Introduction

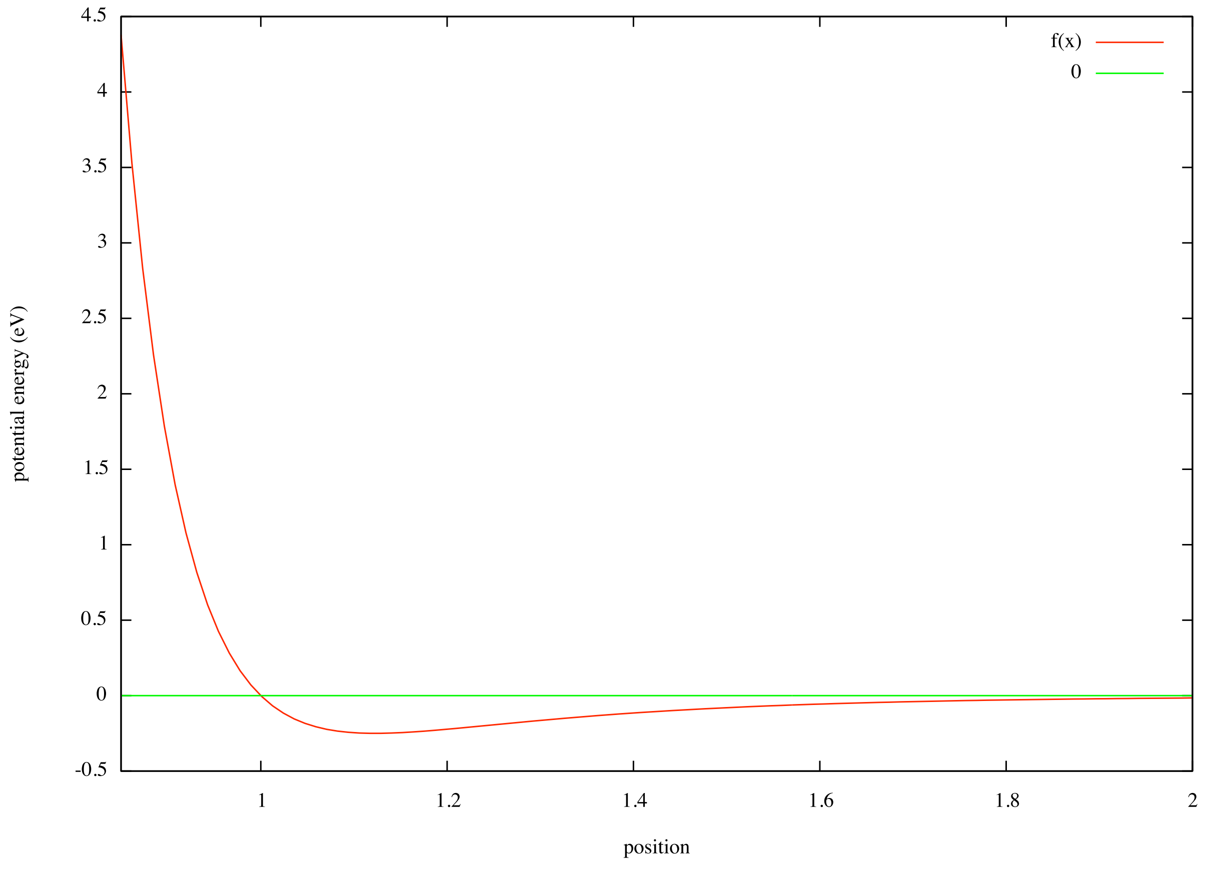
The Lennard-Jones potential is used relentlessly in chemistry and physics[2][3][4] (Just the top few results from Google Scholar). It is often employed to model the van der Waals effect in molecular collision simulations. The form of the Lennard-Jones potential is very similar to:

$$V = c(\frac{1}{x^{12}} - \frac{1}{x^6})$$

which is the potential studied here. Normally this potential repels two particles away from one another when they get too close. Since this is a 1-D quantum mechanics project, I set V up as my potential barrier. In this case x refers to how far into the barrier an electron has flown. Not exactly the same use as molecular collisions, but still interesting. Bound states should still occur at the same energies. A bound state has a particularly favorable resonance within the potential, the lowest bound state is often referred to as the ground state in other projects and is the energy at which a half-length wave fits in the potential and asymptotically approaches zero outside of the well.

Methods

How does someone go about finding the bound states of a potential? That depends on the type of potential being used. In my case it was a potential without asymptotic or periodic boundary conditions. From the equation for my potential it is obvious that it becomes really big really fast as x approaches 0. To avoid any cataclysmic mathematical problems I decided to have my smallest value of x = 0.85 . By setting the minimum x there I ensure a ‘reasonably’ large potential barrier on the left side of the system. Additionally, I set the maximum value as x = 100.0 . Due to the shape of the potential 100.0 seems to be close enough to infinity, a plot of my potential is below.



As can be seen, the potential is already ‘essentially’ zero by the time the electron would get to x = 2, so by x = 100 it should be a fairly reasonable approximation to infinity. Due to the nature of my potential, and having a nice function, I used the sine transform method to find the bound states.

This method works by coupling the wave function with its second derivative and integrating via the Runge-Kutta order 4 method [1].

$$A = e^{kx}$$
$$\frac{dA}{dx} = B$$

$$\frac{dB}{dx} = k^2 A$$

$$k = \sqrt{(E - V(x))\frac{2m}{\hbar}}$$

By integrating the second derivative of A for a fixed energy I obtained values for B. If B at a particular energy and being integrated to x → ∞ is equal to kA then that energy value corresponds to a bound state. This is valid since the wave function (A) should asymptotically go to zero as x → ∞, which means its derivative should do the same.

A couple methods exist for determining if a bound state was in fact calculated. Pictorially is probably the most interesting way since one literally ‘sees’ where the bound states are. This method is done by plotting RK4 determined B values in B vs A coordinates. Additionally two lines, B=kA and B=-kA, are plotted. For all energies save those corresponding to bound states will diverge wildly from the two oblique lines. When a bound state is found, the calculated B values will curve around until they hug the B=-kA line. If the coder were particularly interested in being flashy he could construct a video from the plots for each energy.

The other method, that I am aware of, involves no images. It is much more automated and simply reports the energies at which bound states occur in a data file. This technique works by doing the exact same RK4 process as the pictorial method at a particular energy, but it merely takes the final B value - kA and records that value. If on the next energy level the sign (+-) changes for the new B-kA value then a bound state exists somewhere between those two energies. Then the program continues running until either another sign change occurs and it reports that range too, or it reaches the maximum energy being searched over. For my potential I scanned energies from value at the minimum of the curve up to 0eV. After these energy ranges have been recorded, the same program is re-run with a more fine resolution (1000 steps) over the range and an approximate value for the bound state is picked out. In reality both methods could be employed simultaneously assuming technical difficulties in producing the plots are overcome.

In my RK4 integrator a few constants had to be defined. For the steps in the x direction I used 0.01. This small value should ensure convergence of the calculation.

Results

Finding the bound states for my potential is not particularly interesting unless they are found for a variety of values of c. For this project I chose values of c = 1, 20, 50, and 100. It was an arbitrary selection of values on my part that I thought would give a decent idea of how the bound state(s) change. Besides the original potential I also found bound states at the same c values for:

$$V = c(\frac{1}{x^{13}} - \frac{1}{x^7})$$

and:

$$V = c(\frac{1}{x^{11}} - \frac{1}{x^5})$$

The calculated bound states are in the following table.

x^13 x^7 potenial		**x^12 x^6 potenial**		**x^11 x^5 potenial**	
c=1.0	**bound (eV)**	**c=1.0**	**bound (eV)**	**c=1.0**	**bound (eV)**
ground state	-0.0069826	**ground state**	-0.007293	**ground state**	-0.00689731
c=20.0		**c=20.0**		**c=20.0**	
ground state	-3.481657	**ground state**	-3.5037875	**ground state**	-3.540993
1st excited state	-0.024326	**1st excited state**	-0.0226575	**1st excited state**	-0.020713
c=50.0		**c=50.0**		**c=50.0**	
ground state	-3.189116	**ground state**	-3.2472375	**ground state**	-3.31905
1st excited state	-0.053587	**1st excited state**	-0.046725	**1st excited state**	-0.039104
c=100.0		**c=100.0**		**c=100.0**	
ground state	-2.778687	**ground state**	-2.90456	**ground state**	-3.05527
1st excited state	-0.11967	**1st excited state**	-0.09597	**1st excited state**	-0.07176

Discussion

Interestingly, the energy that the bound states occur for a particular c value does not change much from one potential to another. This is not too surprising though as the potential wells do not change by that much from one potential equation to the next. Something I do not really understand is why at larger c values the two bound states get closer in energy. As c increases the ground state increases in energy and the first excited state decreases. For example, in the x^{13} x^7 potential, as c goes from 20 to 50 the ground state increases in energy by 8.4%. Again, going from c=50 to c=100 there is a change of 12.87%. My idea for why this occurs is that the potential well is getting stretched too much to support the ground state at lower energies, that a half-wave literally would not fit in the width of the potential. This is supported by the negative values of the well always starting at x>1 and the minimum is always in the same location, x=1.12246. Regardless of why, the values in the table show the energies at which regular wave functions will be located.

So now what? The next thing to be done is to modify the program to actually solve the Lennard-Jones potential instead of this similar potential. The only difference between the two is instead of one coefficient c, there are two. One would be c^{12} while the other would be c^6, both being divided by x of the same power. Due to the popularity of Lennard-Jones in essentially all molecular dynamics models fully understanding this potential is rather important.

References

[1] Gilmore, Robert *Elementary Quantum Mechanics in One Dimension*, Johns Hopkins University Press, 2004, particularly chapter 36

[2] W. Kob and C. Donati and S. J. Plimpton and P. H. Poole and S. C. Glotzer, *Dynamical Heterogeneities in a Supercooled Lennard-Jones Liquid,* Physical Review Letters, 1997, 10.1103/PhysRevLett.79.2827

[3] L. Verlet, *Computer "Experiments" on Classical Fluids. I. Thermodynamical Properties of Lennard-Jones Molecules*, Physical Review, 1967, 10.1103/PhysRev.159.98

[4] Amal Lotfi; Jadran Vrabec; Johann Fischer, Vapour liquid equilibria of the Lennard-Jones fluid from the *NpT* plus test particle method, Molecular Physics, Volume 76, Issue 6, August 1992 , pages 1319 - 1333

Appendix

Here is the code used during this project. It was written in Python. A bash script was used to run this code repeatedly with iterated energies.

```
import sys
from numpy import *
#order for arguments is p,q,E
#the variables for the numerators in the potential
#p is for r6
#q is for r12
p = float(sys.argv[1])
q = float(sys.argv[2])

#big_pow and small_pow are the exponents for the lenard jones pot.
#i used variables since i want to do this for various versions
#of the potential, not just the standard exponents
big_pow = 11.0
small_pow = 5.0
#the value for emin here was for a fine tune scan, normally it would be the equation
#commented out right after it
emin =0.0841846535 #abs(p/pow(1.122462048,big_pow)-q/
#pow(1.122462048,small_pow))
#again, the long decimal was specific to a fine tune scan
E = -emin + float(sys.argv[3])*0.0280615512/1000.0
c=0.26265625
#output file
data = open('data_fine.txt','a')
#rmax is the largest r value to be used
#dr is the increment stepsize
rmax = 100.0
dr = 1.0/100.0

#I can't start at zero due to division by zero, so 0.5 gave
#a reasonible energy, about 4000eV
rmin = 0.85
N = (rmax-rmin)/dr

#data is an array of all the points reached
#during the integration so i can make a plot
#of what each curve looks like
data_t = zeros(int(N+1), dtype=float)
data_w = zeros(int(N+1), dtype=float)
data_x = zeros(int(N+1), dtype=float)
#here is my runge-kutta process since the built in one
```

```
#doesn't save the intermediate steps, which are needed
#to make a pretty picture
#the algorithm is from 'Numerical Analysis' by Richard Burden, page 278
#all of the variables are defined there
alpha = exp(sqrt(abs(E-p/pow(rmin,12)+q/pow(rmin,6))*c)*rmin)
h = (rmax-rmin)/N
#i use t here instead of the more natural r
#to follow the notation in the book being used
#making me less likely to get confused and screw it up
t = rmin
w = alpha
data_t[0]= t
data_w[0] = w
data_x[0]=rmin

for i in range(1,int(N)+1):
    data_x[i]=rmin+dr*i
    tempt = t
    tempw = w
    k1 = h*(-c*(E-p/pow(tempt,big_pow)+q/pow(tempt,small_pow)))*tempw

    tempt = t + h/2.0
    tempw = w + k1/2.0
    k2 = h*(-c*(E-p/pow(tempt,big_pow)+q/pow(tempt,small_pow)))*tempw

    tempt = t + h/2.0
    tempw = w + k2/2.0
    k3 = h*(-c*(E-p/pow(tempt,big_pow)+q/pow(tempt,small_pow)))*tempw

    tempt = t + h
    tempw = w + k3
    k4 = h*(-c*(E-p/pow(tempt,big_pow)+q/pow(tempt,small_pow)))*tempw

    w = w +(k1 + 2.0*k2 + 2.0*k3 + k4)/6
    t = rmin + i*h

    data_t[i] = t
    data_w[i] = w
   # print >> data, t, w
ka = sqrt(c*abs((E-p/pow(rmax,big_pow)+q/pow(rmax,small_pow))))*exp(sqrt(c*abs((E-
p/pow(rmax,big_pow)+q/pow(rmax,small_pow))))*rmax)
print >> data, E ,w,ka,w-ka
data.close()
```