# Calculating Wavefunctions and Spectrum for an Electron in a Gaussian Well

Jonathan Bunton

December 15, 2017

**Abstract**

The calculation of wavefunctions in various potential proves incredibly useful in atomic models, where these wavefunctions represent electron orbitals behaving in various atomic potentials. In this work, we consider an electron within a Gaussian potential function, defined by:

$$V(x) = -De^{-a\frac{x^2+y^2+z^2}{l^2}} \tag{1}$$

If we consider only the region bound by the interval $[-l, l]$ and assume this potential repeats, this problem loosely resembles an atomic lattice structure in three dimensions. To solve this potential for the energy spectrum and wavefunctions, we build our solutions from the basis functions of the form:

$$f_n(x) = \begin{cases} \sqrt{\frac{L}{2}}\sin\left(\frac{n\pi x}{l}\right) & n > 0 \\ \sqrt{\frac{L}{2}}\cos\left(\frac{n\pi x}{l}\right) & n \leq 0 \end{cases} \tag{2}$$

These basis functions form the solutions to the classic infinite square well problem in quantum mechanics, which form a complete orthonormal basis for our wavefunction to be constructed from. Using a specified number of these basis functions (as infinite would be exact but computationally impossible), we construct our Hamiltonian matrix by calculating the expectation values of the potential and kinetic energy for each pair of terms. These expectation values are found with integrals numerically evaluated using Boole's Rule. [2] We then use LAPACK to calculate the eigenvalues and eigenvectors, which correspond to the energies and appropriate weightings of each basis function. [1] From the resulting eigenvectors, we can calculate values for the wavefunctions of interest in our region, and their corresponding eigenvalues indicate the energies. [3]

The resulting spectra indicates that for the parameters $a = 4$, $l = 1 \times 10^{-10}$, and $A = -2.179 \times 10^{-17}$, we receive approximately 20 bound states, building our energy

eigenvectors from only the five lowest energy basis functions. This simulation extracts ten of these states for four of the eight symmetry possibilities, with the other ten appearing as a result of the degeneracy and symmetry of the potential to rotation. For bound states, the wavefunctions have negative energies, and also exhibit higher peaks in at least one dimension in the center of the well, as the probability of the electron being within the well is much higher for bound states. With the energy eigenvalues, this code effectively builds the spectrum for the lowest energy states in this potential.

## Results

For visualization and conceptual understanding, this code is done in both the 1D case, and the 3D case. Due to potential function's spherical symmetry, any of the results from the 1D case have analogs in 3D, making it interesting to discuss. The parameters for the potential function used to create an "interesting" impact on the wavefunctions were $a = 4$, $l = 1 \times 10^{-10}$, and $A = -2.179 \times 10^{-17}$.

### 1-D Case

In one dimension, the potential energy function reduces to a Gaussian in one coordinate:

$$V(x) = De^{-a\frac{x^2}{l^2}}, \qquad x \in [-l, l]$$

The resulting function looks as shown in fig. 1, with a large downward dip centered about the origin. Due to its shape, we can expect the resulting wavefunctions to appear similar to those produced by the quantum harmonic oscillator potential $V(x) = x^2$, which makes since as the first term in the Taylor expansion of $e^x$ is also an $x^2$. Based on the parameters used for our problem, we may see a varied number of bound states. The deeper and wider the potential well is, (i.e. larger $D$ and $a$ terms in our function) the more bound states would be created for an electron within the potential. [3]

In the single dimensional case, there are only two basis function combinations: the sine terms in $f_n(x)$, and the cosine terms in $f_n(x)$. This code splits the Hamiltonian for this system into two smaller Hamiltonians that represent those specific basis functions, and evaluates them individually. This results in solutions in the form of effectively Fourier sine and cosine series, separately. The lowest three wavefunctions in each basis are shown in figs. 2 and 3. The system emits three bound states given the potential parameters, which are characterized by the negative energies shown in the spectra, table 1. The wavefunctions found in the 1D case have analogs in three-dimensions, where various modes (values of $n$ in $f_n(x)$ as in eq.

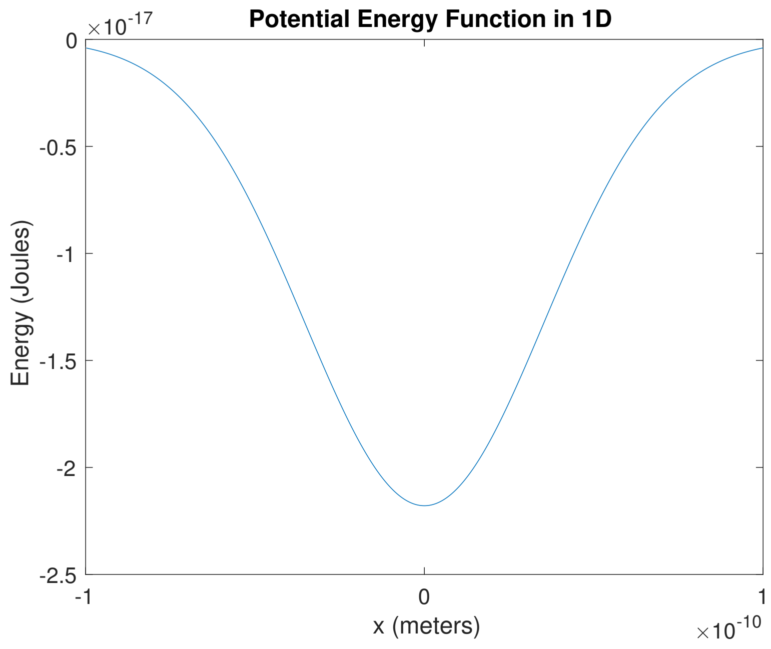

Figure 1: A plot of the potential energy function for the problem in question, in one dimension. For the three dimensional case, this potential function exists across each dimension.

2) will contribute to the construction of a $f_{n,m,o}(x, y, z) = f_n(x)f_m(y)f_o(z)$. These cause various 1D solutions to occur across each dimension in 3D space due to the symmetry in the problem.

## 3D Case

The true problem in question is the three-dimensional case, where the potential takes the form laid out in eq. 1. This results in four-dimensional data, with our wavefunctions being of form $f_{nmo}(x, y, z)$. As in the 1D case we expect our wavefunctions to look reminiscent of the quantum harmonic oscillator solutions, however, these will take varying $n$ values for each $f_n$ in each dimension, causing the axes to take the form of differing solutions to the 1D equation laid out above.

In the 1D case we considered only the two basis functions, however, for three dimensions, each of the three dimensions can take either sine or cosine bases functions from $f_n$, meaning if we attempted to split the Hamiltonian into individual portions, we would need to analyze $2^3 = 8$ basis combination Hamiltonians. Because of the symmetry inherent in the potential, we instead only consider the combinations which produce wavefunctions that are unique under rotations, namely the combinations $sin/sin/sin$, $sin/sin/cos$, $sin/cos/cos$, and $cos/cos/cos$. Because of the nature of the symmetry, there is some inherent degeneracy

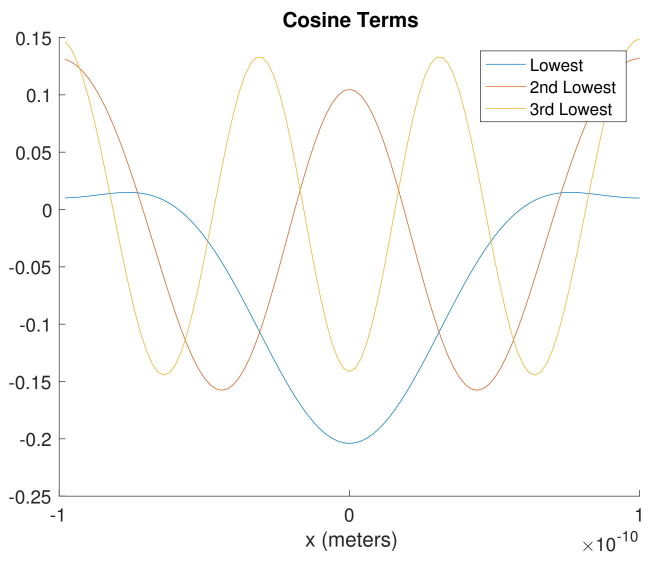

Figure 2: A plot of the three lowest energy cosine basis wavefunctions in one dimension. The two lowest energy wavefunctions (pictured in blue and orange) are bound states, visualized by the maximums and minimums occurring in the center of the potential well, creating a higher probability of the electron being found within the potential well. These bound states are easily identified by negative energies, which are approximately -5.01$\times 10^{-17}$ J and -1.62$\times 10^{-19}$ J.

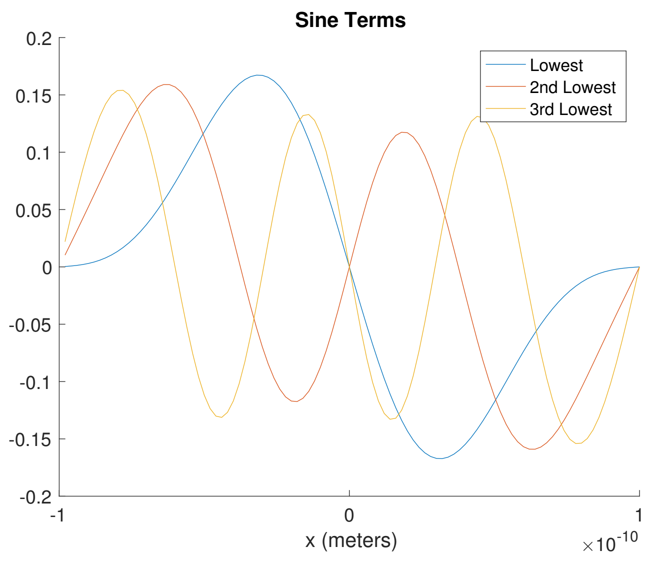

Figure 3: A plot of the first three lowest energy sine basis wavefunctions in one dimension. Only the lowest energy term (pictured in blue) is a bound state, visualized by the wavefunction's maximums and minimums occurring in the center of the potential well, creating the highest probability of the electron's location being within the well. In addition, this bound state has a negative energy of approximately -1.74$\times 10^{-17}$ J.

| Energy Level | Energy (Joules) | Symmetry (sin/cos) |
|---|---|---|
| 1 | $-5.01\times10^{-17}$ | $cos$ |
| 2 | $-1.74\times10^{-17}$ | $sin$ |
| 3 | $-1.62\times10^{-19}$ | $cos$ |
| 4 | $6.97\times10^{-18}$ | $sin$ |
| 5 | $3.02\times10^{-17}$ | $cos$ |
| 6 | $3.60\times10^{-17}$ | $sin$ |
| 7 | $7.78\times10^{-17}$ | $sin$ |
| 8 | $8.98\times10^{-17}$ | $cos$ |
| 9 | $1.33\times10^{-16}$ | $sin$ |
| 10 | $1.74\times10^{-16}$ | $cos$ |

Table 1: A table containing the values for the lowest 10 energy eigenfunctions for the 1D Gaussian potential. The negative energy terms correspond to bound states within the potential.

in the energy spectrum. This is shown in the resulting energy spectrum for the lowest 20 eigenstates, table 2.

To properly visualize the 4-dimensional data, we take 3-dimensional slices of the data to plot. The lowest energy wavefunction is shown with each $x$, $y$, and $z$ held constant at 0 separately in figs. 4, 5, and 6. Bound states exhibit wavefunction deviation toward minimums and maximums in the regions at the center of the Gaussian potential (point $(0, 0, 0)$), whereas free states show no change as they pass over the potential well. In addition, these solutions are periodic, which means we can picture them repeating over all three dimensions to represent an electron in a full 3D lattice. Bound states would indicate that in at least one dimension, the electron is effectively "trapped" in the wells, and will not be shared with the other atoms in the lattice. For free states, the electrons have enough energy to be shared with other atoms within the lattice.

To illustrate the difference between bound and free states, the plots of the 19th wavefunction from table 2 are shown in figs. 7, 8, and 9. In contrast, these wavefunctions display no minimums or maximums around the center of the Gaussian potential, which indicates the electron is "less likely" to be found in this location.

There are ultimately four more combinations of basis trig functions available in each dimension, where the sine and cosine terms are permuted through the $x$, $y$, and $z$ axes, but these will yield identical shapes to those in the initial four symmetries in this paper, transformed only by a rotation. These will also show degeneracy in the energies accordingly. On this basis, we need only examine the first four combinations of basis functions to understand the behavior of the remaining permutations.

| Energy State | Energy (Joules) | Symmetry (x-trig/y-trig/z-trig) |
|---|---|---|
| 1 | -7.15$\times 10^{-31}$ | $sin/cos/cos$ |
| 2 | -3.75$\times 10^{-31}$ | $sin/cos/cos$ |
| 3 | -3.185$\times 10^{-31}$ | $sin/cos/cos$ |
| 4 | -1.74$\times 10^{-31}$ | $sin/cos/cos$ |
| 5 | -1.44$\times 10^{-31}$ | $sin/cos/cos$ |
| 6 | 1.76$\times 10^{-31}$ | $cos/cos/cos$ |
| 7 | 1.20$\times 10^{-31}$ | $cos/cos/cos$ |
| 8 | 1.20$\times 10^{-31}$ | $cos/cos/cos$ |
| 9 | 1.20$\times 10^{-31}$ | $cos/cos/cos$ |
| 10 | 2.41$\times 10^{-17}$ | $cos/cos/cos$ |
| 11 | 2.41$\times 10^{-17}$ | $sin/sin/cos$ |
| 12 | 3.61$\times 10^{-17}$ | $sin/sin/cos$ |
| 13 | 3.61$\times 10^{-17}$ | $sin/sin/sin$ |
| 14 | 6.02$\times 10^{-17}$ | $sin/sin/cos$ |
| 15 | 6.02$\times 10^{-17}$ | $sin/sin/cos$ |
| 16 | 7.23$\times 10^{-17}$ | $sin/sin/cos$ |
| 17 | 7.23$\times 10^{-17}$ | $sin/sin/sin$ |
| 18 | 7.23$\times 10^{-17}$ | $sin/sin/sin$ |
| 19 | 7.23$\times 10^{-17}$ | $sin/sin/sin$ |
| 20 | 1.08$\times 10^{-16}$ | $sin/sin/sin$ |

Table 2: A table describing the energies of the 20 lowest energy wavefunctions with unique (under rotation) wavefunctions. Even among these 20 wavefunctions there is clear degeneracy. In addition, negative energy terms indicate bound states, leading to the conclusion that there are five bound states with unique symmetries for these potential parameters.

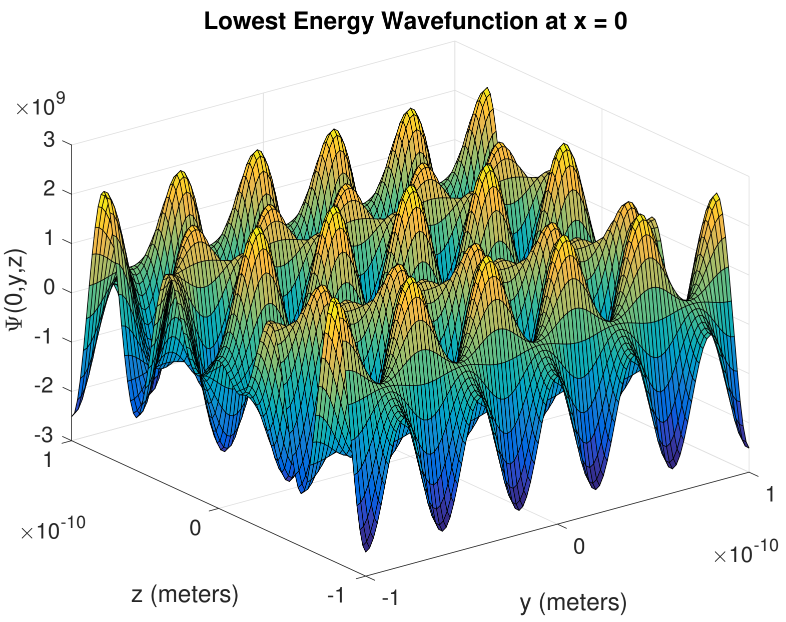

Figure 4: The lowest-energy wavefunction plotted with $x = 0$. The function shows no variation as it nears the center of the Gaussian potential.

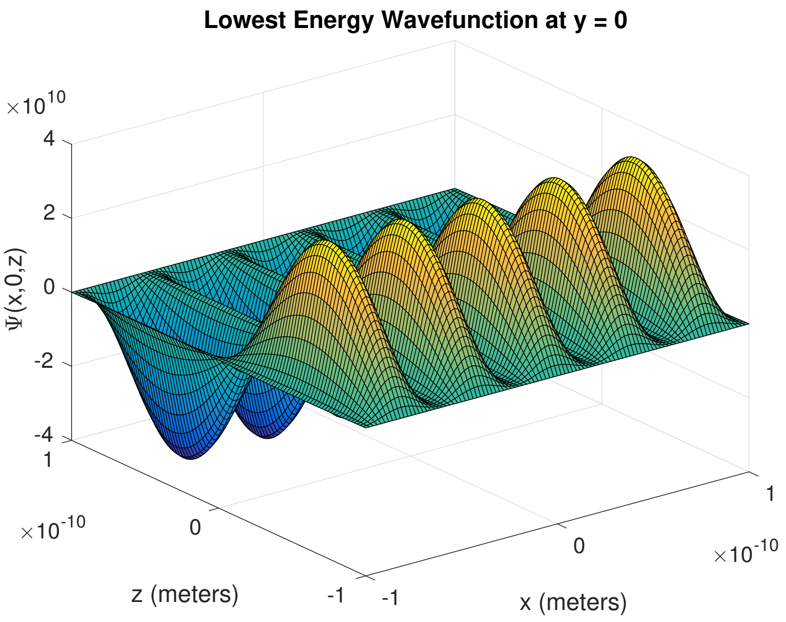

Figure 5: The lowest-energy wavefunction plotted with $y = 0$. The function again shows no variation near the center of the Gaussian potential.

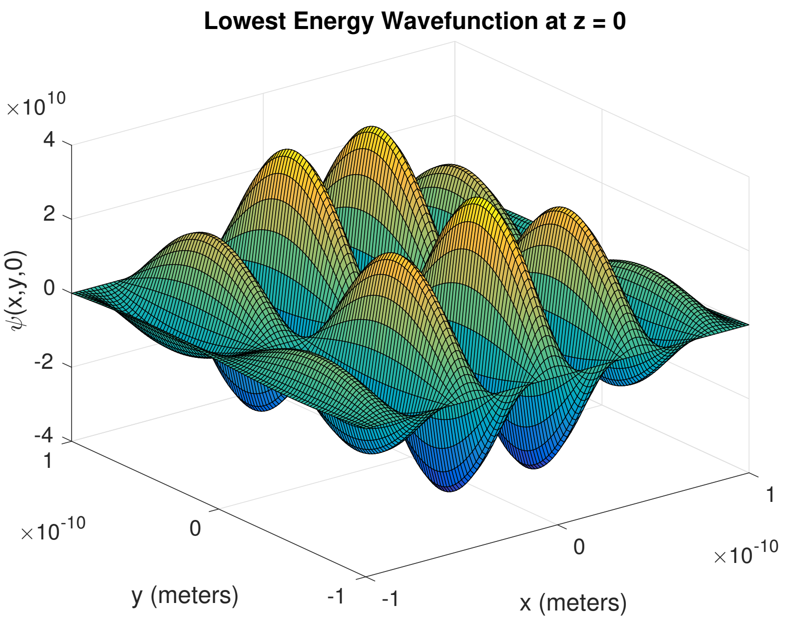

Figure 6: The lowest-energy wavefunction plotted with $z = 0$. The maximum in the wavefunction toward the center of the Gaussian potential, which points to the bound-state nature of this energy state.

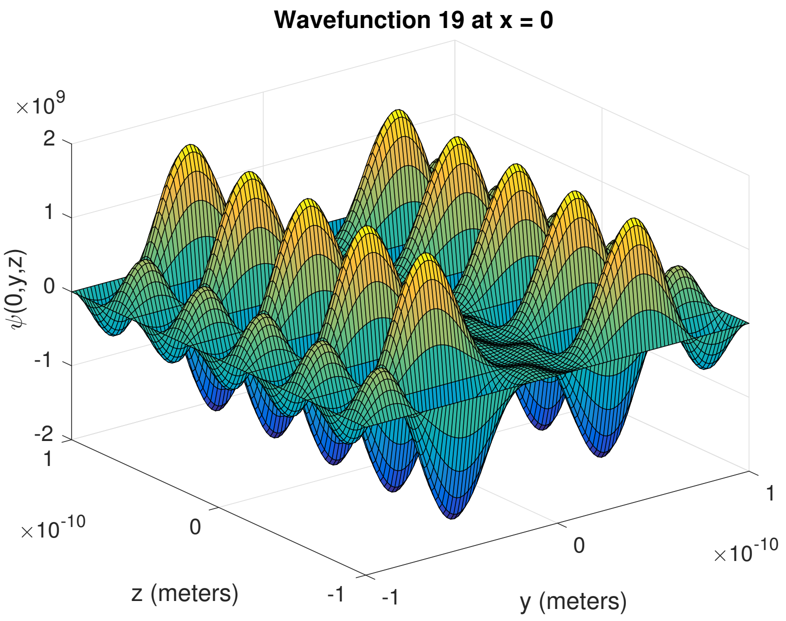

Figure 7: Wavefunction 19 from table 2, plotted at $x = 0$. This state is a free state, with positive energy, which is clear from the lack of wavefunction minimum or maxima at the center of the Gaussian well.

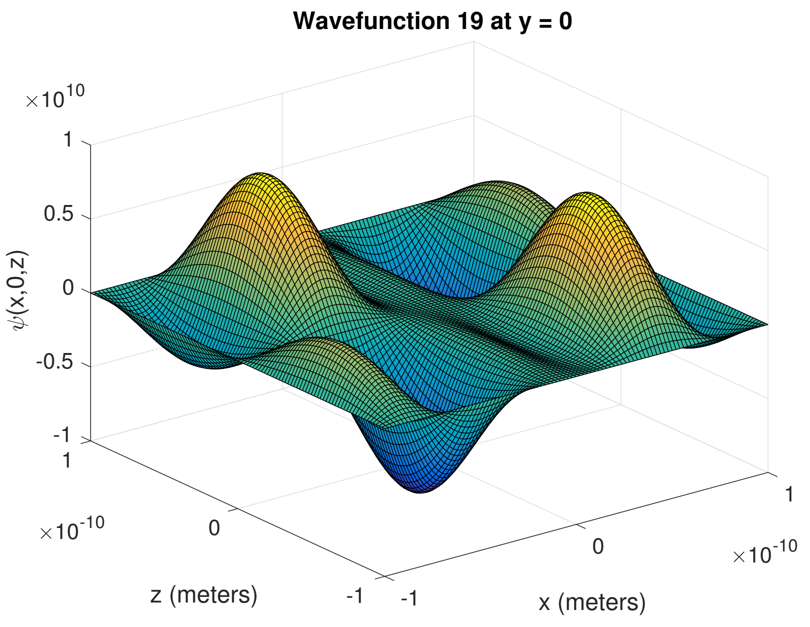

Figure 8: Wavefunction 19 from table 2, plotted at $y = 0$. This state is a free state, with positive energy, which is clear from the lack of variation in the wavefunction at the center of the Gaussian well.

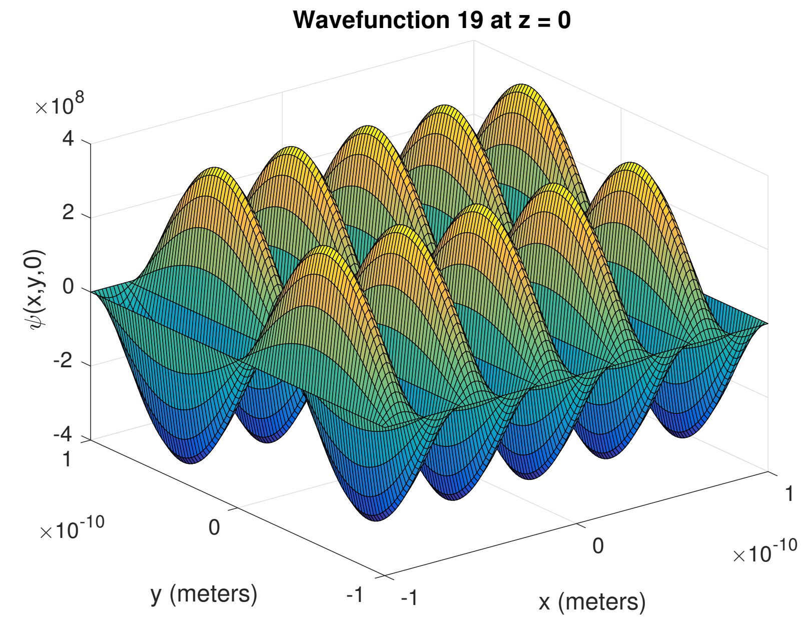

Figure 9: Wavefunction 19 from table 2, plotted at $z = 0$. This state is a free state, with positive energy, which is clear from the lack of variation in the wavefunction at the center of the Gaussian well.

## Discussion

The resulting energy spectrum shows that for the given parameter values for a Gaussian potential in three dimensions, $a = 4$, $l = 1 \times 10^{-10}$, and $A = -2.179 \times 10^{-17}$, there are ten bound states. Building these resulting wavefunctions from a series of only five terms yields the energy spectrum in three dimensions shown in table 2. Looking at the solutions to the one-dimensional case shows clear analogies to the three dimensional functions, where the eigenfunctions make up the functions along each axis. The three-dimensional eigenvalues show some degeneracy, where multiple eigenfunctions can yield identical energies due to ability to assign each basis function's energy to different axes without changing its value.

Bound wavefunctions exhibit maximums and minimums around the center of the Gaussian well, indicating a higher probability of the electron's position being within the well. In contrast, free states show no change over the well, or no minimums and maximums at the center. In the case of an atomic lattice, bound states indicate the electrons have insufficient energy to be shared between the atoms. Conversely, free states indicate the electron may be free to move between the various atoms that comprise the lattice.

Using Boole's Rule to calculate the inner products required to construct the Hamiltonian matrices and LAPACK to calculate the eigenvectors and values of said matrices, we have accurately approximated the energy spectrum for an electron in our specified Gaussian potential well, and also retrieved approximations for the wavefunctions in the lowest twenty eigenstates.

## Acknowledgements

I would like to thank Dr. Justin Oelgoetz for assistance formulating the simulation code, and for providing insightful explanations and troubleshooting along the way.

All graphs and plots made in this paper were produced using MATLAB computational software.

## References

[1] E. Anderson, Z. Bai, C. Bischof, S. Blackford, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, and D. Sorensen. *LAPACK Users' Guide*. Society for Industrial and Applied Mathematics, Philadelphia, PA, third edition, 1999.

[2] George Boole. *Calculus of Finite Differences*. Chelsea Pub. Co, 5th edition, 1970.

[3] David J. Griffiths. *Introduction to Quantum Mechanics*. Pearson, 2nd edition, 2005.

## Appendix

Attached is the source code for this project, written in Fortran.

### Main

```
program quantumwell
use constants
use integrateme
implicit none
real*8, allocatable, dimension(:,:,:,:) :: wfns
real*8, allocatable, dimension(:,:) :: H
real*8, allocatable, dimension(:,:) :: oneDeewfns
real*8, allocatable, dimension(:) :: x
real*8 :: dx
integer, parameter :: resolution = 100
integer :: k1x, k1y, k1z, k2x, k2y, k2z
integer :: m1, m2, m3, n, a, b, i
character:: oned

! LAPACK DSYEV ARGUMENTS
integer :: numel, info
real*8, allocatable, dimension(:) :: eigenvalues, work
integer :: lwork
character*1 :: jobz = 'V', uplo = 'U'

print*, '1D?'
read(*,*) oned

if(oned == 'n') then

print*,'Starting sin/sin/sin terms...'

!-----------------------3D ATTEMPT--------------------!
! x: sin
! y: sin
! z: sin
```

```
allocate(H(nmax**3,nmax**3), wfns(elevels,0:resolution,0:
   resolution,0:resolution))

do k1x = 1,nmax
   do k1y = 1, nmax
      do k1z = 1, nmax
         do k2x = k1x, nmax
            do k2y = k1y, nmax
               do k2z = k1z, nmax
               a = (k1x-1)*nmax**2 + (k1y-1)*nmax + k1z
               b = (k2x-1)*nmax**2 + (k2y-1)*nmax + k2z

               H(a,b) = integral(k1x,k2x)*integral(k1y,k2y)*
                  integral(k1z,k2z)*(2/L)**3
               H(b,a) = H(a,b)
               end do
            end do
         end do
      H(a,a) = ((pi*hbar/l)**2)/m *(k1x**2 + k1y**2 + k1z**2) + H(
         a,a)
      end do
   end do
end do

numel = size(H,1)
lwork = 3*numel
allocate(eigenvalues(numel), work(lwork),x(0:resolution))
call dsyev(jobz,uplo,numel,H,numel,eigenvalues,work,lwork,info)
wfns = 0.0d0

dx = 2*L/(size(x)-1)
do i = 0,resolution
x(i) = -L + float(i)*dx
end do
print*, dx
```

```
do n = 1,elevels
    do k1x = 1,nmax
        do k1y = 1,nmax
            do k1z = 1,nmax
            a = (k1x-1)*nmax**2 + (k1y-1)*nmax + k1z
                do m1 = 0, resolution
                    do m2 = 0, resolution
                         do m3 = 0, resolution
                         wfns(n,m1,m2,m3) = wfns(n,m1,m2,m3) + (2/l)
                             **(3/2)*H(n,a)*sin(k1x*pi/l*x(m1))&
                            *sin(k1y*pi*x(m2)/l)*sin(k1z*pi*x(m3)/l)
                         end do
                    end do
                end do
            end do
        end do
    end do
end do

open(unit=100,file='sinsinsinvectors.txt')
print*,"Writing␣sin/sin/sin␣data..."
do m1 = 1,elevels
    do k1x = 0,resolution
        do k1y = 0,resolution
            do k1z = 0, resolution
            !Columns: energy eigenvalue, i, j, k, psi(x,y,z)
            write(100,*) eigenvalues(m1), k1x, k1y, k1z, wfns(m1,k1x,
                k1y,k1z)
            end do
        end do
    end do
end do
deallocate(H,eigenvalues,work,wfns)
```

```
!------------------------3D ATTEMPT----------------------!
! x: sin
! y: sin
! z: cos
print*,'Starting sin/sin/cos terms...'
allocate(H((nmax+1)*nmax**2,(nmax+1)*nmax**2), wfns(elevels,0:
   resolution,0:resolution,0:resolution))

do k1x = 1,nmax
   do k1y = 1, nmax
      do k1z = 0, -nmax, -1
         do k2x = k1x, nmax
            do k2y = k1y, nmax
               do k2z = k1z, -nmax,-1
               a = (k1x-1)*nmax*(nmax+1) + (k1y-1)*(nmax+1) - k1z
                  + 1
               b = (k2x-1)*nmax*(nmax+1) + (k2y-1)*(nmax+1) - k2z
                  + 1

               H(a,b) = integral(k1x,k2x)*integral(k1y,k2y)*
                  integral(k1z,k2z)*(2/L)**3
               H(b,a) = H(a,b)
               end do
            end do
         end do
      H(a,a) = ((pi*hbar/l)**2)/m *(k1x**2 + k1y**2 + k1z**2) + H(
         a,a)
      end do
   end do
end do

numel = size(H,1)
lwork = 3*numel
allocate(eigenvalues(numel), work(lwork))
call dsyev(jobz,uplo,numel,H,numel,eigenvalues,work,lwork,info)
wfns = 0.0d0
```

```
do n = 1,elevels
   do k1x = 1,nmax
      do k1y = 1,nmax
         do k1z = 0,-nmax,-1
         a = (k1x-1)*nmax*(nmax+1) + (k1y-1)*(nmax+1) - k1z + 1
            do m1 = 0, resolution
               do m2 = 0, resolution
                   do m3 = 0, resolution
                   wfns(n,m1,m2,m3) = wfns(n,m1,m2,m3) + (2/l)
                      **(3/2)*H(n,a)*sin(k1x*pi/l*x(m1))&
                     *sin(k1y*pi*x(m2)/l)*cos(k1z*pi*x(m3)/l)
                   end do
               end do
            end do
         end do
      end do
   end do
end do

open(unit=101,file='sinsincosvectors.txt')
print*,"Writing␣sin/sin/cos␣data..."
do m1 = 1,elevels
   do k1x = 0,resolution
      do k1y = 0,resolution
         do k1z = 0, resolution
         !Columns: energy eigenvalue, i, j, k, psi(x,y,z)
         write(101,*) eigenvalues(m1), k1x, k1y, k1z, wfns(m1,k1x,
            k1y,k1z)
         end do
      end do
   end do
end do
close(101)
deallocate(H,eigenvalues,work,wfns)
```

```
!------------------------3D ATTEMPT------------------------!
! x: sin
! y: cos
! z: cos
print*,'Starting sin/cos/cos terms...'
allocate(H(((nmax+1)**2)*nmax,((nmax+1)**2)*nmax), wfns(elevels,0:
   resolution,0:resolution,0:resolution))

do k1x = 1,nmax
   do k1y = 0, -nmax, -1
      do k1z = 0, -nmax, -1
         do k2x = k1x, nmax
            do k2y = k1y, -nmax,-1
               do k2z = k1z, -nmax,-1
               a = (k1x-1)*(nmax+1)**2 + (-k1y)*(nmax) - k1z + 1
               b = (k2x-1)*(nmax+1)**2 + (-k2y)*(nmax) - k2z + 1

               H(a,b) = integral(k1x,k2x)*integral(k1y,k2y)*
                  integral(k1z,k2z)*(2/L)**3
               H(b,a) = H(a,b)
               end do
            end do
         end do
      H(a,a) = ((pi*hbar/l)**2)/m *(k1x**2 + k1y**2 + k1z**2) + H(
         a,a)
      end do
   end do
end do

numel = size(H,1)
lwork = 3*numel
allocate(eigenvalues(numel), work(lwork))
call dsyev(jobz,uplo,numel,H,numel,eigenvalues,work,lwork,info)
wfns = 0.0d0
```

```
do n = 1,elevels
    do k1x = 1,nmax
        do k1y = 0,-nmax,-1
            do k1z = 0,-nmax,-1
            a = (k1x-1)*(nmax+1)**2 + (-k1y)*(nmax) - k1z + 1
                do m1 = 0, resolution
                    do m2 = 0, resolution
                         do m3 = 0, resolution
                         wfns(n,m1,m2,m3) = wfns(n,m1,m2,m3) + (2/l)
                             **(3/2)*H(n,a)*sin(k1x*pi/l*x(m1))&
                           *cos(k1y*pi*x(m2)/l)*cos(k1z*pi*x(m3)/l)
                         end do
                    end do
                end do
            end do
        end do
    end do
end do

open(unit=102,file='sincoscosvectors.txt')
print*,"Writing sin/cos/cos data..."
do m1 = 1,elevels
    do k1x = 0,resolution
        do k1y = 0,resolution
            do k1z = 0, resolution
            !Columns: energy eigenvalue, i, j, k, psi(x,y,z)
            write(102,*) eigenvalues(m1), k1x, k1y, k1z, wfns(m1,k1x,
                k1y,k1z)
            end do
        end do
    end do
end do
close(102)
deallocate(H,eigenvalues,work,wfns)
```

```
!----------------3D ATTEMPT---------------!
! x: cos
! y: cos
! z: cos
print*,'Starting cos/cos/cos terms...'
allocate(H((nmax+1)**3,(nmax+1)**3), wfns(elevels,0:resolution,0:
   resolution,0:resolution))

do k1x = 0,-nmax,-1
   do k1y = 0, -nmax, -1
      do k1z = 0, -nmax, -1
         do k2x = k1x, -nmax,-1
            do k2y = k1y, -nmax,-1
               do k2z = k1z, -nmax,-1
               a = (-k1x)*(nmax+1)**2 + (-k1y)*(nmax+1) - k1z + 1
               b = (-k2x)*(nmax+1)**2 + (-k2y)*(nmax+1) - k2z + 1

               H(a,b) = integral(k1x,k2x)*integral(k1y,k2y)*
                  integral(k1z,k2z)*(2/L)**3
               H(b,a) = H(a,b)
               end do
            end do
         end do
      H(a,a) = ((pi*hbar/l)**2)/m *(k1x**2 + k1y**2 + k1z**2) + H(
         a,a)
      end do
   end do
end do

numel = size(H,1)
lwork = 3*numel
allocate(eigenvalues(numel), work(lwork))
call dsyev(jobz,uplo,numel,H,numel,eigenvalues,work,lwork,info)
```

```
wfns = 0.0d0


do n = 1,elevels
   do k1x = 0,-nmax,-1
      do k1y = 0,-nmax,-1
         do k1z = 0,-nmax,-1
         a = (-k1x)*(nmax+1)**2 + (-k1y)*(nmax+1) - k1z + 1
            do m1 = 0, resolution
               do m2 = 0, resolution
                   do m3 = 0, resolution
                   wfns(n,m1,m2,m3) = wfns(n,m1,m2,m3) + (2/l)
                       **(3/2)*H(n,a)*cos(k1x*pi/l*x(m1))&
                     *cos(k1y*pi*x(m2)/l)*cos(k1z*pi*x(m3)/l)
                   end do
               end do
            end do
         end do
      end do
   end do
end do

open(unit=104,file='coscoscosvectors.txt')
print*,"Writing cos/cos/cos data..."
do m1 = 1,elevels
   do k1x = 0,resolution
      do k1y = 0,resolution
         do k1z = 0, resolution
         !Columns: energy eigenvalue, i, j, k, psi(x,y,z)
         write(104,*) eigenvalues(m1), k1x, k1y, k1z, wfns(m1,k1x,
            k1y,k1z)
         end do
      end do
   end do
end do
close(104)
```

```
deallocate(H,eigenvalues,work,wfns)

deallocate(x)


else

!--------------------1 - DIMENSIONAL CASE--------------------!
!1-D TRY

!Sin Terms
allocate(H(nmax,nmax), oneDeewfns(elevels,resolution))
H = 0.0D0
oneDeewfns = 0.0D0

do k1z = 1, nmax
   do k2z = k1z, nmax
   a = k1z
   b = k2z
   H(a,b) = integral(k1z,k2z)*(2.0/L)
   H(b,a) = H(a,b)
   end do
   H(a,a) = ((pi*hbar*k1z/l)**2)/(2*m) + H(a,a)
end do

numel = size(H,1)
lwork = 3*numel

open(unit=800, file='before.txt')
do a = 1,nmax
   write(800,*) (H(a,b), b = 1,nmax)
end do
close(800)

allocate(eigenvalues(numel), work(lwork),x(resolution))
call dsyev(jobz,uplo,numel,H,numel,eigenvalues,work,lwork,info)
```

```
onedeewfns = 0.0d0
if(info .ne. 0) then
print*,'LAPACK error:', info
endif

dx = 2*L/resolution
do i = 1,resolution
x(i) = -L + float(i)*dx
end do

open(unit=200, file='eigendata.txt')
do a = 1,nmax
   write(200,*) (H(a,b), b = 1,nmax)
end do
close(200)

do n = 1,elevels
   do k1z = 1,nmax
        do m3 = 1,resolution
        oneDeewfns(n,m3) = oneDeewfns(n,m3) + sqrt(2.0/l)*H(n,k1z)*
           sin(dble(k1z)*pi*x(m3)/l)
        end do
   end do
   oneDeewfns(n,:) = oneDeewfns(n,:)/(norm2(oneDeewfns(n,:)))
end do

open(unit=100,file='sinvectors.txt')
print*,"Writing sin data..."
do m1 = 1,elevels
   do k1z = 1, resolution
   !Columns: energy eigenvalue, i, j, k, psi(x,y,z)
   write(100,*) m1, eigenvalues(m1),  k1z, x(k1z), oneDeewfns(m1,
      k1z)
   end do
end do
deallocate(H,eigenvalues,work,onedeewfns,x)
```

```
close(100)



!Cos terms
allocate(H(0:nmax,0:nmax), oneDeewfns(elevels,resolution))

do k1z = 0, -nmax, -1
   do k2z = k1z, -nmax,-1
   a = abs(k1z)
   b = abs(k2z)
   H(a,b) = integral(k1z,k2z)*(2.0/L)
   H(b,a) = H(a,b)
   end do
end do

do k1z = 0,-nmax, -1
   a =  abs(k1z)
   H(a,a) = ((pi*hbar*k1z/l)**2.0)/m + H(a,a)
end do

numel = size(H,1)
lwork = 3*numel
open(unit=200, file='eigendata.txt')
do a = 0,nmax
   write(200,*) (H(a,b), b = 0,nmax)
end do
close(200)

allocate(eigenvalues(numel), work(lwork),x(resolution))
call dsyev(jobz,uplo,numel,H,numel,eigenvalues,work,lwork,info)

if(info .ne. 0) then
print*,'LAPACK␣error:', info
endif
```

```
onedeewfns = 0.0d0
dx = 2*L/resolution

do i = 1,resolution
x(i) = -L + float(i)*dx
end do



do n = 1,elevels
   do k1z = 0,-nmax,-1
        do m3 = 1,resolution
        oneDeewfns(n,m3) = oneDeewfns(n,m3) + (2.0/l)**(1/2)*H(n,
           abs(k1z))*cos(dble(k1z)*pi*x(m3)/l)
        end do
   end do
   oneDeewfns(n,:) = oneDeewfns(n,:)/(norm2(oneDeewfns(n,:)))
end do

open(unit=400,file='cosvectors.txt')
print*,"Writing␣cos␣data..."
do m1 = 1,elevels
   do k1z = 1, resolution
   !Columns: energy eigenvalue, i, j, k, psi(x,y,z)
   write(400,*) m1, eigenvalues(m1),  k1z, x(k1z), oneDeewfns(m1,
      k1z)
   end do
end do

open(unit=500, file='potential.txt')
do i = 1,resolution
   write(500,*) x(i), depth*dexp(-sd*(x(i)/l)**2)
end do
close(500)

deallocate(H,eigenvalues,work,onedeewfns,x)
```

```
end if
close(400)
open(unit=300, file='info.txt')
write(300,*) nmax
write(300,*) depth
write(300,*) sd
write(300,*) l
write(300,*) resolution
write(300,*) dx
close(300)
end program quantumwell
```

## Integration Module

```
module integrateme
use constants
implicit none
contains

function integral(n, m)
real*8 :: integral
integer, intent(in) :: n, m
real*8 :: x, dx
real*8, dimension(5) :: fx
integer :: nsteps, i, j

if ((n .gt. 0) .and. (m .le. 0)) then
   integral = 0.0d0
   return
else if ((m .gt. 0)  .and. (n .le. 0)) then
   integral = 0.0d0
   return
end if

nsteps = 4*2*nmax
dx = l/(4*dble(nsteps))
```

```
integral = 0.0
x = 0.0

if (n .gt. 0) then
   do i = 1,nsteps
      x = dble(i-1)*4.0*dx
      do j = 1,5
      fx(j) = depth*dsin(dble(n)*pi*x/l)*dsin(dble(m)*pi*x/l)*dexp
         (-sd*(x/l)**2)
      x = x + dx
      end do
   integral = integral + (4.0/45.0)*dx*(7.0*fx(1) + 32.0*fx(2) +
      12.0*fx(3) + 32.0*fx(4) + 7.0*fx(5))
   end do
else
   do i = 1,nsteps
      x = dble(i-1)*4.0*dx
      do j = 1,5
      fx(j) = depth*dcos(dble(n)*pi*x/l)*dcos(dble(m)*pi*x/l)*dexp
         (-sd*(x/l)**2)
      x = x + dx
      end do
   integral = integral + (4.0/45.0)*dx*(7.0*fx(1) + 32.0*fx(2) +
      12.0*fx(3) + 32.0*fx(4) + 7.0*fx(5))
   end do
end if
return
end function

end module integrateme
```

## Constants Module

```
module constants

integer, parameter :: nmax = 5 ! number of basis function terms to
    include
```

```
integer, parameter :: elevels = 10
real*8, parameter :: pi = dacos(-1.0d0)
real*8, parameter :: sd = 4!0.1!1000!1D10 ! well width
real*8, parameter :: depth = -2.179e-17!-3! -2.179D-5 ! well depth
real*8, parameter :: l = 1D-10    ! period of basis functions and
    width of well
real*8, parameter :: hbar = 1.0545718D-34
real*8, parameter :: m = 9.1094D-31

!V(x) = depth*exp(-sd*(x/l)**2)

end module constants
```

## MATLAB Plotting Script

Code can be run while MATLAB is in the directory containing result files and will create a .gif file of positions, total energy and momentum plots, an initial velocity sampling histogram, average displacement magnitude vs. time, and initial and final scatter plots of position and save them to the directory.

```
% IMPORT LAST RUN DETAILS
info = importdata('info.txt');
nmax = info(1);
depth = info(2);
sd = info(3);
l = info(4);
resolution = info(5);
%dx = info(6);
dx = 2E-12;
elevels = 5;
[x y] = meshgrid(-l:dx:l);



% IMPORT SIN/SIN/SIN TERMS
A = importdata('sinsinsinvectors.txt');
sinsinsin = zeros(resolution+1,resolution+1,resolution+1,elevels);
sinsinsinenergies = zeros(3,1);
count = 1;
```

```
for elevel = 1:elevels
    sinsinsinenergies(elevel) = A(count,1);
for i = 1:resolution+1
    for j = 1:resolution+1
        for k = 1:resolution+1
            sinsinsin(A(count,2)+1,A(count,3)+1,A(count,4)+1,
               elevel) = A(count,5);
            count = count+1;
        end
    end
end
end

%IMPORT SIN/SIN/COS TERMS
A = importdata('sinsincosvectors.txt');
sinsincos = zeros(resolution+1,resolution+1,resolution+1,elevels);
sinsincosenergies = zeros(3,1);
count = 1;
for elevel = 1:elevels
    sinsincosenergies(elevel) = A(count,1);
for i = 1:resolution+1
    for j = 1:resolution+1
        for k = 1:resolution+1
            sinsincos(A(count,2)+1,A(count,3)+1,A(count,4)+1,
               elevel) = A(count,5);
            count = count+1;
        end
    end
end
end

%IMPORT SIN/COS/COS TERMS
A = importdata('sincoscosvectors.txt');
sincoscos = zeros(resolution+1,resolution+1,resolution+1,elevels);
sincoscosenergies = zeros(3,1);
count = 1;
```

```
for elevel = 1:elevels
    sincoscosenergies(elevel) = A(count,1);
for i = 1:resolution+1
    for j = 1:resolution+1
        for k = 1:resolution+1
            sincoscos(A(count,2)+1,A(count,3)+1,A(count,4)+1,
               elevel) = A(count,5);
            count = count+1;
        end
    end
end
end

%IMPORT COS/COS/COS TERMS
A = importdata('coscoscosvectors.txt');
coscoscos = zeros(resolution+1,resolution+1,resolution+1,elevels);
coscoscosenergies = zeros(3,1);
count = 1;
for elevel = 1:elevels
    coscoscosenergies(elevel) = A(count,1);
for i = 1:resolution+1
    for j = 1:resolution+1
        for k = 1:resolution+1
            coscoscos(A(count,2)+1,A(count,3)+1,A(count,4)+1,
               elevel) = A(count,5);
            count = count+1;
        end
    end
end
end
```