University of Hertfordshire UH

Self-Consistent Field Problems in Quantum Mechanics

Alex Barber-Lynch

Department of Physics, Astrophysics and Maths
University of Hertfordshire
April 2020

Contents

Abstract

Computers have allowed Hartree equations to be solved through the use of the Self-Consistent Field Method. One such example is two interacting particles bound in an infinite well. This paper discusses the solutions to two particles trapped in a one-dimensional infinite square well for the systems ground and first excited state, as well as the solutions to two particles trapped in a one-dimensional quantum harmonic oscillator for the systems ground and first excited state. The results show how the two particles interact with each other under two different conditions.

1 Introduction

The problem of solving a single particle in an infinite potential well is a common example in quantum mechanics. The Schrödinger equation can be solved analytically for this single particle problem. Therefore, an expansion of this problem is to include a second, identical particle and have both particles interact. The identity of the two particles is interchangeable between any two interacting fermions and two interacting bosons as both can exist in the same energy level when only two particles are present in the system. For the purposes of calculations and examples we will use electrons. The working of the Schrödinger equation when an interaction potential is included leads directly to the Hartree equations which we discuss in the next section. Finally, a computer will be used to solve the equations using the self-consistent field method developed by Hartree and Fock (D. R. Hartree 1927)(F.R.S. and W. Hartree 1935) and applied by J.S. Bolemon and D.J. Etzold (Jay S. Boleman 1974).

2 Initial Equations

2.1 Single Particle

An appropriate starting point is to show the Schrödinger equation solution for a single particle in an infinite potential well.

The Time-Independent Schrödinger Equation:

$$-\frac{\hbar^2}{2m}\frac{d^2}{dx^2}\psi(x) + V_{well}(x)\psi(x) = E\psi(x) \quad (1)$$

When

$$V_{well}(x) = \begin{cases} 0, & \text{if } x < |L|, \\ \infty, & \text{if } x > |L|. \end{cases} \quad (2)$$

the Schrödinger equation can be solved to the form:

$$\psi(x) = Asin(kx) + Bcos(kx) \quad , \quad k = \frac{\sqrt{2mE}}{\hbar} \quad (3)$$

where the values of the constants A and B can be calculated (Paul C. W. Davies 1994).

2.2 Two Interacting Particles

The introduction of a second interacting particle means that the Schrödinger equations must be modified to include the interaction potential. We add the term $V_{int}(|x_1 - x_2|)$, which shows that the potential energy that each particle exerts on the other is dependent on the separation between the two particles. We now attempt to write a general term for the energy of the system, and, as we did previously, we set the potential of the well to be:

$$V_{well}(x) = \begin{cases} 0, & \text{if } x < |L|, \\ \infty, & \text{if } x > |L|. \end{cases} \quad (4)$$

The Time-Independent Schrödinger Equation becomes

$$[-\frac{\hbar^2}{2m}(\frac{d^2}{dx_1^2} + \frac{d^2}{dx_2^2}) + Vint(|x_1 - x_2|)]\psi(x_1, x_2) = E\psi(x1, x2) \quad (5)$$

where $\psi(x_1, x_2)$ is the total wavefunction of the system and E is the total energy of the system.

Although this equation conserves total energy in the system, exact solutions to specific problems will be rare (Moshinsky 1968). The interaction term means that the eigenfunction of one particle will always be bound to the eigenfunction of the other particle. This means deriving exact particle equations for either particle will be impossible. A solution to this is to average the interaction term over both particles' positions and substitute the wavefunction of the collected particles with a sum of the wavefunctions of both particles. This can be shown logically. If you average the interaction term over both particles' positions, the equation resembles a form where it has two non-interacting particles in the system. As each particle in the non-interacting system moves independently of each other, it can be inferred that each particle moves independently in the averaged field of the other particle, thus, allowing the wavefunction of the collected particles to be substituted with a sum of the wavefunctions of both particles. (Jay S. Boleman 1974)

$$\left[-\frac{\hbar^2}{2m}(\frac{d^2}{dx_1^2} + \frac{d^2}{dx_2^2}) + \iint |\psi_1(x_1)|^2 |\psi_2(x_2)|^2 \, V_{int}(|x_1 - x_2|) dx_2 \, dx_1 \right] \times \psi_1(x_1)\psi_2(x_2) = E\psi_1(x_1)\psi_2(x_2) \quad (6)$$

Separating E into $E_1 + E_2$ leads to two equations (one for each particle)

$$-\frac{\hbar^2}{2m}\frac{d^2}{dx_1^2}\psi_1(x_1) + \int |\psi_2(x_2)|^2 \, V_{int}(|x_1 - x_2|) \, dx_2 \, \psi_1(x_1) = E_1\psi_1(x_1) \quad (7)$$

and

$$-\frac{\hbar^2}{2m}\frac{d^2}{dx_2^2}\psi_2(x_2) + \int |\psi_1(x_1)|^2 \, V_{int}(|x_2 - x_1|) \, dx_1 \, \psi_2(x_2) = E_2\psi_2(x_2) \quad (8)$$

These are the Hartree equations. As you can see, the eigenvalues and wavefunction of the first particle are dependent on the eigenvalues and wavefunction of the second, which are in turn themselves dependent on

the eigenvalues and wavefunction of the first particle. These two equations are coupled through their integrals and are called coupled integrodifferential equations. These are the equations we will be using in the Self-Consistent Field method.

3 Self-Consistent Field Method

3.1 History

In 1927, D. R. Hartree introduced a procedure, which he called the self-consistent field method, to calculate approximate wavefunctions and energies for particles of a quantum many-body system in a stationary state. (D. R. Hartree 1927) His idea was to use an initial field to generate a distribution of charge which he called the final field, then use the final field as a new initial field to generate a new final field. If the final field is the same as the initial field, the field will be called 'self-consistent' and the characteristics of a particle can be discovered. This method involved three main approximations: 1) no inclusion of relativity and spin effects; 2) no inclusion of exchange effects; and, 3) each particle is replaced with a statistical average distribution in calculating its effects on the other particle. (F.R.S. and W. Hartree 1935) Subsequently, in 1930 V.A. Fock introduced a method to account for exchange effects. The practical calculation of Fock's method however was too abstract for contemporary physicists to understand and implement, but in 1935 Hartree reformulated Fock's method into the Hartree-Fock method. (ibid.) This method is what we now know as the self-consistent field method and is shown in Figure 2. Although this new method was more suitable for the purpose of calculations, it was still too complex to calculate until the introduction of computers. In 1974 J.S. Boleman and D.J. Etzold published a paper outlining how the self-consistent field method could be used with a computer to solve quantum mechanical problems in one dimension (Jay S. Boleman 1974).

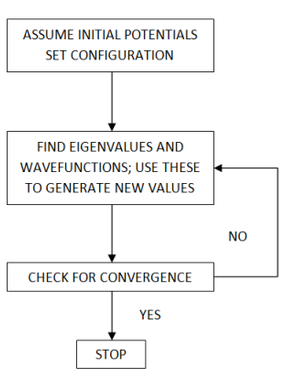

Figure 1: Self Consistent Field flow chart (Jay S. Boleman 1974)

3.2 Application

Due to the coupled integrals, equations (5) and (6) cannot be solved normally. In order to calculate $\psi_1(x_1)$ we need to know $\psi_2(x_2)$ and to calculate $\psi_2(x_2)$ we need to know $\psi_1(x_1)$ etc... We must, therefore, use the self-consistent field method (SCF) for this purpose. An initial guess for the eigenvalues and wavefunction must first be generated. The system must remain in the same configuration during the process i.e. to calculate the ground state and first excited state of the system, separate SCF calculations must be made. The initial guess will then be used to generate new eigenvalues and a new wavefunction for each particle in the system and these, in turn, will be used to generate new values. Provided the initial guess we used was appropriate, the eigenvalues and wavefunctions used in the process will generate themselves after a small number of iterations. Once the answers have reached a suitable convergence, the process can be halted and the convergent answers can be presented as the correct answers.

4 Infinite Potential Well

4.1 Setting up the SCF method

We place two particles in the ground state initially and if they are fermions we assume they have opposite spin. The interaction potential that we choose is a Gaussian following Bolemon and Etzold, because the symmetry of the Gaussian fits the symmetry of the system and the particles can pass through each other without experiencing an infinite force, i.e. avoiding singularities (ibid.).

$$Vint(|x_1 - x_2|) = Ce^{\frac{(x_1-x_2)^2}{B}} \tag{9}$$

The values C=30 and B=0.04 are chosen due to the appearance of the graphs. The effects of the interaction potential can be clearly seen and the results can be calculated relatively quickly due to a low iteration number (ibid.). These values determine the strength of the interaction potential. Graphs with varying values of C and B are included in the appendix.

Due to the wavefunction becoming zero at the edges of the well, we can use the shooting method to calculate solutions for each iteration of the SCF method and an initial guess for the energy of the particle is used to calculate the wavefunction. If the wavefunction is non-zero at the edge of the well, the energy is adjusted depending on whether the wavefunction is greater than or less than zero and the process is run again until the correct value is generated. This value is then used in the SCF method.

To generate a ground state plot of the system, we use the wavefunction and eigenvalues of a single particle in its ground state as our 'initial guess' for both particles.

4.2 Results

The effect of the interaction potential is shown in figures 2 and 3, where the probability density of the two interacting particles is clearly different to that of the single particle. Figure 2 shows the system in its ground state when both particles are in their ground state. Their mutual interaction potential forces them away from the centre of the well where their probability densities are greatest when they have no interaction. This results in the interacting

First excited state C=30 B=0.04	ground state C=30 B=0.04	First excited state C=50 B=0.5	ground state C=50 B=0.5
		E1= 13.9 E2=, 45.848	E1= 13.9 E2=, 25.868
E1=10.667 E2=40.044	E1=10.667 E2=13.907	E1= 22.647 E2=, 45.778	E1= 25.274 E2=, 25.156
E1=12.369 E2=40.037	E1=13.742 E2=13.755	E1= 22.715 E2=, 45.77	E1= 25.287 E2=, 25.198
E1=12.369 E2=40.037	E1=13.754 E2=13.754	E1= 22.697 E2=, 45.772	E1= 25.254 E2=, 25.219
	E1=13.754 E2=13.754	E1= 22.701 E2=, 45.771	E1= 25.24 E2=, 25.227
		E1= 22.7 E2=, 45.771	E1= 25.235 E2=, 25.23
			E1= 25.233 E2=, 25.233
			E1= 25.233 E2=, 25.233

Table 1: Table showing the number of iterations required to complete a calculation for two interacting particles in the infinite square well. All results are in eV.

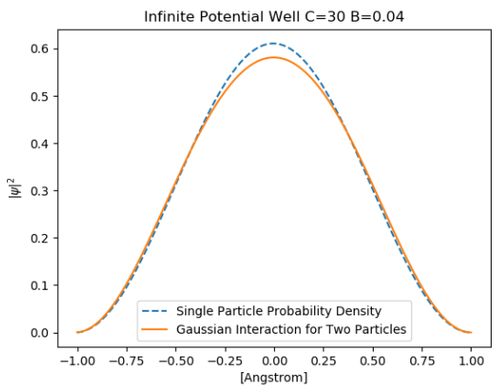

Figure 2: The ground state of the system. Both electrons are in their ground state and have the same probability density. The dotted line represents the single non-interacting particle result for the ground state of an electron.

particles having a lower probability density at the centre of the well and a slightly higher probability density near the edges of the well than the non-interacting particles.

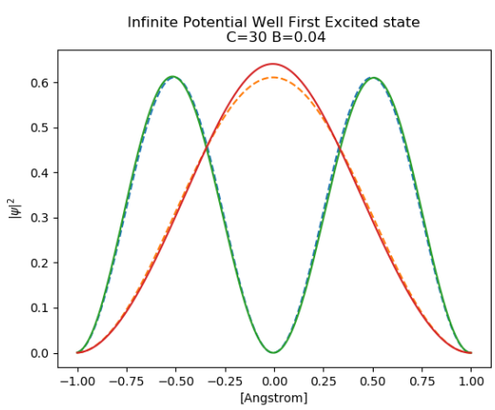

Figure 3: The first excited state of the system. The first electron is in its ground state while the second electron is in its first excited state. The dotted lines represent the single non-interacting particle results for the ground and first excited states of an electron.

Figure 3 shows the system in its first excited state where the first particle is in its ground state while the second particle is in its first excited state. The second particle's interaction potential forces the first particle into the centre of the well while the first particle's interaction potential forces the second particle away from the centre of the well. This results in the probability density of particle one being greater than the probability density of a non-interacting particle in its ground state at the centre of the well. Additionally the probability density maximums of particle two are more shifted away from the centre of the well than the probability density maximums of a non-interacting particle in its first excited state.

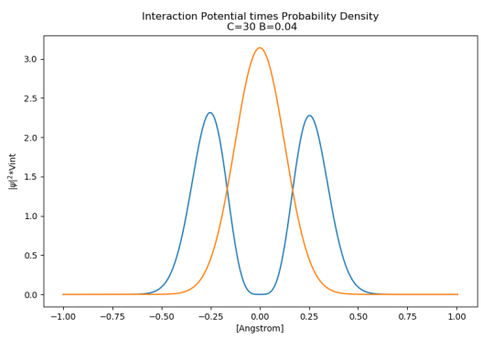

Figure 4: Interaction potential each particle experiences when the other is at position x=0 and the system is in its first excited state. Orange line is interaction potential of particle in ground state. Blue line is interaction potential particle in first excited state.

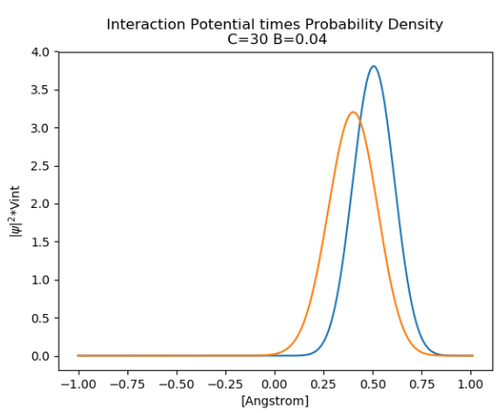

Figure 5: Interaction potential each particle experiences when the other is at position x=0.5 and the system is in its first excited state. Orange line is interaction potential of particle in ground state. Blue line is interaction potential particle in first excited state.

As shown in figures 4 and 5, the interaction potential experienced by one particle from the other changes as the position of the particle changes. At the centre of the well the particle in the ground state has the greatest interaction potential and as such, forces the particle in the first excited state away from the centre. Further away from the centre, at positions x=0.5 and x=-0.5, the particle in its first excited state has a greater interaction potential and as such, forces the particle in ground state towards the centre of the well. These results coincide with the interpretation

of figure 3.

5 Simple Harmonic Oscillator

5.1 Setting up the SCF method

Again, we initially place the two particles in the ground state and if they are fermions we assume they have opposite spin. We define the potential of the well as $V_{well}(x) = \frac{1}{2}x^2$ and the interaction potential as:

$$V_{int}(|x_1 - x_2|) = D(x_1 - x_2)^2 \tag{10}$$

The interaction potential is attractive and modelled as a quadratic with attraction increasing as the particles are separated, this results in the same effect as the quadratic well potential. When $x_1 = x_2$ the interaction potential is set to decrease to zero to avoid singularities. We chose the value $D = \frac{1}{2}$ following Bolemon and Etzold, due to the same reasons as described in chapter 4.1 for values C and B (Jay S. Boleman 1974).

Due to the wavefunction becoming zero at the edges of the well, the same method can be used as described in chapter 4.1, the shooting method.

Again, like in the infinite potential well example, to generate a ground state plot of the system, we use a single particle's ground state wavefunction and eigenvalues as our 'initial guess' for both particles.

5.2 Results

Figure 6 shows the system in its ground state when both particles are in their ground states. Due to the interaction potential being attractive, the mutual interaction potential pulls the two particles towards the centre of the well. This can be seen by the fact that the probability density of the two interacting particles at the centre of the well is greater than the probability density of the single particle at the centre of the well.

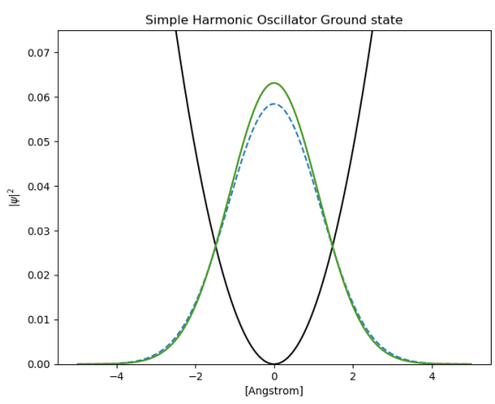

Figure 6: The ground state of the system. Both electrons are in their ground state and have the same probability density. The dotted line represents the single non-interacting particle result for the ground state of an electron. Black line is well potential

Figure 7 shows the system in its first excited state where particle one is in its ground state and particle two is in its first excited state. The interaction potential particle two experiences from particle one pulls the particle towards the middle of the well. This can be seen by the peaks of the two probability density maximums being slightly closer to the centre of the well than the peaks of the non-interacting particle in its first excited state. The interaction potential particle one experiences from particle two also pulls the particle towards the centre of the well. Due to the interaction potential increasing with separation of the particles, the furthest probability density peak of particle two will have the greatest attraction, which is shown in figure 9. Due to the symmetry of the system, particle one is pulled towards the centre of the well, which is shown by the probability density at the centre of the well being greater for particle one than the probability density at the centre of the well for the non-interacting particle in its first excited state.

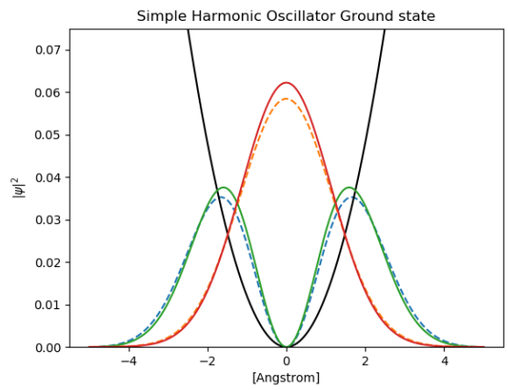

Figure 7: The first excited state of the system. The first electron is in its ground state while the second electron is in its first excited state. The dotted lines represent the single non-interacting particle results for the ground and first excited states of an electron. Black line is well potential

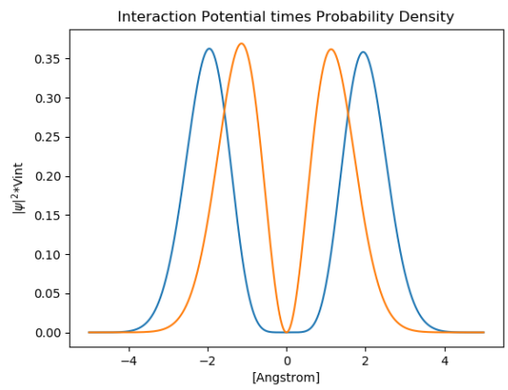

Figure 8: Interaction potential each particle experiences when the other is at position x=0 when the system is in its first excited state. Orange line is interaction potential of particle in ground state. Blue line is interaction potential of particle in first excited state.

Figures 8 and 9 show how the interaction potential experienced by one particle from the other changes as the position of the particle changes. Due to the interacting potential being zero when $x_1 = x_2$, figure 8 has the effect of two peaks for both the particle in its ground state and first excited state. Due to the two peaks for the particle in its ground state being localized to the centre of the well, and the peak remaining localized to the centre of the well

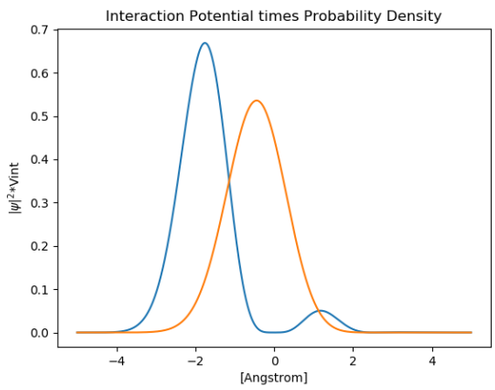

Figure 9: Interaction potential each particle experiences when the other is at position x=2.5 when the system is in its first excited state. Orange line is interaction potential of particle in ground state. Blue line is interaction potential of particle in first excited state.

when the other particle is moved, as seen in Figure 9, the interacting potential from particle one pulls particle two towards the centre of the well. Figure 9 shows the effect of the interaction potential being a quadratic that increases with distance. Particle one is placed at position x=2.5 and feels a pull from the furthest peak of the particle in its first excited state. If particle one were instead placed at position x=-2.5, due to the symmetry of the system, the graph would be reflected. Particle one would still experience the greatest pull from particle twos furthest peak. This results in particle one always being drawn towards the centre of the well, as can be seen in figure 7.

6 Conclusion

The self-consistent field method is an effective method for solving Hartree equations with the use of computers. Decreasing the size of dx, the integration step size, increases the accuracy of the resulting graphs. For example, in Figure 9, the height of the peaks for each particle on either side of the centre of the well is the same value. Due to the small values of the graph, the errors resulting from using a large dx become more visible. The solution would be to use a smaller value of dx, but the time required for the process to finish becomes much longer. Due to my technological limits, I used the smallest value of dx without the computational time becoming too lengthy.

An expansion would be to add additional particles to the interaction. This could easily be achieved by adding multiple interaction terms. The model could also be applied to more real world problems such as multi-electron atoms, for example, helium.

7 References

F.R.S., D. R. Hartree and W. Hartree (1935). "Self-Consistent Field, with Exchange, for Beryllium". In:

Hartree, D. R. (1927). "The Wave Mechanics of an Atom with a Non-Coulomb Central Field. Part II. Some Results and Discussion". In:

Jay S. Boleman, David J. Etzold (1974). "Enriching Elementary Quantum Mechanics with the Computer: Self-Consistent Field Problems in One Dimension". In: *American Journal of Physics*.

Moshinsky, M. (1968). "How Good is the Hartree-Frock Approximation". In: *American Journal of Physics* 36.52.

Paul C. W. Davies, David S. Betts (1994). *Quantum Mechanics*. Second Edition. Nelson Thornes Ltd.

8 Appendix/Python Code

8.1 Graphs of the infinite potential well with varying values of C and B

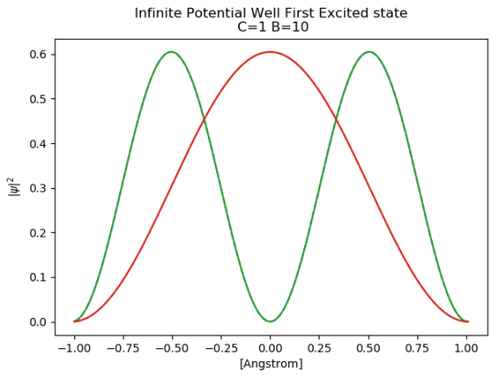

Figure 10: Caption

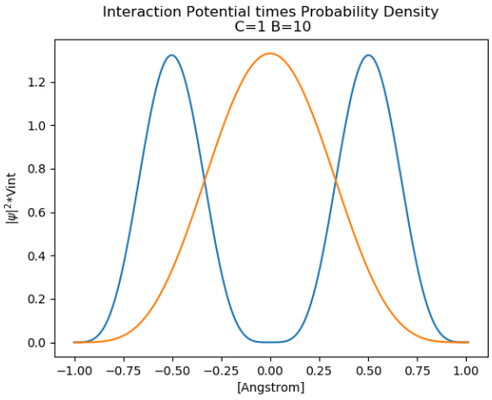

Figure 11: Caption

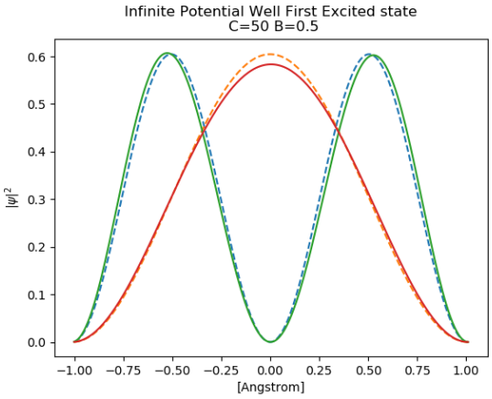

Figure 12: Caption

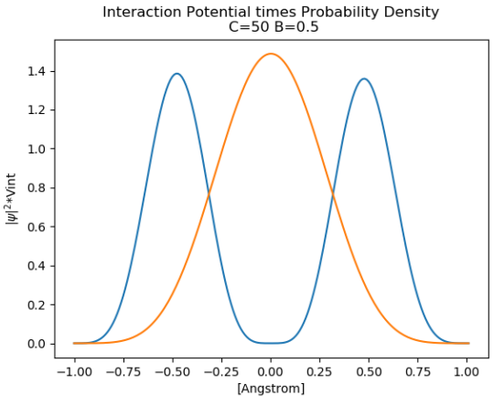

Figure 13: Caption

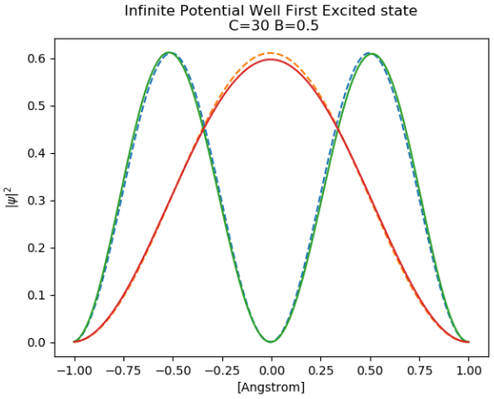

Figure 14: Caption

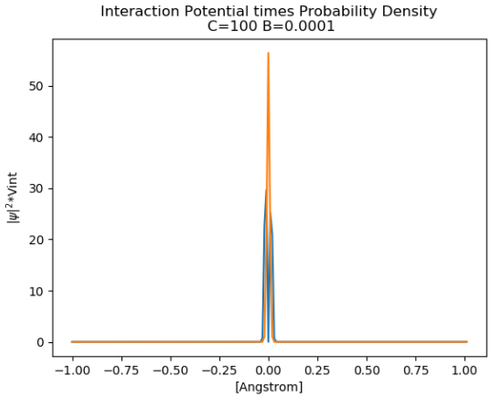

Figure 15: Caption

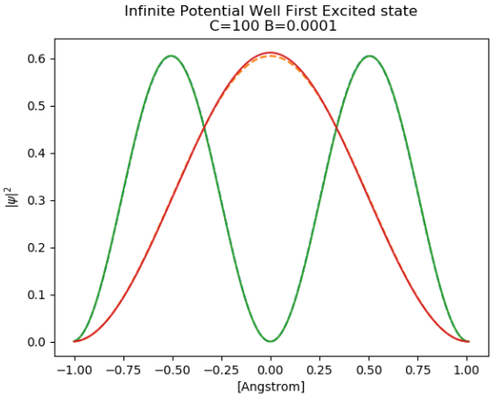

Figure 16: Caption

8.2 Two Interacting Particles, Infinite Potential Well

```
import matplotlib.pyplot as plt
import numpy as np
import sys



def wave(E,PSI2,X2):    #calculates final values of psi for specific E
    X=[]                #set empty X list
    PSI=[]              #set empty PSI list
    x1=-L
    psi=0
    dpsi=1
    while x1<=L:                #calculate values of psi along every dx step
        x1=round(x1,10)
        ddpsi=-0.26246*(E-Vint(X2,x1,PSI2))*psi    #includes interaction
                                                    potential. -0.26246 is value
                                                    of -hbar^2/2m
        dpsi=dpsi+ddpsi*dx
        psi=psi+dpsi*dx
        X.append(x1)            #append value of x to end of X list
        PSI.append(psi)         #append value of psi to end of PSI list
        x1=x1+dx
    psi=round(psi,10)
    return psi,PSI,X            #outputs final value of psi and filled lists PSI
and X

def guesswave(E):   #calculates values of psi for specific E. This is
                     the guess wavefunction.
    X=[]            #set empty X list
    PSI=[]          #set empty PSI list
    x1=-L
    psi=0
    dpsi=1
    while x1<=L:                #calculate values of psi along every dx step
        x1=round(x1,10)
        ddpsi=-0.26246*(E)*psi        # -0.26246 is value of -hbar^2/2m
        dpsi=dpsi+ddpsi*dx
        psi=psi+dpsi*dx
        X.append(x1)            #append value of x to end of X list
        PSI.append(psi)         #append value of psi to end of PSI list
        x1=x1+dx
    psi=round(psi,10)
    return psi,PSI,X            #outputs final value of psi and filled lists PSI
and X

def VintEq (k,X2,x1,PSI2):   #interaction potential equation
    return (PSI2[k]**2)*C*np.e**(-(((x1-X2[k])**2)/B))

def Vint (X2,x1,PSI2):      #calculate total value of potential a
                             particle feels from the other at a
                             specific value of x
    Varea=0
    for k in range(1,len(PSI2)-1):
        Varea+=dx*VintEq(k,X2,x1,PSI2)
    Varea+=0.5*dx*(VintEq(0,X2,x1,PSI2)+VintEq(len(PSI2)-1,X2,x1,PSI2))
    Varea=round(Varea,10)
    return Varea

def search (PSI2,X2,e):         #find value of E for a single particle when
it is effected by interaction potential
                                  of second particle. This is the
                                  shooting method.
#    global PSI2,X2
    E=e
    if wave(E,PSI2,X2)[0]>0:
        while wave(E,PSI2,X2)[0]!=0:        #calculate final value of psi at
specific energy.
            if wave(E,PSI2,X2)[0]>0:       #adjust value of E depending on if
                                            final value of psi is greater than
                                            or less than zero
```

```
                pE=E
                E=E*2
            if wave(E,PSI2,X2)[0]<0:      #recalculate final value of psi with
                                           new value of E. repeat until
                                           correct value of E is found.
                ppE=pE
                pE=E
                E=pE-(pE-ppE)/2
                if wave(E,PSI2,X2)[0]>0:
                    ppE=pE
                while wave(E,PSI2,X2)[0]!=0:
                    if wave(E,PSI2,X2)[0]<0:
                        pE=E
                        E=pE-(pE-ppE)/2
                        if wave(E,PSI2,X2)[0]>0:
                            ppE=pE
                    if wave(E,PSI2,X2)[0]>0:
                        pE=E
                        E=pE-(pE-ppE)/2
                        if wave(E,PSI2,X2)[0]<0:
                            ppE=pE
        return E,wave(E,PSI2,X2)[1],wave(E,PSI2,X2)[2]
    if wave(E,PSI2,X2)[0]<0:
        while wave(E,PSI2,X2)[0]!=0:
            if wave(E,PSI2,X2)[0]<0:
                pE=E
                E=E*2
            if wave(E,PSI2,X2)[0]>0:
                ppE=pE
                pE=E
                E=pE-(pE-ppE)/2
                if wave(E,PSI2,X2)[0]<0:
                    ppE=pE
                while wave(E,PSI2,X2)[0]!=0:
                    if wave(E,PSI2,X2)[0]>0:
                        pE=E
                        E=pE-(pE-ppE)/2
                        if wave(E,PSI2,X2)[0]<0:
                            ppE=pE
                    if wave(E,PSI2,X2)[0]<0:
                        pE=E
                        E=pE-(pE-ppE)/2
                        if wave(E,PSI2,X2)[0]>0:
                            ppE=pE
        return E,wave(E,PSI2,X2)[1],wave(E,PSI2,X2)[2]
                        #once a correct value of E is found it is returned


def guesssearch(e):            #find value of E for a single particle. This is
the shooting method and is used in
                                calculating the guess

    #global PSI2,X2,E
    E=e
    if guesswave(E)[0]>0:
        while guesswave(E)[0]!=0:    #calculate final value of psi at
specific energy.
            if guesswave(E)[0]>0:  #adjust value of E depending on if
                                    final value of psi is greater than
                                    or less than zero
                pE=E
                E=E*2
            if guesswave(E)[0]<0:     #recalculate final value of psi with
                                       new value of E. repeat until
                                       correct value of E is found.
                ppE=pE
                pE=E
                E=pE-(pE-ppE)/2
                if guesswave(E)[0]>0:
                    ppE=pE
                while guesswave(E)[0]!=0:
                    if guesswave(E)[0]<0:
```

```
                        pE=E
                        E=pE-(pE-ppE)/2
                        if guesswave(E)[0]>0:
                            ppE=pE
                    if guesswave(E)[0]>0:
                        pE=E
                        E=pE-(pE-ppE)/2
                        if guesswave(E)[0]<0:
                            ppE=pE
        return guesswave(E)[1],guesswave(E)[2]
    if guesswave(E)[0]<0:
        while guesswave(E)[0]!=0:
            if guesswave(E)[0]<0:
                pE=E
                E=E*2
            if guesswave(E)[0]>0:
                ppE=pE
                pE=E
                E=pE-(pE-ppE)/2
                if guesswave(E)[0]<0:
                    ppE=pE
                while guesswave(E)[0]!=0:
                    if guesswave(E)[0]>0:
                        pE=E
                        E=pE-(pE-ppE)/2
                        if guesswave(E)[0]<0:
                            ppE=pE
                    if guesswave(E)[0]<0:
                        pE=E
                        E=pE-(pE-ppE)/2
                        if guesswave(E)[0]>0:
                            ppE=pE
        return guesswave(E)[1],guesswave(E)[2]
                        #once a correct value of E is found it is returned

def normalize (PSI):                #function that normalizes the input
    area=0
    for i in range(1,len(PSI)-1):
        area+=abs(dx*PSI[i])
    area+=abs(0.5*dx*(PSI[0]+PSI[len(PSI)-1]))
    C=1/area
    for j in range(len(PSI)):
        PSI[j]=C*PSI[j]

global L,dx,C,B
                         #set initial values.
L=1.                     #length of box is -L to L
dx=0.005
C=30.
B=0.04
psi=1.
plt.figure()

PSI2,X2=guesssearch(10)[0],guesssearch(10)[1]      #calculate 'guess'
#function (one electron infinite square well)

#PSI2,X2=guesswave(1)[1],guesswave(1)[1]
normalize(PSI2)
for i in range(0,len(PSI2)):
    PSI2[i]=PSI2[i]**2
plt.plot(X2,PSI2,label="Single Particle Probability Density 1st Excited
                        #plot single particle result. this is also used
                        as the guess
State",linestyle='dashed')
#E=format(E,".5g")
E1=0                     #initial values of E and Previous E so while loop runs
E2=0
PE1=1
PE2=1

while E1!=PE1 and E2!=PE2:        #SCF method. process loops until calculated
                                   values equal previous values
```

```
    PE1=E1                                  #set E1 as Previous E1 before
calulating new E1
    e=1
    E,PSI1,X2=search(PSI2,X2,e)[0],search(PSI2,X2,e)[1],search(PSI2,X2,e)[2]
                        #calulate E1 using previous values of PSI, X
    E1=format(E,".5g")    #Reduce E1 to 5 significant figures
    normalize(PSI1)
    PE2=E2          #set E2 as previous E2 before calculating new E2
    e=20
    E,PSI2,X2=search(PSI1,X2,e)[0],search(PSI1,X2,e)[1],search(PSI1,X2,e)[2]
                        #calculate E2 using previous values of PSI, X
    E2=format(E,".5g")      #Reduce E2 to 5 significant figures
    normalize(PSI2)
    print("E1=",E1,"eV   E2=",E2,"eV")    #output calculated values of
                                            E1 and E2 for each iteration


PSI=guesssearch(1)[0]
print(len(X2),len(PSI))
normalize(PSI)
for i in range(0,len(PSI)):
    PSI[i]=PSI[i]**2
plt.plot(X2,PSI,label="Single Particle Probability Density Ground
State",linestyle='dashed') #plot single particle result

for i in range(0,len(PSI1)):    #calculates psi^2
    PSI1[i]=PSI1[i]**2

for i in range(0,len(PSI2)):    #calculates psi^2
    PSI2[i]=PSI2[i]**2

E1=str(E1)            #plotting the graph
E2=str(E2)
plt.plot(X2,PSI2,label="Particle 1 in Ground State")
plt.plot(X2,PSI1,label="Particle 2 in 1st Excited State")
#plt.legend(loc=1)
plt.title("Infinite Potential Well First Excited state"+"\n"+" C=30 B=0.04")
plt.xlabel("[Angstrom]")
plt.ylabel("$|\psi|^2$")
#plt.yticks([])
plt.show()


plt.figure()          #plotting the interaction potentials time probability
                        density of each particle for specific value of x
Int=[]
for k in range (0,len(PSI2)):
    Int.append(VintEq (k,X2,0,PSI2))
normalize(Int)
plt.plot(X2,Int)
plt.title("Interaction Potential times Probability Density"+"\n"+" C="+str(C)+" B="+str(B))
plt.xlabel("[Angstrom]")
plt.ylabel("$|\psi|^2$*Vint")

Int=[]
for k in range (0,len(PSI1)):
    Int.append(VintEq (k,X2,0,PSI1))
normalize(Int)
plt.plot(X2,Int)
plt.show()

plt.figure()            #plotting the interaction potential for each
                          particle
y=[]
for i in range (len(X2)):
    y.append(C*np.e**(-(((0-X2[i])**2)/B)))
plt.plot(X2,y)
plt.title("Interaction Potential"+"\n"+" C="+str(C)+" B="+str(B))
plt.xlabel('[Angstrom]')
plt.ylabel('Vint')
plt.show()
```

8.3 Two Interacting Particles, Simple Harmonic Oscillator

```
import matplotlib.pyplot as plt
import numpy as np
import sys



def wave(E,PSI2,X2):    #calculates final values of psi for specific E
    X=[]                #set empty X list
    PSI=[]              #set empty PSI list
    x1=-L
    psi=0
    dpsi=1
    while x1<=L:              #calculate values of psi along every dx step
        x1=round(x1,10)
        ddpsi=-0.26246*(E-(x1**2)/2-Vint(X2,x1,PSI2))*psi
#includes interaction
                                                    potential. -0.26246 is value
                                                    of -hbar^2/2m
        dpsi=dpsi+ddpsi*dx
        psi=psi+dpsi*dx
        X.append(x1)            #append value of x to end of X list
        PSI.append(psi)         #append value of psi to end of PSI list
        x1=x1+dx
    psi=round(psi,10)
    return psi,PSI,X            #outputs final value of psi and filled lists PSI
and X

def guesswave(E):   #calculates final values of psi for specific E
    X=[]            #set empty X list
    PSI=[]          #set empty PSI list
    x1=-L
    psi=0
    dpsi=1
    while x1<=L:              #calculate values of psi along every dx step
        x1=round(x1,10)
        ddpsi=-0.26246*(E-(x1**2)/2)*psi      #-0.26246 is value
                                                    of -hbar^2/2m
        dpsi=dpsi+ddpsi*dx
        psi=psi+dpsi*dx
        X.append(x1)            #append value of x to end of X list
        PSI.append(psi)         #append value of psi to end of PSI list
        x1=x1+dx
    psi=round(psi,10)
    return psi,PSI,X            #outputs final value of psi and filled lists PSI
and X

def VintEq (k,X2,x1,PSI2):      #interaction potential equation
    return (PSI2[k]**2)*(((x1-X2[k])**2)/2)

def Vint (X2,x1,PSI2):          #calculate total value of potential a
                                particle feels from the other at a
                                specific value of x
    Varea=0
    for k in range(1,len(PSI2)-1):
        Varea+=dx*VintEq(k,X2,x1,PSI2)
    Varea+=0.5*dx*(VintEq(0,X2,x1,PSI2)+VintEq(len(PSI2)-1,X2,x1,PSI2))
    Varea=round(Varea,10)
    return Varea

def search (PSI2,X2,e):         #find value of E for a single particle when
it is effected by interaction potential
                                  of second particle. This is the
                                  shooting method.
#    global PSI2,X2
    E=e
    if wave(E,PSI2,X2)[0]>0:
        while wave(E,PSI2,X2)[0]!=0:        #calculate final value of psi at
specific energy.
            if wave(E,PSI2,X2)[0]>0:        #adjust value of E depending on if
                                            final value of psi is greater than
```

```
                                        or less than zero
                pE=E
                E=E*2
            if wave(E,PSI2,X2)[0]<0:        #recalculate final value of psi with
                                            new value of E. repeat until
                                            correct value of E is found.
                ppE=pE
                pE=E
                E=pE-(pE-ppE)/2
                if wave(E,PSI2,X2)[0]>0:
                    ppE=pE
                while wave(E,PSI2,X2)[0]!=0:
                    if wave(E,PSI2,X2)[0]<0:
                        pE=E
                        E=pE-(pE-ppE)/2
                        if wave(E,PSI2,X2)[0]>0:
                            ppE=pE
                    if wave(E,PSI2,X2)[0]>0:
                        pE=E
                        E=pE-(pE-ppE)/2
                        if wave(E,PSI2,X2)[0]<0:
                            ppE=pE
        return E,wave(E,PSI2,X2)[1],wave(E,PSI2,X2)[2]
    if wave(E,PSI2,X2)[0]<0:
        while wave(E,PSI2,X2)[0]!=0:
            if wave(E,PSI2,X2)[0]<0:
                pE=E
                E=E*2
            if wave(E,PSI2,X2)[0]>0:
                ppE=pE
                pE=E
                E=pE-(pE-ppE)/2
                if wave(E,PSI2,X2)[0]<0:
                    ppE=pE
                while wave(E,PSI2,X2)[0]!=0:
                    if wave(E,PSI2,X2)[0]>0:
                        pE=E
                        E=pE-(pE-ppE)/2
                        if wave(E,PSI2,X2)[0]<0:
                            ppE=pE
                    if wave(E,PSI2,X2)[0]<0:
                        pE=E
                        E=pE-(pE-ppE)/2
                        if wave(E,PSI2,X2)[0]>0:
                            ppE=pE
        return E,wave(E,PSI2,X2)[1],wave(E,PSI2,X2)[2]
                        #once a correct value of E is found it is returned

def guesssearch (e):            #find value of E for a single particle. This is
the shooting method and is used in
                                calculating the guess ground state
    #global PSI2,X2,E
    E=e
    if guesswave(E)[0]>0:
        while guesswave(E)[0]!=0:   #calculate final value of psi at
specific energy. specific energy.
            if guesswave(E)[0]>0:   #adjust value of E depending on if
                                        final value of psi is greater than
                                        or less than zero
                pE=E
                E=E*2
            if guesswave(E)[0]<0:   #recalculate final value of psi with
                                        new value of E. repeat until
                                        correct value of E is found.
                ppE=pE
                pE=E
                E=pE-(pE-ppE)/2
                if guesswave(E)[0]>0:
                    ppE=pE
                while guesswave(E)[0]!=0:
                    if guesswave(E)[0]<0:
                        pE=E
```

```
                        E=pE-(pE-ppE)/2
                        if guesswave(E)[0]>0:
                            ppE=pE
                    if guesswave(E)[0]>0:
                        pE=E
                        E=pE-(pE-ppE)/2
                        if guesswave(E)[0]<0:
                            ppE=pE
        return guesswave(E)[1],guesswave(E)[2]
    if guesswave(E)[0]<0:
        while guesswave(E)[0]!=0:
            if guesswave(E)[0]<0:
                pE=E
                E=E*2
            if guesswave(E)[0]>0:
                ppE=pE
                pE=E
                E=pE-(pE-ppE)/2
                if guesswave(E)[0]<0:
                    ppE=pE
                while guesswave(E)[0]!=0:
                    if guesswave(E)[0]>0:
                        pE=E
                        E=pE-(pE-ppE)/2
                        if guesswave(E)[0]<0:
                            ppE=pE
                    if guesswave(E)[0]<0:
                        pE=E
                        E=pE-(pE-ppE)/2
                        if guesswave(E)[0]>0:
                            ppE=pE
        return guesswave(E)[1],guesswave(E)[2]
                    #once a correct value of E is found it is returned

def normalize (PSI):      #function that normalizes the input
    area=0
    for i in range(1,len(PSI)-1):
        area+=abs(dx*PSI[i])
    area+=abs(0.5*dx*(PSI[0]+PSI[len(PSI)-1]))
    C=1/area
    for j in range(len(PSI)):
        PSI[j]=C*PSI[j]

global L,dx,C,B
                        #set initial values
L=5.                    #length of box is -L to L
dx=0.1
C=30.
B=0.04
psi=1.
plt.figure()

PSI2,X2=guesssearch(3)[0],guesssearch(3)[1]      #calculate 'guess' function (one electron
#PSI2,X2=guesswave(1)[1],guesswave(1)[1]
normalize(PSI2)
for i in range(0,len(PSI2)):
    PSI2[i]=PSI2[i]**2
plt.plot(X2,PSI2,label="Single Particle Probability Density 1st Excited
State",linestyle='dashed')
                            #plot single particle result. this is also used
                            as the guess
E=format(E,".5g")
E1=0                        #initial values of E and Previous E so while loop runs
E2=0
PE1=1
PE2=1

while E1!=PE1 and E2!=PE2:          #SCF method. process loops until calculated
                                    values equal previous values
    PE1=E1                      #set E1 as Previous E1 before calulating new E1
    e=1
    E,PSI1,X2=search(PSI2,X2,e)[0],search(PSI2,X2,e)[1],search(PSI2,X2,e)[2]
```

```
                        #calulate E1 using previous values of PSI, X
    E1=format(E,".5g")  #Reduce E1 to 5 significant figures
    normalize(PSI1)
    PE2=E2
    e=1
    E,PSI2,X2=search(PSI1,X2,e)[0],search(PSI1,X2,e)[1],search(PSI1,X2,e)[2]
                        #calulate E2 using previous values of PSI, X
    E2=format(E,".5g")  #Reduce E2 to 5 significant figures
    normalize(PSI2)
    print("E1=",E1,"eV   E2=",E2,"eV")


PSI=guesssearch(1)[0]
print(len(X2),len(PSI))
normalize(PSI)
#for i in range(0,len(PSI)):
#    PSI[i]=PSI[i]**2
plt.plot(X2,PSI,label="Single Particle Probability Density Ground State",linestyle='dashed

"""
for i in range(0,len(PSI1)):
    PSI1[i]=PSI1[i]**2

for i in range(0,len(PSI2)):
    PSI2[i]=PSI2[i]**2
"""

E1=str(E1)                  #plotting the graph
E2=str(E2)
plt.plot(X2,PSI2,label="Particle 1 in Ground State")
plt.plot(X2,PSI1,label="Particle 2 in 1st Excited State")
#plt.legend(loc=1)
plt.title("Simple Harmonic Oscillator First Excited state")
plt.xlabel("[Angstrom]")
plt.ylabel("$|\psi|^2$")
#plt.yticks([])
plt.show()

plt.figure()            #plotting the interaction potential times probability
                          density for each particle
Int=[]
for k in range (0,len(PSI2)):
    Int.append(VintEq (k,X2,2.5,PSI2))
normalize(Int)
plt.plot(X2,Int)
plt.title("Interaction Potential times Probability Density")
plt.xlabel("[Angstrom]")
plt.ylabel("$|\psi|^2$*Vint")

Int=[]
for k in range (0,len(PSI1)):
    Int.append(VintEq (k,X2,2.5,PSI1))
normalize(Int)
plt.plot(X2,Int)
plt.show()
```