27-750, A.D. Rollett

Due: 16th February, 2020.

Homework 4, Single Slip, and Multiple Slip Crystal Plasticity

Q1. [30 points] *Single Slip: Calculating Schmid Factors*

In a single crystal tensile test on Ni, the orientation of the crystal is given as (18.89°, 68.58°, 11.31°) in Bunge Euler angles. Assume that an axisymmetric tensile stress is applied along the sample 3 (Z) direction. Note that we apply stress boundary conditions, as discussed in class.

(a) Determine which crystal direction is parallel to the *tensile axis*. Give your answer in the form of (hkl)[uvw] and simplify the numbers to be single digit integers. You should be able to modify the spreadsheet or program that you wrote in earlier homeworks to produce an answer.

(a) The Miller indices for this crystal orientation are: ( 1 5 2 ) [ 3 -1 1 ], based on a standard conversion of Euler angles to matrix, to Miller indices[[1]](#footnote-1)†. Then, obviously, the tensile axis is // [ 1 5 2 ].

(b) Identify which combination of slip plane and direction is active, and calculate the Schmid factor (to 3 significant figures).

(b) ( -1 1 1 ) [ 1 1 0 ]; m = 0.490.

Note: Ni is fcc and can slip on any {111} plane in any <110> direction. You will have to find out which is the most highly stressed slip system, i.e. find the largest Schmid factor. This value of the Schmid factor is what you should use to determine the tensile yield stress because it determines which slip system will be activated first.

(c) Calculate the tensile yield stress (to 3 significant figures) based on a critical resolved shear stress that we will (arbitrarily) set at 100 MPa.

(c) 204 MPa.



*You must submit a copy of the table of results showing how you calculated your answer which must display the maximum Schmid factor and the Miller indices of the activated slip system*.

Q2. [30 points] *Compute Slip Matrices; verify 384 non-singular cases out of 792*

Referring to slides 43 through 55 and #50 in particular, construct all the 5x5 slip matrices and determine which ones are non-singular. Verify that out of the 5C12 combinations, only 384 are valid solutions, i.e., have a non-zero determinant.

*Hints*:

Clearly you have to start by making a list of all 12 slip systems in fcc metals based on {111}<110>. Although this problem is deceptively simple, the main programming challenge is to compute a list of all the possible combinations of five slip systems (as opposed to just the number). Fortunately the web has code fragments that will do this (in C++, Fortran etc.), and Matlab has a built-in function.

*What to submit*:

You should submit your code (e.g., python or Matlab script), as well as the list of 384 non-singular cases.

Answer: See CalcTaylorMatrices.f90, where I chose to use Fortran to perform the calculations. I was able to find code for computing the combinations and already had a matrix inversion code (that Ben Anglin wrote).

Q3. [40 points] *Bishop-Hill Model: Calculation of Taylor Factors for Multiple Slip*

Assume that slip occurs on {111} planes and in <110> directions. For each of the three different strain types, uniaxial tension parallel to the Z axis, plane strain in the +X & -Z, and simple shear in the XZ sense (see the matrices below), calculate the following quantities.

(a) the index of the active stress state (from the Bishop-Hill list);

(b) the Taylor factor (to at least 2 decimal places) for crystals with the following orientations;

(c) the active multi-slip stress state (as in the values of A, B, C, F, G, & H);

 for crystals with the following orientations.

2.1 ( 0 0 1) [1 0 0]

2.2 ( 0 8 10) [1 0 0]

2.3 (-1 1 0) [1 1 1]

2.4 ( 2 1 3 ) [ 3 6 -4 ]

2.5 ( 1 1 2 ) [ 1 1 -1 ].

The strain tensors for the 3 different strain types are as follows.

Uniaxial tension // Z:



Plane strain compression on +X and -Z:



Simple Shear on XZ:



|  |  |  |  |
| --- | --- | --- | --- |
| Question | Orientation | Taylor Factor for Uniaxial Tension //Z | Stress State |
| 2.1a | ( 0 0 1) [1 0 0] | 2.449 | 1.0, -1.0, 0.0, 0.0, 0.0, 0.0 |
| 2.2a |  ( 0 8 10) [1 0 0] | 3.585 | 0.0, 0.0, 0.0, 1.0, 0.0, 0.0 |
| 2.3a |  (-1 1 0) [1 1 1] | 3.674 | (to be determined by you) |

Answers are provided (above) for a few of the cases so that you can check your method. The meaning of the Miller indices is the conventional one discussed earlier in the class: (hkl) // sample-Z (=sample-3), [uvw] // sample-X (= sample-1).

The pseudo-code to obtain the Taylor factor is given in the lecture notes.

If you consider uniaxial tension for the three corners of the unit triangle (i.e. 001, 110 and 111), in which two corners is the inverse Schmid factor equal to the Taylor factor, and in which corner are they different?

Answer: See the description in Hosford, Ch. 3. – his definition of the Taylor factor is rather obscure but the von Mises equivalent strain approach seems to work well enough. The Taylor factors appear to be identical with those in the books. The following results were obtained with Bishop\_Hill.f90.

|  |  |  |  |
| --- | --- | --- | --- |
| Uniaxial Tension | Orientation | Taylor Factor, M | Stress State (A, B, C, F, G, H) |
| 2.1a | (0 0 1) [1 0 0] | 2.449 | 1.0, -1.0, 0.0, 0.0, 0.0, 0.0 |
| 2.2a | (0 8 10) [1 0 0] | 3.585 | 0.0, 0.0, 0.0, 1.0, 0.0, 0.0 |
| 2.3a | (-1 1 0) [1 1 1] | 3.674 | 0.0, 0.0, 0.0,  0.5, -0.5, 0.5 |
| 2.4a | (2 1 3) [3 6 -4] | 3.149 | -0.5, 0.5, 0.0,  0.5, -0.5, 0.0 |
| 2.5a |  |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
| Plane Strain | Orientation | Taylor Factor | Stress State |
| 2.1b | ( 0 0 1) [1 0 0] | 2.121 | 1.0 -1.0 0.0 0.0 0.0 0.0 |
| 2.2b |  ( 0 8 10) [1 0 0] | 2.121 | 0.0 1.0 -1.0 0.0 0.0 0.0 |
| 2.3b | (-1 1 0)[1 1 1] | 3.526 | 0.0 0.0 0.0 0.0 0.0 1.0 |
| 2.4b |  (2 1 3) [3 6 -4] | 3.019 | -0.5 0.5 0.0 -0.5 -0.5 0.0 |
| 2.5b |  |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
| Simple Shear | Orientation | Taylor Factor | Stress State |
| 2.1c | ( 0 0 1) [1 0 0] | 4.243 | 0.0 0.0 0.0 0.0 1.0 0.0 |
| 2.2c | ( 0 8 10) [1 0 0] | 3.313 | 0.0 0.0 0.0 0.0 1.0 0.0 |
| 2.3c | (-1 1 0) [1 1 1] | 1.732 | 0.0 1.0 1.0 0.0 0.0 0.0 |
| 2.4c | (2 1 3) [3 6 -4] | 2.541 | 1.0 -1.0 0.0 0.0 0.0 0.0 |
| 2.5c |  |  |  |

Bishop & Hill stress states (Fortran style); factor of √6 *not* included:

 stress(1,1)=1.

 stress(1,2)=-1.

 stress(1,3)=0.

 stress(1,4)=0.

 stress(1,5)=0.

 stress(1,6)=0.

! # 1

 stress(2,1)=0.

 stress(2,2)=1.

 stress(2,3)=-1.

 stress(2,4)=0.

 stress(2,5)=0.

 stress(2,6)=0.

! # 2

 stress(3,1)=-1.

 stress(3,2)=0.

 stress(3,3)=1.

 stress(3,4)=0.

 stress(3,5)=0.

 stress(3,6)=0.

! # 3

 stress(4,1)=0.

 stress(4,2)=0.

 stress(4,3)=0.

 stress(4,4)=1.

 stress(4,5)=0.

 stress(4,6)=0.

! # 4

 stress(5,1)=0.

 stress(5,2)=0.

 stress(5,3)=0.

 stress(5,4)=0.

 stress(5,5)=1.

 stress(5,6)=0.

! # 5

 stress(6,1)=0.

 stress(6,2)=0.

 stress(6,3)=0.

 stress(6,4)=0.

 stress(6,5)=0.

 stress(6,6)=1.

! # 6

 stress(7,1)=0.5

 stress(7,2)=-1.

 stress(7,3)=0.5

 stress(7,4)=0.

 stress(7,5)=0.5

 stress(7,6)=0.

! # 7

 stress(8,1)=0.5

 stress(8,2)=-1.

 stress(8,3)=0.5

 stress(8,4)=0.

 stress(8,5)=-0.5

 stress(8,6)=0.

! # 8

 stress(9,1)=-1.

 stress(9,2)=0.5

 stress(9,3)=0.5

 stress(9,4)=0.5

 stress(9,5)=0.

 stress(9,6)=0.

! # 9

 stress(10,1)=-1.

 stress(10,2)=0.5

 stress(10,3)=0.5

 stress(10,4)=-0.5

 stress(10,5)=0.

 stress(10,6)=0.

! # 10

 stress(11,1)=0.5

 stress(11,2)=0.5

 stress(11,3)=-1.

 stress(11,4)=0.

 stress(11,5)=0.

 stress(11,6)=0.5

! # 11

 stress(12,1)=0.5

 stress(12,2)=0.5

 stress(12,3)=-1.

 stress(12,4)=0.

 stress(12,5)=0.

 stress(12,6)=-0.5

 stress(13,1)=0.5

 stress(13,2)=0.

 stress(13,3)=-0.5

 stress(13,4)=0.5

 stress(13,5)=0.

 stress(13,6)=0.5

 stress(14,1)=0.5

 stress(14,2)=0.

 stress(14,3)=-.5

 stress(14,4)=-0.5

 stress(14,5)=0.

 stress(14,6)=0.5

 stress(15,1)=0.5

 stress(15,2)=0.

 stress(15,3)=-0.5

 stress(15,4)=0.5

 stress(15,5)=0.

 stress(15,6)=-0.5

 stress(16,1)=0.5

 stress(16,2)=0.

 stress(16,3)=-0.5

 stress(16,4)=-0.5

 stress(16,5)=0.

 stress(16,6)=-0.5

 stress(17,1)=0.

 stress(17,2)=-0.5

 stress(17,3)=0.5

 stress(17,4)=0.

 stress(17,5)=0.5

 stress(17,6)=0.5

 stress(18,1)=0.

 stress(18,2)=-0.5

 stress(18,3)=0.5

 stress(18,4)=0.

 stress(18,5)=-0.5

 stress(18,6)=0.5

 stress(19,1)=0.

 stress(19,2)=-0.5

 stress(19,3)=0.5

 stress(19,4)=0.

 stress(19,5)=0.5

 stress(19,6)=-0.5

 stress(20,1)=0.

 stress(20,2)=-0.5

 stress(20,3)=0.5

 stress(20,4)=0.

 stress(20,5)=-0.5

 stress(20,6)=-0.5

 stress(21,1)=-0.5

 stress(21,2)=0.5

 stress(21,3)=0.

 stress(21,4)=0.5

 stress(21,5)=0.5

 stress(21,6)=0.

 stress(22,1)=-0.5

 stress(22,2)=0.5

 stress(22,3)=0.

 stress(22,4)=-0.5

 stress(22,5)=0.5

 stress(22,6)=0.

 stress(23,1)=-0.5

 stress(23,2)=0.5

 stress(23,3)=0.

 stress(23,4)=0.5

 stress(23,5)=-0.5

 stress(23,6)=0.

 stress(24,1)=-0.5

 stress(24,2)=0.5

 stress(24,3)=0.

 stress(24,4)=-0.5

 stress(24,5)=-0.5

 stress(24,6)=0.

 stress(25,1)=0.

 stress(25,2)=0.

 stress(25,3)=0.

 stress(25,4)=0.5

 stress(25,5)=0.5

 stress(25,6)=-0.5

 stress(26,1)=0.

 stress(26,2)=0.

 stress(26,3)=0.

 stress(26,4)=0.5

 stress(26,5)=-0.5

 stress(26,6)=0.5

 stress(27,1)=0.

 stress(27,2)=0.

 stress(27,3)=0.

 stress(27,4)=-0.5

 stress(27,5)=0.5

 stress(27,6)=0.5

 stress(28,1)=0.

 stress(28,2)=0.

 stress(28,3)=0.

 stress(28,4)=0.5

 stress(28,5)=0.5

 stress(28,6)=0.5

+++++++++++++++++++++++++++++++++++++++

program BishopHill

! gfortran -gdwarf-2 -O0 -o BishopHill Bishop\_Hill.f90

 ! solve the Bishop Hill model, adr 28 iv 01

 ! updated for crystal orientation, 21 iv 05

 ! formatted to f90, x 11

 implicit none

 real scalar

 real rtmp

 real :: stress,strain,dw,dwmax

 integer :: index,i,j,k,l

 dimension stress(28,6),strain(6)

 real :: a(3,3),hkl(3),uvw(3),t1(3)

 real :: eps(3,3),eps\_xtal(3,3)

 real :: strain\_xtal(6)

 real :: rtmp1,rtmp2,tayf

 real :: evm,enorm

 real :: sqr3

 real :: tayf2

 integer :: iq

 real :: phi1 , Phi, phi2

 real :: d1 , d2 , d3 , atranspose(3,3)

 integer :: iopt , ior , kerr

 character :: nomen\*1 = 'B'

 ! code::

 sqr3 = sqrt(3.)

 write(\*,\*) 'Welcome to the Reid version of the BH algorithm'

 print \* , 'Adapted to use the vM strain as the reference'

 write(\*,\*) 'Enter the strain as de11,de22,2\*de23,2\*de31,2\*de12'

 read(\*,\*) strain(1),strain(2),strain(4),strain(5),strain(6)

 strain(3) = 0.-strain(1)-strain(2)

 ! incompressibility

 write(\*,\*) 'Full strain vector: '

 print"(2x,6(2x,f7.3))", (strain(i),i=1,6)

 eps(1,1) = strain(1)

 eps(2,2) = strain(2)

 eps(3,3) = strain(3)

 eps(2,3) = strain(4)/2.

 eps(3,1) = strain(5)/2.

 eps(1,2) = strain(6)/2.

 eps(3,2) = strain(4)/2.

 eps(1,3) = strain(5)/2.

 eps(2,1) = strain(6)/2.

 ! evm = sqrt((2./9.\*((strain(1)-strain(2))\*\*2 +

 ! 1 (strain(2)-strain(3))\*\*2 + (strain(3)-strain(1))\*\*2)) +

 ! 2 ((eps(1,2)\*\*2 + eps(1,3)\*\*2 + eps(2,3)\*\*2)/3.))

 evm = (2./sqr3)\*sqrt(0.5\*(eps(1,1)\*\*2 + eps(2,2)\*\*2 + eps(3,3)\*\*2 &

 + eps(2,1)\*\*2 + eps(3,1)\*\*2 + eps(3,2)\*\*2 + &

 eps(1,2)\*\*2 + eps(1,3)\*\*2 + eps(2,3)\*\*2))

 print"(' eps vM = ',f7.3)",evm

 write(\*,\*)

 print \* , 'Enter orientation as (hkl)[uvw], =0, or as 3 Bunge angles, =1?'

 read \* , iq

 if ( iq == 0 ) then

 write(\*,\*) 'Enter (hkl) // sample.3 as 3 numbers'

 read(\*,\*) hkl(1),hkl(2),hkl(3)

 write(\*,"( 'Plane HKL = ',3(f8.1))") hkl

 call vecnorm(hkl)

10 write(\*,\*) 'Enter [uvw] // sample.1 as 3 numbers'

 read(\*,\*) uvw(1),uvw(2),uvw(3)

 write(\*,"( 'Plane UVW = ',3(f8.1))") uvw

 call vecnorm(uvw)

 rtmp=scalar(hkl,uvw)

 if(abs(rtmp).gt.1e-4) then

 write(\*,\*) 'Sorry, uvw is not perp. to hkl; try again'

 goto 10

 endif

 call vecpro2(hkl,uvw,t1)

 call vecnorm(t1)

 do 20, i=1,3

 a(i,1)=uvw(i)

 a(i,2)=t1(i)

 a(i,3)=hkl(i)

 ! a(i,3)=t1(i)

 ! a(i,2)=hkl(i)

20 enddo ! standard arrangement, not Canova matrix!

 elseif ( iq == 1 ) then

 print \* ,'Enter three Bunge angles in degrees:'

 read \*, phi1 , Phi, phi2

 iopt = 2

 call euler(atranspose,iopt,nomen,phi1 , Phi, phi2,ior,kerr)

 do i = 1,3

 do j = 1,3

 a(i,j) = atranspose(j,i)

 end do

 end do

 else

 print \* , 'invalid choice, stopping ...'

 stop

 end if

 print\*,' Orientation Matrix: '

 do i=1,3

 print"('[ ',3(1x,f7.3),' ]')", (a(i,j),j=1,3)

 enddo

 do i = 1,3

 do j = 1,3

 eps\_xtal(i,j) = 0.

 do k =1,3

 do l = 1,3

 eps\_xtal(i,j) = eps\_xtal(i,j) + a(i,k)\*a(j,l)\*eps(k,l)

 enddo ! L

 enddo ! K

 enddo ! J

 enddo ! I

 print\*,' Strain in Xtal coords: '

 do i=1,3

 print"('[ ',3(1x,f7.2),' ]')", (eps\_xtal(i,j),j=1,3)

 enddo

 evm = (2./sqr3)\*sqrt(0.5\*(eps\_xtal(1,1)\*\*2 + eps\_xtal(2,2)\*\*2 &

 + eps\_xtal(3,3)\*\*2 + &

 eps\_xtal(2,1)\*\*2 + eps\_xtal(3,1)\*\*2 + eps\_xtal(3,2)\*\*2 + &

 eps\_xtal(1,2)\*\*2 + eps\_xtal(1,3)\*\*2 + eps\_xtal(2,3)\*\*2))

 print"(' eps[xtal] vM = ',f7.3)",evm

 enorm = sqrt((eps\_xtal(1,1)\*\*2 + eps\_xtal(2,2)\*\*2 + eps\_xtal(3,3)\*\*2 + &

 eps\_xtal(2,1)\*\*2 + eps\_xtal(3,1)\*\*2 + eps\_xtal(3,2)\*\*2 + &

 eps\_xtal(1,2)\*\*2 + eps\_xtal(1,3)\*\*2 + eps\_xtal(2,3)\*\*2))

 !print"(' eps[xtal] vM = ',f7.3)",evm

 strain\_xtal(1) = eps\_xtal(1,1)

 strain\_xtal(2) = eps\_xtal(2,2)

 strain\_xtal(3) = eps\_xtal(3,3)

 strain\_xtal(4) = eps\_xtal(2,3) + eps\_xtal(3,2)

 strain\_xtal(5) = eps\_xtal(3,1) + eps\_xtal(1,3)

 strain\_xtal(6) = eps\_xtal(1,2) + eps\_xtal(2,1)

 print\*

 print\*,'Strain in crystal axes: '

 print"(2x,6(2x,f7.3))", (strain\_xtal(i),i=1,6)

 stress(1,1)=1.

 stress(1,2)=-1.

 stress(1,3)=0.

 stress(1,4)=0.

 stress(1,5)=0.

 stress(1,6)=0.

 ! # 1

 stress(2,1)=0.

 stress(2,2)=1.

 stress(2,3)=-1.

 stress(2,4)=0.

 stress(2,5)=0.

 stress(2,6)=0.

 ! # 2

 stress(3,1)=-1.

 stress(3,2)=0.

 stress(3,3)=1.

 stress(3,4)=0.

 stress(3,5)=0.

 stress(3,6)=0.

 ! # 3

 stress(4,1)=0.

 stress(4,2)=0.

 stress(4,3)=0.

 stress(4,4)=1.

 stress(4,5)=0.

 stress(4,6)=0.

 ! # 4

 stress(5,1)=0.

 stress(5,2)=0.

 stress(5,3)=0.

 stress(5,4)=0.

 stress(5,5)=1.

 stress(5,6)=0.

 ! # 5

 stress(6,1)=0.

 stress(6,2)=0.

 stress(6,3)=0.

 stress(6,4)=0.

 stress(6,5)=0.

 stress(6,6)=1.

 ! # 6

 stress(7,1)=0.5

 stress(7,2)=-1.

 stress(7,3)=0.5

 stress(7,4)=0.

 stress(7,5)=0.5

 stress(7,6)=0.

 ! # 7

 stress(8,1)=0.5

 stress(8,2)=-1.

 stress(8,3)=0.5

 stress(8,4)=0.

 stress(8,5)=-0.5

 stress(8,6)=0.

 ! # 8

 stress(9,1)=-1.

 stress(9,2)=0.5

 stress(9,3)=0.5

 stress(9,4)=0.5

 stress(9,5)=0.

 stress(9,6)=0.

 ! # 9

 stress(10,1)=-1.

 stress(10,2)=0.5

 stress(10,3)=0.5

 stress(10,4)=-0.5

 stress(10,5)=0.

 stress(10,6)=0.

 ! # 10

 stress(11,1)=0.5

 stress(11,2)=0.5

 stress(11,3)=-1.

 stress(11,4)=0.

 stress(11,5)=0.

 stress(11,6)=0.5

 ! # 11

 stress(12,1)=0.5

 stress(12,2)=0.5

 stress(12,3)=-1.

 stress(12,4)=0.

 stress(12,5)=0.

 stress(12,6)=-0.5

 !

 stress(13,1)=0.5

 stress(13,2)=0.

 stress(13,3)=-0.5

 stress(13,4)=0.5

 stress(13,5)=0.

 stress(13,6)=0.5

 !

 stress(14,1)=0.5

 stress(14,2)=0.

 stress(14,3)=-.5

 stress(14,4)=-0.5

 stress(14,5)=0.

 stress(14,6)=0.5

 !

 stress(15,1)=0.5

 stress(15,2)=0.

 stress(15,3)=-0.5

 stress(15,4)=0.5

 stress(15,5)=0.

 stress(15,6)=-0.5

 !

 stress(16,1)=0.5

 stress(16,2)=0.

 stress(16,3)=-0.5

 stress(16,4)=-0.5

 stress(16,5)=0.

 stress(16,6)=-0.5

 !

 stress(17,1)=0.

 stress(17,2)=-0.5

 stress(17,3)=0.5

 stress(17,4)=0.

 stress(17,5)=0.5

 stress(17,6)=0.5

 !

 stress(18,1)=0.

 stress(18,2)=-0.5

 stress(18,3)=0.5

 stress(18,4)=0.

 stress(18,5)=-0.5

 stress(18,6)=0.5

 !

 stress(19,1)=0.

 stress(19,2)=-0.5

 stress(19,3)=0.5

 stress(19,4)=0.

 stress(19,5)=0.5

 stress(19,6)=-0.5

 !

 stress(20,1)=0.

 stress(20,2)=-0.5

 stress(20,3)=0.5

 stress(20,4)=0.

 stress(20,5)=-0.5

 stress(20,6)=-0.5

 !

 stress(21,1)=-0.5

 stress(21,2)=0.5

 stress(21,3)=0.

 stress(21,4)=0.5

 stress(21,5)=0.5

 stress(21,6)=0.

 !

 stress(22,1)=-0.5

 stress(22,2)=0.5

 stress(22,3)=0.

 stress(22,4)=-0.5

 stress(22,5)=0.5

 stress(22,6)=0.

 !

 stress(23,1)=-0.5

 stress(23,2)=0.5

 stress(23,3)=0.

 stress(23,4)=0.5

 stress(23,5)=-0.5

 stress(23,6)=0.

 !

 stress(24,1)=-0.5

 stress(24,2)=0.5

 stress(24,3)=0.

 stress(24,4)=-0.5

 stress(24,5)=-0.5

 stress(24,6)=0.

 !

 stress(25,1)=0.

 stress(25,2)=0.

 stress(25,3)=0.

 stress(25,4)=0.5

 stress(25,5)=0.5

 stress(25,6)=-0.5

 !

 stress(26,1)=0.

 stress(26,2)=0.

 stress(26,3)=0.

 stress(26,4)=0.5

 stress(26,5)=-0.5

 stress(26,6)=0.5

 !

 stress(27,1)=0.

 stress(27,2)=0.

 stress(27,3)=0.

 stress(27,4)=-0.5

 stress(27,5)=0.5

 stress(27,6)=0.5

 !

 stress(28,1)=0.

 stress(28,2)=0.

 stress(28,3)=0.

 stress(28,4)=0.5

 stress(28,5)=0.5

 stress(28,6)=0.5

 !

 dwmax=0.

 index=0

 do 100, i=1,28

 dw = 0.

 dw = dw - strain\_xtal(1)\*stress(i,2)

 dw = dw + strain\_xtal(2)\*stress(i,1)

 dw = dw + strain\_xtal(4)\*stress(i,4)

 dw = dw + strain\_xtal(5)\*stress(i,5)

 dw = dw + strain\_xtal(6)\*stress(i,6)

 ! dW = -Bde11 + Ade22 + 2Fde23 + 2Gde31 +2Hde12

 if(dw.gt.dwmax) then

 dwmax=dw

 index=i

 print\*,'dw,i ',dw,' ',i

 endif

 if((-1.\*dw).gt.dwmax) then

 dwmax=-1.\*dw

 index=i+28

 print\*,'dw,i ',dw,' ',i

 endif

100 end do

 print\*

 write(\*,\*) 'Index of the multiple slip stress state = ',index

 if(index.gt.28) then

 index = index-28

 write(\*,\*) 'Index of the +/- stress state = ',index

 endif

 print\*,'Stress # A B C F G H'

 write(\*,120) (stress(index,j),j=1,6)

120 format(6x,6(2x,f6.1))

 rtmp1 = 0.

 do i = 1,6

 rtmp1 = rtmp1 + stress(index,i)\*\*2

 enddo

 ! magnitude of the stress

 rtmp2 = 0.

 do i = 1,6

 rtmp2 = rtmp2 + strain\_xtal(i)\*\*2

 enddo

 ! magnitude of the strain in xtal coords.

 ! print\*,'dwmax,rtmp1,rtmp2',dwmax,sqrt(rtmp1),sqrt(rtmp2)

 ! tayf = 2.\* dwmax / sqrt(rtmp2)

 tayf = dwmax / evm

 ! This is the definition of Taylor factor that allows for multi-axial

 ! strain or stress states, as found in LApp

 print\*

 print"('The Taylor factor = ',f7.3,' \*sqrt(6)')",tayf

 print"(' = ',f7.3)",tayf\*sqrt(6.)

 ! these next few lines were for curiosity about other normalizations of strain

 ! tayf2 = dwmax / enorm

 ! print\*

 ! print"('The Taylor factor = ',f7.3,' \*sqrt(6)')",tayf2

 ! print"(' = ',f7.3)",tayf2\*sqrt(6.)

 call exit(0)

end program BishopHill

!

! -------------------

!

subroutine euler(a,iopt,nomen,d1,d2,d3,ior,kerr)

 ! Last revision 20nov90 UFK

 ! common a(3,3),grvol(1152),epsga(5),ist1,ist2,sqr3,sqrh,ident(3,3)

 ! SPECIAL VERSION WITHOUT COMMON BLOCK

 real a(3,3),d1,d2,d3,th,sth,cth,sph,cph,sps,cps,ps,ph,dth,dph,dps

 character nomen

 save kor

 real pi

 parameter ( pi = 3.14159265 )

 real rad

 parameter ( rad = 57.29578 )

 ! CODE::

 ! \*\*\* this subroutine calculates the euler angles associated with the

 ! transformation matrix a(i,j) if iopt=1 and viceversa if iopt=2

 ! \*\*\* Note that a is sample (rows) in terms of crystal (columns);

 ! -- opposite of standard g (e.g.Bunge) - this is Canova's

 ! \*\*\* Note that in this version, the Euler angles are defined symmetrically:

 ! so that interchanging phi and psi means transposing a.

 ! ("Kocks" nomen: defined going from +X to +Y in both COD and SOD)

 ! \*\*\* However, other angle conventions are translated, according to

 ! nomen="K" - Kocks (as internally) -- also sometimes "N"...

 ! "R" - Roe (Psi=psi, Phi=180-phi)

 ! "B" - Bunge (phi1=90+psi, Phi=Theta, phi2=90-phi)

 ! any other - Canova (Theta first, phiC=90+phi, omega=90-psi)

 ! \*\*\* Note: only in symm. notation does a point with all Euler angles

 ! between 0 and 90 deg appear in the +x,+y quadrant!

 ! If you want to see an individual point in this quadrant and are:

 ! using Roe, the third angle must be between 90 and 180;

 ! Bunge, first ;

 ! Canova, second .

 ! \*\*\* Input and output Euler angles d1,d2,d3 in degrees

 !

 goto(5,20),iopt

5 if(abs(a(3,3)) .ge. 0.999) goto 10

 th=acos(a(3,3))

 sth=sin(th)

 !

 if((abs(a(2,3)/sth).lt.1e-35).and. &

 & (abs(a(1,3)/sth).lt.1e-35)) then

 ps=pi/4.

 else

 ps=atan2(a(2,3)/sth,a(1,3)/sth)

 endif

 !

 if((abs(a(3,2)/sth).lt.1e-35).and. &

 & (abs(a(3,1)/sth).lt.1e-35)) then

 ph=pi/4.

 else

 ph=atan2(a(3,2)/sth,a(3,1)/sth)

 endif

 !cccc if it bombs out here, both a-components in one arg. are zero

 ! (this should not be possible, but has happened, probably fixed)

 ! ADR: i 01 - above is the fix!!

 !

 go to 15

 !

10 if((abs(a(1,2)/sth).lt.1e-35).and. &

 & (abs(a(1,1)/sth).lt.1e-35)) then

 ps=pi/4.

 else

 ps=0.5\*atan2(a(1,2),-a(1,1))

 endif

 !

 ph=-ps

 ! The above still have the problem that they give too many

 ! equivalents of the same grain in DIOROUT and density file. Therefore:

 if(kerr.eq.1.and.kor.ne.ior) then

 print\*,'NOTE: grain',ior,' has Theta < 1 deg.: sometimes problems'

 print\*

 kor=ior

 endif

 !

15 dth=th\*rad

 dph=ph\*rad

 dps=ps\*rad

 d1=dps

 d2=dth

 if(nomen.eq.'K'.or.nomen.eq.'k'.or.nomen.eq.'N') then

 d3=dph

 elseif(nomen.eq.'R'.or.nomen.eq.'r') then

 d3=180.-dph

 elseif(nomen.eq.'B'.or.nomen.eq.'b') then

 d1=dps+90.

 d3=90.-dph

 else

 d1=dth

 d2=dph+90.

 d3=90.-dps

 endif

 if(d1.ge.360.) d1=d1-360.

 if(d3.ge.360.) d3=d3-360.

 if(d1.lt.0.) d1=d1+360.

 if(d3.lt.0.) d3=d3+360.

 return

 ! \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

20 dth=d2

 dps=d1

 if(nomen.eq.'K'.or.nomen.eq.'k') then

 dph=d3

 elseif(nomen.eq.'R'.or.nomen.eq.'r') then

 dph=180.-d3

 elseif(nomen.eq.'B'.or.nomen.eq.'b') then

 dps=d1-90.

 dph=90.-d3

 else

 dth=d1

 dph=d2-90.

 dps=90.-d3

 endif

 ph=dph/rad

 th=dth/rad

 ps=dps/rad

 sph=sin(ph)

 cph=cos(ph)

 sth=sin(th)

 cth=cos(th)

 sps=sin(ps)

 cps=cos(ps)

 a(1,1)=-sps\*sph-cph\*cps\*cth

 a(2,1)=cps\*sph-cph\*sps\*cth

 a(3,1)=cph\*sth

 a(1,2)=sps\*cph-sph\*cps\*cth

 a(2,2)=-cph\*cps-sph\*sps\*cth

 a(3,2)=sth\*sph

 a(1,3)=sth\*cps

 a(2,3)=sps\*sth

 a(3,3)=cth

 return

end subroutine euler

! \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

subroutine vecnorm(vec)

 real vec(3),rnorm

 rnorm=0.

 do 10, i=1,3

 rnorm=rnorm+vec(i)\*\*2

10 end do

 if(rnorm.le.0.0) return

 rnorm=sqrt(rnorm)

 do 20, i=1,3

 vec(i)=vec(i)/rnorm

20 end do

 return

end subroutine vecnorm

! --------------------

function scalar(a,b)

 ! return SCALAR PRODUCT of A and B

 real scalar,a(3),b(3)

 scalar=0.

 do 100, i=1,3

 scalar=scalar+a(i)\*b(i)

100 end do

 return

end function scalar

! \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

subroutine vecpro2(v1,v2,vout)

 ! vector product

 real v1(3),v2(3),vout(3)

 vout(3)=v1(1)\*v2(2)-v1(2)\*v2(1)

 vout(1)=v1(2)\*v2(3)-v1(3)\*v2(2)

 vout(2)=v1(3)\*v2(1)-v1(1)\*v2(3)

 return

end subroutine vecpro2

1. † SteveSintay\_Pole\_Figures-26May11.xls; SchmidFactors-2016.xlsx. [↑](#footnote-ref-1)