Guru Jambheshwar University Science and Technology

Department of Physics

Simulation of Electron Orbit in Hydrogen Molecule Ion

Project of Computational Physics

Submitted to : Dr. Sahil Saini

Submitted By

Sourah 220070720010 , Rimit ..12 , Sandeep ..18

November,2023

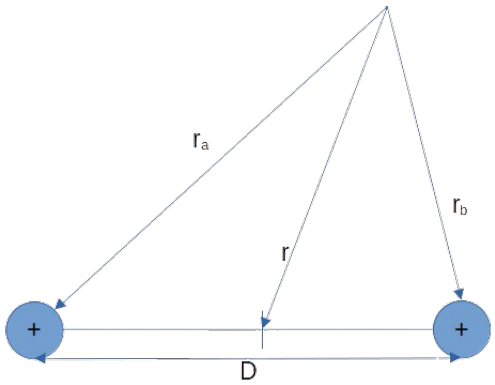
Abstract

This is a our Computational Physics project in which we simulate the Electron Orbit for the ground state of Hydrogen Molecule Ion (HMI). This can be done numerically solving the radial and angular part of the Schrödinger equation in Elliptical Spherical coordinate for the HMI. And processing all the data and join the again radial and angular part, and getting the orbit graph.

Hydrogen Molecule Ion

1.1 Introduction

The HMI is the simplest molecule. It have two proton and one electron, and these protons are bonded with the covalent bond. So the electron is shared between the two proton and the electron have influence of both of the proton by the Coulombian force.



Hydrogen Molecule Ion

Here D is inter nuclear distance, and we easily calculate the potential for electron by using the superposition principle and we can get the Schrödinger wave equation for the electron and its is

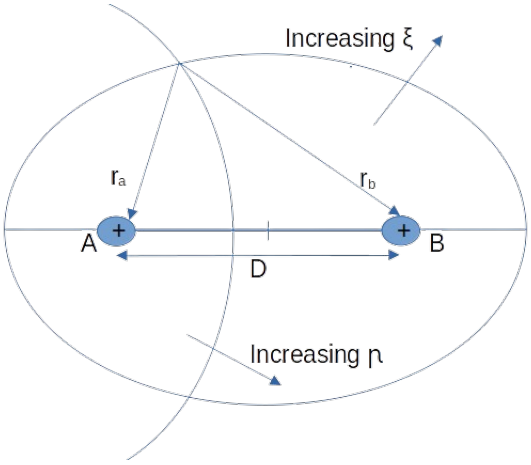
$$(-\boldsymbol{\nabla^2} - \frac{\boldsymbol{2}}{\boldsymbol{r_a}^{\boldsymbol{2}}} - \frac{\boldsymbol{2}}{\boldsymbol{r_b}^{\boldsymbol{2}}})\boldsymbol{\psi(r)} = \boldsymbol{E\psi(r)} \tag{1.1}$$

where r_a and r_b are the distances from the electron to the two protons. Following Slater, we use a system of atomic units in which the unit of energy is the Rydberg or 13.6 eV; the unit of length is the radius of first Bohr orbit, 0.529 Å. Here to solve for wave function we need to separate

the wave function, and then solve for wave function. And we can see that we can't able to separate the radial and angular part of the wave function, and for solve this we need to change the coordinates system then we change into the elliptical spherical coordinates.

1.2 Wave function in Elliptical Coordinate System

It is necessary to introduce prolate confocal elliptic coordinates, $\boldsymbol{\xi = (r_a + r_b)/D, \eta = (r_a – r_a)/D}$, and $\boldsymbol{\phi}$, which is the angle of rotation about the nuclear axis. The inverse relations are $\boldsymbol{r_a = D(\xi + \eta)/2}$ and $\boldsymbol{r_b = D(\xi – \eta)/2}$; we note that $\boldsymbol{2/r_a = 4/[D(\xi + \eta)], 2/r_b = 4/[D(\xi – \eta)]}$. The meaning of these definitions becomes clear when we examine elliptic coordinates in a plane. The lines of constant $\boldsymbol{\xi}$ are ellipses, which share foci A and B. The lines of constant $\boldsymbol{\eta}$ are hyperbolas, again with A and B as foci. These two families form an orthogonal system of curves (see figure below). The variable $\boldsymbol{\xi}$ plays a role analogous to $\boldsymbol{r}$, the distance to the origin, in the usual polar coordinate system. When $\boldsymbol{\eta}$ increases, the point $\boldsymbol{(\xi, \eta)}$ moves around the origin, so that this parameter is similar to the quantity $\boldsymbol{cos(\theta)}$ in polar coordinates. The domains of each variable are $\boldsymbol{-1 \leq \eta \leq 1 and 1 \leq \xi \leq \infty}$.



Wave Function after Transformation:

$$\begin{aligned}&\frac{\partial}{\partial\xi}\left[(\xi^2-1)\frac{\partial\psi}{\partial\xi}\right]+\frac{\partial}{\partial\eta}\left[(1-\eta^2)\frac{\partial\psi}{\partial\xi}\right]\\&+\left[\frac{1}{\xi^2-1}+\frac{1}{1-\eta^2}\right]\frac{\partial^2\psi}{\partial\phi^2}+\left[c^2(\xi^2-\eta^2)+2D\xi\right]\psi=0\end{aligned}\tag{1.2}$$

with $c=\frac{1}{2}D^2E$
Here $\psi=R(\xi)S(\eta)e^{im\phi}$ and by analogy with the atomic case, we have assumed an explicit form of the ψ dependence and introduced a so-called separation constant,m. Now we can remove the ϕ dependence by putting the value of $\frac{\partial^2\psi}{\partial\phi^2}=-m^2\psi$ in equation (1.2) and we get the equation (1.3)

$$\begin{aligned}&\left\{\frac{1}{R}\left[(\xi^2-1)R'\right]'-\frac{m^2}{\xi^2-1}+2D\xi+c^2\xi^2\right\}\\&+\left\{\frac{1}{S}\left[(1-\eta^2)S'\right]'-\frac{m^2}{1-\eta^2}-c^2\eta^2\right\}=0\end{aligned}\tag{1.3}$$

Here we can see that the radial $R(\xi)$ and angular $S(\eta)$ part are in first half and second half of the equation and there sum is equal to the zero it happened only when the both numerically equal to constant with different sign and let the constant is Λ and now we can separate the equations an get the separated equation ready to solution.
Radial equation is

$$\left\{\frac{1}{R}\left[(\xi^2-1)R'\right]'-\frac{m^2}{\xi^2-1}+2D\xi+c^2\xi^2-\Lambda\right\}=0\tag{1.4}$$

After some rearrangement in equation (1.4) and simple solving the derivative we get this equation (1.5)

$$R''=\frac{1}{\xi^2-1}\left\{\left[\frac{m^2}{\xi^2-1}-2D\xi-c^2\xi^2+\Lambda\right]S-2\xi R'\right\}\tag{1.5}$$

And this is the equation (1.5) we used to solve for the radial part of the wave function
And Angular equation is

$$\frac{1}{S}\left[(1-\eta^2)S'\right]'+\left[\Lambda-\frac{m^2}{1-\eta^2}-c^2\eta^2\right]=0\tag{1.6}$$

After some rearrangement in equation (1.6) and simple solving the derivative we get this equation (1.7)

$$\boldsymbol{S'' = \frac{1}{1-\eta^2}\left\{\left[\frac{m^2}{1-\eta^2} + c^2\eta^2 - \Lambda\right] S + 2\eta S'\right\}} \tag{1.7}$$

Here the equation (1.5) and (1.7) is used for solve the radial and angular part of the wave function using Boundary conditions.

1.2.1 Boundary Conditions and Constants Relations

As we see the equation (1.5) and (1.7) are the two second order differential equation with four constants $\boldsymbol{(m, D, E, \Lambda)}$ and for solving the equation we need the boundary condition and the value for constants.
Here $\boldsymbol{D}$ is inter nuclear distance we can take any value by our self, but we need to find its value where the system is most stable and for $\boldsymbol{m}$ we can take it equal to zero because we are finding the orbit for ground state and this is taken from the Hydrogen Atomic Model.And the finding the value of energy is part of the problem, and we're done it in programming.
And we have two relations between $\boldsymbol{c^2}$ and $\boldsymbol{\Lambda}$ value it given as

$$\boldsymbol{\Lambda = 0.3127477 * c2 - 0.0231669 * (c^2)^2 - 0.0005110 * (c^2)^3 - 0.0000045 * (c^2)^4} \tag{1.8}$$

$$\boldsymbol{\Lambda = 2 + 0.6043499 * c^2 - 0.006188 * (c^2)^2 - 0.0000589 * (c^2)^3} \tag{1.9}$$

And we use equation (1.9) to solve the problem because it gives good result for lower value of energy. And this relation is coming form the solving angular part for case when protons are very close to each other.

Boundary Condition for Angular Part: As we see in angular part of the wave equation (in 1.7) and we get the solution of this equation like that

$$\boldsymbol{S(\eta) = (1-\eta^2)^{\frac{m}{2}} f(\eta)} \tag{1.10}$$

And $\boldsymbol{f(\eta)}$ is an unknown function , and we put the value of $\boldsymbol{S(\eta)}$ in the equation (1.6) and get this equation

$$\boldsymbol{(1-\eta^2)f'' - 2(m+1)\eta f' - [m(m+1) - \Lambda - c^2\eta^2] = 0} \tag{1.11}$$

And we also know that the $-1 \leq \eta \leq 1$ and $-1 \leq S(\eta) \leq 1$ because $S(\eta)$ have similar behavior as $cos(\theta)$ we discuss it above. Then from this we can say that $f(1) = 1$ and $f(-1) = \overset{+}{-} 1$ and this depended upon that wave function is symmetric or anti-symmetric (we talk about it later) and putting the both value in equation (1.11) we get the $f'(1)$ and $f'(-1)$ such as

$$f'(1) = \frac{m(m+1) - \Lambda + c^2}{2(m+1)} f(1) \quad f'(-1) = \overset{+}{-} \frac{m(m+1) - \Lambda + c^2}{2(m+1)} f(1) \tag{1.12}$$

From the text above we get the boundary condition for the Angular part of the wave equation and we solve for the symmetric wave function, so we use $f(-1) = 1$ and we solve for ground state, so we take $m = 0$ in programs.

Boundary Condition for Radial Part : As we in upper section we solve the Angular equation same as we solve the Radial equation we can write the radial equation as such

$$R(\xi) = g(\xi)(\xi^2 - 1)^{\frac{m}{2}} \tag{1.13}$$

Here $g(\xi)$ is unknown function and when we put this value in the radial equation (1.4) we get

$$(\xi^2 - 1)g'' + 2(m+1)\xi g' + [2D\xi + c^2\xi^2 + m(m+1) - \Lambda]g = 0 \tag{1.14}$$

we also know $1 \leq \xi \leq \infty$, so the lowest value of ξ is 1 and for $\xi = 1$ we take the $g(1) = 1$ because the $g(\xi)$ have maximum value at $\xi = 1$ and for normalization we can take it equal to 1. And we put these value in equation(1.14) we get

$$g'(1) = -\frac{2D + c^2 + m(m+1) - \Lambda}{2(m+1)} \tag{1.15}$$

And from here we get all the boundary condition and constant value for solve the problem.

1.3 Bonding and Anti-Bonding of the HMI

In upper section we encounter a term bonding and anti-bonding, in this section we explain what is bonding and anti-bonding in HMI.

As we know that HMI is made up of two hydrogen atoms and the wave function of HMI is the linear sum of the wave functions of the two hydrogen atom and how the two wave function overlap each other cause the bonding and anti-bonding, and this can be seen in this equation

$$\boldsymbol{\psi(HMI) = \psi(A) \overset{+}{-} \psi(B)} \tag{1.16}$$

If sign is positive then bonding is happened or if sign is negative then anti-bonding is happened. And that create a big change in the resultant wave function, here we solve for only Bonding state, but we plot anti-bonding by applying some operation in bonding result.

Methodology

All we in upper chapter is just the theoretical study of the HMI, we actually study it from the books, and now we are actually solving the problem using numerical method. For that first we need to reconsider the problem what it is?

2.1 What's the Real Problem to solve?

We need to numerically solve wave function for HMI and create the orbit for the electron or probability distribution for electron, and this very complex problem we need to break into the five simple problem that can be solved easily with what we learn in the upper chapter. And there is the list of part of this problem we have :

1. Finding the relation between the inter-nuclear distance and total energy of the system.
2. Solve for the Energy eigenvalue for a given inter nuclear distance and solve for the radial and angular wave function.
3. Solve the complete radial part for HMI(radial equation can't do that).
4. Now combine the both radial and angular part, and make the whole wave function bonding and anti-bonding.
5. Make the electron cloud for the HMI (in 2D).

2.2 How to solve the problem

Actually the list we see in upper section are the order to solve the problem and some of them are very similar, and we can do it step by step:

2.2.1 1^{st}, Total energy vs Inter Nuclear Distance

For find the relation between inter-nuclear distance and total energy we need to shooting method and then solve the radial roughly and get good approximate value for which the solution gives radial distribution as similar to Hydrogen atom, and we need to just do it for very small range with very small increments, and we can program as the loop for energy values until we find the Energy eigenvalue.

2.2.2 2^{nd}, Solve Radial and Angular Wave Function

This seems similar to the upper step, but this time very are solve Radial and Angular part for very refined value of energy eigenvalue, and this time we take a very large number of steps for find the very accurate wave function and energy eigenvalue, and program for this similar to upper, but we just increase the number of points where we find the wave function and that gave very accurate result, and don't do it in upper program because it make program computational hungry and we need it fast, so we do it in two steps and that give a very significant performance improvement.

2.2.3 3^{rd}, Solve the complete Radial Wave Function

In this section we just process the data we get form the $\mathbf{2^{nd}}$ step, and we use the Python programming language for this(because Python is best for data operation). In this step we just make the mirror image of the radial function and this similar to hydrogen atom radial distribution and do it for both protons and join then with different inter nuclear distance and also for both bonding and anti-bonding state.

2.2.4 4^{th}, Solve the complete Wave Function

This is done by 3D mapping of the complete radial wave function we get form the $\mathbf{3^{rd}}$ step and also join it with its angular form and this is also

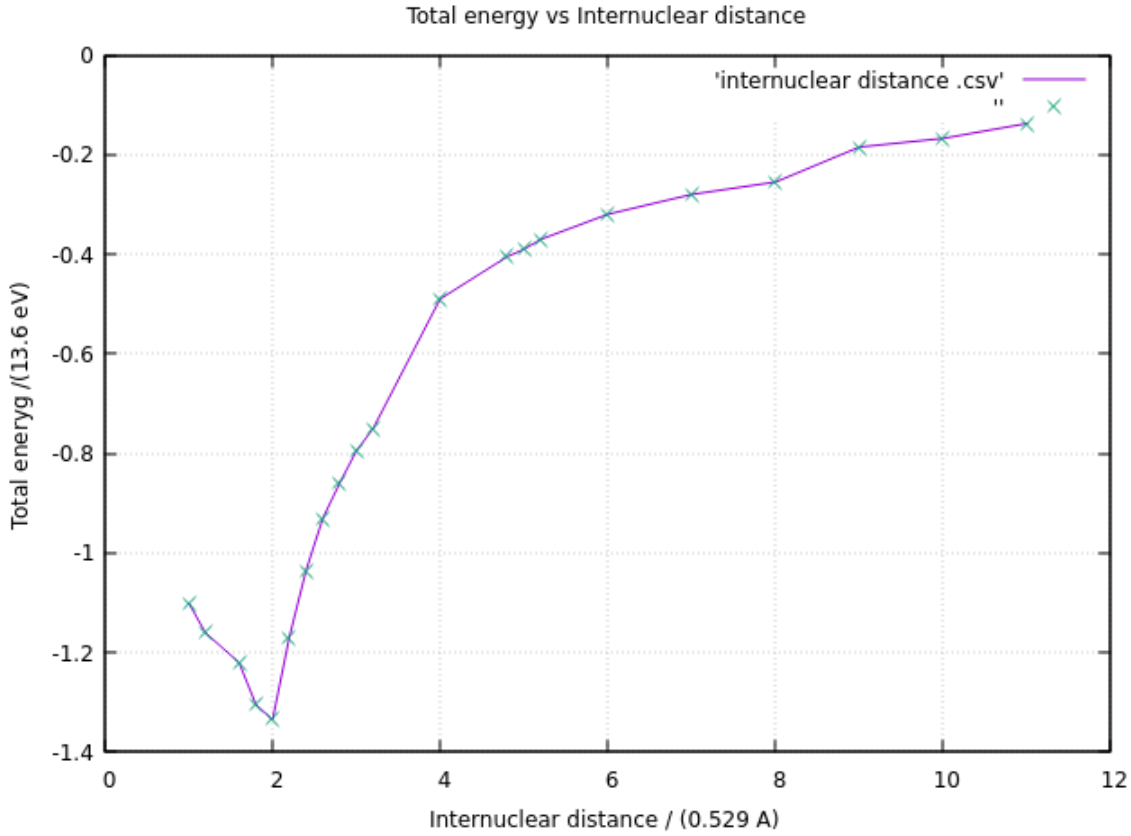
done in python and in this step we need a good amount of computational power because we need to map it in 3D and to reduce time to done and memory usage we decrease the number of steps in used in solving in radial and angular equation.

2.2.5 5^{th}, Making the electron cloud for HMI

It is done in the FORTRAN in this we use Monte Carlo Method for generating the electron cloud, and it's done by using the radial function distribution to generate the random numbers, the number of the random number we generate for a region is directly proportional to the probability distribution of the wave function.

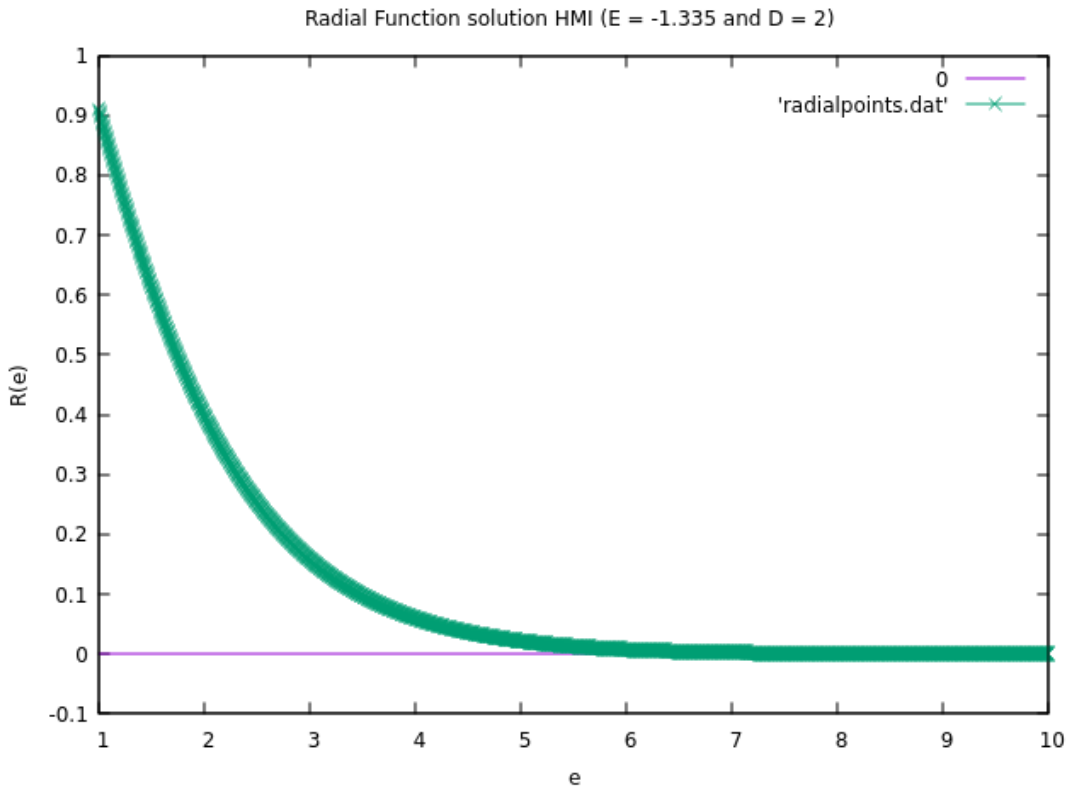
Results

3.1 Total Energy vs Inter Nuclear Distance

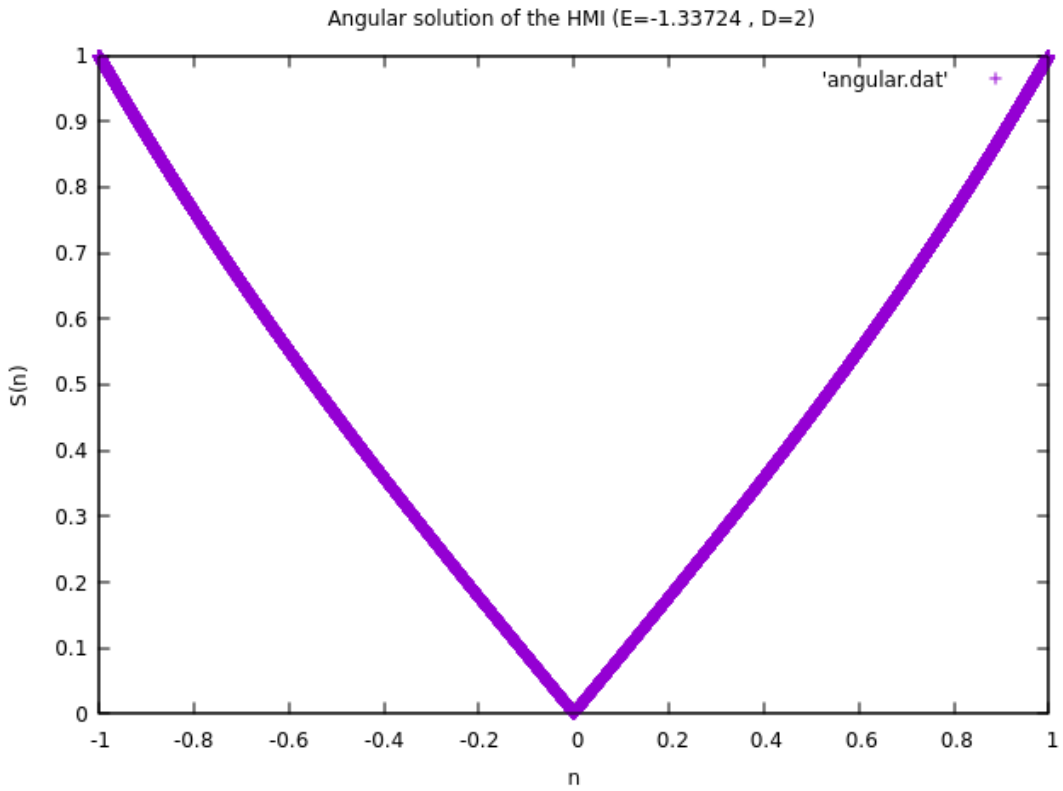


Conclusion: For inter nuclear distance 2 (x 0.529A) the systems is most stable with energy -1.335 (x 13.6eV)

3.2 Solution of Radial part of the Wave Function



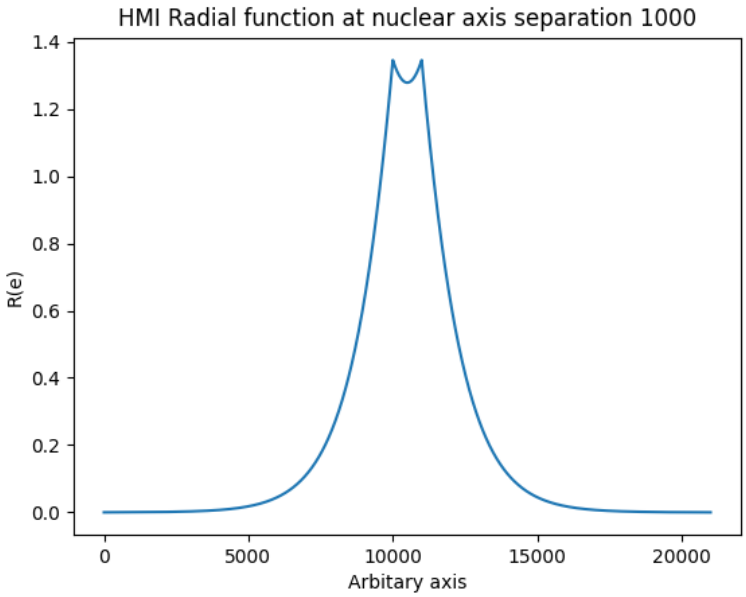
3.3 Solution of Angular part of the Wave Function

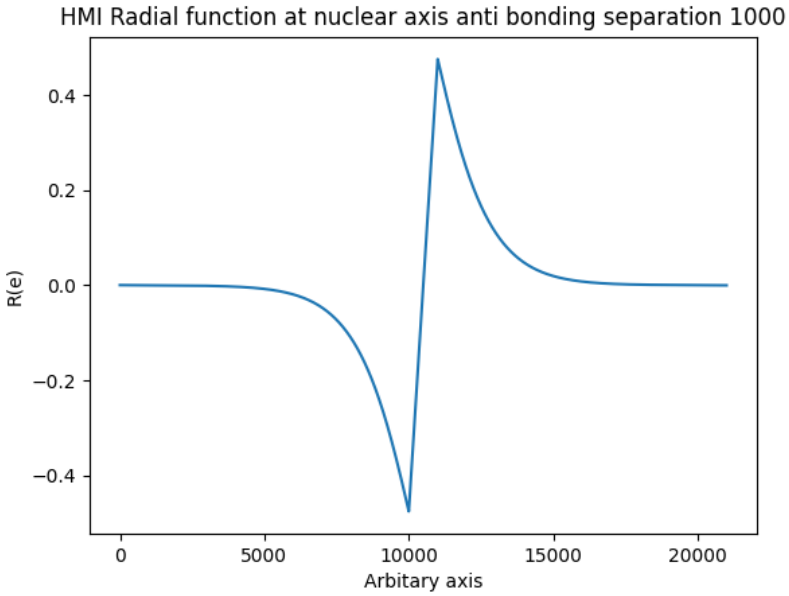


Disclaimer: This is not very accurate, but it shows the nature the angular wave function.

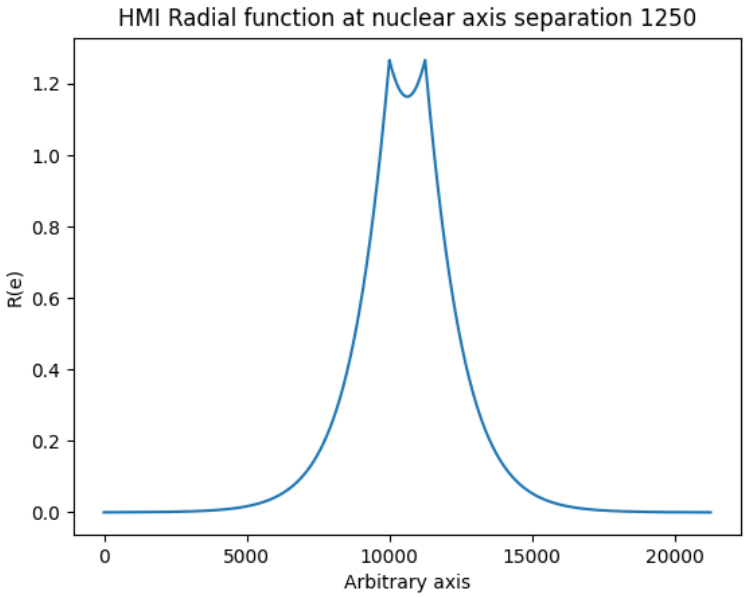
3.4 Wave Function along Inter Nuclear Axis

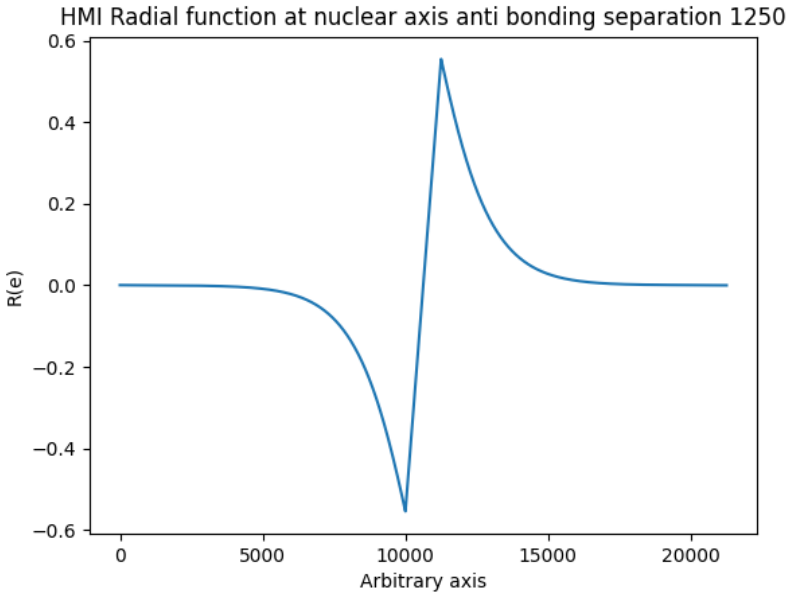
Disclaimer: The x-axis is just any arbitrary axis it just gives visual representation only of inter nuclear distance.
As we're going down the inter nuclear distance is increasing.



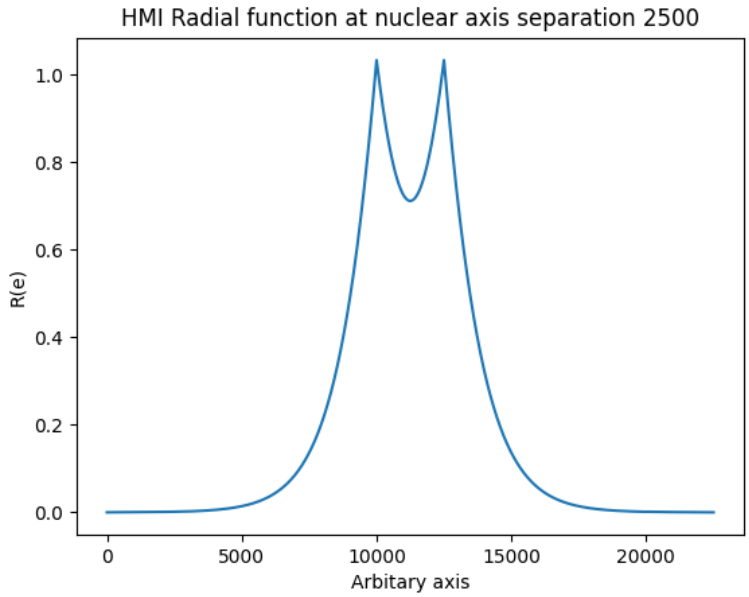


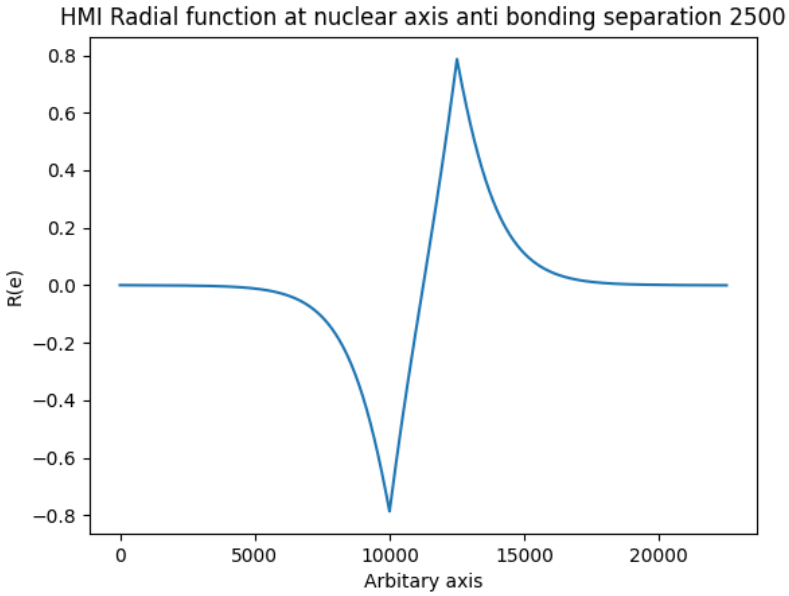
First for **Bonding** and second for **Anti Bonding** of HMI along inter nuclear Axis.



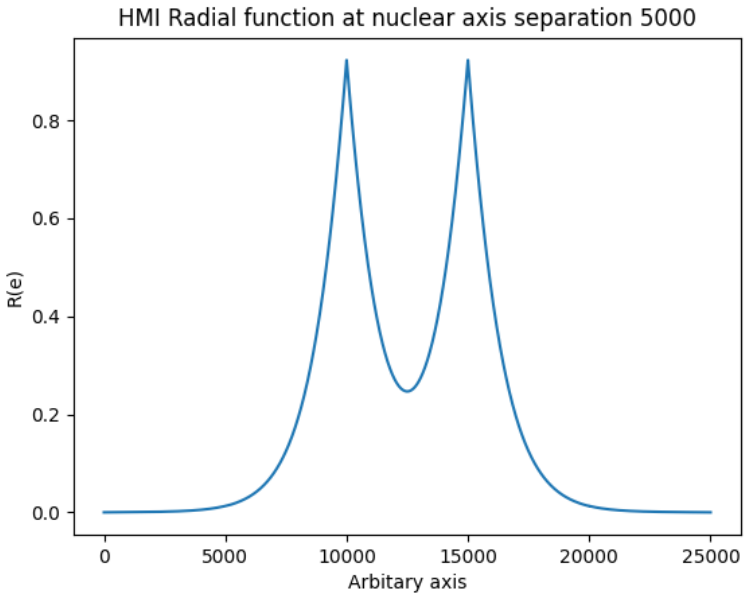


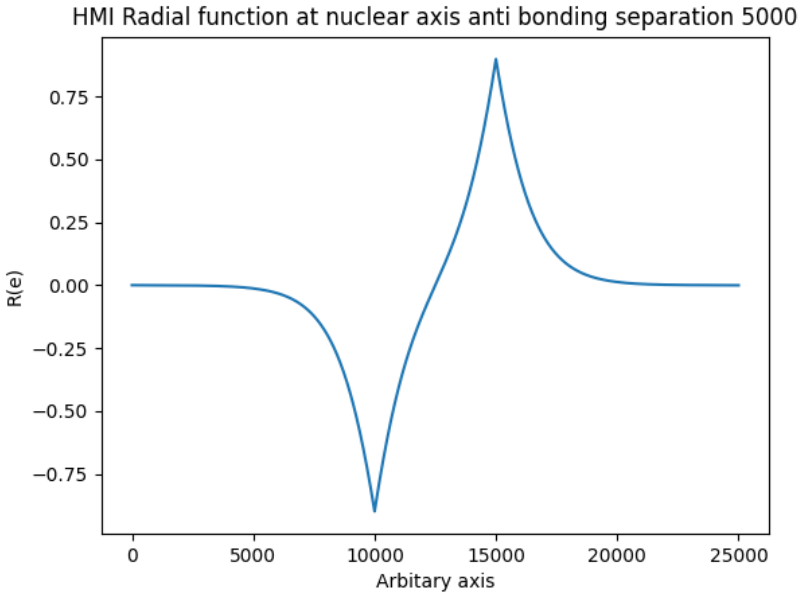
First for **Bonding** and second for **Anti Bonding** of HMI along inter nuclear Axis.





First for **Bonding** and second for **Anti Bonding** of HMI along inter nuclear Axis.

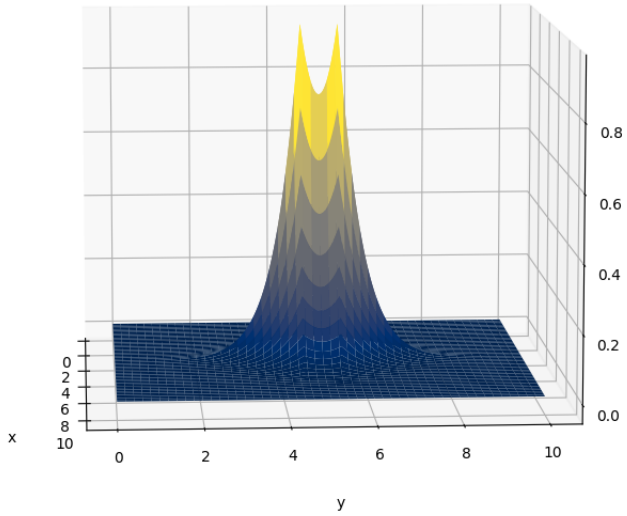


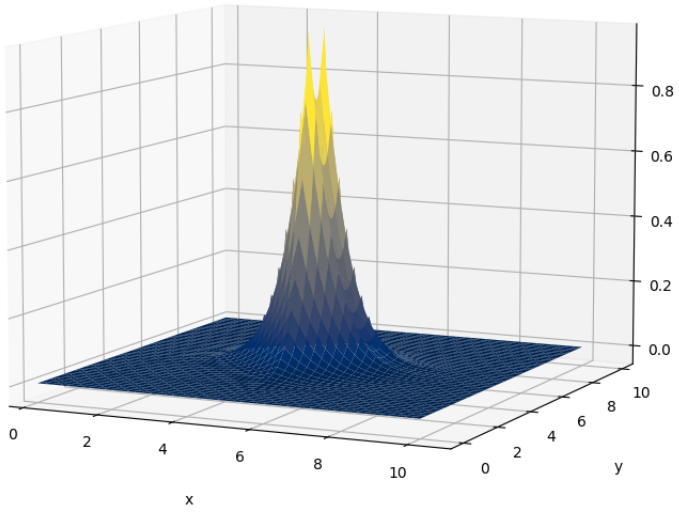


First for **Bonding** and second for **Anti Bonding** of HMI along inter nuclear Axis.

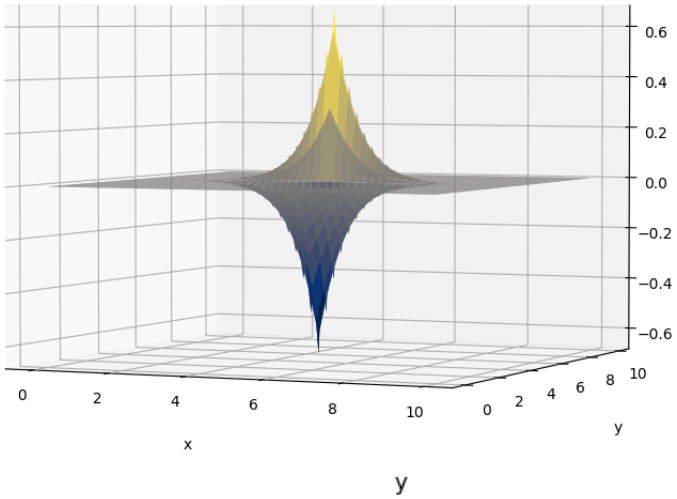
3.5 Wave Function of electron of HMI

Disclaimer: The x-axis and y-axis are just any arbitrary axis it just gives visual representation only of inter nuclear distance.

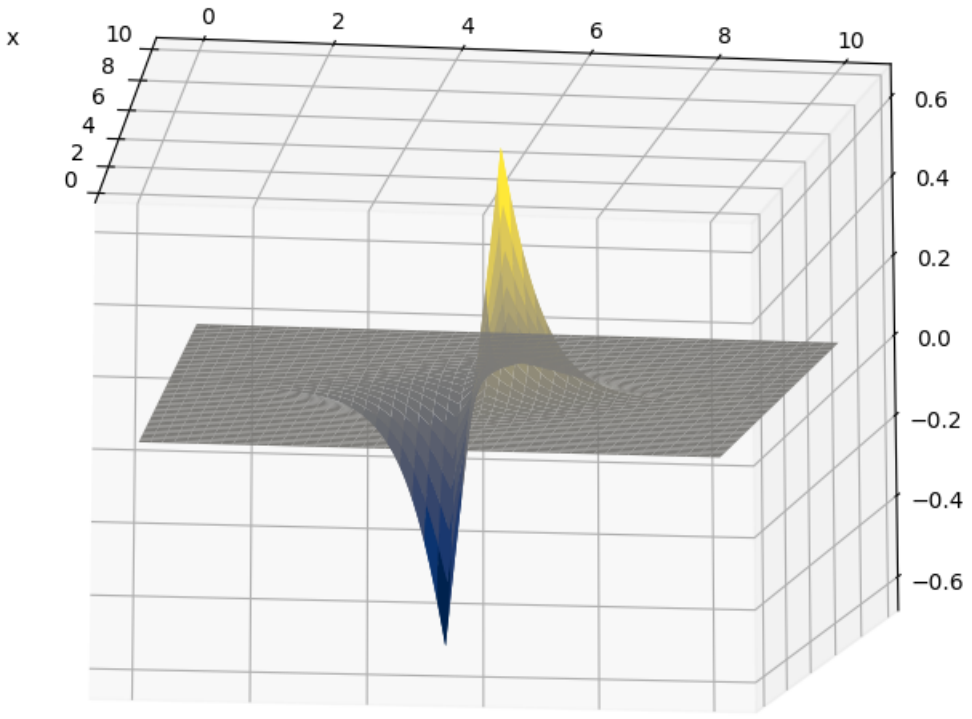




Bonding

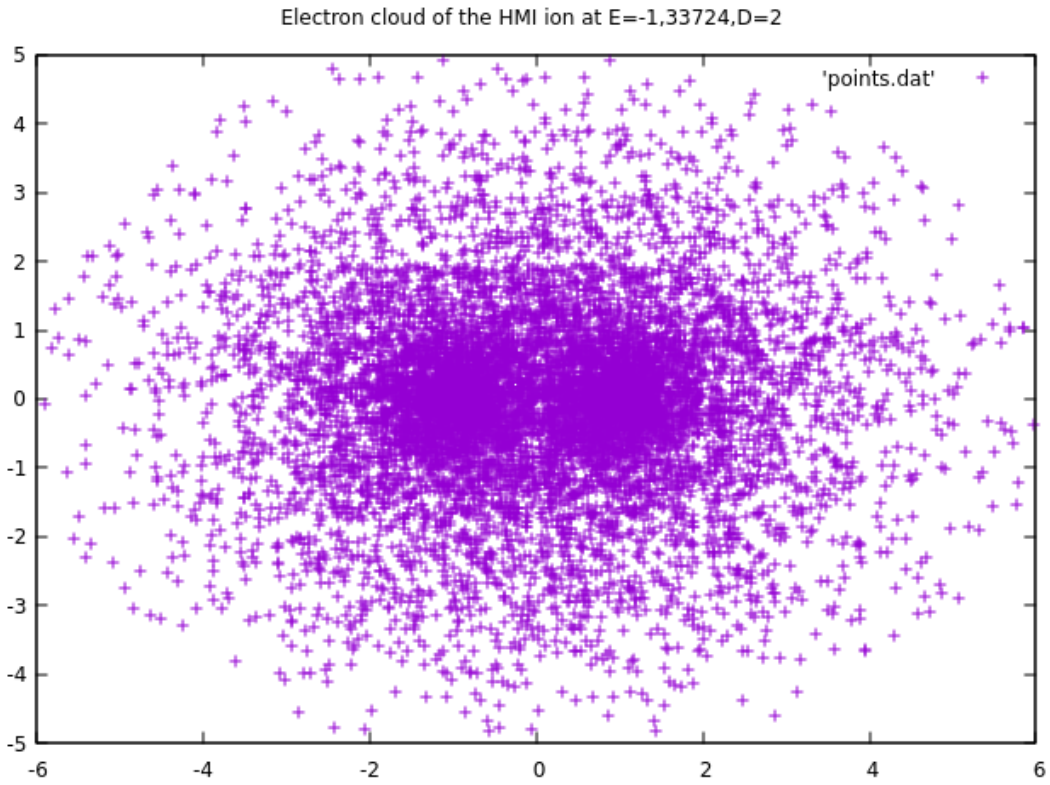


y



Anti Bonding

3.6 Electron Cloud of HMI in 2D



Here x and y-axis both have distance in multiple of 0.526A,and proton are at 1 and -1

References

- **Paper:**
  https://home.uni-leipzig.de/~physik/sites/mona/wp-content/uploads/sites/3/2017/04/the_hydrogen_molecular_ion_revisited.pdf
- **Source Code:**
  https://github.com/sourabh945/Hydrogen-Molecule-Ion.git

Appendix:

5.1 Scheme use for declaring the variables:

- $\boldsymbol{\xi} = \mathrm{e}$
- $\boldsymbol{\eta} = \mathrm{n}$
- $\boldsymbol{\Lambda} = \mathrm{c\_s}$
- $\boldsymbol{c^2} = \mathrm{c2}$

5.2 Source-Code

5.2.1 Program for finding the energy eigenvalue of the HMI using Shooting Method:

```
program main
  implicit none

  ! here we are only find the energy eigenvalue by using the
  shooting method and for this we
  ! we go from a range of energy and solve the radial equation
  for all the value of the energy and
  ! when the solution of radial equation behave like we wanted
  then we print this energy value and
  ! use the value to getting a very good radial and angular
  solution and also the electron cloud

   double precision , allocatable :: x(:) , si(:) , si1(:)
```

```
10      integer :: num_of_intervals , j , k
        double precision :: x_upper , x_lower , h , energy,
    step_size   , normalizer , c2 , d , c_s
12      double precision :: energy_upper , energy_step
        logical :: f1 , zero = .false.
14
        print*, "Enter the distance "  ! here we getting the inter
    nuclear distance
16      read*, d

18      print*, "Enter the value of the energy : "  ! upper limit of
         the energy
        read*, energy_upper
20                                          ! here we divide energy
    into 200 steps ,u can increase it
        energy_step = energy_upper/200     ! if u have a very good
    computer
22
        inquire(file="points.dat",exist=f1) ! opening the file for
    the store the data for plot
24      if(f1) then
        open(1,file="points.dat",status="replace")
26      else
        open(1,file="points.dat",status="new",action="write")
28      endif

30      do k = 1,200  ! this is our main loop for the energy
    eigenvalue

32          energy = -energy_step*k ! calculating the energy c2 and
    c_s

34          c2 = d*d*energy/4.0
            c_s = 0.6043499*c2 - 0.006188*c2*c2 - 0.0000589*c2*c2*c2
     +2
36          !c_s = 0.3127477*c2 - 0.0231669*c2*c2 - 0.0005110*(c2
    **3) - 0.0000045*(c2**4)

38          num_of_intervals = 10000 ! it for the radial equation
            x_upper = 10
40          x_lower = 1 + 1.0e-7
            h = (x_upper - x_lower)/real(num_of_intervals)
42
            allocate(x(num_of_intervals+1),si(num_of_intervals+1),
    si1(num_of_intervals+1))
```

```
44
        x(1) = x_lower  ! all the initial values and boundary
    condition
46      si(1) = 1
        si1(1) = -(2*d + c2 - c_s)/2.0
48      normalizer = 0   ! here we also normalize the function
        do j = 2,num_of_intervals
50          call rkmethod(j)       ! here we calculate the
    radial function for given energy
            normalizer = normalizer + 0.5*h*(si(j)*si(j) + si(j
    -1)*si(j-1))
52      enddo    ! we use Trapezoidal method for integration

54      normalizer = sqrt(normalizer)
        do j = 1, num_of_intervals
56          si(j) = si(j)/normalizer   ! in this condition we
    calculate that the function is
            if((abs(si(j)) < 1.0e-4 ) .and. (abs(si(j-10)) < 1.0
    e-4)) zero=.true.
58          write(1,*)x(j),si(j),si1(j),si(j)**2 , energy  !
    dying or not when go away
        enddo                          ! if the function is
    dying then we get the energy value
60      write(1,*)""                   ! we print it's energy
    and if it's not then repeat for another value
        if (zero) then
62          print*, "The solution of the equation is the founded
     at " , energy , " and at ", k , "point"
            deallocate(x,si,si1)
64          stop
        endif
66      deallocate(x,si,si1)
    enddo
68

    close(1)
70  print*,'simulation is end'
    stop
72

    contains
74

        double precision function func(val_x,val_y,val_y1) !
    here the radial equation
76          double precision , intent(in) :: val_x,val_y,val_y1
            double precision :: result , mid_1 , mid_2
78          mid_1 = ((2*d*val_x + c2*val_x*val_x - c_s)*val_y)
```

```
            mid_2 = 2*val_x*val_y1
            result = (-mid_1 - mid_2)/(val_x*val_x - 1)
            func = result
            return
        end function func

        ! here we calculate the value by the rk method and this is for 2nd order equation

        subroutine rkmethod(i)    ! here we use 4th order rk method for 2nd order ODE
            integer , intent(in) :: i
            double precision :: k1,k2,k3,k4,k11,k12,k13,k14, result
            k11 = h*si1(i-1)
            k1 = h*func(x(i-1),si(i-1),si1(i-1))
            k12 = (si1(i-1) + k1/2.0)*h
            k2 = h*func((x(i-1)+h/2.0),(si(i-1)+k11/2.0),(si1(i-1)+k1/2.0))
            k13 = (si1(i-1) + k2/2.0)*h
            k3 = h*func((x(i-1)+h/2.0),(si(i-1)+k13/2.0),(si1(i-1)+k2/2.0))
            k14 = (si1(i-1) + k3)*h
            k4 = h*func((x(i-1)+h),(si(i-1)+k14),(si1(i-1)+k3))
            x(i) = x(i-1) + h
            si(i) = si(i-1) + (k11+2*k12+2*k13+k14)/6.0
            si1(i) = si1(i-1) + (k1+2*k2+2*k3+k4)/6.0
            return
        end subroutine rkmethod
end program
```

"Energy eigenvalue finder.f95"

5.2.2 Program for solve for Radial and Angular wave equation, and also find the Electron Cloud for HMI:

```
program main
implicit none !declaring the variable for the program

double precision , allocatable :: x(:) , si(:) , si1(:) ! here we use for both radial and angular part
double precision :: prob(10)
```

```
integer :: num_of_intervals , j  ,count = 0              ! by
    just reallocate the array
double precision :: x_upper , x_lower , h , energy, normalizer ,
     c2 , d , c_s
logical :: f1 , f2 , angular = .false.
real , parameter :: pi = 3.141592653589793

d = 2.0 ! it is the inter nuclear distance

print*, "Enter the value of the energy : "
read*, energy

num_of_intervals = 10000

c2 = d*d*energy/4.0  ! c_s can be taken by both value it depend
    upon the what type of system it is.
c_s = 0.6043499*c2 - 0.006188*c2*c2 - 0.0000589*c2*c2*c2 +2
!c_s = 0.3127477*c2 - 0.0231669*c2*c2 - 0.0005110*(c2**3) -
    0.0000045*(c2**4)

! for radial part solution

inquire(file="radial.dat",exist=f1) ! opening the file for the
    store the data for plot
if(f1) then
    open(1,file="radial.dat",status="replace")
else
    open(1,file="radial.dat",status="new",action="write")
endif


x_upper = 10          ! This is the upper limit we can extent it
    also for higher energies
x_lower = 1 + 1.0e-7 ! we can't use the 1 because the function
    is not defined at 1
h = (x_upper - x_lower)/real(num_of_intervals)

allocate(x(num_of_intervals+1),si(num_of_intervals+1),si1(
    num_of_intervals+1))

! here x = e , si = R(e) , si1 = R'(e)

x(1) = x_lower
si(1) = 1
```

```
si1(1) = -(2*d + c2 - c_s)/2.0 ! this is the boundary condition we derive with m=0
normalizer = 0  ! normalizer to use to normalize the function it use Trapezoidal method integration

do j = 2,num_of_intervals
    call rkmethod(j)       ! here we call the rk method to solve for next value
    normalizer = normalizer + 0.5*h*(si(j)*si(j) + si(j-1)*si(j-1))
enddo

normalizer = sqrt(2*normalizer) ! we multiply it with 2 because we find only one side of the radial fxn
                                ! And for we getting exact value when we do for both side

do j = 1, num_of_intervals

    if (j == 1000*count+1) then  ! the prob is use for make the electron cloud
        prob(count+1) = si(j)    ! here prob store the value at the some points
        count = count + 1        ! that have max in range we can sum for all value
    endif                        ! but i choose this way and it works fine

    si(j) = si(j)/normalizer
    write(1,*)x(j),si(j),si1(j),si(j)**2,si(j)*x(j)
enddo

deallocate(x,si,si1) ! here we deallocate the array for use in angular part

close(1)

! for Angular wave function solution

angular = .true.  ! this change the func into angular equation

inquire(file='angular.dat',exist=f2)
if(f2) then
    open(2,file="angular.dat",status="replace")
else
```

```
76      open(2,file="angular.dat",status="new",action="write")
endif
78  ! x = n , si = S(n) , si1 = S'(n)
allocate(x(num_of_intervals+1),si(num_of_intervals+1),si1(
    num_of_intervals+1))
80
x_upper = 0     ! here we done for half of the and generate the
    other value using these values
82  x_lower = -1 + 1.0e-4
h = (x_upper - x_lower)/real(num_of_intervals)
84
x(1) = x_lower
86  si(1) = 1
si1(1) = (c2-c_s)/2  ! here m = 0 and we get this equation in
    Boundary condition section
88  normalizer = 0        ! normalizer is same as the radial part but
     we not using it.

90  do j = 2,num_of_intervals
      call rkmethod(j)
92    normalizer = normalizer + 0.5*h*(si(j)*si(j) + si(j-1)*si(j
    -1))
enddo
94
normalizer = sqrt(2*normalizer)
96  do j = 1,num_of_intervals        ! this gives value between -1 to
     0
      !si(j) = si(j)/normalizer
98    write(2,*)x(j),si(j),si1(j)/normalizer,si(j)/x(j)
enddo
100 do j = num_of_intervals,1,-1     ! here we generate the value b/w
     0 to 1
      !si(j) = si(j)/normalizer    ! using the other values
102   write(2,*)-x(j),si(j),si1(j)/normalizer,si(j)/x(j)
enddo
104
deallocate(x,si,si1) ! here we finally deallocate the memory
106 close(2)

108     ! Making the electron cloud by calling the subroutine orbit

110 call orbit(prob)

112 print*,'simulation is end'
stop
```

```
114
contains  ! func is the wave equations
116
        double precision function func(val_x,val_y,val_y1)
118         double precision , intent(in) :: val_x,val_y,val_y1
            double precision :: result , mid_1 , mid_2
120         if(angular) then
                mid_1 = (c2*val_x*val_x - c_s)*val_y ! angular
    function
122             mid_2 = (2*val_x*val_y1)
                result = (mid_1 + mid_2)/(1 - val_x*val_x)
124             else
                mid_1 = ((2*d*val_x + c2*val_x*val_x - c_s)*
    val_y)
126             mid_2 = 2*val_x*val_y1                          !
     radial function
                result = (-mid_1 - mid_2)/(val_x*val_x - 1)
128             endif
                    func = result
130         return
        end function func
132
        ! here we calculate the value by the rk method and this
    is for 2nd order equation
134     subroutine rkmethod(i)
            integer , intent(in) :: i
136         double precision :: k1,k2,k3,k4,k11,k12,k13,k14,
    result
            k11 = h*si1(i-1)
138         k1 = h*func(x(i-1),si(i-1),si1(i-1))  ! here we use
    4th order rk method for 2nd order ODE
            k12 = (si1(i-1) + k1/2.0)*h
140         k2 = h*func((x(i-1)+h/2.0),(si(i-1)+k11/2.0),(si1(i
    -1)+k1/2.0))
            k13 = (si1(i-1) + k2/2.0)*h
142         k3 = h*func((x(i-1)+h/2.0),(si(i-1)+k13/2.0),(si1(i
    -1)+k2/2.0))
            k14 = (si1(i-1) + k3)*h
144         k4 = h*func((x(i-1)+h),(si(i-1)+k14),(si1(i-1)+k3))
            x(i) = x(i-1) + h
146         si(i) = si(i-1) + (k11+2*k12+2*k13+k14)/6.0
            si1(i) = si1(i-1) + (k1+2*k2+2*k3+k4)/6.0
148         return
        end subroutine rkmethod
150
```

```
    ! the logic behind this program is simple we just
calculate the radial equation
    ! Because i am not good in file operation in FORTRAN
and i am still learning
    ! so i just calculate the radial distribution and from
that calculate the probability
    ! finding the electron and multiply it with 100 or any
other number and generate the
    ! resultant number random numbers and plot and we get
the electron cloud

    subroutine orbit(probability)
        double precision , dimension(10) , intent(in)::
probability
        integer :: i,k,points
        double precision :: t,r,x1,y1
        logical :: f2

        inquire(file="orbit.dat",exist=f2)
        if(f2) then
            open(3,file="orbit.dat",status="replace")
        else
            open(3,file="orbit.dat",status="new",action="
write")
        endif

        do k = 1,5
            points = probability(k)*100
            do i = 1, 1000
                if (points < i) then ! here we generate the
random no. in circle like
                    call random_number(t)  ! the random
position of electron and as the
                    call random_number(r)  ! probability of
find the electron increase the
                    r = r*real(k)          ! the number of
the points are also increasing
                    x1 = r*cos(t*2*pi) + 1 ! and we get what
 we wanted and repeat this
                    y1 = r*sin(2*pi*t)     ! process for
both the protons
                    write(3,*)x1,y1
                    x1 = r*cos(t*2*pi) - 1
                    y1 = r*sin(2*pi*t)
                    write(3,*)x1,y1
```

```
                    endif
184                 enddo
                    write(3,*)""
186             enddo
                close(3)
188             return
        end subroutine orbit
190 end program
```

"main.f95"

5.2.3 Python Program for Operate on the files and generate the graphs for the HMI orbit:

```
from matplotlib import pyplot as plt
2 import csv
import numpy as np
4 from math import pow
import os
6 from mpl_toolkits import mplot3d

8 ''' this program we use to operate on the data file we generate
    through
the fortran program here we are plot the graph with help of this
     program
10 here we make all the plots except some of the are made using the
     gnuplot
'''
12 path = "./processed_csv_files"

14 e = []
n = []
16 angular = []
radial = []
18
data = []
20
os.system(f"touch {path}/radial.csv")
22 os.system(f"touch {path}/angular.csv")
os.system(f"{path}/Hydrogen_Radial.csv")
24


26 with open('radial.dat','r') as file :
```

```
    with open(f'{path}/radial.csv','a') as csv_file:
        writer = csv.writer(csv_file)
        for line in file :
            data += [line.split()]
            e.append(float(data[-1][0]))
            radial.append(float(data[-1][1]))
            writer.writerow([e[-1],radial[-1]])
        csv_file.close()
    file.close()

with open('angular.dat','r') as file :
    with open(f'{path}/angular.csv','a') as csv_file:
        writer = csv.writer(csv_file)
        for line in file:
            data += [line.split()]
            n.append(float(data[-1][0]))
            angular.append(float(data[-1][1]))
            writer.writerow([n[-1],angular[-1]])
        csv_file.close()
    file.close()

Radial = []
E = []
step = 20/200000

with open(f'{path}/Hydrogen_Radial.csv',"a") as file :
    writer = csv.writer(file)
    for i in range(len(e)-1,-1,-1):
        Radial.append(radial[i])
        writer.writerow([radial[i]])
    for i in range(0,len(e)):
        Radial.append(radial[i])
        writer.writerow([radial[i]])

    file.close()

result = []
separation = 200

with open(f'{path}/HMI_Radial.csv','a') as file :
    writer = csv.writer(file)
    for i in range (0,len(Radial)+separation):
        if i > separation and i < len(Radial):
            result.append(Radial[i]+Radial[i-separation])
        elif i>=len(Radial):
```

```
            result.append(Radial[i-separation])
        else:
            result.append(Radial[i])
        writer.writerow([result[i]])

resultab = []

with open(f'{path}/HMI_Radial_antibonding.csv','a') as file :
    writer = csv.writer(file)
    for i in range (0,len(Radial)+separation):
        if i > separation and i < len(Radial):
            resultab.append(-Radial[i]+Radial[i-separation])
        elif i>=len(Radial):
            resultab.append(Radial[i-separation])
        else:
            resultab.append(-Radial[i])
        writer.writerow([resultab[i]])

Result = np.array(result)

Resultab = np.array(resultab)


plt.plot(Radial)
plt.title('Hydrogen Radial Function from HMI Radial function')
plt.ylabel('R(e)')
plt.xlabel('Arbitrary axis')
plt.savefig(f"{path}/Hydrogen Radial Function")
plt.close()
plt.plot(result)
plt.title(f'HMI Radial function at nuclear axis separation {
    separation}')
plt.ylabel('R(e)')
plt.xlabel('Arbitrary axis')
plt.savefig(f"{path}/HMI Radial Function{separation}")
plt.close()
plt.plot(resultab)
plt.title(f'HMI Radial function at nuclear axis anti bonding
    separation {separation}')
plt.ylabel('R(e)')
plt.xlabel('Arbitrary axis')
plt.savefig(f"{path}/HMI Radial Function ab {separation}")
plt.close()

step = 10/len(Result)
```

```
step_y = 10/len(Radial)
x = np.arange(0,10,step_y)
y = np.arange(0,10,step)
X,Y = np.meshgrid(x,y)

fig = plt.figure(figsize=(12,10))
ax = plt.axes(projection='3d')

Result = np.array(Result)
print('multiply start')
Z = []
with open(f'{path}/HMI_Radial.csv','a') as file :
    writer = csv.writer(file)
    for i in range(0,len(Result)):
        mid = []
        for j in range(0,len(Radial)):
            mid.append((Result[i]*Radial[j]))
        writer.writerow(mid)
        Z.append(mid)

Z = np.array(Z)
print('complete')

surf = ax.plot_surface(X,Y,Z,cmap=plt.cm.cividis)

plt.title('HMI bonding state')

ax.set_xlabel('x', labelpad=20)
ax.set_ylabel('y', labelpad=20)
ax.set_zlabel('R(e)', labelpad=20)

fig.colorbar(surf, shrink=0.5, aspect=8)

plt.savefig(f"HMI bonding state {separation*10}.png")

plt.show(block=True)

step = 10/len(Resultab)
step_y = 10/len(Radial)
x = np.arange(0,10,step_y)
y = np.arange(0,10,step)
X,Y = np.meshgrid(x,y)

fig = plt.figure(figsize=(12,10))
ax = plt.axes(projection='3d')
```

```
160
Resultab= np.array(Resultab)
162 print('multiply start')
Z = []
164 with open(f'{path}/HMI_Radial.csv','a') as file :
    writer = csv.writer(file)
166     for i in range(0,len(Resultab)):
        mid = []
168         for j in range(0,len(Radial)):
            mid.append((Resultab[i]*Radial[j]))
170         writer.writerow(mid)
        Z.append(mid)
172
Z = np.array(Z)
174 print('complete')

176 surf = ax.plot_surface(X,Y,Z,cmap=plt.cm.cividis)

178 plt.title('HMI anti bonding state')

180 ax.set_xlabel('x', labelpad=20)
ax.set_ylabel('y', labelpad=20)
182 ax.set_zlabel('R(e)', labelpad=20)

184 fig.colorbar(surf, shrink=0.5, aspect=8)

186 plt.savefig(f"HMI anti bonding state {separation*10}.png")

188 plt.show(block=True)
```

processingfile.py