = 1, 3, 5, \ldots$

6 Conclusions

This report shows that the shooting method coupled with numerical integration using Forward Euler can produce satisfactory approximations of eigenfunctions and eigenvalues for the TISE. The code accompanying this report not only solve the 1D finite well with results very close to analytical solutions, but is also modular and easily adaptable to different potentials. This was illustrated by the relatively small amount of effort required to reconfigure our program to solve the infinite well potential and quantum harmonic oscillator potential. In both of these latter examples, numerical techniques achieved results in close agreement with analytical solutions.

By decreasing the step size used for numerical integration, the error introduced by Forward Euler should converge towards zero, yielding a flexible method to numerically integrate open-form differential equations. Coupled with the plotting routines presented, this set of methods allows us to conveniently approximate energy eigenvalues and eigenstates for arbitrary 1D potentials and plot the results.

Nonetheless, our method is far from perfect. We saw in section 3.3.1 that Forward Euler violates the conservation of energy and can result in unstable asymptotic behavior. Decreasing the step size is one solution to this problem, but doing so also increases the amount of computation required to carry out the simulation. Other methods of increasing the rate of error convergence to 0 as $\Delta U \to 0$ include using a higher-order implicit/symplectic solver for the numerical integration process. To improve the precision of ϵ, the repetitive guess-refine feedback loop necessary for the shooting method can be itself made a callable bisection search over the ϵ parameter with some termination condition.