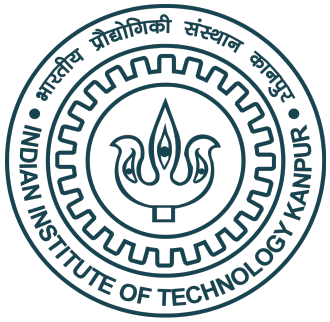
Project Report - CHM684A

Normal Mode Analysis Of A Molecule

Chitresh Bhaskar Chaudhari



Indian Institute of Technology, Kanpur

Contents

Problem Statement

In this project I wrote a code to perform vibrational analysis for the Lennard-Jones cluster with 6 argon atoms and visualised the vibrational motions of cluster using MOLDEN.

1 Introduction

A normal mode is a vector, along which the motion of the atoms of a molecule is uncoupled to the motion along other normal modes. The vibrational/rotational/translational motion of the atoms in a molecule can be always represented using linear combination of all the normal modes. Motion along a particular normal mode will change only a part to the potential, and the sum of all such contributions will then give the total potential.

1.1 Lennard Jones Potential

The Lenard Jones potential is given by:-

$$U(R_1, R_2, R_3, ..., R_n) = 4\epsilon \sum_{I<J} [(\frac{\sigma}{R_{ij}})^{12} - (\frac{\sigma}{R_{ij}})^6]$$

$$R_{ij} = |Ri - Rj|$$

ϵ=Depth of potential
σ=Distance before which the repulsive term in the potential contributes

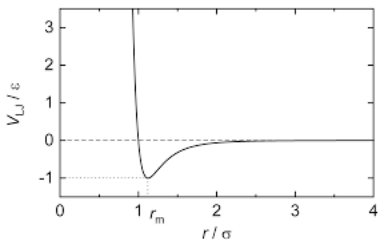


Figure 1: Lennard-Jones potential

1.2 BFGS based Quasi-Newton-Raphson Method for optimisation

Steps involved in this method are as Follows:

- Choosing random starting point x_0

- Estimating inverse of Hessian H^{-1} using BFGS(Broyden–Fletcher–Goldfarb–Shanno) Method.
- Calculating step size λ using line-search methods. In this project Backtraking line search is used.
- calculating new x using following equation

$$x_{n+1} = x_n + \lambda H^{-1} p$$

- Iterating till convergence.

1.3 Force Constant Matrix

Force constant matrix **K** is defined as

$$\mathbf{K} = M^{-1/2} H M^{-1/2}$$

Here, **H** is a $3N \times 3N$ hessian matrix and **M** is a $3N \times 3N$ mass matrix. Eigenvectors of this matrix **K** are normal modes for given molecule. Frequency corresponding to particular mode is calculated as

$$\nu_i = \frac{1}{2\pi}\sqrt{\lambda_i}$$

Here, λ_i is the eigenvalue.

2 Methods and Models

2.1 Steps Involved

- Using BFGS based Quasi-Newton-Raphson Method optimise the given initial structure to lowest energy state. Code for this part was already given. Only subroutines for calculating energy and gradient of energy are written by me(Explained in further section).
- Once we get optimised structure from previous state, calculating hessian matrix and thus force-constant matrix for optimised structure.
- Obtaining eigenvalues and corresponding eigenvectors of **K** matrix using qr factorisation module.
- Calculating frequencies of normal mode using above equation.
- Printing 'vib.log' and 'vib.nmd' using provided subroutines.
- Visualising 'vib.log' in MOLDEN and 'vib.nmd' in VMD.

2.2 Explanation of written Code

2.2.1 MODULE Function_Derivative

In this module, first function calculates potential of a system using lennar-jones equation. Code iterates over all possible pairs, calculates their contribution in potential and keeps adding it in final potential.

```
!This is a function which calculates total energy
  FUNCTION func(x) RESULT(f)
    USE kinds
    USE system
    USE distance

    IMPLICIT NONE
    REAL(KIND=dp) :: f
    REAL(KIND=dp), DIMENSION(:), POINTER :: x
    real(KIND = dp) :: d, d2, d6, d12
    integer :: i, j
    f = 0.d0
    do i = 1, n-1
      do j = i+1, n
        d = x((i-1)*3+1:3*i) .distance. x((j-1)*3+1:3*j)
        d2=1.d0/(d**2)
        d6=d2*d2*d2
        d12=d6*d6
        f=f+d12-d6
      end do
    end do
    f = f*4.d0
  END FUNCTION func
```

Figure 2: Function to calculate potential energy of system

Another subroutine in same module takes positions of atoms as argument and returns gradient of energy with respect to each atom in each direction. In this subroutine loop iterates over all combination for each atom and calculates gradient of energy wrt to each atom in each direction. Code is given in Figure 3.

2.2.2 Program to calculate K and normal modes

At first this program, read optimised coordiantes of atoms from 'optimise.xyz' file and saves it. After this Hessian matrix is calculated as per following formula

$$H_{ij} = \frac{1}{2\Delta q}(g_j^+ - g_j^-)$$

Here $\Delta q = 0.01\sigma$ and g_j is gradient wrt j^{th} element. After this, program calculates eigenvalue and eigenvectors using MODULE qr. It prints all eigenvalues and eigenvectors on screen after calculation is done. Now, frequencies are calculated and unit conversion is done. Next, program calls subroutine to print 'vib.log' and 'vib.nmd' files. Code of this programm is attached in appendix section.

```
!This is a subroutine which calculates gradient vector "g"
!x: Input Coordinates. It should be an array of rank 1 of size 3N
!g: Output Gradients.  It is an array of rank 1 of size 3N
  SUBROUTINE grad(x,g)
    USE kinds
    USE system
    USE distance
    IMPLICIT NONE
    REAL(KIND=dp), POINTER, DIMENSION(:) :: g, x
    real(kind=dp) :: d, d2, d6, d12, dum(3)
    integer :: i, j

    g(1:3*n) = 0.d0
!WRITE IT YOURSELF
    do i = 1, n
      do j = 1, n
        if(i .ne. j) then
        d = x((i-1)*3+1:3*i) .distance. x((j-1)*3+1:3*j)
        dum(1:3) = x((i-1)*3+1:3*i) - x((j-1)*3+1:3*j)
        d2=1.d0/(d**2)
        d6=d2*d2*d2
        d12=d6*d6
        g((i-1)*3+1:3*i) = g((i-1)*3+1:3*i) + (-2.D0*d12+d6)*dum(:)*d2
        dum(:) = 0
        end if
      end do
    end do
  END SUBROUTINE grad

END MODULE
```

Figure 3: Subroutine to calculate gradient of energy

3 Results

3.1 Structure optimisation

Potential energy for initial structure = 95.287849
Potential energy for optimised structure = -12.712062
Norm of gradient initially = 14.305681
Norm of gradient finally = 0.000040
Coordinates for Initial and optimised structure is shown in Figure 4.

```
LJ 0.000000 0.000000 0.000000
LJ 0.000000 0.000000 1.000000
LJ 1.100000 0.000000 0.000000
LJ 1.100000 -0.002094 1.199998
LJ 0.646341 0.541161 0.708052
LJ 0.453487 -0.542540 0.707107
```

(a) Initial structure

```
LJ      0.024366      0.029688      0.013287
LJ     -0.037842      0.040848      1.128979
LJ      1.137762     -0.041998      0.076048
LJ      1.075592     -0.030853      1.191750
LJ      0.601026      0.787853      0.597510
LJ      0.498923     -0.789011      0.607582
```

(b) Optimised structure

Figure 4: Coordinates of atoms

3.2 Normal mode Frequencies

Mode	Frequency in cm^{-1}	Mode	Frequency in cm^{-1}	Mode	Frequency in cm^{-1}
Mode 1	65.8208	Mode 7	47.2631	Mode 13	-0.9434
Mode 2	57.6206	Mode 8	32.9682	Mode 14	-0.9398
Mode 3	57.6405	Mode 9	32.9658	Mode 15	-0.5587
Mode 4	57.6417	Mode 10	31.9769	Mode 16	-6.4829×10^{-7}
Mode 5	47.2434	Mode 11	32.3588	Mode 17	4.7096×10^{-7}
Mode 6	47.2754	Mode 12	31.3349	Mode 18	-3.6762×10^{-7}

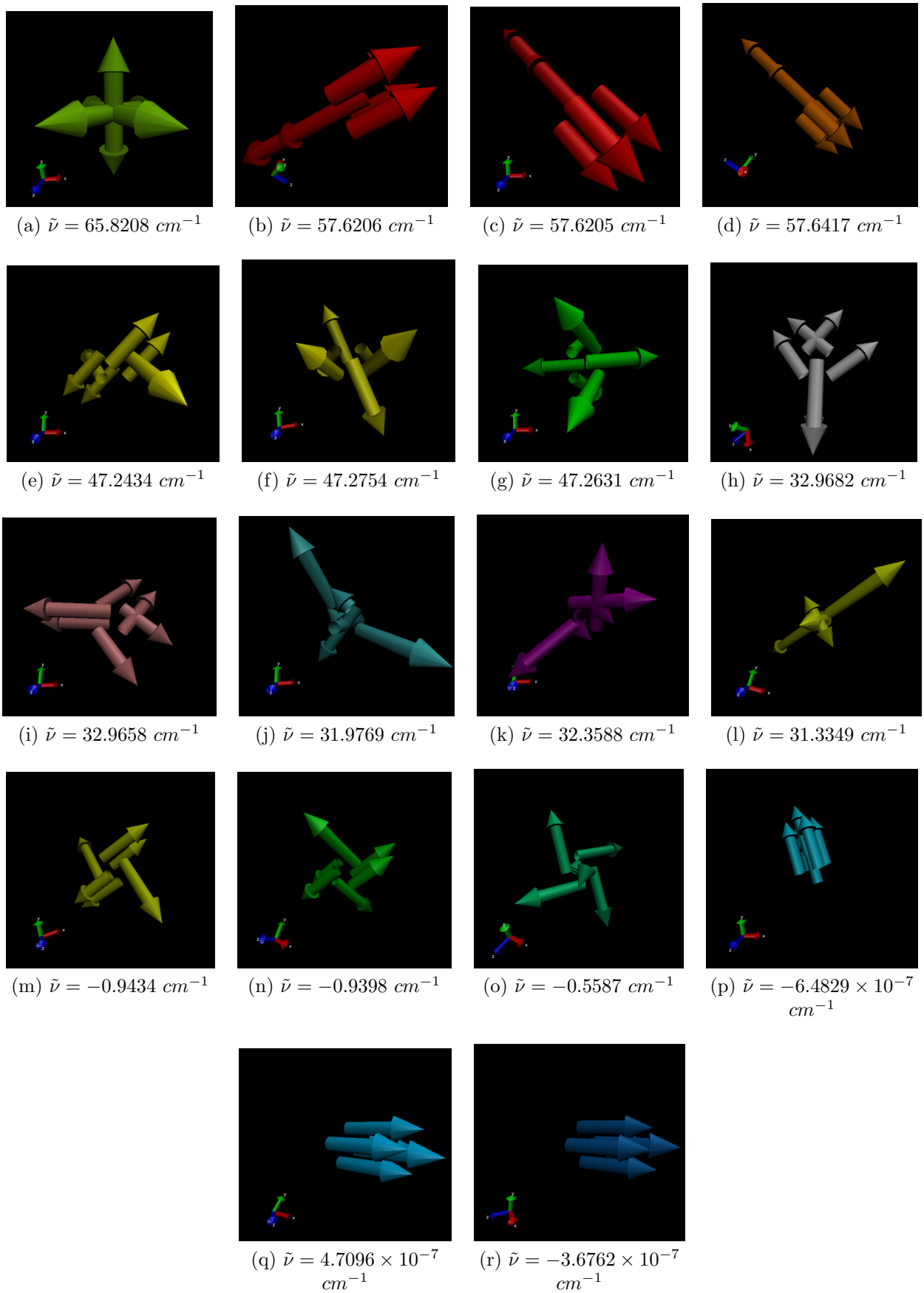
Table 1: Normal mode Frequncies

3.3 Visualisation of Normal modes

Visualisation of all the normal modes in VMD is as shown as in Figure 5 on next page. We can observe that, last three modes corresponds to translational motion of cluster along same direction. This gives explanation to nearly zero eigenvalues for these modes. Animation of all these normal modes can be seen here.

4 Conclusion

In this project, I successfully optimised the given cluster of atoms to have minimum energy. Obtained normal modes and normal mode frequencies of cluster using force constant matrix and visualised them using MOLDEN and VMD.



(a) $\tilde{\nu} = 65.8208\ cm^{-1}$ (b) $\tilde{\nu} = 57.6206\ cm^{-1}$ (c) $\tilde{\nu} = 57.6205\ cm^{-1}$ (d) $\tilde{\nu} = 57.6417\ cm^{-1}$

(e) $\tilde{\nu} = 47.2434\ cm^{-1}$ (f) $\tilde{\nu} = 47.2754\ cm^{-1}$ (g) $\tilde{\nu} = 47.2631\ cm^{-1}$ (h) $\tilde{\nu} = 32.9682\ cm^{-1}$

(i) $\tilde{\nu} = 32.9658\ cm^{-1}$ (j) $\tilde{\nu} = 31.9769\ cm^{-1}$ (k) $\tilde{\nu} = 32.3588\ cm^{-1}$ (l) $\tilde{\nu} = 31.3349\ cm^{-1}$

(m) $\tilde{\nu} = -0.9434\ cm^{-1}$ (n) $\tilde{\nu} = -0.9398\ cm^{-1}$ (o) $\tilde{\nu} = -0.5587\ cm^{-1}$ (p) $\tilde{\nu} = -6.4829 \times 10^{-7}\ cm^{-1}$

(q) $\tilde{\nu} = 4.7096 \times 10^{-7}\ cm^{-1}$ (r) $\tilde{\nu} = -3.6762 \times 10^{-7}\ cm^{-1}$

Figure 5: Normal modes of cluster of 6 Ar atoms

Acknowledgement

I would like to thank Prof. Nisanth Nair and Dr. Saurabh Srivastav for Motivating and guiding me throughout the project.

Appendix

Complete code can be accessed here: MODE-ANALYSIS-OF-A-MOLECULE

```
program project3
use kinds
use system
use function_derivative
use qr
use molden_output
use nmd
implicit none

integer :: i, j, k
real(KIND = dp), pointer :: x(:), H(:,:), gplus(:), gminus(:), xdum(:),
                            A(:,:), q(:,:), r(:,:), u(:,:), frq(:)
real(KIND = dp), pointer :: xc(:), yc(:), zc(:), mass(:), M(:,:)
character(len=2) :: at

open(3, file = "optimize.xyz")
read(3,*)n
rewind(3)
print *, "Number of atoms: ", n
allocate(x(3*n), H(3*n,3*n), gplus(3*n), gminus(3*n), xdum(3*n),
        A(3*n,3*n), q(3*n,3*n), r(3*n,3*n), u(3*n,3*n), frq(3*n))
allocate(M(3*n,3*n))

do i = 1, 36
    read(3,*)
    read(3,*)
    do j = 1, n
        read(3,*)at, x(3*(j-1)+1:3*j)
    end do
end do
close(3)

H(:,:) = 0.0d0
xdum(:) = x(:)
do i = 1, 3*n
    do j = 1, 3*n
```

```
            xdum(i) = x(i) + 0.01d0
            call grad(xdum, gplus)
            xdum(i) = x(i) - 0.01d0
            call grad(xdum, gminus)
            H(i,j) = (gplus(j)-gminus(j))/(2*0.01d0)
            xdum(:) = x(:)
        end do
    end do

    print *,"Hessian Matrix = "
    do i = 1, 3*n
        print "(18F10.6)", H(i,1:3*n)
    end do

    M(:,:) = 0.d0
    do i = 1, 3*n
        M(i,i) = (6.63E-26)**(-0.5)
    end do
    A = MATMUL(M, H)
    A = MATMUL(A, M)

    print *, "------------------------------------------------------------------------"

    !A(:,:) = H(:,:)
    DO k=1,20
        CALL qrd(a,q,r)
        a=MATMUL(r,q)
        IF(k>1)THEN
            r(:,:)=u(:,:)
            u=MATMUL(r,q)
        ELSE
           u(:,:)=q(:,:)
        END IF
    END DO

    PRINT *, "********************eigenvalues****************"
    PRINT "(18F10.6)", (A(k,k), k=1,3*n)
    PRINT *, "********************eigenvectors****************"
    do i=1,3*n
        PRINT "(18F10.6)", u(i,1:3*n)
    end do

    print *, "------------------------------------------------------------------------"
```

```
do i = 1, 3*n
    if(A(i,i) .lt. 0.d0)then
        A(i,i) = -1*A(i,i)
        frq(i) = -1*dsqrt(A(i,i))/(2*pi)
    else
        frq(i) = dsqrt(A(i,i))/(2*pi)
    end if
end do


frq(:) = frq(:)/(3E10)

print *, "Frequencies: "
do i = 1, 3*n
print "(2ES16.6)", frq(i)
end do
!-----------------------------------------------------------------------

allocate(al(n), xc(n), yc(n), zc(n), mass(n))

al(1:n) = "Ar"
do i = 1, n
    xc(i) = x(3*i - 2)
    yc(i) = x(3*i -1)
    zc(i) = x(3*i)
end do
mass(1:n) = 39.948d0

call create_molden_vib_output(frq,u,al,xc,yc,zc,mass)

call print_nmd(al, x, frq, u)

end program project3
```