CyXtal: C(P)ython Package for Crystal Plasticity Data Analysis

development note

Chen Zhang

January 14, 2016

1 Module: ctools

2 Module: cxtallite

3 External Module: ext_aps

3.1 parsers.py

This particular module include several function that help facilitate the analysis of data collected at beamline-34-ID-E at Advanced Photon Source in Argonne National Lab.

1. **parser_xml(. . .)**

2. **strain_refine(. . .)**
   The algorithm of strain refinement is based on the software package "LaueGo" developed and maintained by Dr. Tischler at APS, ANL. [1]

   The strain refinement is not a direct (deterministic) measurement of strain using DAXM characterization. Instead, the strain tensor extracted from the Laue diffraction pattern is more of a "guessed value". The general algorithm of extracting strain tensor from DAXM scan data is as follows:

   - Acquire the crystal orientation by forward prediction of the Laue diffraction pattern. Due to small strain and experimental measurement error, the simulated Laue pattern will only partially match the measured pattern. Here the word "partially" has two levels of meanings:

[1] http://www.aps.anl.gov/Sectors/33_34/microdiff/

  - Only some of the diffraction spots are used to identify the crystal orientation of the voxel as it is not possible to match all diffraction spots in the pattern due to non-ideal deconvolution process. [2]
  - The matching between the calculated diffraction spots and measured diffraction spots are not perfect. The difference between calculation and measurements can be attributed to experimental error (uncontrollable and hopefully small) and the residual strain present in the crystal lattice, which distort the diffraction spots from its ideal position.

- Using the crystal orientation obtained by ignoring the lattice distortion (allowing relatively big mismatch of Laue patterns), it is possible to calculate the ideal [reciprocal] lattice vectors for given voxel. Perturbing the ideal [reciprocal] lattice vectors (or lattice constants) and recalculating the diffraction spots, one can theoretically stumble upon the real lattice vectors with lattice strain present. Comparing the transformation matrix between the ideal and real, the deformation gradient can be found through

  $$\mathbf{F} = \mathbf{V}_{\text{ideal}}\mathbf{V}_{\text{real}}^T$$

  where $\mathbf{V}$ is the bases formed by [reciprocal] lattice vectors.

  With the deformation gradient known, it is fairly easy to calculate the strain tensor through the Green strain tensor,

  $$\begin{aligned}\mathbf{C} &= \mathbf{F}^T\mathbf{F} \\ &= (1+\epsilon)^2 \\ &\approx 1 + 2\epsilon + O(\epsilon^2)\end{aligned}$$

  So the strain tensor for given voxel can be approximated by

  $$\epsilon = \frac{1}{2}(\mathbf{F}^T\mathbf{F} - \mathbf{I})$$

- Since it is not possible to directly compute $\mathbf{V}_{\text{real}}$, an automated optimization process is used to find a well "guessed" $\mathbf{V}_{\text{real}}$. However, it is possible that different optimization algorithm and initial guessing conditions will lead to different $\mathbf{V}_{\text{real}}$, which is why the strain tensor extracted using this method is not deterministic.

Another point worth mentioning here is that the full strain tensor can only be "guessed" when the beam energy is known. In other words, without the knowledge of the length of $\mathbf{q}$ (diffraction vector), it is not possible to inferred the whole strain tensor. However, if the volume of the unit cell is assumed to remain constant through deformation, the

[2] The shadows from neighboring grains and echoes from sample internal cannot be fully eliminated during the deconvolution of the diffraction pattern, which is used to acquire Laue pattern for each voxel during post processing of DAXM data.

deviatoric component of the strain tensor is not tied to $||\mathbf{q}||$. Thus, it is necessary to add this assumption to the optimization if white beam calibration is not done, which add one more step to get the deviatoric strain component,

$$\epsilon_{dev} = \epsilon - \frac{1}{3}\mathrm{tr}(\epsilon)$$

4 External Module: ext_damask

5 External Module: ext_vtk