PHY-563-PROJECT

Title- "Checking mechanical stability for crystal symmetries and Calculation of mechanical properties of materials"

Vivek Variar-1710110394

Malay Singh-1710110195

Aim of the program:

Given the space group or lattice vector matrix of a compound , and the elasticity coefficient matrix of the compound , the program does the following :

1. Identifies the crystal lattice/symmetry type of the compound.

2. Checks mechanical stability of the compound.

3. The Reuss bound, Voigt bounds of the Bulk modulus and Shear modulus of the compound are calculated and displayed. Two different sets of formulae are used to find the bounds of these moduli, and the results are compared.

4. Using the Reuss-Voigt-Hill approximations, values of the Bulk modulus , Shear modulus, Young's modulus and Poisons ratio of the compound are calculated and displayed

Theory:

Given the lattice vector array matrix of a compound, one can find the crystal symmetry of the compound. Each crystal symmetry has a distinct general lattice vector array matrix, which can be used to identify the crystal symmetry.

Each of 230 space groups (in 3 dimensions) fall into one of the 7 crystal systems (triclinic , Monoclinic , Orthorhombic , Tetragonal , Trigonal , Hexagonal , Cubic).

Mechanical stability conditions: These are a set of conditions which the elements of the C matrix of a compound have to satisfy in order for the compound to be mechanically stable. These conditions vary for different crystal systems. The conditions for cubic phase system are shown below as an example. The conditions are taken from- [2]

$$C_{11} > 0, \quad C_{44} > 0, \quad C_{11} > |C_{12}|, \quad (C_{11} + 2C_{12}) > 0.$$

Voigt bound : It is the upper limit of the effective moduli. It is obtained by the average polycrystalline moduli based on assumption of uniform strain throughout a polycrystal.

Reuss bound: It is the lower limit of the effective moduli. It is based on the assumption of uniform stress throughout the polycrystal.

One set of formulae used to calculate the reuss and voigt bounds of the moduli are the following. These formulae are valid for all types of crystal systems. Formulae from – [4]

$$B_R = [S_{11} + S_{22} + S_{33} + 2(S_{12} + S_{13} + S_{23})]^{-1}$$

$$B_V = \frac{1}{9}[C_{11} + C_{22} + C_{33} - 2(C_{12} + C_{13} + C_{23})]$$

$$G_R = 15[4(S_{11} + S_{22} + S_{33}) - 4(S_{12} + S_{13} + S_{23}) + 3(S_{44} + S_{55} + S_{66})]^{-1}$$

$$G_V = \frac{1}{15}[C_{11} + C_{22} + C_{33} - (C_{12} + C_{13} + C_{23}) + 3(C_{44} + C_{55} + C_{66})]$$

The second set of the formulae used vary according to the type of crystal system. The formulae for orthorhombic systems are shown below as an example. The formulae are taken from – [2]

$$B_V = (1/9)[C_{11} + C_{22} + C_{33} + 2(C_{12} + C_{13} + C_{23})],$$

$$G_V = (1/15)[C_{11} + C_{22} + C_{33} + 3(C_{44} + C_{55} + C_{66}) - (C_{12} + C_{13} + C_{23})],$$

$$B_R = \Delta[C_{11}(C_{22} + C_{33} - 2C_{23}) + C_{22}(C_{33} - 2C_{13}) - 2C_{33}C_{12} + C_{12}(2C_{23} - C_{12}) + C_{13}(2C_{12} - C_{13}) + C_{23}(2C_{13} - C_{23})]^{-1},$$

$$G_R = 15\{4[C_{11}(C_{22} + C_{33} + C_{23}) + C_{22}(C_{33} + C_{13}) + C_{33}C_{12} - C_{12}(C_{23} + C_{12}) - C_{13}(C_{12} + C_{13}) - C_{23}(C_{13} + C_{23})]/\Delta + 3[(1/C_{44}) + (1/C_{55}) + (1/C_{66})]\}^{-1},$$

$$\Delta = C_{13}(C_{12}C_{23} - C_{13}C_{22}) + C_{23}(C_{12}C_{13} - C_{23}C_{11}) + C_{33}(C_{11}C_{22} - C_{12}^2).$$

Voigt-Reuss-Hill approximations: The average of Voigt and Reuss bound. This approximation is used to obtain the B (Bulk modulus) , G (Shear modulus) . Using B and G obtained , E (Young's modulus) and v (Possion's ratio) are calculated.

In terms of the Voigt-Reuss-Hill approximations,[52] $M_H = (1/2)(M_R + M_V)$, $M = B, G$. Young's modulus E and Possion's ratio ν are obtained by the following formulas:

$$E = 9BG/(3B + G), \quad \nu = (3B - 2G)/[2(3B + G)].$$

Code explanation:

Subroutine initialize: Fills in matrix C, the values of the elasticity coefficients for different compounds

Subroutine symmetryidentifier : Takes input of the lattice vector array matrix from a text file, identifies the crystal system and outputs an integer which represents the type of crystal system , using if-else statements.

Subroutine spacegroup : Takes input of the space group of the compound from the user , and identifies the crystal system to which the particular space group belongs. A text file contains all 230 space groups in matrix form (46 into 5). The input taken from the user is compared with every space group in the text file and if a match is found, the position of the matched space group is filled in h, k . If else statements are then used to match the h,k to the correct crystal system. Subroutine outputs an integer representing the crystal system to which the space group belongs.

Subroutine findinverse: Takes C , and dimension of C as input and outputs its inverse matrix (elastic compliance coefficient matrix). Online resources were used to understand L-U decomposition and fix some errors present in the subroutine.

Subroutine stability: The purpose of this subroutine was to check whether the given lattice was in a stable configuration or not. We used the conditions given in [2] for this. These include stability conditions for the following 5 lattices: *Cubic, Hexagonal, Tetragonal, Orthorhombic* and *Monoclinic*.
We inputted an integer value 'st' which denotes the lattice type and printed if the lattice is stable or not. We initialized a flag variable to zero which was changed to one only if a stable condition was satisfied. At the end of the subroutine, a flag zero indicated unstable compound whereas a flag one indicated a stable compound.

Subroutine reuss: A simple subroutine which followed [4] and used the simple formula for Bulk modulus and Shear modulus via Reuss method. It doesn't have different equations for different types of lattice.

Subroutine voigt: Like the previous subroutine, it is also a simple subroutine which has one equation each for Bulk and Shear modulus using Voigt method.

Subroutine reussphysrev: This used the equations given in [2]. There were different equations for Bulk and Shear modulus for different lattice type (the 5 types for which stability conditions have been calculated).

Subroutine voigtphysrev: This also used the equation given in [2] for the Voigt method to calculate Bulk and Shear modulus of different lattice type (the 5 types for which stability conditions have been calculated).

In the main program we average the values of Bulk and shear modulus obtained via Reuss and Voigt method. We do this separately for paper [2] and [4]. Then we calculate the values of Young's modulus and Poisson's ratio separately for both papers.

Results

The results were as we expected them to be. We checked for various Li-Sn compounds using data given in [4]. The results matched and the percentage difference between the values obtained by following equations given in [2] and [4] was of the order of 10^-6 or less. Thus we were able to successfully code a program using multiple subroutines to fulfill the aim of detecting the lattice type of given compound either by lattice symmetry or space group entered by user. We identified if the compound was stable and output the values of 4 properties of the compound namely- Bulk modulus, shear modulus, Young's modulus and Poisson's ratio.

Sources / References :

[1]- **Necessary and sufficient elastic stability conditions in various crystal systems** by F´elix Mouhat and Franc¸ois-Xavier Coudert.
[2]- **Crystal structures and elastic properties of superhard IrN_2 and IrN_3 from first principles** by Zhi-jian Wu.
[3]- https://www.youtube.com/watch?v=m3EojSAgIao , https://ww2.odu.edu/~agodunov/computing/programs/book2/Ch06/Inverse
[4]- **Understanding the Lithiation of Sn Anode for High Performance Li-ion Batteries with Exploration of Novel Li-Sn Compounds at Ambient and Moderately High Pressure** by Raja Sen and Priya Johari.

Additional:

LU decomposition method used to find inverse of a test 3 by 3 matrix :

(From next page)

- Find Inverse of matrix A using LU decomposition method

- Finding the L and U matrices:

$$A = \begin{pmatrix} 25 & 5 & 1 \\ 64 & 8 & 1 \\ 144 & 12 & 1 \end{pmatrix}$$

$$L = \begin{pmatrix} 1 & 0 & 0 \\ - & 1 & 0 \\ - & - & 1 \end{pmatrix}$$

, L_{21}, L_{31}, L_{32} we will find and fill

U should be of the form:

$$U = \begin{pmatrix} - & - & - \\ 0 & - & - \\ 0 & 0 & - \end{pmatrix}$$

Now for getting U,

$$\begin{pmatrix} 25 & 5 & 1 \\ 64 & 8 & 1 \\ 144 & 12 & 1 \end{pmatrix}$$ $R_2 \rightarrow R_2 - Cf.R_1$, Cf such that element (2,1) is 0

$$= \begin{pmatrix} 25 & 5 & 1 \\ 0 & -4.8 & -1.56 \\ 144 & 12 & 1 \end{pmatrix}$$, $Cf = \frac{64}{25} = 2.56$

$L_{21} = 2.56$

$$\begin{pmatrix} 25 & 5 & 1 \\ 0 & -4.8 & -1.56 \\ 144 & 12 & 1 \end{pmatrix}$$

$R_3 \to R_3 - Cf.\ R_1$ such that element $(3,1)$ is 0.

$$\to \begin{pmatrix} 25 & 5 & 1 \\ 0 & -4.8 & -1.56 \\ 0 & -16.8 & -4.76 \end{pmatrix}$$

$Cf = \frac{144}{25} = 5.76$

$L_{31} = 5.76$

$$\to \begin{pmatrix} 25 & 5 & 1 \\ 0 & -4.8 & -1.56 \\ 0 & -16.8 & -4.76 \end{pmatrix}$$

$R_3 \to R_3 - Cf.\ R_2$ s.t element $(3,2)$ is 0

$$\to \begin{pmatrix} 25 & 5 & 1 \\ 0 & -4.8 & -1.56 \\ 0 & 0 & 0.7 \end{pmatrix}$$

$Cf = \frac{-16.8}{-4.8} = 3.5$

$L_{32} = 3.5$

$\therefore$ $$U = \begin{pmatrix} 25 & 5 & 1 \\ 0 & -4.8 & -1.56 \\ 0 & 0 & 0.7 \end{pmatrix}$$

$$L = \begin{pmatrix} 1 & 0 & 0 \\ 2.56 & 1 & 0 \\ 5.76 & 3.5 & 1 \end{pmatrix}$$

Let B is inverse matrix of A,

$$B = \begin{pmatrix} B_{11} & B_{12} & B_{13} \\ B_{21} & B_{22} & B_{23} \\ B_{31} & B_{32} & B_{33} \end{pmatrix}$$

Now, we know

$$\begin{pmatrix} & A & \end{pmatrix}\begin{pmatrix} B_{11} \\ B_{21} \\ B_{31} \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}$$

$$\rightarrow \begin{pmatrix} 25 & 5 & 1 \\ 64 & 8 & 1 \\ 144 & 12 & 1 \end{pmatrix}\begin{pmatrix} B_{11} \\ B_{21} \\ B_{31} \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}$$

$A = LU$ $\quad (X) \quad \cdot \quad (Y)$

$$LUX = Y$$

Taking $UX = Z$

$$LUX = Y \rightarrow LZ = Y$$

$$LZ = Y \Rightarrow \begin{pmatrix} 1 & 0 & 0 \\ 2.56 & 1 & 0 \\ 5.76 & 3.5 & 1 \end{pmatrix}\begin{pmatrix} Z_1 \\ Z_2 \\ Z_3 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}$$

(Forward substitution)

$z_1 = 1$

$2.56 z_1 + z_2 = 0 \rightarrow z_2 = -2.56$

$5.76 z_1 + 3.5 z_2 + z_3 = 0$

$\rightarrow 5.76 + 3.5(-2.56) + z_3 = 0$

$\rightarrow z_3 = 3.2$

$$\rightarrow \begin{pmatrix} z_1 \\ z_2 \\ z_3 \end{pmatrix} = \begin{pmatrix} 1 \\ -2.56 \\ 3.2 \end{pmatrix}$$

$UX = Z$

$$\rightarrow \begin{pmatrix} 25 & 5 & 1 \\ 0 & -4.8 & -1.56 \\ 0 & 0 & 0.7 \end{pmatrix} \begin{pmatrix} X_1 \\ X_2 \\ X_3 \end{pmatrix} = \begin{pmatrix} 1 \\ -2.56 \\ 3.2 \end{pmatrix}$$

(Back substitution)

$0.7 X_3 = 3.2 \rightarrow X_3 = 4.571$

$-4.8 X_2 - 1.56 X_3 = -2.56$

$$\rightarrow X_2 = \frac{-2.56 + 1.56(4.571)}{-4.8} = -0.952$$

$25 X_1 + 5 X_2 + X_3 = 1$

$$\rightarrow X_1 = \frac{1 - 5(-0.952) - 4.571}{25} = 0.0476$$

$$\begin{pmatrix} X_1 \\ X_2 \\ X_3 \end{pmatrix} = \begin{pmatrix} 0.0476 \\ -0.952 \\ 4.571 \end{pmatrix} = \begin{pmatrix} B_{11} \\ B_{21} \\ B_{31} \end{pmatrix}$$

For 2nd column of B

$$\begin{pmatrix} 25 & 5 & 1 \\ 64 & 8 & 1 \\ 144 & 12 & 1 \end{pmatrix} \begin{pmatrix} B_{12} \\ B_{22} \\ B_{32} \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix}$$

$\uparrow$ (X) $\quad$ $\uparrow$ (Y)

Same process as for column 1 is followed :

$$\begin{pmatrix} X_1 \\ X_2 \\ X_3 \end{pmatrix} = \begin{pmatrix} 0.0833 \\ 1.417 \\ -5.0 \end{pmatrix} = \begin{pmatrix} B_{12} \\ B_{22} \\ B_{32} \end{pmatrix}$$

For 3rd column of B,

$$\begin{pmatrix} 25 & 5 & 1 \\ 64 & 8 & 1 \\ 144 & 12 & 1 \end{pmatrix} \begin{pmatrix} B_{13} \\ B_{23} \\ B_{33} \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}$$

$\uparrow$ (X) $\quad$ $\uparrow$ (Y)

$$\begin{pmatrix} X_1 \\ X_2 \\ X_3 \end{pmatrix} = \begin{pmatrix} 0.0357 \\ -0.4643 \\ 1.429 \end{pmatrix} = \begin{pmatrix} B_{13} \\ B_{23} \\ B_{33} \end{pmatrix}$$

$$B = A^{-1} = \begin{pmatrix} 0.0476 & 0.0833 & 0.0357 \\ -0.952 & 1.417 & -0.4643 \\ 4.571 & -5.0 & 1.429 \end{pmatrix}$$

Code:

```
program Project
   implicit none
   integer::ch
   real,dimension(6,6)::C,S
   real:: B,G,E,v,Br,Gr,Bv,Gv,Br1,Gr1, Bv1, Gv1, B1,G1, E1, v1,diffb,diffg,diffe,diffv
   integer::i,j, st,n=6
   logical::stab !used in stability subroutine


   C=0
   S=0


   call initialize(C)
   call findinverse(C,S,n)


   print*,"Enter :"
   print*,"1. For input of lattice vector array of material"
   print*,"2. For input of space group of material"


   read*,ch
   if(ch==1) then
     call symmetryidentifier(st)
      !print*,"Lattice number is ",st
   else if(ch==2)then
      call spacegroup(st)
   end if
```

```
   call stability(C,st,stab)

   call voigt(C,Bv,Gv)

   call voigtphysrev(C,Bv1,Gv1,st)

   call reuss(S,Br,Gr)

   call reussphysrev(C,Br1,Gr1,st)


   if (stab .eqv. .true.) then

      print*,"Lattice is stable."

   else if(stab .eqv. .false.) then

      print*,"Lattice is not stable."

   end if


   print*, "Voigt method. Bulk and Shear modulus from Physical review paper Zhi-jian
Wu:",Bv1, Gv1

   print*, "Voigt method. Bulk and Shear modulus from Ma'am's paper:",Bv, Gv

   print*, "Reuss method. Bulk and Shear modulus from Physical review paper Zhi-jian Wu:",
Br1,Gr1

   print*, "Reuss method. Bulk and Shear modulus from Ma'am's paper:", Br,Gr

   B1 = (Bv1 + Br1)/2

   G1 = (Gv1 + Gr1)/2

   E1 = 9*B1*G1/(3*B1 + G1)

   v1 = (3*B1 - 2*G1)/(2*(3*B1 + G1))


   B = (Bv + Br)/2

   G = (Gv + Gr)/2
```

```
   E = 9*B*G/(3*B + G)

   v = (3*B - 2*G)/(2*(3*B + G))

   print*, "The four mechanical properties using Physical review paper by Zhi-jian Wu are as follows: "

   print*,"Bulk Modulus: ",B1,"GPa"

   print*,"Shear Modulus: ",G1,"GPa"

   print*,"Young's modulus: ",E1,"GPa"

   print*,"Poisson's ratio: ",v1


   print*, "The four mechanical properties using Priya ma'am's paper are as follows: "

   print*,"Bulk Modulus: ",B,"GPa"

   print*,"Shear Modulus: ",G,"GPa"

   print*,"Young's modulus: ",E,"GPa"

   print*,"Poisson's ratio: ",v



   diffb= abs((B-B1))*100/B1

   print*,"Percetage difference between Bulk modulus what Ma'am's paper gives vs Zhi-jian Wu's paper gives:", diffb

   diffg= abs((G-G1))*100/G1

   print*,"Percetage difference between Bulk modulus what Ma'am's paper gives vs Zhi-jian Wu's paper gives:", diffg

   diffe= abs((E-E1))*100/E1

   print*,"Percetage difference between Bulk modulus what Ma'am's paper gives vs Zhi-jian Wu's paper gives:", diffe

   diffv= abs((v-v1))*100/v1

   print*,"Percetage difference between Bulk modulus what Ma'am's paper gives vs Zhi-jian Wu's paper gives:", diffv
```

```
end program Project


subroutine spacegroup(t) !returns an integer st indicating which lattice type given
spacegroup belongs to
   implicit none
   integer::choice
   integer,intent(out):: t
   character(len=17)::charch
   integer::i,j,h,k,flag=0
   character(len=17)::c(46,5)
   open(1857,file='spacegroupinputvvf.txt',status='old',action='read')
   print*,"Enter Space group"
   read*,charch
   do i=1,46
         read(1857,*)c(i,:)


   end do
   do i=1,46
      do j=1,5
         if(c(i,j)==charch) then
            h=i
            k=j
```

```
        end if
    end do
end do
if((h==1).AND.(k.GE.1).AND.(k.LE.2)) then
    print*,"Lattice is Triclinic "
    t=6
    flag=1

else if((h==1).AND.(k.GE.3).AND.(k.LE.5)) then
    print*,"Lattice is Monoclinic "
    t=5
    flag=1
else if((h.GE.2).AND.(h.LE.3).AND.(k.GE.1).AND.(k.LE.5)) then
    print*,"Lattice is Monoclinic "
    t=5
    flag=1
else if((h.GE.4).AND.(h.LE.14).AND.(k.GE.1).AND.(k.LE.5)) then
    print*,"Lattice is Orthorhombic "
    t=4
    flag=1
else if((h==15).AND.(k.GE.1).AND.(k.LE.4)) then
    print*,"Lattice is Orthorhombic "
    t=4
    flag=1
else if((h==15).AND.(k==5)) then
```

```
    print*,"Lattice is Tetragonal "
    t=3
    flag=1
else if((h.GE.16).AND.(h.LE.28).AND.(k.GE.1).AND.(k.LE.5)) then
    print*,"Lattice is Tetragonal "
    t=3
    flag=1
else if((h==29).AND.(k.GE.1).AND.(k.LE.2)) then
    print*,"Lattice is Tetragonal "
    t=3
    flag=1
else if((h==29).AND.(k.GE.3).AND.(k.LE.5)) then
    print*,"Lattice is Trigonal "
    t=7
else if((h.GE.30).AND.(h.LE.33).AND.(k.GE.1).AND.(k.LE.5)) then
    print*,"Lattice is Trigonal "
    t=7
    flag=1
else if((h==34).AND.(k.GE.1).AND.(k.LE.2)) then
    print*,"Lattice is Trigonal "
    t=7
    flag=1
else if((h==34).AND.(k.GE.3).AND.(k.LE.5)) then
    print*,"Lattice is Hexagonal "
    t=2
    flag=1
```

```
   else if((h.GE.35).AND.(h.LE.38).AND.(k.GE.1).AND.(k.LE.5)) then
      print*,"Lattice is Hexagonal "
      t=2
      flag=1
   else if((h==39).AND.(k.GE.1).AND.(k.LE.4)) then
      print*,"Lattice is Hexagonal "
      t=2
      flag=1
   else if((h==39).AND.(k==5)) then
      print*,"Lattice is Cubic "
      t=1
      flag=1
   else if((h.GE.40).AND.(h.LE.46).AND.(k.GE.1).AND.(k.LE.5)) then
      print*,"Lattice is Cubic "
      t=1
      flag=1
   end if

   if(flag==0) then
      print*,"Invalid spacegroup entered."
      stop
   end if

end subroutine spacegroup

subroutine symmetryidentifier(st)
```

```
implicit none
integer,intent(out)::st
integer::j,i
real,allocatable, dimension(:,:)::L
real::v
allocate(L(3,3))
open(193,file='latticevinputf.txt',status='old',action='read')
v=3.0
do j= 1,3
   read(193,*)L(j,:)
end do
print*,"Lattice array:"
do j= 1,3
   print*,L(j,:)
end do

if(L(1,2)==L(1,3).AND.L(1,3)==L(2,1).AND.L(2,1)==L(2,3).AND.L(2,3)==L(3,1).AND.L(3,1)==L(3,
2).AND.L(1,2)==0) then
    if(L(1,1)==L(2,2).AND.L(2,2)==L(3,3)) then
    print*,"Lattice is sc"
    i=1
    else if((L(1,1)==L(2,2)).AND.(L(2,2).NE.L(3,3))) then
    print*,"Lattice is tetragonal P"
    i=5
    else if((L(1,1).NE.L(2,2)).AND.(L(2,2).NE.L(3,3)).AND.(L(1,1).NE.L(3,3))) then
    print*,"Lattice is orthorombic P"
    i=7
```

```
    end if
else if((L(1,1).NE.L(2,2)).AND.(L(2,2).NE.L(3,3)).AND.(L(1,1).NE.L(3,3))) then
    if(L(1,1)==(-L(2,2)).AND.(L(3,2).NE.0)) then
    print*,"Lattice is fcc"
    i=2
    else if((L(1,2)==0).AND.(L(3,2).NE.0)) then
    print*,"Lattice is orthorombic fc"
    i=9
    else if(L(1,2)==L(2,2)) then
       if(L(2,3)==L(3,3)) then
          print*,"Lattice is orthorombic body centered"
          i=10
       else
          print*,"Lattice is orthorombic base centered"
          i=8
       end if

    else if((L(1,1)==(sqrt(v)/2*L(1,1))).AND.(L(2,1)==(-L(2,2)*sqrt(v)))) then
       print*,"Lattice is Hexagonal"
          i=4
    else if((L(1,2)==L(1,3)).AND.(L(1,3)==L(2,3)).AND.(L(2,3)==0)) then
       if(L(2,1)==0) then
       print*,"Lattice is Monoclinic p , unique axis b "
       i=13
       else if((L(3,1)==0).AND.(L(3,2)==0)) then
       print*,"Lattice is Monoclinic p , unique axis c "
```

```
        i=11
        else if(L(2,1).NE.0) then
        print*,"Lattice is Triclinic "
        i=14
        end if

    else if(L(1,1)==L(3,1).AND.(L(1,3)==(-L(3,3)))) then
        print*,"Lattice is Monoclinic base centered "
        i=12
    else if(L(1,1)==(-L(3,1)).AND.(L(1,3)==(-L(2,2)/2))) then
        print*,"Lattice is Trigonal R , 3 fold axis c  "
        i=15

    else
        print*,"Unidentified lattice type"
        i=20

    end if
else if((L(1,1)==L(2,2)).AND.(L(2,2)==L(3,3)).AND.(L(1,3).NE.0).AND.(L(2,1).NE.0)) then
    print*,"Lattice is bcc"
          i=3
else if((L(1,2)==L(1,3)).AND.(L(1,3)==L(2,3)).AND.(L(2,3)==0)) then
   if(L(2,1)==0) then
    print*,"Lattice is Monoclinic p , unique axis b "
    i=13
   else if((L(3,1)==0).AND.(L(3,2)==0)) then
```

```
     print*,"Lattice is Monoclinic p , unique axis c "
     i=11
    else if(L(2,1).NE.0) then
     print*,"Lattice is Triclinic "
     i=14
     else
        print*,"Unidentified lattice"
        i=20
     end if
else if(L(1,1)==L(3,1).AND.(L(1,3)==(-L(3,3)))) then
     print*,"Lattice is Monoclinic base centered "
     i=12
else if(L(1,1)==(-L(3,1)).AND.(L(1,3)==(-L(2,2)/2))) then
     print*,"Lattice is Trigonal R , 3 fold axis c  "
     i=15
else if((L(1,1)==L(2,2)).AND.(L(1,2)==(-L(2,1)))) then
     print*,"Lattice is Trigonal I,bct   "
     i=6
else
     print*,"Unidentified lattice"
     i=20
end if

if((i.GE.1).AND.(i.LE.3)) then
     st=1
else if(i==4) then
```

```
      st=2
  else if(i==5) then
      st=3
  else if((i.GE.7).AND.(i.LE.10)) then
      st=4
  else if((i.GE.11).AND.(i.LE.13)) then
      st=5
  else
      st=100
  end if



end subroutine symmetryidentifier


subroutine stability(coeff,st, stab)
   implicit none
   integer,intent(in):: st
   real,dimension(6,6),intent(in):: coeff
   logical,intent(out)::stab
   stab= .false.

   !Taking a value s. If s=1 then its cubic, s=2 implies hexagonal, s=3 implies tetragonal,s=4 implies orthorhombic,s=5     implies monoclinic
   if(st==1) then

      if (((coeff(1,1)-coeff(1,2)) > 0) .and. ((coeff(1,1)+2*coeff(1,2))>0) .and. ((coeff(4,4))>0)) then
```

```
          stab=.true.
        else
          stab=.false.

        end if
    else if (st==2) then
       if ((coeff(1,1)> abs(coeff(1,2))) .and. ((2*(coeff(1,3)**2))
<coeff(3,3)*(coeff(1,1)+2*coeff(1,2))) &
        .and. (coeff(4,4)>0 ) ) then

           stab=.true.
        else
          stab=.false.
        end if

    else if(st==3) then
        if ((coeff(1,1)> abs(coeff(1,2))) .and. ((2*(coeff(1,3)**2))
<coeff(3,3)*(coeff(1,1)+coeff(1,2))) &
        .and. (coeff(4,4)>0 ) .and. (coeff(6,6)>0) ) then

          stab=.true.
        else
          stab=.false.
        end if

    else if(st==4) then
        if (( coeff(1,1)>0) .and. (coeff(2,2)>0) .and.   (coeff(3,3)>0) .and. (coeff(4,4)>0) &
```

```
        .and.   (coeff(5,5)>0) .and. (coeff(6,6)>0) .and. &
        (coeff(1,1)+coeff(2,2)+ coeff(3,3) +2*(Coeff(1,2) +Coeff(1,3)+Coeff(2,3))>0) .and. &
        ((coeff(1,1)+coeff(2,2)-2*coeff(1,2))>0) .and. (Coeff(1,1)+Coeff(3,3)-2*Coeff(1,3)>0) &
        .and. (coeff(2,2)+coeff(3,3)-2* coeff(2,3)>0)  ) then

        stab=.true.
      else
        stab=.false.

      end if

    else if(st==5) then !monoclinic
      if ( (coeff(1,1)> 0) .and. (Coeff(2,2)>0) .and. (Coeff(3,3)> 0) .and. (Coeff(4,4)>0) &
        .and. (Coeff(5,5)> 0) .and. (Coeff(6,6)> 0) .and. &
        (Coeff(1,1) + Coeff(2,2) + Coeff(3,3) + 2*(Coeff(1,2) + Coeff(1,3) + Coeff(2,3))>0) .and.
((Coeff(3,3)*Coeff(5,5) &
        - Coeff(3,5)**2)>0) .and. ((Coeff(4,4)*Coeff(6,6)- Coeff(4,6)**2)>0) .and. &
         ((Coeff(2,2) + Coeff(3,3) -2*Coeff(2,3))>0)) then
       stab=.true.
      else
        stab=.false.
      end if
    else
      print*, "Stability condition and formulas for various modulus not given in paper."
      stop
    end if
```

```
end subroutine stability


subroutine initialize(C)
   implicit none
   real,dimension(6,6),intent(inout)::C
   integer:: i,j
   !inputting C matrix for different compounds given in Ma'am's research paper
   !3 belongs to trigonal which we do not have case for
   print*,"Enter 1- Sn, 2- Li2Sn5, 3- Li4Sn1, 4- Li, 5-Li5Sn1 , 6-Enter C matrix through text file."
   read*, i

   if (i==1) then
      C(1,1)=56.12
      C(2,2)=56.12
      C(3,3)=56.12
      C(4,4)=43.26
      C(5,5)=43.26
      C(6,6)=43.26
      C(1,2)=26.60
      C(1,3)=26.60
      C(2,1)=26.60
      C(2,3)=26.60
      C(3,1)=26.60
      C(3,2)=26.60
```

```
else if (i==2) then
    C(1,1)=79.03
    C(3,3)=95.24
    C(4,4)=19.85
    C(6,6)=14.93
    C(1,2)=37.11
    C(2,1)=37.11
    C(2,2)=79.03
    C(2,3)=15.45
    C(1,3)=15.45
    C(3,1)=15.45
    C(3,2)=15.45
    C(5,5)=19.85
else if(i==3) then
    C(1,1)=73.59
    C(2,2)=73.59
    C(3,3)=75.37
    C(4,4)=8.40
    C(5,5)=8.40
    C(6,6)=0

    C(1,2)=9.14
    C(1,3)=0.60
    C(1,4)=-9.28
    C(2,1)=9.14
```

C(2,3)=0.6

C(2,4)=9.28

C(3,1)=0.6

C(3,2)=0.6

C(4,1)=-9.28

C(4,2)=9.28

C(5,6)=-9.28

C(6,5)=-9.28

else if (i==4) then

C(1,1)=15.09

C(2,2)=15.09

C(3,3)=15.09

C(4,4)=7.26

C(5,5)=7.26

C(6,6)=7.26

C(1,2)=13.38

C(1,3)=13.38

C(2,1)=13.38

C(2,3)=13.38

C(3,1)=13.38

C(3,2)=13.38

else if (i==5) then

C(1,1)=58.33

C(2,2)=65.85

```
    C(3,3)=45.84
    C(4,4)=16.74
    C(5,5)=19.63
    C(6,6)=7.35
    C(1,2)=-5.87
    C(1,3)=12.71
    C(1,5)=1.41
    C(2,1)=-5.87
    C(2,3)=6.65
    C(2,5)=6.22
    C(3,1)=12.71
    C(3,2)=6.65
    C(3,5)=8.10
    C(5,1)=1.41
    C(5,2)=6.22
    C(5,3)=-8.10
    C(6,4)=8.22

else if(i==6) then
     open(1599,file='Cinputvf.txt',status='old',action='read')
     do j=1,6
       read(1599,*)C(j,:)
     end do
     print*,"C matrix from text file is :"
     do j=1,6
       print*,C(j,:)
```

```
        end do



    end if
end subroutine


subroutine findinverse(C,MI,n) !calculates and returns S matrix
    implicit none
    integer,intent(in)::n
    real,dimension(n,n), intent(in)::C
    real,dimension(n,n):: M
    real,intent(out)::MI(n,n)
    real::L(n,n),U(n,n)
    real::x(n),y(n),z(n)
    real::fac
    integer::i,j,k
    M=C
    y=0.0
    U=0.0
    L=0.0
    do k=1,n-1
       do i=k+1,n
          fac=M(i,k)/M(k,k)
          L(i,k)=fac
             do j=k+1,n
                M(i,j)=M(i,j)-(fac*M(k,j))
```

```
        end do
    end do
end do



do j=1,n
    do i=1,j
        U(i,j)=M(i,j)
    end do
end do
do i=1,n
    L(i,i)=1.0
end do

do k=1,n
    y(k)=1.0
    z(1)=y(1)
    do i=2,n
        z(i)=y(i)
        do j=1,i-1
            z(i)=z(i)-(L(i,j)*z(j))
        end do
    end do
    x(n)=z(n)/U(n,n)
    do i=n-1,1,-1
        x(i)=z(i)
```

```
            do j=n,i+1,-1
            x(i)=x(i)-(U(i,j)*x(j))
            end do
            x(i)=x(i)/U(i,i)
        end do

    do i=1,n
        MI(i,k)=x(i)
    end do
    y(k)=0.0

    end do

end subroutine findinverse

subroutine reuss(S,Br,Gr) !subroutine which return Br and Gr value according to Ma'am's paper
    implicit none
    real,dimension(6,6),intent(in)::S
    real,intent(inout)::Br,Gr
    real:: st
    integer:: i

    Br = 1/(S(1,1) + S(2,2) + S(3,3) + 2*(S(1,2) + S(1,3) + S(2,3)))
    Gr = 15/(4*(S(1,1) + S(2,2) + S(3,3))- 4*(S(1,2) + S(1,3) + S(2,3)) + 3*(S(4,4) + S(5,5) + S(6,6)))
```

```
end subroutine


subroutine reussphysrev(C,Br1,Gr1,st) !subroutine which return Br and Gr value according to Zhi's paper
    implicit none
    real,dimension(6,6)::C
    real,intent(inout)::Br1,Gr1
    real::M,C2,Bv1,a,b,c1,d,e,f,g, del
    integer,intent(in):: st
    !If st=1 then its cubic, st=2 implies hexagonal, st=3 implies tetragonal,st=4 implies orthorhombic,st=5 implies monoclinic
    if (st==1) then
        Br1 = (C(1,1) + 2*C(1,2))/3
        Gr1=(5*(C(1,1)-C(1,2))*C(4,4))/(4*C(4,4)+3*(C(1,1)-C(1,2)))
    else if (st==2) then
        C2=(C(1,1) +C(1,2))*C(3,3)-2*C(1,3)**2
        M=C(1,1) +C(1,2) + 2*C(3,3) -4*C(1,3)
        Bv1=(2*(C(1,1) + C(1,2)) + 4*C(1,3) + C(3,3))/9
        Br1=C2/M
        Gr1=(5/2)*(C2*C(4,4)*C(6,6))/(3*Bv1*C(4,4)*C(6,6) + C2*(C(4,4) + C(6,6)))
    else if(st==3) then
        C2=(C(1,1) +C(1,2))*C(3,3)-2*C(1,3)**2
        M=C(1,1) +C(1,2) + 2*C(3,3) -4*C(1,3)
        Bv1=(2*(C(1,1) + C(1,2)) + 4*C(1,3) + C(3,3))/9
        Br1=C2/M
        Gr1=15/((18*Bv1/C2 +(6/(C(1,1) - C(1,2))) + (6/C(4,4)) + (3/C(6,6))))
```

```
   else if(st==4) then

      del= C(1,3)*(C(1,2)*C(2,3) - C(1,3)*C(2,2) + C(2,3) * (C(1,2)*C(1,3)-C(2,3)*C(1,1))
+C(3,3)*(C(1,1)*C(2,2) -C(1,2))**2)

      Br1= del/(C(1,1)*C(2,2)+C(3,3) - 2*C(2,3)) + C(2,2)*(C(3,3)- 2*C(1,3)) - 2*C(3,3)*C(1,2) +
C(1,2)*(2*C(2,3) -c(1,2)) + &

         C(1,3)*(2*C(1,2) - C(2,3)*(2*C(1,3) - C(2,3)))

      Gr1= 15/(4*(C(1,1)*(C(2,2) + C(3,3) + C(2,3)) + C(2,2)*(C(3,3) + C(1,3)) + C(3,3)*C(1,2) -
C(1,2)*(C(2,3) + C(1,2)) &

         - C(1,3)*(C(1,2) +C(1,3)) - C(2,3)*(C(1,3)+C(2,3)))/del + 3*((1/C(4,4)) +(1/C(5,5))
+(1/C(6,6))))

   else if(st==5) then

      a= C(3,3)*C(5,5) - C(3,5)**2

      b= C(2,3)*C(5,5) - C(2,5)*C(3,5)

      c1= C(1,3)*C(3,5) - C(1,5)*C(3,3)

      d= C(1,3)*C(5,5)-C(1,5)*C(3,5)

      e= C(1,3)*C(2,5) - C(1,5)*C(2,3)

      f= C(1,1)*(C(2,2)*C(5,5) - C(2,5)**2) - C(1,2)*(C(1,2)*C(5,5) - C(1,5)*C(2,5)) +
C(1,5)*(C(1,2)*C(2,5) &

         - C(1,5)*C(2,2)) + C(2,5)*(C(2,3)*C(3,5) - C(2,5)*C(3,3))

      g = C(1,1)*C(2,2)*C(3,3) - C(1,1)*C(2,3)**2 - C(2,2)*C(1,3)**2 - C(3,3)*C(1,2)**2 +
2*C(1,2)*C(1,3)*C(2,3)

      del=2*(C(1,5)*C(2,5)*(C(3,3)*C(1,2)-C(1,3)*C(2,3))+C(1,5)*C(3,5)*(C(2,2)*C(1,3)-
C(1,2)*C(1,3))+ &

         C(2,5)*C(3,5)*(C(1,1)*C(2,3)-C(1,2)*C(1,3))) -(C(1,5)**2*(C(2,2)*C(3,3)-C(2,3)**2) &

         +C(2,5)**2*(C(1,1)*C(3,3) - C(1,3)**2) +C(3,5)**2 *(C(1,1)*C(1,2)-C(1,2)**2))
+g*C(5,5)

      Br1 = del/(a*(C(1,1) + C(2,2)- 2*C(1,2)) + b*(2*C(1,2)- 2*C(1,1) - C(2,3)) + c1*(C(1,5) -
2*C(2,5)) +d*(2*C(1,2) &
```

```
          + 2*C(2,3) - C(1,3) -2*C(2,2)) + 2*e*(C(2,5) - C(1,5))+f)

       Gr1 = 15/(4*(a*(C(1,1)+C(2,2)+C(1,2))+b*(C(1,1)-C(1,2)-C(2,3)) +c1*(C(1,5)+C(2,5))
+d*(C(2,2)-C(1,2)-C(2,3)-C(1,3))+ &

          e*(C(1,5)- c(1,5)) +f)/del + 3*((g/del) +(C(4,4) +C(6,6))/(C(4,4)*C(6,6)- C(4,6)**2)))


    end if



end subroutine


subroutine voigt(C,Bv,Gv) !subroutine which return Bv and Gv value according to Ma'am's
paper


    implicit none

    real,dimension(6,6),intent(in)::C

    real,intent(out)::Bv,Gv

  !  print*,C(1,1),C(2,2),C(3,3),C(1,2),C(1,3),C(2,3)

    Bv = (C(1,1) + C(2,2) + C(3,3)+ 2*(C(1,2) + C(1,3) + C(2,3)))/9

    Gv = (C(1,1) + C(2,2) + C(3,3)- (C(1,2) + C(1,3) + C(2,3)) + 3*(C(4,4) + C(5,5) + C(6,6)))/15


end subroutine


subroutine voigtphysrev(C,Bv1,Gv1,st) !subroutine which return Bv and Gv value according
to Zhi's paper

    implicit none

    real,dimension(6,6)::C

    real,intent(inout)::Bv1,Gv1
```

```
real::M
integer,intent(in):: st

!If st=1 then its cubic, st=2 implies hexagonal,
!st=3 implies tetragonal,st=4 implies orthorhombic,st=5 implies monoclinic
if (st==1) then
   Bv1 = (C(1,1) + 2*C(1,2))/3
   Gv1= (C(1,1) - C(1,2) + 3*C(4,4))/5

 else if (st==2) then
   M=C(1,1) +C(1,2) + 2*C(3,3) -4*C(1,3)
   Bv1=(2*(C(1,1) + C(1,2)) + 4*C(1,3) + C(3,3))/9
   Gv1 = (M + 12*C(4,4) + 12*C(6,6))/30
 else if (st==3) then
   M= C(1,1) +C(1,2) + 2*C(3,3) -4*C(1,3)
   Bv1=(2*(C(1,1) + C(1,2)) + 4*C(1,3) + C(3,3))/9
   Gv1= (M + 3*C(1,1)- 3*C(1,2)+ 12*C(4,4) + 6*C(6,6) )/30

 else if(st==4) then
   Bv1 = (C(1,1) + C(2,2) + C(3,3) + 2*(C(1,2) + C(1,3) + C(2,3)))/9
   Gv1 = (C(1,1) +C(2,2) + C(3,3) + 3*(C(4,4) +C(5,5) +C(6,6)) - (C(1,2) +C(1,3)+C(2,3)))/15
 else if(st==5) then
   Bv1 = (C(1,1) + C(2,2) + C(3,3) + 2*(C(1,2) + C(1,3) + C(2,3)))/9
   Gv1 = (C(1,1) +C(2,2) + C(3,3) + 3*(C(4,4) +C(5,5) +C(6,6)) - (C(1,2) +C(1,3)+C(2,3)))/15
end if
```

end subroutine