PHYS2210 Homework Problem 26 Shooting Method

Problem 2 - A) Edit the shooting method program below to find the lowest three state energies for a particle in a symmetric box of width a=2.0 and depth V0=-10. Report your numerical answers in the units of this program in the next cell down from here.

B) Describe how you could use this program to find the states of an asymmetric well where $V(x<0)=\infty$, $V(x>a/2)=0$, and $V(insidebox)=-V_0$). (What are the conditions you might place on the eigenfunctions?) Give your answer in the next cell down from here.

Note that the shooting method is discussed in Problem 4.15 in Townsend. Note that you can change the symmetry of the state you are seeking by commenting out one or the other of lines 9 or 10 in the function "def acquire_well_state(a,V0, x, dx, Eguess)".

A.
-9.1823 n =1

-6.911 n=2

-3.075 n=3

B. You could calculate the energy levels of one half of the well, guess the energy levels and plug those energy levels into a simulation of the other half of the well. The wavefunction would asymtopically approach infinity at both sides, and be continuous and smooth without, including at the junction point between the two well depths

The Time Independent Schrodinger equation is

$$E\Psi = \left(\frac{-\hbar^2}{2m}\nabla^2 + V(x)\right)\Psi$$

where $\Psi(x,t) = \psi(x)exp(iE/\hbar)$. In one dimension, we can rearrange the equation to read

$$\psi''(x) = -\frac{2m}{\hbar^2}\left[E - V(x)\right]\psi(x)$$

We'll set $m=\hbar=1$, so that we must solve

$$\psi''(x) = 2\left[V(x) - E\right]\psi(x)$$

This means that you must keep track of your energy units when converting in both

Finite Difference Methods

On a computer, we must work with discretized spatial grids. Since the potential is symmetric, the eigenstates will be either symmetric or anti-symmetric, so we only have to solve for "half" of the wavefunction, from $x = 0$ to some large $x = L$. We will solve for the wavefunction on a discrete grid with N points, defined by

$$x_i = i \times \delta x$$
$$\delta x = \frac{L}{N}$$

Furthermore, we need to use finite differencing to calculate derivatives numerically on a discrete grid. For example, the forward finite difference scheme for the first derivative is

$$\psi'(x_i) \approx \frac{\psi(x_{i+1}) - \psi(x_i)}{\delta x}$$

While the backward differencing scheme is

$$\psi'(x_i) \approx \frac{\psi(x_i) - \psi(x_{i-1})}{\delta x}$$

By using both the forward and backward differencing schemes, we can obtain what's called the second-order central difference approximation to the second derivative

$$\psi_i'' \approx (\psi_i')' = \left(\frac{\psi_{i+1} - \psi_i}{\delta x}\right)' = \frac{(\psi_{i+1} - \psi_i) - (\psi_i - \psi_{i-1})}{\delta x^2}$$

$$\psi_i'' \approx \frac{\psi_{i+1} - 2\psi_i + \psi_{i-1}}{\delta x^2}$$

where $\psi_i = \psi(x_i)$.

Shooting Method

Let's plug our our expression for the discrete second order derivative into the Schrodinger equation. After rearranging a bit, we get

$$\psi_{i+1} = 2\psi_i - \psi_{i-1} + 2\delta x^2 \left[V(x_i) - E\right]\psi_i$$

The solutions are either symmetric or anti-symmetric, so either ψ_0 or ψ_0' will be zero. Since

$$\psi_1 \approx \psi_0 + \delta x \psi_0'$$

We can start with either ψ_0 or ψ_0', and then use the previous two formulas to construct the rest of the wavefunction. This is called the Shooting Method.

Note: How do we know ψ_0 or ψ_0'? Turns out, it doesn't matter! We just guess a value, and the only effect the value has is on the normalization of the resulting wavefunction, which we can easily correct.

We know from Townsend Ch 4.1 that the ground state for a symmetric well is symmetric, with $\psi_0' = 0$, while the first excited state will be anti-symmetric, with $\psi_0 = 0$.

Obtaining the Energy

Wait, but what about the energy, E? We don't know that either, but it appears as a parameter in the Shooting Method. Furthermore, although we can solve the Schodinger equation for any E, only very particular energies will yield physical, normalizable states.

We'll find the allowed energies using an inefficient method that gives some insight. It's called "guess and check" (by me). If the energy is not that of an eigenstate, the wavefunction we find by finite difference will blow up for large x. We seek an energy that is reasonably close to the non-blow-up point. With this method, we'll never get exactly the right energy, but we can be be so (arbitrarily) close to the right energy that the wavefunction near the well will be extremely close to the right one. First we need our basic packages

In [25]:
```
%matplotlib inline
import matplotlib
import matplotlib.pyplot as plt
import numpy
import math
```

Small functions to get the double well potential and normalize wavefunctions.

1. *Potential* takes an array of the grid points and returns an array of the same length with the values of the potential at each point, $V(x_i) = V_i$.

- *norm* takes an array of values for the wavefunction $\psi(x_i) = \psi_i$ and returns its norm, $\int \psi^2 dx \approx \sum_i \psi_i^2 \delta x$.
- *normalize* will normalize the wavefunction so that $\sum_i \psi_i^2 \delta x = 1$.

In [26]:
```
def Potential(x, a, V0):
    V=numpy.zeros(len(x))
    V[(abs(x)<=a/2.)] = -V0
    return V
```

This is the meat of the program. Starting with the value of the wavefunction and its derivative at $x = 0$, we use the shooting method to calculate the rest of the wavefunction iteratively, changing the energy until we find an acceptable solution.

In [27]:
```
def forward_shooting(V, x, dx, f0, df0, Eguess):
    """
    V - array with the values of the potential at the points ir
    x - the grid points
    dx - the spacing of the grid
    f0 - value of the function at x=0
    df0 - value of the derivative of the function at x=0
    Eguess is the energy you previously guessed.
    """
    E = Eguess
    psi = numpy.zeros(len(x))
    psi[0]=f0


    for i, x_ in enumerate(x[0:-1]):
        if i==0:
            psi[i+1] = psi[i] + dx*df0
        else:
            psi[i+1] = -psi[i-1] + 2.*( 1 + dx**2 * (V[i] - E)

    if(abs(psi[i+1]) > 100):
        print ("big psi for", i+1)
    return (psi, E)
```

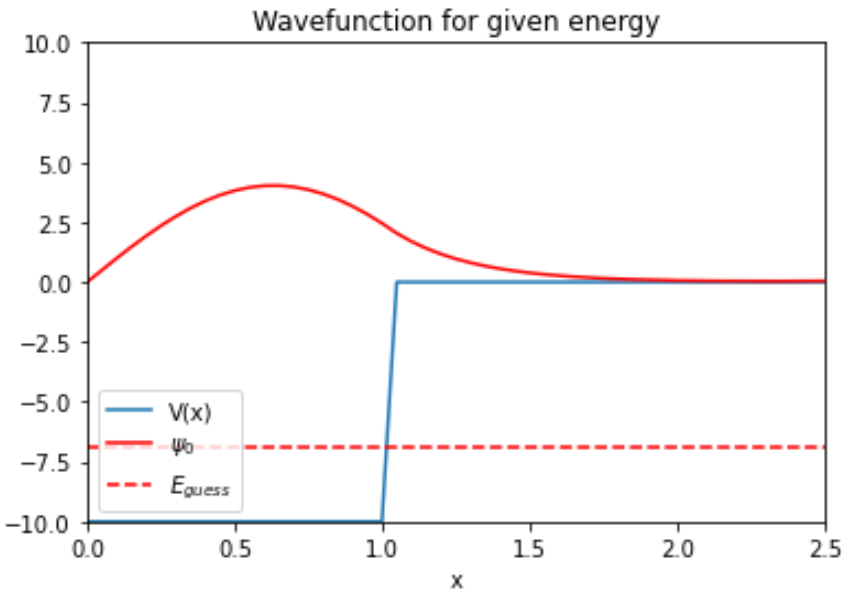
When we call the function below with the well parameters, we get the wavefunction back.

In [119]:
```
def acquire_well_state(a,V0, x, dx, Eguess):
    V = Potential(x, a, V0)

    #If you want a symmetric state the fourth and fifth paramet
    #If antisymmetric, change them to 0,1.
    #I chose V0 as the original amplitude (fourth parameter) ir
    #of V and psi the same scale
    # ************************************************************
#     (psi_0, Eguess) = forward_shooting(V, x, dx, V0, 0, Egues
    (psi_0, Eguess) = forward_shooting(V, x, dx, 0, V0, Eguess)
    # ************************************************************
    return (x, psi_0, Eguess)
```

In the following cell you input the parameters for your calculation. For a finite square well you need the width of the well, a,the depth of the well, V_0, and the step size, dx. You will also input a guess for the energy, Eguess, in the same units as V_0. The input variable L allows you to set the range of your calculation and plot.

In [147]:
```
# ***********************************************************
a=2.0
V0=10
dx=0.05
Eguess=-6.911 #HERE IS WHERE YOU INPUT YOUR GUESS FOR THE ENERG
# THE UNIT FOR ENERGY SHOULD BE THE SAME AS THE UNITS YOU USED
# ***********************************************************
L=5*a
L_plot=5
num= numpy.rint(L/dx+1) # number of points to calculate
num=num.astype('int32')
x=numpy.linspace(0,L,num,True)

(x, Psi_0, energy_0) = acquire_well_state(a, V0, x, dx, Eguess)

print ('Guess energy=', Eguess)

plt.subplots()
plt.plot(x, Potential(x, a, V0), label="V(x)")

plt.plot(x, Psi_0, 'r',label="$\psi_0$")
plt.plot([0,L_plot], [energy_0,energy_0],'r--',label="$E_{guess

plt.xlim([0,L_plot/2])
plt.ylim([-V0,V0])
plt.legend(loc=3)
plt.title('Wavefunction for given energy')
plt.xlabel('x')
plt.show()
```

```
big psi for 200
Guess energy= -6.911
```



References

1. Shooting Method for Double Well (https://nbviewer.jupyter.org/urls/www.numfys.net/media/notebooks/eigenenergies_double_well_potential.ipynb), Henning G. Hugdal, Magnus H-S Dahle and Peter Berg

Note that the wavefunction you will get here is not normalized. In order to normalize, you need to integrate the probability density over a reasonable range and find the factor that yields an integral of one.

I will also supply you with a more sophisticated program written by Joe Paki (RPI alum 2012(?)) and based on the reference above. Joe's program seeks the best energy iteratively, removing the slow human in the middle. We will use his program next to explore the behavior of the states of a double well.

In [ ]: