Name: Asif Iqbal Bhatti
Assignment # 1
Course Title: Computational methods of Electronic structure theory (CMEST)
***Submitted to*:** Andris Gulans

Solution to Hydrogen atom by LAPACK subroutines

1 Hydrogen atom radial plot

Six lowest hydrogen energy levels are:

n	l	Energy (a.u)
1	0	-0.4988
2	0	-0.1248
2	1	-0.1251
3	0	-0.0555
3	1	-0.0556
3	2	-0.0312

The corresponding figures are plotted below.

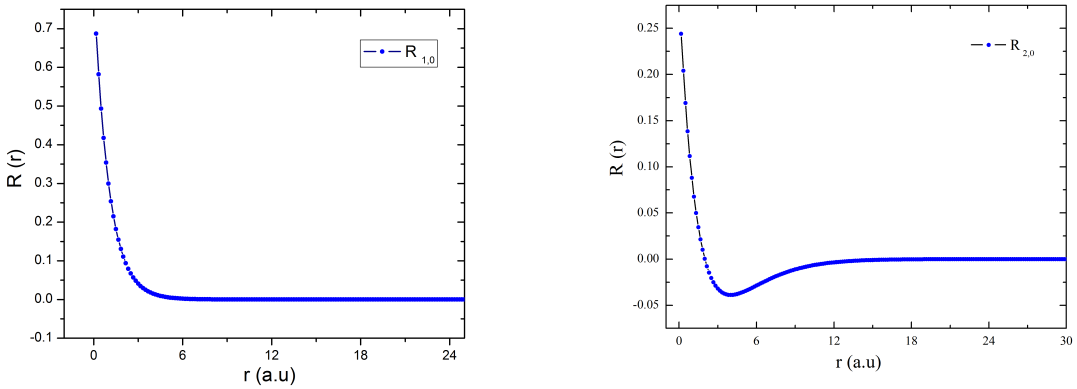

Figure 1.1:

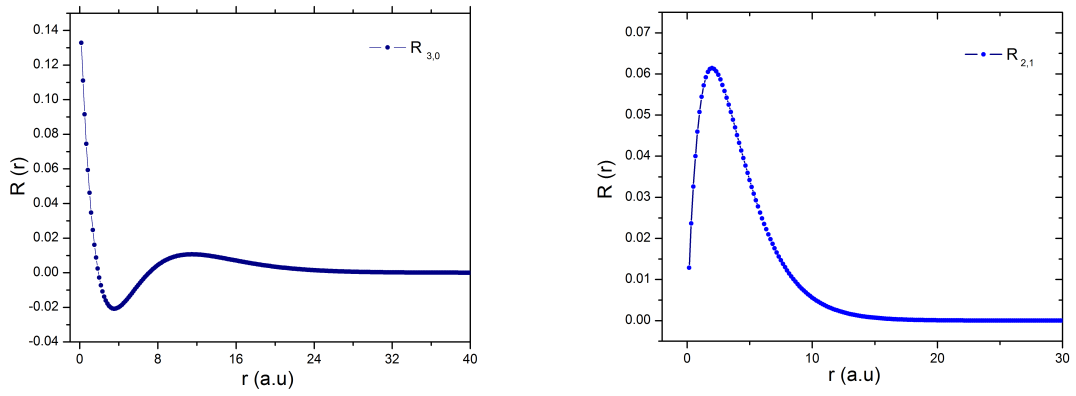

Figure 1.2:

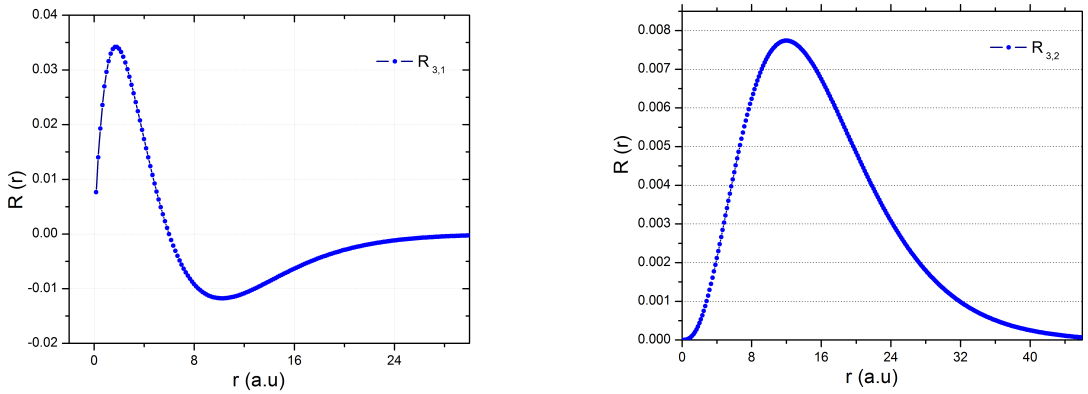

Figure 1.3:

Appendix

The difference among these solvers dsyev, dsbev and dstebz resides in the execution time. In case for dstebz, the execution time is faster as seen by local CPU_time. Since, in this routine eigenvalues can be called in the given range, which implies no need for extra calculation. If compared to dsbev subroutine the diagonal and sub or super-diagonal term are stored in another matrix of these dimensions which is a little faster. Lastly, dsyev just take the given matrix and solves for eigenvalues and optionally eigenvectors. Over here it doesn't do any extra effort of assigning matrix to other dimension and solve. So, if by execution time (by calling *CPU_time subroutine*) it could be stated in this way

dstebz < dsbev < dsyev

Exercise # 1: DSYEV.f90 subroutine

Functions/Subroutines:

```
call dsyev(JOBZ, UPLO, N, A, LDA, W, WORK, LWORK, INFO)
```

DSYEV computes the eigenvalues and, optionally, the left and/or right eigenvectors for symmetric matrices,

```
1  program SE
2    implicit none
3   double precision                                  :: t, start_time, stop_time
4   double precision, parameter                       :: rmax = 50
5   integer                                           :: i, j, k, N, M, INFO, LWORK, l
6   double precision, dimension(:), allocatable       :: r, W, WORK
7   double precision, dimension(:,:), allocatable     :: H
8   intrinsic  INT, MIN, MAX
9
10  print*, "Enter the size of a grid N > 2 " ; read*, N
11  print*, "Enter the value for angular momemtum l = " ; read*, l
12
13  LWORK = 3*N-1
14  allocate(WORK(LWORK),r(N-1),H(N-1,N-1),W(N))
15  !*************************Intialization*******************************
16
17  t = rmax/dble(N)
18  r = 0d0
19  H = 0d0
20  !***************Constructing the corresponding Hamiltonian**************
21  !*********************for 1D schrodinger equation************************
22
23  do i = 1, N-2
24   r(i)     = i*t
25   H(i,i)   = 2*(1 - (1/r(i))*t*t + (dble(l)*(dble(l)+1)) &
26                    *(1/(r(i)**2))*((t*t)/2) )
27   H(i+1,i) = -1d0
28   H(i,i+1) = -1d0
29  enddo
30  H(N-1,N-1) = 2*(1-(1/(r(N-2)+t)*t*t) + (dble(l)*(dble(l)+1))* &
31                  (1/(r(N-2)+t)**2)*((t*t)/2) )
32  H = H/(2*t**2)
33  !***********************************************************************
34  !***************Eigenvalues & Eigen vectors*****************************
35
36  call cpu_time(start_time)
37  call dsyev('V', 'L', M, H, M, W, WORK, LWORK, INFO)
38  call cpu_time(stop_time)
39  if( INFO.gt.0 ) then
40   write(*,*)'The algorithm failed to compute eigenvalues.'
41   stop
42  endif
43
44  !***********************************************************************
45  print*, ":: The Eigenvalues for l = ", l
46  write(*,'(9X,100F8.4)') (W(i), i = 1, 4)
47  write (*,*) "The corresponding Eigenvectors are"
48
49  open(unit = 20, file = 'radialplot.dat', action = 'write', status = 'replace')
```

```
50   write(20,'(2X,I4)') N-2  do i = 1, N-2
51   write(20,'(2X, F8.4,2X,F8.4)') r(i), -H(i,2)*(1/r(i))
52   enddo
53   close(20)
54
55   print *, "Original loop time:", stop_time - start_time, "seconds"
56
57  end program SE
```

Exercise # 1: DSBEV.f90 subroutine

DSBEV computes all the eigenvalues and, optionally, eigenvectors of a **real symmetric band matrix**.

```
call dsbev(JOBZ, UPLO, N, KD, AB, LDAB, W, Z, LDZ, WORK, INFO)
```

```
1   program SE
2     implicit none
3    double precision                                    :: t
4    double precision, parameter                         :: rmax = 40
5    integer                                             :: i, j, k, N, M, INFO, kd, LDAB, l
6    double precision, dimension(:), allocatable         :: r, W, WORK
7    double precision, dimension(:,:), allocatable       :: H, Z, ab
8    character(1), parameter                             :: uplo = 'U'
9    intrinsic  INT, MIN, MAX
10   double precision                                    :: st_time, sp_time
11
12   print*, "Enter the size of a grid N > 5: " ;  read*, N
13   print*, "Enter the value for angular momentum l = " ; read*, l
14
15   kd = 1 !******* For H we have only one super/sub diagonal vector*******
16  LDAB = kd + 1
17   allocate(r(N-1),H(N-1,N-1),W(N),Z(N,N),ab(LDAB,N),WORK(3*N-2))
18
19   !**********************Intialization**********************************
20   t = rmax/dble(N)
21   r = 0d0
22   H = 0d0 !***************Constructing the corresponding Hamiltonian**************
23   !*********************for 1D schrodinger equation************************
24   do i = 1, N-2
25    r(i)      = i*t
26    H(i,i)    = 2*(1 - (1/r(i))*t*t + (l*(l+1))*(1/r(i)**2)*((t*t)/2) )
27    H(i+1,i) = -1d0
28    H(i,i+1) = -1d0
29   enddo
30   H(N-1,N-1)  = 2*(1 - (1/(r(N-2)+t)*t*t) + (l*(l+1))*(1/(r(N-2)+t)**2)&
31                  *((t*t)/2) )
32   H = H/(2*t**2)
33   !***************************************************************************
34   !********Upper or lower SYM BAND matrix is stored in ******************
35   if (uplo == 'U') then
36   do i = 1, N-1
37    do j = i, MIN(N-1, i+kd)
38     ab(kd+1+i-j,j) = H(i,j)
39       enddo
40   enddo
41   else if (uplo == 'L') then
42    do i = 1, N-1
43     do j = i, MAX(1, i-kd)
44      ab(1+i-j,j) = H(i,j)
45     enddo
46    enddo
47   endif
48   !***************************************************************************
49   N = N -1 !************************ (N-1)x(N-1) matrix
50   !*******calling a LAPACK dsbev subroutine*********************************
51
52   call cpu_time(st_time)
53   call dsbev('V',uplo, N, kd, ab, LDAB, W, Z, N, WORK, INFO)
54   call cpu_time(sp_time)
55   !**********************printing the eigenvalues**************************
56   print*, ":: The Eigen values for l =  ", l
57   write(*,'(1x,1000f12.5)'), (W(i), i = 1, 5)
```

```
58   print*, ":: The correspoding Eigenvector are"
59   !********Saving the radial part in file to plot**********************
60
61   open(unit = 20, file = 'radialWF.dat', action = 'write', status = 'replace')
62   write(20,*) N-1
63   do i = 1, N-1
64    write(20,'(1X,1000F12.5)') r(i), H(i,1)*(1/r(i)), H(i,2)*(1/r(i)), H(i,3)*(1/r(i)), H(i,4)*(1/r(i))
65   enddo
66   close(20)
67   write(*,*) "Wall Time", sp_time - st_time, "seconds"
68  end program SE
```

Exercise # 1: DSTEBZ.f90 subroutine

```
call dstebz(RANGE, ORDER, N, VL, VU, IL, IU, ABSTOL, D, E, M, NSPLIT, &
            W, IBLOCK, ISPLIT, WORK, IWORK, INFO)
```

DSTEBZ computes the eigenvalues of a symmetric tridiagonal matrix T. The user may ask for all eigenvalues, all eigenvalues in the half-open interval $(VL, VU]$, or the $IL-th$ through $IU-th$ eigenvalues.

```
1   program eigendstebz
2    USE SE_mod ! module is given below this program for constructing Hamiltonian
3    implicit none
4   character(1), parameter                    :: RANGE = 'A', ORDER = 'E'
5   integer                                    :: N, M, NSPLIT, INFO, i, j, l
6   integer, parameter                         :: IL = 0, IU = 0
7   integer, dimension(:), allocatable         :: IBLOCK, ISPLIT, IWORK
8   double precision, parameter                :: VL = 0, VU = 0
9   double precision, parameter                :: ABSTOL = -1
10  double precision, dimension (:), allocatable    :: W, WORK, D, E
11  double precision, dimension (:,:), allocatable  :: A
12  double precision                           :: start_time, stop_time
13
14  !********Requesting matrix dimension*************************************
15  print*, "Enter the matrix dimension > 2:  "; read*, N
16  print*, "Enter the corresponding l = "; read*, l
17  call hamil(A,N,l)
18  allocate (IBLOCK(N), ISPLIT(N), IWORK(3*N), W(N), WORK(4*N), D(N), E(N-1))
19  N = N - 1
20
21  open (unit = 10, file = 'hamilmatrix', action = 'write', status = 'replace')
22  do i = 1, N
23   write(10,'(1x,1000000F12.6)') (A(i,j), j= 1, N)
24  enddo
25  close(10)
26  !*****assigning diagonal and sub/super diagonal terms******************
27  do i = 1, N
28   D(i) = A(i,i)
29  enddo
30  do j = 1, N-1
31   E(j) = A(j,j+1)
32  enddo
33  !******************All Eigenvalues is requested**********************
34
35  call cpu_time(start_time)
36  call DSTEBZ(RANGE, ORDER, N, VL, VU, IL, IU, ABSTOL, D, E, M, &
37  NSPLIT, W, IBLOCK, ISPLIT, WORK, IWORK, INFO)
38  call cpu_time(stop_time)
39
40  if (INFO.NE.0) then
41   write(*,*) 'DSTEBZ', INFO
42  else
43   write(*,*)
44   write(*,*) "Number of Eigen values found", M, NSPLIT
45   Write (*, *) 'Eigenvalues are::'
46   Write (*, *) (W(j), j= 1, 5)
47  endif
48  print*, "Original loop time", start_time - stop_time, "seconds"
49  deallocate(IBLOCK,ISPLIT,IWORK,W,WORK,D,E,A)
50 end program eigendstebz
51 !!!!!!!!!!!!!!!!!!!!MODULE FOR constructing Hamiltonian!!!!!!!!!!!!
52 MODULE SE_mod
```

```
53    implicit none
54    contains
55   subroutine hamil(H,N,l)
56   double precision                                        :: t
57   double precision, parameter                             :: rmax = 100
58   integer                                                 :: i, j, k, M, g
59   double precision, dimension(:), allocatable             :: r
60   double precision, dimension(:,:), allocatable, intent(inout)    :: H
61   integer, intent(in)                  :: N, l
62   allocate(r(N-1),H(N-1,N-1))
63   !*************** Intialization ********************************
64   t = rmax/dble(N)
65   r = 0d0
66   H = 0d0
67   !*************** Constructing the corresponding
68   Hamiltonian**************
69   !********************* for 1D schrodinger equation************************
70   !************ Theoretical Energy level -1/2n**2*************************
71   do i = 1, N-2
72    r(i)      = i * t
73    H(i,i)    = 2*(1 - (1/r(i))*t*t + (l*(l+1))*(1/r(i)**2)*((t*t)/2) )
74    H(i+1,i) = -1d0
75    H(i,i+1) = -1d0
76   enddo
77   H(N-1,N-1)    = 2*(1 - (1/(r(N-2)+t)*t*t) + (l*(l+1))*((1/r(N-2)+t)**2)*((t*t)/2) )
78   H = H/(2*t**2)
79   !***********************************************************************
80   end subroutine hamil
81 END MODULE SE_mod
```