CTEAS: Technical Documentation

Zachary A. Jones

Draft - 11/17/2011

Contents

Part I
CTEAS - Math Analysis

A detailed overview of the math used to calculate thermal expansion tensors within CTEAS is presented. The supporting functions are split into sections containing a list of inputs, the actions performed within the function, outputs, and subfunction descriptions when applicable. Where necessary, descriptions of the math steps have been added for the purpose of understanding the sometimes-cryptic syntax of programming languages. No descriptions of Matlab or LabVIEW base functions are given. It is the hope of the author that the user will have this document and the matlab function under scrutiny open side-by-side to gather the required understanding of the function. Note that the CTERun functions collectively contain 1907 lines of code and 616 lines of internal documentation. Only the mathematical operations for obtaining thermal expansion matrices and data for plotting will be discussed.

1 Calculating and Plotting a Thermal Expansion Ellipsoid

The following subsections describe the functions used to calculate the x, y, and z coordinates that can be plotted as a three-dimensional ellipsoid that represents thermal expansion at a given temperature or range of temperatures.

1.1 getPtTemp.m

Adjust temperatures using Room Temperature Platinum Lattice Constant obtained from JADE files. If RT Plat. Const. doesn't exist, Temperatures get edited manually. Set RT Plat. Lat. Const. to 0 in CTEAS GUI.

Inputs

- RefTval = RT Platinum Lattice Constant
- datin = data read in from JADE files.

Actions

- RefT = 20
  - ▷ Room Temperature (20°C)
- PtDaa = $\left(\frac{aVal-RefTval}{RefTval}\right) \times 100$
  - ▷ (NOTE: If RefTval=0, PtDaa = aval)
- PtDaaErr = $\left(\frac{aVal+aErr-RefTval}{RefTval}\right) \times 100$
  - ▷ (NOTE: If RefTval=0, PtDaaErr = $aVal + aErr$
- Tcal = $\left(3.8274 \times PtDaa^3\right) - \left(130.5739 \times PtDaa^2\right) + \left(1101.6564 \times PtDaa\right) + RefT$
  - ▷ (Temperature Correction, no documentation given)
- Terr = $\left(3.8274 \times PtDaaErr^3\right) - \left(130.5739 \times PtDaaErr^2\right) + \left(1101.6564 \times PtDaaErr\right) + RefT - Tcal$
  - ▷ (Error in Temperature Calculation, no documentation given again)

Outputs

- datout = JADE data with corrected Temperature.

1.2 xtalCTEprep.m

Prepare data for calculating the 2nd rank Thermal Expansion Tensor per temperature step.

Inputs

- datin = Data from input files with corrected temperatures
- phasename = Name of phase to be analyzed
- ordn = Polynomial fit order

Actions

- lsfits = Polynomial fits to the lattice parameters using lpexpand subfunction.
- entries = List of formatted data from input files containing the phase "phasename." Data are sorted using the organizedata1 function within xtalCTEprep.m.
  - ▷ entries.T = Temperature (°C)
  - ▷ entries.d = d-spacing in Angstroms
- tmp1 = polyfit(entries.T-T_room,log(entries.d),order);
  - ▷ Polynomial line fit data at a given temperature
- params = list of results from the polyfit.

Outputs

- lsfits = polynomial fits of the lattice parameters.
- entries = Temperature and d-spacing data for the phase to be analyzed.
- params = list of results from the polyfit.

Subfunctions

1.2.1 organizedata1

- Inputs
  - ▷ datin = Data from input files that contains all read data.
  - ▷ phasename = Name of phase to be analyzed.
- Actions
  - ▷ Compares each entry in datin with the phasename. If it matches, the entry is organized and stored as an entry in datout. The temperature and d-spacing data are stored as an entry in entries.
- Outputs
  - ▷ datout = Data from input files that corresponds only to the phase that will be analyzed (lattice parameters, temperatures, and associated errors)
  - ▷ entries = Temperature and d-spacing data that corresponds only to the phase that will be analyzed.

1.2.2 lpexpand

- Inputs
  - ▷ latparams = datout from organizedata1
  - ▷ ordn = polynomial fit order
- Actions
  - ▷ T0=25;
    - ◦ Room temperature (25°C in this file)
  - ▷ fita=polyfit(latparams.T-T0,log(latparams.a),ordn);
    - ◦ Polynomial fit to each of the lattice parameters (fita, fitb, fitc, etc).
- Outputs
  - ▷ fita, fitb, fitc, fitaplha, fitbeta, fitgamma
    - ◦ polynomial fits of the lattice parameters (stored in lsfits)

1.3 xtalCTE.m

Calculates the thermal expansion tensor at a temperature, T based on the polyfits outlined in xtalCTEprep.m.

Inputs

- lsfits = polynomial fits of the lattice parameters.
- entriesT = Temperature and d-spacing data for the phase to be analyzed.
- params = list of results from the polyfit in xtalCTEprep.m.
- T = Temperature
- system = String containing the system to be analyzed.

Actions

Calculate conversion matrix and recprocal conversion matrix. Initialize dtce, dcteErr, xyz1, and xyznrm to finite lengths for memory allocation purposes. Depricated option for deciding if error weighting should be applied (set to 'yes' by default via Ryan's opinion). Calculate the thermal expansion for each lattice parameter using getCTETempWErr.

Outputs

- ten1 = CTE Tensor at input Temperature.
- conMat1 = Conversion matrix at input Temperature.

Subfunctions

1.3.1 getconversionmat

Calculate conversion matrix for use in converting points and vectors from the crystal lattice to the orthonormal system.

- Inputs
  - ▷ fita, fitb, fitc, fitalpha, fitbeta, fitgamma = lsfits
  - ▷ T = Temperature (°C)
- Actions
  - ▷ aval = getLPTemp(fita, T);
    - ◦ Same for all others in lsfits.
  - ▷ $m_{11} = aval \times sin\left(\frac{betaval \times \pi}{180}\right)$
  - ▷ $m_{12} = bval\left[\frac{cos\left(\frac{gammaval \times \pi}{180}\right) - cos\left(\frac{betaval \times \pi}{180}\right) \times cos\left(\frac{alphaval \times \pi}{180}\right)}{sin\left(\frac{betaval \times \pi}{180}\right)}\right]$
  - ▷ $m_{22} = bval\left[sin\left(\frac{alphaval \times \pi}{180}\right)^2 - \frac{m_{12}^2}{bval^2}\right]^{\frac{1}{2}}$
  - ▷ $m_{31} = aval \times cos\left(\frac{betaval \times \pi}{180}\right)$
  - ▷ $m_{32} = bval \times cos\left(\frac{alphaval \times \pi}{180}\right)$
  - ▷ $m_{33} = cval$
  - ▷ $conMatOut = \begin{bmatrix} m_{11} & m_{12} & 0 \\ 0 & m_{22} & 0 \\ m_{31} & m_{32} & m_{33} \end{bmatrix}$
- Outputs
  - ▷ conMatOut = calculated conversion matrix at the given input temperature.
  - ▷ mcalcsout = m_{**} calculations.

1.3.2 getRecipConMatrix

Calculate the reciprocal conversion matrix.

- Inputs
  - ▷ fita, fitb, fitc, fitalpha, fitbeta, fitgamma = lsfits
  - ▷ T = Temperature (°C)
  - ▷ mcalcs = m_{**} calculations from getconversionmat.
- Actions
  - ▷ aval = getLPTemp(fita, T);
    - ◦ Same for all others in lsfits.
  - ▷ $Vol_1 = aval \times bval \times cval$
  - ▷ $Vol_2 = 1 - cos\left(\frac{alphaval \times \pi}{180}\right)^2 - cos\left(\frac{betaval \times \pi}{180}\right)^2 - cos\left(\frac{gammaval \times \pi}{180}\right)^2$
  - ▷ $Vol_3 = 2\left[cos\left(\frac{alphaval \times \pi}{180}\right) cos\left(\frac{betaval \times \pi}{180}\right) cos\left(\frac{gammaval \times \pi}{180}\right)\right]$

- ▷ $Vol = Vol_1 \left[Vol_2 + Vol_3\right]^{\frac{1}{2}}$
  - ○ Note: Vol was split into three parts for formatting. Only the variable "Vol" exists in the code.
- ▷ $n_{11} = \frac{m_{22}m_{33}}{Vol}$
- ▷ $n_{13} = \frac{-m_{31}m_{22}}{Vol}$
- ▷ $n_{21} = \frac{-m_{12}m_{33}}{Vol}$
- ▷ $n_{22} = \frac{aval \times cval \times sin\left(\frac{betaval \times \pi}{180}\right)}{Vol}$
- ▷ $n_{23} = \frac{m_{31}m_{12} - m_{11}m_{32}}{Vol}$
- ▷ $n_{33} = \frac{m_{11}m_{22}}{Vol}$
- ▷ $matOut = \begin{bmatrix} n_{11} & 0 & n_{13} \\ n_{21} & n_{22} & n_{23} \\ 0 & 0 & n_{33} \end{bmatrix}$

- Outputs
  - ▷ matOut = Reciprocal Conversion Matrix at given Temperature.

1.3.3 getLPTemp

Calculate a lattice parameter for a given input temperature based on the polyfit of that lattice parameter and room temperature.

- Inputs
  - ▷ fit1 = Lattice parameter fit (fita, fitb, fitc, etc)
  - ▷ T1 = Temperature ($^{\circ}C$)
- Actions
  - ▷ T0 = 25
    - ○ Room Temperature (°C)
  - ▷ lpoutln = lpoutln+fit1(end-ctr1+1)*((T1-T0).^(ctr1-1));
    - ○ In a loop: for ctr1=1:length(fit1)
  - ▷ lpout=exp(lpoutln);
- Outputs
  - ▷ lpout = Calculated lattice parameter based on polyfit.

1.3.4 getCTETemp

Calculate the thermal expansion at a temperature of a lattice direction based on the polyfit of the lattice parameter and room temperature.

- Inputs
  - ▷ fit1 = Lattice parameter fit (fita, fitb, fitc, etc)
  - ▷ T1 = Temperature ($^{\circ}C$)
- Actions
  - ▷ alphaout = alphaout+((fit1(end-ctr1)*(ctr1))*((T1-T0)^(ctr1-1)));s
    - ○ In a loop: for ctr1=1:length(fit1(1:end-1))
- Outputs
  - ▷ alphaout = thermal expansion of a lattice direction at a temperature

1.3.5 getCTETempWErr

Calculate the thermal expansion and its error at a temperature of a lattice direction based on the polyfit of the lattice parameter and room temperature.

- Inputs
  - ▷ fit1 = Lattice parameter fit (fita, fitb, fitc, etc)
  - ▷ T1 = Temperature ($°C$)
- Actions
  - ▷ alphaout = alphaout+((fit1(end-ctr1)*(ctr1))*((T1-T0)^(ctr1-1)));
    - ◦ In a loop: for ctr1=1:length(fit1(1:end-1))
  - ▷ [dVal,dErr]=polyval(fit1.p,T1-T0,fit1.S);
    - ◦ Obtain the error in measurement
  - ▷ alphaErr=alphaout*dErr/dVal;
    - ◦ Calculate the error in the thermal expansion.
- Outputs
  - ▷ alphaout = thermal expansion of a lattice direction at a temperature
  - ▷ alphaErr = error in alphaout

1.3.6 fitCTETensor

Build and fit a rank two thermal expansion tensor based on the thermal expansions calculated in previous steps.

- Inputs
  - ▷ xyzin =
  - ▷ dctein =
  - ▷ system = Symmetry to be analyzed.
- Actions
  - ▷ Assign x1, tentemp, and tenout values based on the crystal system analyzed.
  - ▷ triclinic:
    - ◦ x1= $\begin{bmatrix} xyzin(:,1)'.*xyzin(:,1)'; \\ xyzin(:,2)'.*xyzin(:,2)'; \\ xyzin(:,3)'.*xyzin(:,3)'; \\ xyzin(:,1)'.*xyzin(:,2)'; \\ xyzin(:,1)'.*xyzin(:,3)'; \\ xyzin(:,2)'.*xyzin(:,3)'; \end{bmatrix}$
    - ◦ tentemp = x1\dctein';
    - ◦ tenout = $\begin{bmatrix} tentemp(1) & \frac{tentemp(4)}{2} & \frac{tentemp(5)}{2} \\ \frac{tentemp(4)}{2} & tentemp(2) & \frac{tentemp(6)}{2} \\ \frac{tentemp(5)}{2} & \frac{tentemp(6)}{2} & tentemp(3) \end{bmatrix}$
  - ▷ monoclinic:

- x1= $\begin{bmatrix} xyzin(:,1)'.*xyzin(:,1)'; \\ xyzin(:,2)'.*xyzin(:,2)'; \\ xyzin(:,3)'.*xyzin(:,3)'; \\ xyzin(:,1)'.*xyzin(:,3)'; \end{bmatrix}$
- tentemp = x1\dctein';
- tenout = $\begin{bmatrix} tentemp(1) & 0 & \frac{tentemp(4)}{2} \\ 0 & tentemp(2) & 0 \\ \frac{tentemp(4)}{2} & 0 & tentemp(3) \end{bmatrix}$

▷ tetragonal:

- x1= $\begin{bmatrix} xyzin(:,1)'.*xyzin(:,1)'+xyzin(:,2)'.*xyzin(:,2)' \\ xyzin(:,3)'.*xyzin(:,3)'; \end{bmatrix}$
- tentemp=inv(x1'*x1)*x1'*dctein';
- tenout = $\begin{bmatrix} tentemp(1) & 0 & 0 \\ 0 & tentemp(1) & 0 \\ 0 & 0 & tentemp(2) \end{bmatrix}$

▷ hexagonal:

- x1= $\begin{bmatrix} xyzin(:,1)'.*xyzin(:,1)'+xyzin(:,2)'.*xyzin(:,2)' \\ xyzin(:,3)'.*xyzin(:,3)'; \end{bmatrix}$
- tentemp=inv(x1'*x1)*x1'*dctein';
- tenout = $\begin{bmatrix} tentemp(1) & 0 & 0 \\ 0 & tentemp(1) & 0 \\ 0 & 0 & tentemp(2) \end{bmatrix}$

▷ orthorhombic:

- x1= $\begin{bmatrix} xyzin(:,1)'.*xyzin(:,1)'; \\ xyzin(:,2)'.*xyzin(:,2)'; \\ xyzin(:,3)'.*xyzin(:,3)'; \end{bmatrix}$
- tentemp = x1\dctein';
- tenout = $\begin{bmatrix} tentemp(1) & 0 & 0 \\ 0 & tentemp(2) & 0 \\ 0 & 0 & tentemp(3) \end{bmatrix}$

▷ cubic

- x1= $\begin{bmatrix} xyzin(:,1)'.*xyzin(:,1)'; \\ xyzin(:,2)'.*xyzin(:,2)'; \\ xyzin(:,3)'.*xyzin(:,3)'; \end{bmatrix}$
- tentemp=inv(x1'*x1)*x1'*dctein';
- tenout = $\begin{bmatrix} tentemp(1) & 0 & 0 \\ 0 & tentemp(2) & 0 \\ 0 & 0 & tentemp(3) \end{bmatrix}$

▷ trigonal:

- x1= $\begin{bmatrix} xyzin(:,1)'.*xyzin(:,1)'+xyzin(:,2)'.*xyzin(:,2)' \\ xyzin(:,3)'.*xyzin(:,3)'; \end{bmatrix}$
- tentemp=inv(x1'*x1)*x1'*dctein';
- tenout = $\begin{bmatrix} tentemp(1) & 0 & 0 \\ 0 & tentemp(1) & 0 \\ 0 & 0 & tentemp(2) \end{bmatrix}$

▷ rhombohedral:

- x1= $\begin{bmatrix} xyzin(:,1)'.*xyzin(:,1)'+xyzin(:,2)'.*xyzin(:,2)' \\ xyzin(:,3)'.*xyzin(:,3)'; \end{bmatrix}$

- tentemp=inv(x1'*x1)*x1'*dctein';
- tenout = $\begin{bmatrix} tentemp(1) & 0 & 0 \\ 0 & tentemp(1) & 0 \\ 0 & 0 & tentemp(2) \end{bmatrix}$

- Outputs
  - tenout = rank two thermal expansion tensor at input temperature.

1.3.7 fitCTETensorWeighted

Build and fit a weighted rank two thermal expansion tensor based on the thermal expansions calculated in previous steps.

- Inputs
  - xyzin
  - dctein
  - dcteErr
  - system = Symmetry to be analyzed.
- Actions
  - W1=diag(1./(dcteErr.*dcteErr));
    - Weighting matrix equal to the inverse of the variance for each CTE, with the assumption that the error given is the standard deviation of the CTE.
  - tentemp=inv(x1'*W1*x1)*x1'*W1*dctein';
  - Assign x1 and tenout values based on the crystal system analyzed.
  - triclinic:
    - x1= $\begin{bmatrix} xyzin(:,1)'.*xyzin(:,1)'; \\ xyzin(:,2)'.*xyzin(:,2)'; \\ xyzin(:,3)'.*xyzin(:,3)'; \\ xyzin(:,1)'.*xyzin(:,2)'; \\ xyzin(:,1)'.*xyzin(:,3)'; \\ xyzin(:,2)'.*xyzin(:,3)'; \end{bmatrix}$
    - tenout = $\begin{bmatrix} tentemp(1) & \frac{tentemp(4)}{2} & \frac{tentemp(5)}{2} \\ \frac{tentemp(4)}{2} & tentemp(2) & \frac{tentemp(6)}{2} \\ \frac{tentemp(5)}{2} & \frac{tentemp(6)}{2} & tentemp(3) \end{bmatrix}$
  - monoclinic:
    - x1= $\begin{bmatrix} xyzin(:,1)'.*xyzin(:,1)'; \\ xyzin(:,2)'.*xyzin(:,2)'; \\ xyzin(:,3)'.*xyzin(:,3)'; \\ xyzin(:,1)'.*xyzin(:,3)'; \end{bmatrix}$
    - tenout = $\begin{bmatrix} tentemp(1) & 0 & \frac{tentemp(4)}{2} \\ 0 & tentemp(2) & 0 \\ \frac{tentemp(4)}{2} & 0 & tentemp(3) \end{bmatrix}$
  - tetragonal:
    - x1= $\begin{bmatrix} xyzin(:,1)'.*xyzin(:,1)' + xyzin(:,2)'.*xyzin(:,2)' \\ xyzin(:,3)'.*xyzin(:,3)'; \end{bmatrix}$

- ∘ tenout = $\begin{bmatrix} tentemp(1) & 0 & 0 \\ 0 & tentemp(1) & 0 \\ 0 & 0 & tentemp(2) \end{bmatrix}$

▷ hexagonal:

- ∘ x1= $\begin{bmatrix} xyzin(:,1)'.*xyzin(:,1)' + xyzin(:,2)'.*xyzin(:,2)' \\ xyzin(:,3)'.*xyzin(:,3)'; \end{bmatrix}$
- ∘ tenout = $\begin{bmatrix} tentemp(1) & 0 & 0 \\ 0 & tentemp(1) & 0 \\ 0 & 0 & tentemp(2) \end{bmatrix}$

▷ orthorhombic:

- ∘ x1= $\begin{bmatrix} xyzin(:,1)'.*xyzin(:,1)'; \\ xyzin(:,2)'.*xyzin(:,2)'; \\ xyzin(:,3)'.*xyzin(:,3)'; \end{bmatrix}$
- ∘ tenout = $\begin{bmatrix} tentemp(1) & 0 & 0 \\ 0 & tentemp(2) & 0 \\ 0 & 0 & tentemp(3) \end{bmatrix}$

▷ cubic

- ∘ x1= $\begin{bmatrix} xyzin(:,1)'.*xyzin(:,1)'; \\ xyzin(:,2)'.*xyzin(:,2)'; \\ xyzin(:,3)'.*xyzin(:,3)'; \end{bmatrix}$
- ∘ tenout = $\begin{bmatrix} tentemp(1) & 0 & 0 \\ 0 & tentemp(2) & 0 \\ 0 & 0 & tentemp(3) \end{bmatrix}$

▷ trigonal:

- ∘ x1= $\begin{bmatrix} xyzin(:,1)'.*xyzin(:,1)' + xyzin(:,2)'.*xyzin(:,2)' \\ xyzin(:,3)'.*xyzin(:,3)'; \end{bmatrix}$
- ∘ tenout = $\begin{bmatrix} tentemp(1) & 0 & 0 \\ 0 & tentemp(1) & 0 \\ 0 & 0 & tentemp(2) \end{bmatrix}$

▷ rhombohedral:

- ∘ x1= $\begin{bmatrix} xyzin(:,1)'.*xyzin(:,1)' + xyzin(:,2)'.*xyzin(:,2)' \\ xyzin(:,3)'.*xyzin(:,3)'; \end{bmatrix}$
- ∘ tenout = $\begin{bmatrix} tentemp(1) & 0 & 0 \\ 0 & tentemp(1) & 0 \\ 0 & 0 & tentemp(2) \end{bmatrix}$

- Outputs
  - ▷ tenout = Weighted rank two thermal expansion tensor at input temperature.

1.4 plotEllipsoid.m

Calculate and output x, y, z, and colormap coordinates for plotting a thermal expansion ellipsoid using the eigenvectors and eigenvalues of a thermal expansion tensor at a temperature.

- Inputs
  - ▷ V1 - Eigenvectors of the thermal expansion tensor at a temperature
  - ▷ e1s = Eigenvalues of the thermal expansion tensor at a temperature
- Actions

- ▷ e1 = diag(e1s)
  - Use only the diagonal components of the eigenvectors
- ▷ q1=q2=[0:0.02:2]*pi;
  - Arrays of points from zero to 2π.
- ▷ [q1,q2]=meshgrid(q1,q2);
  - Reassign q1 and q2 to be a series of values that form a sphere in 3-dimensional space.
- ▷ r1=e1(1)*(cos(q1).^2).*(sin(q2).^2)+e1(2)*(sin(q1).^2).*(sin(q2).^2)+e1(3)*(cos(q2).^2);
  - Determine the radial values for each point in the q1, q2 mesh.
- ▷ x1 = r1.*cos(q1).*sin(q2);
- ▷ y1 = r1.*sin(q1).*sin(q2);
- ▷ z1 = r1.*cos(q2);
  - Cartesian coordinates of each point in the q1,q2 mesh.
- ▷ C1 = eye(3)
  - Definition of the original axis.
- ▷ rot1 = getRotationMat(V1,C1);
  - Determine rotation matrix between the Eigenvectors and the C1 axis.
- ▷ xs=rot1(1,1)*x1+rot1(1,2)*y1+rot1(1,3)*z1;
- ▷ ys=rot1(2,1)*x1+rot1(2,2)*y1+rot1(2,3)*z1;
- ▷ zs=rot1(3,1)*x1+rot1(3,2)*y1+rot1(3,3)*z1;
  - Using the rotation matrix, define new x, y, and z coordinates to be the points on the thermal expansion ellipsoid.

- Outputs
  - ▷ xs, ys, zs, cs = x, y, and z coordinates of points that construct a thermal expansion ellipsoid. cs is a colormap.

2 Cross-Sections of the Thermal Expansion Ellipsoid

This section details the functions used in CTEAS that manipulate, determine, and plot two-dimensional cross-sections of a thermal expansion ellipsoid. The mathematics involved will be a primary focus.

2.1 EllipseWrap.m

A wrapper function that contains steps to calculate and plot a cross-section of a thermal expansion ellipsoid. This is loosely documented because most documentation will occur with the given functions.

- Inputs
  - ▷ TensorSet = Range of CTE Tensors.
  - ▷ ConvMatrx = Calculated conversion matrix between crystallographic and cartesian coordinates.
  - ▷ TempRanges = Range of Temperatures at which the CTE Tensors have been calculated.
  - ▷ aval, bval, cval = hkl or UVW to plot perpendicular to.
  - ▷ lsfits = Lattice parameter polynomial fits.
  - ▷ xscreen, yscreen = Size of figures to be plotted in pixels.
  - ▷ inUVW2 = Boolean to determine if using hkl or uvw output.

- ▷ isrhombohedral = Boolean to determine if a symmetry is true rhombohedral or analyzed in a hexagonal cell.

- Actions
  - ▷ Create a figure and use plotEllipse.m in a loop throughout the range of temperatures in TempRanges. Update color maps by an equal division of colors from blue to red for each call of plotEllipse.
  - ▷ Add arrows for axes if in-plane or eigenvectors else using addarrow4.m.
  - ▷ Name arrows with hkl and uvw conventions.
  - ▷ Add descriptors to plot (title, x and y labels)
  - ▷ Set axis tick labels.

- Outputs
  - ▷ twodvals = List of plotted CTE Ellipse values per temperature.

2.2 plotEllipse.m

From a CTE matrix and an hkl or uvw vector, determines and plots a two-dimensional cross-section of thermal expansion perpendicular to the input vector.

- Inputs
  - ▷ M1 = Rank two thermal expansion tensor at a temperature.
  - ▷ p1 = hkl or uvw vector to plot perpendicular to.
  - ▷ ConMat = Conversion matrix between crystal lattice and orthonormal axis.
  - ▷ cmap = Color map for visual temperature differentiation.

- Actions
  - ▷ [t1,t2] = ϕ and θ, respectively. Calculated from hklToAng.m function.
  - ▷ [valout,Mf] = Thermal expansion along vector and complete thermal expansion matrix, respectively. Calculated from getCTEVect.m function.
  - ▷ Mplane = Mf(1:2,1:2), the two-dimensional component (located in the plane in question) for thermal expansion.
  - ▷ [V,D] = Calculated eigenvalues and eigenvectors of Mplane.
  - ▷ t3 = [0:0.01:2]*pi, an array containing values from zero to 2π in 0.01 radian steps.
  - ▷ r1 = Array of calculated point distances from the center of the plot based on Mplane.
    - r1 = (Mplane(1,1)*(cos(t3)).^2)-(2*Mplane(1,2)*cos(t3).*sin(t3))+(Mplane(2,2)*(sin(t3)).^2);
  - ▷ x1, y1 = x and y components of the r1 vector using the values of t3 as angles.
    - x1=r1.*cos(t3);
    - y1=r1.*sin(t3);
  - ▷ Plot x1 and y1, tidy up plot with built-in Matlab functions.

- Outputs
  - ▷ V, D, valout, t1, t2, x1, y1 = Described above.

2.3 addarrow4.m

Add an arrow to the thermal expansion ellipsoid cross-section calculated by plotEllipse.m if the direction of the arrow lies within the plane.

- Inputs
  - ▷ M1 = CTE Tensor at temperature.
  - ▷ p1 = hkl direction in question.
  - ▷ a1 = Reference vector (usually c-axis).
  - ▷ axis = Label for axis if applicable.
  - ▷ fontsize = Size of fonts (in points) to be displayed for labeling.
- Actions
  - ▷ Convert a1 to orthonormal coordinates using hklToorth.m.
    - ◦ a2=hklToorth(ConMat,a1);
  - ▷ Check to see if the arrow will be in the plane by calculating the cosine between the axis in question (converted to orthonormal coordinates) and a2. If it is equal to zero, the arrow does not lie in the plane. Stop function and inplane = 0.
  - ▷ If the cosine is not equal to zero, inplane = 1 and the ϕ and θ angles of the a1 vector are calculated and stored as phi1 and theta1.
    - ◦ [phi1,theta1] = hklToAng(a1,ConMat);
  - ▷ The magnitude of the a1 vector can be calculated as the valout portion of the getCTEVect.m function.
    - ◦ [valout,Mf]=getCTEVect(M1,phi1,theta1);
  - ▷ Since a2 is in the direction of a1 but of unit length, it can be multiplied by valout to obtain the right length.
    - ◦ orth1=a2*valout;
  - ▷ The ϕ and θ angles between the p1 vector and the z-axis must be determined.
    - ◦ [t1,t2]=hklToAng(p1,ConMat);
  - ▷ A new conversion matrix, M2 must be determined for creation of a rotation matrix. Note that the negative values have been added because of the conventions Matlab uses.
    - ◦ m1=rotateVect([-1;0;0],t1,t2);
    - ◦ m2=rotateVect([0;1;0],t1,t2);
    - ◦ m3=rotateVect([0;0;-1],t1,t2);
    - ◦ M2=[m1';m2';m3'];
    - ◦ r1=getRotationMat(eye(3),M2);
  - ▷ Rotate a2 by multiplying it by r1, then extend it by multiplying it's unit length by valout.
    - ◦ a3=(a2'*r1)'*valout;
  - ▷ a3 is the set of coordinates in orthonormal space that comprises the magnitude of thermal expansion in the a1 hkl. Plot the components on the 2D cross-section plot.
- Outputs
  - ▷ inplane = 1 if the arrow lies within the plane, 0 if it does not.

2.4 rotateVect.m

Rotate a vector by ϕ and θ.

- Inputs
  - a1 = An orthonormal vector.
  - T1 = ϕ
  - T2 = θ
- Actions
  - Create rotation matrices about Y and Z axes, combine them.
  - t1=T1*pi/180;
  - t2=T2*pi/180;
  - Ry=[cos(t1),0,-sin(t1);0 1 0;sin(t1),0,cos(t1)];
  - Rz=[cos(t2),sin(t2),0;-sin(t2),cos(t2),0;0,0,1];
  - Rfv=Ry*Rz;
  - Rotate about the Z-axis and then the Y-axis, then multiply a1 by the resulting rotation matrix.
    - as1 = (a1'*Rfv)';
- Outputs
  - as1 = Rotated vector.

2.5 antirotateVect.m

Rotate a vector by θ, then ϕ. These angles are usually specified as negative values.

- Inputs
  - a1 = An orthonormal vector.
  - T1 = ϕ
  - T2 = θ
- Actions
  - Create rotation matrices about Y and Z axes, combine them.
  - t1=T1*pi/180;
  - t2=T2*pi/180;
  - Ry=[cos(t1),0,-sin(t1);0 1 0;sin(t1),0,cos(t1)];
  - Rz=[cos(t2),sin(t2),0;-sin(t2),cos(t2),0;0,0,1];
  - Rfv=Rz*Ry;
  - Rotate about the Y-axis and then the Z-axis, then multiply a1 by the resulting rotation matrix.
    - as1 = (a1'*Rfv)';
- Outputs
  - as1 = Rotated vector.

2.6 hklToAng.m

Convert a vector composed of h, k, and l components to angles ϕ and θ in spherical coordinates with negligable length. Length is not important because of scaling that takes place in other functions.

- Inputs
  - ▷ hkl = vector composed of h, k, and l components
  - ▷ CM1 = Conversion matrix
- Actions
  - ▷ Convert hkl to orthornormal coordinates using hkltoorth(CM1,hkl).
  - ▷ Convert orthonormal coordinates to angles using orthToAng(vtmp).
- Outputs
  - ▷ phi, theta = components of unit vector in spherical coordinates

2.7 getCTEVect.m

Calculate the thermal expansion in a direction where ϕ is the angle between a vector and the Z-axis and θ is the angle between the projection of the vector on the X-Y plane and the X-axis.

- Inputs
  - ▷ M1 = CTE Tensor at temperature.
  - ▷ phi, theta = components of unit vector in spherical coordinates
- Actions
  - ▷ Ry=[cos(phi1),0,-sin(phi1);0 1 0;sin(phi1),0,cos(phi1)];
  - ▷ Rz=[cos(theta1),sin(theta1),0;-sin(theta1),cos(theta1),0;0,0,1];
  - ▷ Rfm=Ry*Rz;
  - ▷ Mf=Rfm*M1*Rfm';
  - ▷ valout=Mf(3,3);
- Outputs
  - ▷ valout = Scaling factor for unit vector.
  - ▷ Mf = 2D component of rotated CTE Tensor perpendicular to the vector that phi and theta were determined from.

2.8 hklToorth.m

Convert a vector composed of h, k, and l components to orthonormal x, y, and z coordinates.

- Inputs
  - ▷ CM1 = Conversion matrix at temperature.
  - ▷ hkl = hkl in question.
- Actions
  - ▷ v=CM1*hkl/norm(CM1*hkl);
- Outputs
  - ▷ v = Orthonormal vector that is equal to the hkl in length, position, and direction in space.

2.9 getRotationMat.m

Determine a rotation matrix for converting between axis systems.

- Inputs
  - ▷ M1, M2 = Matrices that describe an axis system using vector components. eg: [1 0 0; 0 1 0; 0 0 1] for x, y, and z.
- Actions
  - ▷ Get the cosine between each comparative point in both matrices using:
    - ○ rotMat=[getCos(M1(:,1),M2(:,1)),getCos(M1(:,2),M2(:,1)),getCos(M1(:,3),M2(:,1));getCos(M1(:,1),M2(:,2)),g
  - ▷ function val1=getCos(x1,x2)
    - ○ val1=x1'*x2/(norm(x1)*norm(x2));
- Outputs
  - ▷ rotMat = Rotation matrix between axis systems.

3 Determination of Thermal Expansion Along a Vector

3.1 HKLExpansion.m

Calculate and plot thermal expansion wrt temperature along an hkl or uvw. Output expansion values for further manipulation.

- Inputs
  - ▷ conMat = Calculated conversion matrix between crystallographic and cartesian coordinates.
  - ▷ Tensors = CTE Tensors at a range of temperatures.
  - ▷ hkl = hkl direction in question.
  - ▷ Temps = Range of temperatures at which CTE Tensors exist.
  - ▷ xscreen, yscreen = Size of figure to be plotted in pixels.
  - ▷ lsfits = Polynomial fits to lattice parameters.
  - ▷ inUVW3 = Boolean to determine if hkl or uvw should be used for output.
- Actions
  - ▷ Determine thermal expansion in the direction of the given hkl over the range of temperatures described by Temps using getCTEVect.m
  - ▷ Configure and plot CTE data with labeling.
- Outputs
  - ▷ Exp = Thermal expansion in the direction of the hkl at the range of temperatures described by Temps.

3.2 orthToAng.m

Convert orthonormal x, y, and z coordinates to ϕ and θ angles with unit length, as length will be re-scaled later.

- Inputs
  - ▷ v1 = Vector described by cartesian coordinates.
- Actions
  - ▷ if abs(v1(1))<.000001 && abs(v1(2))<.000001
    - ◦ if v1(3)<0 && abs(v1(3))>.000001
    - ◦ phi=180; theta=0;
  - ▷ elseif v1(3)>0.000001
    - ◦ phi=0;
    - ◦ theta=0;
  - ▷ else
    - ◦ error('vector has no length and therefore no direction');
  - ▷ end
  - ▷ else
    - ◦ theta=atan2(v1(2),v1(1))*180/pi;
    - ◦ phi=acos(v1(3)/norm(v1))*180/pi;
  - ▷ end
- Outputs
  - ▷ phi, theta = Angular components of cartesian vector.

4 Miscellaneous Mathematical Operations

4.1 UVWtoHKL.m

Convert a vector composed of u, v, and w components to a vector composed of h, k, and l components at a given temperature.

- Inputs
  - ▷ lsfits = Polynomial fits to lattice parameters.
  - ▷ UVW = uvw in question.
  - ▷ Temp = Temperature to determine hkl relation at.
- Actions
  - ▷ Hden=(U*a^2+V*a*b*cos(gamma)+W*c*a*cos(beta));
  - ▷ Kden=(U*a*b*cos(gamma)+V*b^2+W*b*c*cos(alpha));
  - ▷ Lden=(U*c*a*cos(beta)+V*b*c*cos(alpha)+W*c^2);
  - ▷ %H/Hden=K/Kden=L/Lden, assume H+K+L==1 to solve for an HKL.
  - ▷ H=1/(1+(Kden/Hden)+(Lden/Hden));
  - ▷ K=(H/Hden)*Kden;
  - ▷ L=(H/Hden)*Lden;
  - ▷ /rcHKL=[H;K;L];
- Outputs
  - ▷ HKL = hkl calculated from input UVW.

4.2 HKLtoUVW.m

Convert a vector composed of h, k, and l components to a vector composed of u, v, and w components at a given temperature.

- Inputs
  - ▷ lsfits = Polynomial fits to lattice parameters.
  - ▷ HKL = hkl in question.
  - ▷ Temp = Temperature to determine UVW relation at.
- Actions
  - ▷ S11=b^2*c^2*sin(degtorad(alpha))^2;
  - ▷ S12=a*b*c^2*(cos(degtorad(alpha))*cos(degtorad(beta))-cos(degtorad(gamma)));
  - ▷ S13=a*b^2*c*(cos(degtorad(gamma))*cos(degtorad(alpha))-cos(degtorad(beta)));
  - ▷ S22=a^2*c^2*sin(degtorad(beta))^2;
  - ▷ S23=a^2*b*c*(cos(degtorad(beta))*cos(degtorad(gamma))-cos(degtorad(alpha)));
  - ▷ S33=a^2*b^2*sin(degtorad(gamma))^2;
  - ▷ Uden=(HKL(1)*S11+HKL(2)*S12+HKL(3)*S13);
  - ▷ Vden=(HKL(1)*S12+HKL(2)*S22+HKL(3)*S23);
  - ▷ Wden=(HKL(1)*S13+HKL(2)*S23+HKL(3)*S33);
  - ▷ %solve by stating that U+V+W==1
  - ▷ U=1/(1+(Vden/Uden)+(Wden/Uden));
  - ▷ V=(U/Uden)*Vden;
  - ▷ W=(U/Uden)*Wden;
  - ▷ UVW=[U;V;W];
- Outputs
  - ▷ UVW = UVW calculated from input HKL.