

“Cracked” crystal

Important

The LiveUSB can be booted without changing the computer. Data prepared during this exercise are not stored. If you want to save the results, you have to copy them to your computer or to the cloud.

Hints

The window manager of the LiveUSB is called Xfce. It is most convenient for this exercise to have the focus following the mouse.

Blue icon in the top-left → Settings → Window Manager → Focus → Focus follows mouse

Additionally it is convenient to have smart placement of new windows

Blue icon in the top-left → Settings → Window Manager Tweaks → Placement → Minimum size of windows to trigger smart placement: put bar to the very left

Preparation

Open a terminal (use the blue icon in the top-left of the screen)

- `mkdir I0930a` (→ create directory for today's example)
- `cd I0930a`
- `ln -s /data/I0930a/*.sfrm .` (→ logical links from files in the repository to here)

Initial analysis of raw data

- `renameimages` (→ gives files a more convenient name)
 - default answers: yes
- `scancheck`
- `scancheckplot` (→ graphical output of the analysis)
- `scandb` (→ creates files with default parameters extracted from the raw images)
- `view read s01f0001.sfrm plot` (→ graphical display of the first image)
 - *Note:* you should see two new windows (total and zoom).
 - `exit` (→ leave the program “view”)

Peak search

- `low3` (→ creates artificial low image, use default answers)
- `buildsearch` (→ creates macro file for the program “view”)
 - prefix name: i
 - search type: 3
 - Nr of peaks in one image: 10
 - Minimum I/σ : 3.0 (→ determine only the strongest reflections)
 - Total number of expected peaks: 1000 (we can use the default, here)
 - Minimum resolution: 1.5
 - Maximum resolution: 27.5
 - Peak size in mm: 2.1
 - Display consecutive images: 1

- Ignore plots for n intermediate images: 9 (→ this is only for the graphical output)
 - Enter name of the first image: s01f0001.sfrm
 - More: no
- view @isearch (→ execute the macro which was just created)
 - exit (→ leave the program “view”)
- Two output files are created: i1.drx (for “dirax”), i1.pk (for “peakref”)

Indexing with dirax

- dirax read i1.drx (→ reads the file which was just created)
 - go (→ start the indexing)
 - go (→ another round of refinement)
 - acl (→ repeat the previous table)
 - the first column “acl” gives an identifier for every solution
 - the second column “nH” shows the number of fitting reflections
 - acl number (→ chose the number the solution)
 - savemat ia.rmat
 - exit

Search for second component

- buildrest
 - script name: restpeaks
 - Enter *.rmat filename: ia.rmat (→ this is the orientation matrix which we just determined)
 - Minimum resolution: 5.0
 - Maximum resolution: 25.0
 - max nr of extra peaks in one frame: 5
 - peak size in mm: 1.2
 - minimum values: 5.0
 - first framenummer: 1
 - number of frames: 250
- view @restpeaks (→ execute the macro which was just created)
 - exit

Indexing with dirax (2)

- dirax read rests01f.drx (→ reads the file which was just created)
 - go (→ start the indexing)
 - go (→ another round of refinement)
 - acl (→ repeat the previous table)
 - the first column “acl” gives an identifier for every solution
 - the second column “nH” shows the number of fitting reflections
 - acl number (→ chose the number the solution)
 - savemat ib.rmat
 - exit

Graphical display of orientation matrices

- view read s01f0001.sfrm (→ check the results graphically)

- rmat ia.rmat
- rmat2 ib.rmat
- plot datcol
- next plot datcol (→ repeat these commands several times)
- exit

Reduce unit cells (higher Bravais symmetry)

- rmatrix axcrit=0.5 ia.rmat write=iar.rmat
- rmatrix axcrit=0.5 ib.rmat write=ibr.rmat

Note: the true symmetry is monoclinic 2/m. You might need to change the axis criterion.

Exact relation between twin components

- 2view iar.rmat ibr.rmat
 - Write exactly rotated matrix into file iarr.rmat
 - *Note:* use a rotation in the laboratory system xyz. Suitable values for vector and rotation angle are proposed by the program.
 - After writing, leave the program with the command “q” at any stage

Unit cell refinement

- peakref
 - rmat iar.rmat
 - rmat2 iarr.rmat
 - pk i1.pk (→ file from initial peak search)
 - status (how many reflections do fit according to the default criteria?)
 - free rmat*
 - go3 (→ three cycles of refinement)
 - save detalign.vic (→ detector alignment)
 - savermat first.rmat (→ first twin component)
 - savermat2 second.rmat (→ second twin component)
 - exit

Cellplot

- cellplot first.rmat second.rmat
Press <enter> until you can select option 0 (“unit cell plots”)

Pymol output

- rmat2pdb 6 first.rmat (the 6 stands for six unit cells in each direction)
- rmat2pdb 6 second.rmat

Output files in direct space (uvw) and reciprocal space (hkl) are created. View the result in the program pymol (Display → Orthoscopic view). You may use all pymol features, for example to change colours.

Simulated precession images

- mprecession
 - select first.rmat (→ this is important)
 - mark the option “subtract low images”
 - press the button “Save & Run”
- view
 - rmat first.rmat (→ this is important)
 - rmat2 second.rmat
 - contour prec-0kl.cnt
 - colour
 - plot (two windows appear: total and zoom)
 - *Note:* with the command “go” you can get additional features. Right mouse button returns to the command line.
 - datcol
 - contour prec-h0l.cnt
 - plot
 - datcol
 - etc, etc, etc...
 - exit

Create box files

- rm *low* (we do not use the low image, here)
- bulddatcol (creates macro file for the program “view”)
 - script name: datcol
 - Enter *.rmat filename: first.rmat
 - rmat 2: second.rmat
 - rmat 3: <enter> (→ we do not have a third component, here)
 - Minimum resolution: 1.5
 - Maximum resolution: 27.5
 - Box size in mm: 3.5
 - Box depth in frames: 9 (→ always use an odd number)
 - Maximum duration: 5 (→ omit reflection with too long duration on Ewald sphere)
 - Minimum nr of reflections in one boxfile: 1000
 - Compress type: Z
 - Create boxfiles for second matrix: yes
 - Display images every n frames: 120
- View @datcol

Parallel processing

Boxfiles can be integrated in parallel. On a computer cluster you could install a queuing system. On a laptop or cluster you may use a simple batch system. For this purpose, create with a text editor the file ~/hosts

Example: content of the file ~/hosts for four cores:

localhost localhost localhost localhost

Integrate intensities

- `cd first-second`
- `cp /usr/local/calibration/smart10022786/*.pic .`
- `buildeval15`
 - crystal dimension: 0.2
 - mosaic: 1.2 (→ poor crystal quality)
- `eval15` (→ look interactively at the profiles)
 - `read s01f001.shoe.Z`
 - reflection number: 100
 - `pq10` (→ search for a strong reflection)
 - `refine`
 - `slicewrite` (→ write file “slices.dat” for program `refl3d`)
 - `pq10`
 - `refine`
 - etc, etc, ...
 - `exit`
- `refl3d` (→ graphical output of file “slices.dat”)
- `batchsetup`
- `batchstart` (→ integrate all images in batch mode)
- `cd ../second-first`
(→ repeat everything as for the first twin component)

Results

- `cd ~/l0930a`
- `any`
 - `read first-second/final.y.gz` (→ overlapping reflections are forbidden by default)
 - `allow overlapsum`
 - `hklf5`
 - `read second-first/final.y.gz` (→ overlapping reflections are forbidden by default)
 - `hklf5`
 - `exit`

Note: This results in a HKLF5 file for SHELXL. If you have a license for the program TWINABS, you can replace the command “`hklf5`” with the command “`twinabs`”. This gives an input file for TWINABS.