

First steps in LaueView

by Marius Schmidt
Vers1: April 00
Vers2: July 00

A. Installing LaueView on SGI-machines with IRIX operating system. LaueView can be installed in any directory, which is called \$LaueHome in the following. The following environment variables can be set, they are supposed to be overwritten by *.def files:

setenv CRYSTALNAME \$crystalname	this defines the crystal file (*.xtl) used
setenv CRYSTALINFO \$crystalinfopath	.xtl files are stored there
setenv WORK \$workpath	directory, where your *.log files etc. are.
setenv LAUEVIEWHOME \$LaueHome	the actual directory where LaueView is stored

```
setenv SCREENX 1024
setenv SCREENY 1024
```

care for correct CRYSTALNAME and CRYSTALINFO names. Otherwise, predictions (calculated reflex position) will be undefined.

B. Preparing detector images for input to LaueView:

B.1. Mar356 images (APS): These images are too large for LaueView. Image.mar356 images have to be unpacked and cropped for input. The unpacking is done by a Maresearch c-program called marcvt.c (for compilation care for the correct .h files). This program is executed with a simple cropping program (Zhong Ren) by the shell-script trimImage.mtf. In LaueView use the option <file><external image><16bit> and specify 2560 words and 2560 records. Note ! some motif files require further cropping. Since these images are already truncated there is no need for severe cropping. Just leave out an edge of approx. 10 to 20 pixels (15 recommended)

B.2. Quantum 4 (APS, Adxv) files: Not entirely tested yet for data quality. Modify the Q4 images with script convert.mtf. This uses a program called Q4_2_lv2 which gets rid of the header. If further swaps bytes and multiplies the pix value by 2 (this is necessary because LaueView divides every pix value by 2). In LaueView use 16 bit mode and just type in words and records (words 2304, records 2304). Cut 10 pixels from the edges.

B.3. Image intensifier (Thompson, ESRF). Same 16 bit option as for the Mar3560 scanner but with 1152 words and 1242 records. If the correction is changed the ESRF data can also exist in a Mar scanner like format (not yet tried). In motif files which require cropping cut out about 50 pixels.

B.4. MAR CCD (ESRF): Swap bytes, multiply by 2 (program SwapMCCD, motif convertMCCD.mtf). Read into LaueView with 4096 words and 4096 records in 16 bit mode.

C. Some words about the motif files.

Motif files are a combination of macros for LaueView and C-shell scripts. They are not meant to be a rigid construct. They can be customized and edited. Indexing of the first frame should be the final step of interactive operation of LaueView (all rejections are done by LauePlot). To customize and operate the motif files it is necessary to know basic

Unix C-shell programming and the logic behind the LaueView commands. The macro part of the motif files can be understood and changed by recording actions which you may perform interactively with LaueView. Exemplary motif files are stored in directory /maris/laueview/motifs.

1. Need to read in an image File:

<main> <File> <External image> <Read>

the problem may be that the filename is wrong because the setup differs (see A. Installing LaueView). You have to set the directory and the filename. After confirming <no> goto <Directory> and press <0> in order to input a new directory name. Put in a new name in slot 1 for example. After that select the name in slot 1 by pressing 1. You can always press <dir> for selecting or changing a directory name. The same logic holds for <filenames>. With <0> you can input a new filename in a selected slot, the other numbers select a filename. In general (but there are exceptions for example for image files and see also step 18) all file names may not be given with extension. Normally, the extension is generated within the program and if the filename matches an existing one, the file will be read in. With <view> <view> image files are displayed. Set words and records (pixel in x and y) in <main><file><external_image><16-bit>. Care for the correct settings of files and directories from the very beginning. If for example the data-set is not set or set wrong some motifs (for example integrate.mtf) will not run.

Filename conventions:

1. image
2. image
3. data-set

Directory conventions:

1. image directory
2. working directory (.def files etc.)
3. crystal info directory

note, put the right names into the first *.def file you were able to index a frame.

2. create a new xtal_info file:

First enter new crystal name:

<index> <crystal> <name> <name>

set the cell parameters accordingly

set the directory where the crystalinfo xtalname.xtl will be saved

goto <name> again and save.

(note: avoid the <LaueSim> menu)

3. index the image

if you cannot index an image proceed to another one. Take care, that you set the right phi angle for that frame. Otherwise you have to fight with odd phi settings in subsequent and previous frames (<main><index><goniometer>).

a. pick a new center <main> <pick> <center>

b. pick spots on ellipses: <pick a reflection and store in slot>, <pick>, enter slot number, do this for approx. 4 ellipses.

c. refine the angles of ellipses: <main> <ellipse> <angles only> you have to assign a slot to the ellipse, the parameters of the ellipse will be stored in this particular slot together with the already picked reflections: <slot# >, <pick an ellipse>, <refine>. note !!! before you start refinement, the distance has to be set right, enter <index><manual><refine><distance> !!

d. refine center and angles by using ellipses: <main> <ellipse> <angles and center>, chose the previous refined ellipses stored in slots: <slot #>, <data from slot> N-times. <refine>

e. index with ellipses: <main> <index>, <method> <ellipse>, hit <index> then give in the number of the slots you want to use for indexing. After <quit> indexing starts, with <accept> you can accept your favorite solution (the best solution will be automatically accepted). Hit <rev>iew and <su>perimpose and the pattern will be displayed.

f. Indexing by nodal-spots is similar and more easy. Pick nodal spots into a slot. Use the indexing method <nodal>. Specify the slot into which you picked the spots and start indexing.

Note, if you are indexing a frame different from frame #1 care for the correct PHI setting (<index><goniometer><phi>). Otherwise you will have trouble to assign the right phi value to previous frames since they differ from your notes.

4. Refine parameters

a. change goniometer settings if the superimposed pattern slightly differs from the data. This is necessary because the refinement routines are not robust enough: <main> <index> <goniometer>, change phi, chi and omega values. Set the wavelength range with <wav>.

b. refine setting: <main><index><refine>: First find spots <spot><find>, 600 spots with sigmacut of 10 is mostly sufficient. However, need to reduced sigmacut to 2.5 with sparse undulator data. Store spots in slot number 11 !!!

c. set <box> to 15 in <index><refine>, activate error plot: <error plot> <plotting Function on>. Refine orientation matrix by subsequent lowering the box size. Box size should be approx. 2 times the initial residual. Switch on paramters <refine> <paramters> in the following order:

beam center, distance, horizontal (or vertical, not both) pixel size, F, S (T) of detector, E, F, fudge factors, cell constants (one constant has to be fixed). Reduce the box accordingly. It should be about 2 times the initial residual. At the end of the refinement, a box of size 0.5 is expected.

5. Create subsequent .def files from the first image .def file. This is done by the motif default.mtf. You can also directly change the phi-angle setting with this shell script. Go through all images and check whether the superimposed simulated reflections fit the observed spots. If not, correct the diffractometer angles phi, omega and chi <main><index><goniometer>. This is particularly necessary for diffractometer which are not 100 % accurate or with crystals which were hit by the laser very hard and therefore moved in the capillary a bit. Store the changed settings in the corresponding .def file. This is tedious work, a good diffractometer is highly desirable. For subsequent changes of the *.def files the shell script default_modify.mtf exists.

6. Find spots in all frames. Typical sigmacut of 10 is sufficient for 600 spots in wiggler data. For undulator data the sigmacut may be released down to 2.5. Also box size can play a role, use 11 within crowded images, 9 with small, non streaky spots.

Examples:

- Wiggler Dec. 1999 ESRF, myoglobin crystal (Mb_24, small 300x100x50 μm^3), 600 spots with sigmacut of 6 (64 bunches).
- Undulator Nov. 1998 ESRF, myoglobin crystal (Mb_17, small 200x200x50 μm^3), only 245 spots with sigmacut of 2.5 (64 bunches).
- Wiggler April 2000 APS, myoglobin crystal (Mb_25, small 420x180x80 μm^3), 530 spots with sigmacut of 10 (100 sextets).
- Wiggler April 2000 APS, pyp crystal (PYP_B4, beam size restricted the size of crystal to 500x80x80 μm^3), 775 spots with sigmacut 10 (100 sextets).

7. After the crude setting of diffractometer parameter by hand (or presumably, if a good diffractometer is available you can set them automatically) all parameter that influence the calculated spot pattern have to be refined. This can be done in 2 ways: The refined parameter from frame N are transfered to frame N+1 and refined further for this particular frame. The danger is that the parameter, especially the cell constants, fade through the number of frames compensating for some fudge distortions. As a result they are completely off at the last frame and the data will be unreliable and possibly mis-indexed. The second way is more stable and robust and can be used primarily for data collected on detectors for which correction is not fully understood. This method still indexes right but puts the uncorrected detector distortions in some fudge factors (including cell constants) which are, intentionally, not transferred further. This way is recommended and uses the motif file refinedef.mtf. A final refinement with search box sizes up to 0.5 mm finishes the geometric refinement.

8. The set.mtf assigns a number to each image. This information is stored in a file called \$dataset.set. It also contains the refined image center (x,y coordinates).

9. For profile sampling the spot overlaps must be determined. Overlapping spots must be excluded from standard profiles. This is done by the motif spoverlap_rdb.mtf. The L-curve is not determined yet. Therefore, a box function with values of 1.0 over the wavelength range of interest should be put in. Note, the box function may not exceed Lmin and Lmax. Otherwise, a bus error will result. If a L-curve has been determined after scaling the whole process should be repeated starting with spoverlap_rdb and using this more realistic L-curve. This is necessary because the spot overlaps are determined by using resolution dependent bandpass. This means that the wings of the L curve are assumed not to contribute to the highest resolution data. Consequently, the bandpass will be deminished as a function of resolution. Take care that all filenames specified are correct. Don't interchange capitals with small letters. Try to be consistent. The extension after the dot is always written in small letters. Example: Mb17_ref.lam, PYP_6.lam etc.. Care for consistent spot radius. In crowded and streaky images it is sometimes advisable to select clean spots by using a relatively large spot_radius in spoverlap_rdb and selectsam.mtf. On the other hand, if not enough spots for the sampling can be found, the spot size can be reduced. If integration is executed in the next step with a different spot

size the `spoverlap_rdb` motif should be executed again with the same and correct spot radius. Check the spot size on a representative image with `<LaueView><window><open>` by counting the pixel of a spot. This does not effect the sample profiles because these are determined and stored in the previous steps. Typical spot radii are 12 pixel for typical, non-streaky data. Typically 1000 non-overlapping reflections will be selected by the next motif file `selectsam.mtf`. Attention ! Check out “spots checked”. Selected spots must not exceed 1000 spots. Otherwise the list of spots will simply be truncated to 1000. The excluded spots are not uniformly distributed over the detector space. Avoid this problem by setting `sigmacut` to higher values. (Example: `sigmacut 9` in good wiggler data). The selected spots will be profile fitted by `sampling.mtf` and used for further processing. Eliminate poorly determined profiles with `reject_sam.mtf`: `<plot> #1 #2. <set> error bits by flanking (min and max) values of #1 and #2. <reject><column>, #1 <return>, #2 <return>`. At the end hit `<output>` and then enter `<stop>`.

short list of options:

0: % rejected	
1: x	10: off-prediction dx
2: y	11: off-prediction dy
3: half-long axis a	12: slope px
4: half-short axis b	13: slope py
5: abnormal-sa	14: non-Gaussian-correction-ga
6: abnormal-ta	15: non-Gaussian-correction-gb
7: abnormal-sb	16: chi-square
8: abnormal-tb	17: abnormal-ta/abnormal-sa
9: non-radial correction epsilon	18: abnormal-tb/abnormal-sb

If you make rejections don't forget to record the values you specify and the resulting rejections for future examination.

10. Integration: uses output of motifs `spoverlap_rdb` and the sample profiles of sampling to integrate non overlapping and spatially (not harmonically) overlapping reflections. Specify same spot radius as in `spoverlap`. If spot radius is different repeat `spoverlap_rdb` with same spot radius. Then run `integration_rdb.mtf`. If the data set is not set and stored correctly with `<file>` in previous steps (preferable in `default.mtf`) `integrate.mtf` will not work. Integration should also be repeated if a box function was initially used for the bandpass.

11. Some very poorly integrated reflections (near the background) will be rejected by the script `rejectsht_bg.mtf` (the crystal file in `$crystalinfo` is of importance, check for the correct file name) and a subsequent interactive rejection cleans the list of integrated reflections (`rejectsht_1x1.mtf`). Don't forget to edit the correct crystal file into the `rejectsht_1x1` motif file. The command `rejectsht_1x1` will be executed by the following:

`rejectsht_1x1.mtf dataset outputfile`. Example: `rejectsht_1x1.mtf PYP_6 tmp`

Parameter prone to reject:

1: H	10: resolution
2: K	11: wavelength
3: L	12: background (count)

4: pattern #	13: intensity
5: multiplicity	14: sigma(I)
6: spatial overlap	15: intensity/sigma(I)
7: x (pixel)	16: height-background
8: y (pixel)	17: wavelength(A)/resolution(A)
9: log height (counts)	18: error code

Not all are worthwhile to plot. Use 10/15 to judge limit of resolution, you can also plot 17/15 to chop away low I/σ (remember wavelength/resolution = $\sin\theta$, a measure of the scattering angle, high I/σ are expected at low scattering angles and due to the scattering cross section at larger wavelength). V. Srajer found out that cutting in the *.sht files is superior over cutting in the *.fct files (rejectfct). Throwing away more reflection in the rejectsht_1x1 will usually result in better data sets. Typically I/σ will be cut between 0.2 and 0.5 which is dependent on the scatter plot. Check also 9,16 and 12,14. If nothing is rejected note the maximal and minimal values since they are used for the deconvolution algorithm and should be input to deconvolution.mtf. The result of this motif file is a single data set of unscaled integrated intensities (mv tmp.sht dataset.sht).

12. The unscaled integrated intensities will be scaled and the L-curve determined by scale.mtf. Scale.mtf has to be run several times. In the first run, no scale factors (.sca file) and no .fct file is available. Therefore use option .sht in the motif file. The name of the initial and subsequent L-curves is constructed from the data-set name (\$dataset.lam). In the first run, use the options weighting, Lorenz, refine L-cure and # of Chebyshev. Typical # of Chebyshev are 16 or 32 for the first run. This number can be expanded to 64 and 128 and more in subsequent runs. Look at the scatter plots whether you find ripples. This could be an indication for a poor or over-defined L-curve. For sharp single line undulator spectra 64 Chebyshevs are already sufficient. Subsequently activate isotropic scaling and B-factor scaling. After each scaling deactivate each parameter. Then increase number of Chebyshevs. It takes too long to refine all parameters together. Observe the R-value. Don't use anisotropic scaling or anisotropic B-values unless you have a huge number of frames (no idea how long it refines, with 96 Chebyshevs, maybe over night). To display the L-curve read in the scale file. <main><file><scale>. To write out the L-curve file (*.lam) use <main><file><X-ray spectrum><write>. Enter 0.01 for wavelength interval. To plot the L-curve in LaueView use <main><Scale><current><plot><X-ray spectrum><plot>.

After the initial successful determination of a L-curve of a particular X-ray beam go back to spoverlap_rdb, sampling and integration_rdb for the determination of new, improved sample profiles and improved unscaled intensities. If a L-curve was already determined by another data-set this step is not necessary. Between subsequent identical scale.mtf runs reject reflections with the motif rejectfct.mtf. Try to match the weighted and unweighted R_{sym} -value by rejecting bad reflections. This step makes rejections in order to obtain accurate reflections to determine an improved L-curve. The actual rejections on the data-set will be done after applying the scale factors to the .sht files. This step is controlled by motif file sht2fct1.mtf. In this way, additional necessary rejections on the merged reflections may be done after the sht2fct1 step. Write down the rejections you made in the

scaling procedure and try to reject consistently in further steps. According to the strategy to throw away more reflections in the *.sht files the rejections should not exceed 15%.

options for rejection:

1: H	18: f(wavelength)
2: K	19: f(scale)
3: L	20: f(temperature)
4: pattern #	21: f(absorption)
5: x [Pixel]	22: f(nonuniformity)
6: y [Pixel]	23: f(nonlinearity)
7: wavelength [Å]	24: f(radiation-damage)
8: intensity I	25: I/sigma(I)
9: sigma(I)	26: $F^2/\sigma(F^2)$
10: F^2	27: $\langle F^2 \rangle / \sigma \langle F^2 \rangle$
12: sigma(F^2)	28: $(F^2 - \langle F^2 \rangle) / F^2$
13: $\langle F^2 \rangle$	29: $(F^2 - \langle F^2 \rangle) / \sigma(F^2)$
14: $F^2 - \langle F^2 \rangle$	30: $(F^2 - \langle F^2 \rangle) / \langle F^2 \rangle$
15: f(general)	31: $(F^2 - \langle F^2 \rangle) / \sigma \langle F^2 \rangle$
16: f(Lorenz)	32: resolution
17: f(polarization)	

plot 29, 31, 30 and 28 against wavelength (7).

13. sht2fct1: applies scale factors to the *.sht files. All rejection made during scaling will be void since they affect the *.fct file in the previous step. Therefore, the rejections made after the sht2fct1 step affect the quality of the data set and are therefore of great importance. Make rejections with rejectfct.mtf. The options are the same as in step 12 since the motif file is the same. Do not reject too much (typically 8 % – 10 %, max 15 %).

14. apply: merge reflection to a mean value. Only singles are written out. Therefore, the data set which is put out is called *.sin.hkl for singles only. The R-values on I and F and the overall redundancy is computed in this step. The harmonic reflections will be written out in the next step.

15. deconvolution.mtf: Edit the mtf files with the min/max values determined in the rejectsht_1x1 step. Program deconvolutes harmonically overlapping reflections and writes them out to a file called *.har.hkl. Therefore, two data-sets exist, one with the singles, *.sin.hkl (calculated in apply) and one with the harmonics *.har.hkl. In a next step the sigma of the harmonic reflections has to be scaled to the singles and both data sets have to be merged.

16. final: merging of singles and harmonics. Don't start final on machines with IRIS 6.4 and larger. The exec format of the binary 'scalesig' is wrong. "*.sin.hkl" will be destroyed (you have to repeat apply). Execute shell script with final.mtf \$dataset.

17. for each *.har.hkl file and *.sin.hkl file reject outliers with rejecthkl.mtf. Combine harmonics and singles again with final.mtf.

examples:

```
rejecthkl.mtf PYP_6.sin
```

```
rejecthkl.mtf PYP_6.har
```

18. calculate completeness:

Start LaueView with a representative *.def file. Hit <main><LaueSim><Hkl><Load>, specify the entire filename with extension: for example PYP_6.hkl. Hit <completeness>. Toggle between <shell> and <accumulative>

Scaling of data sets together

(monochromatic scaling)

Scaling of Laue data to monochromatic or calculated structure factors may be difficult. Using LaueView, data which extend beyond the resolution limit tend to stubbornly resist rejection criteria (over-reduction of data). Therefore, highest resolution data may be wrong and a B-factor model for scaling does not apply. Consequently, Wilson scaling will be useless and even introduce an error. In order to find a suitable resolution limit the following can be done: Scale Laue data set to calculated amplitudes (or to monochromatic data set) by applying scale factor and isotropic B value. After scaling calculate scaling factors in resolution shells. Data start to become critical at a resolution where the scaling factors deviate from 1 (the resolution dependent scale factors correct for poorly determined intensities which cannot be scaled by a temperature factor model). Repeat scaling with lower resolution data.

The data sets must be indexed and sorted following standard convention. For myoglobin for example it can happen that a diagonal was used as a unit vector for indexing. Therefore reflections should be flipped before scaling. The scaling process is similar to that for Laue data. First determine the scale factors (in a file *.sca). The quality of the scale factors can be improved by rejections. Then recover all data (inclusive the rejections) and apply the scale factors.

1. As an example myoglobin diagonal settings are flipped to standard settings, input is a 5 column standard LaueView .hkl file, output is a 6 column standard LaueView .hkl file, note! you need the 6th column for phasing:

```
awk '{print -$1-$3,-$2,$3,$4,$5," 0.00"}' in.hkl > out.hkl
```

2. Sorting by LauePlot after changing order:

```
lp2.1 *.hkl -l phs.lab +u -rjo tmp -d -i $CRYSTAL
```

the file phs.lab contains the column labels for plotting, normally:

```
-----
lin h
lin k
lin l
lin F
lin sigF
lin phs
```

 \$CRYSTAL is the crystal-file currently used. All output to tmp. Further sorting on tmp using UNIX command sort:

```
sort +0n tmp > tmp1
lp2.1 tmp1 -duo tmp2
mv tmp2 *.hkl
```

3. Scale sigmas together. Since the scaling is weighted, the distribution of sigmas should be the same for all data-sets which are supposed to be scaled together:

```
scalesigma.mtf dataset1 dataset2 ..... datasetN
sigma but not |F| of dataset2 (to datasetN) will be changed.
```

4. Scale |F_s| together. Weighted by sigma(F).

```
mscale_init.mtf dataset dataset1 ... dataset100.
```

first isotropic scale factor will be determined, then the isotropic B value and in a next step both will be refined. Inputs are dataset1 to dataset100 Output is the dataset.fct (.fct files contain I_s or F_s, here they contain F_s) and a dataset.min file with contains all |F_s| starting with dataset1 and a scale factor file (dataset1.sca) which contains the isotropic scale and B-factors for all datasets.

The .min file looks like a .hkl file. It has numbers in the first column that corresponds to the number of dataset used in the scaling followed by h, k, l, F, and sig(F).

for example:

```
1  8 0 1  164.50 1.71
2  8 0 1   6.40 0.02
```

5. Use script rejectfct.mtf to reject data.

6. Use mscale_repe.mtf to repeat the scaling process since rejections change the outcome and the computation of the scaling parameters. mscale_repe uses the .fct file (all data-sets are written into one giant file) that was output by mscale_init.

7. rejectfct.mtf if necessary followed by mscale_repe.mtf

8. once scaling has converged (and improved with the rejections) you can apply the scale factors. First write back all the data, since the rejections were made to improve the scaling parameters and you might want to keep some of them in the final output. Use script mscale_reje.mtf. After this step you are able to reject some of those reflections you already have rejected for scaling. Do not reject more (would effect the scaling factors in a subsequent scaling step).

9. mscale_appl applies scale factors to the data in the *.fct file, merges the individual measurements and writes out the scaled data to separate .hkl files. Input is one .fct file (dataset.fct) containing all measurements, output are dataset1.hkl...dataset100.hkl.

```
mscale_appl.mtf dataset dataset1 dataset2 dataset3.
```

Merged data will be written out to dataset.min. dataset1.hkl to dataset100.hkl will contain scaled F_s (or I_s for monochromatic scale). Note, throughout the scaling the datasets *.hkl, *.fct and *.min files contain F_s. The only exception is the *.apply.hkl file which contains the square root of whatever is input to mscale_appl: In order to produce this file, equivalent measurements are merged together and the square root is taken. This only make sense if monochromatic data are reduced and scaled together since before merging these data are usually intensities.

10. scaling with CCP4:

- Laue Data into *.mtz file (Laue2mtz.sh):

rotaprep

Label H K L F **SIGF** (LaueData are Fs)

Format '(3F4.0,2F12.4)'

sortmtz

H K L M/ISYM BATCH (uses Unix sort with these keys)

truncate

(no wilson scaling, simply take square root)

labin lmean=I SIGIMEAN=SIGI (labels are automatically set by rotapr)

- scale with scaleit (refine anisotropic)

Calculation of structure factors and other useful stuff

1. Scaling of two data sets: see chapter Scaling of data sets together

2. Correlation between 2 data sets.

- join data sets by means of h k l. Use motifs joinfiles?.mtf. These scripts use Unix commands "join" to combine data sets and "awk" to manipulate them.

- read resulting file into LauePlot.

LauePlot -l join.lab jointfile.hkl -r -w

(-r in LauePlot gives a larger menu to chose from, -w makes white background)

3. Structure factors and electron density maps:

a: add line 6 to the *.hkl file you want to transform

awk '{print \$1,...,\$5,"0"}' input > output

b: add phase information:

stfact pdb-file input_without_phases.hkl input_with_phases.hkl

c: generate difference structure factors:

you can use fofcphs.mtf script:

fofcphs.mtf scaled_amplitudes1.hkl scaled_amplitudes2.hkl file_with_phases.hkl 1-2_with_phases.hkl

4. Calculation of λ -curves in distance shells (2 ϑ -shells) from the center of the detector:

Use the *.sht file after rejectsh_1x1. Copy the *.sht file to as many *.sht files as distance shells you want to analyze. With rejectsh_1x1 you can reject (or select) data in wav/res bins. Use following to calculate concentric shells of equivalent area on the detector:

Guess the maximum 2 ϑ angle or the maximum distance in mm from the center of the detector where you still expect reasonably good data. For example, for a resolution of 1.6 Å and a bandwidth of 0.7 to 1.8 Å you can expect data on a Q4 detector (d=100 mm distance) up to an r_{\max} of 70 mm-80 mm from the beam center. First shell radius and the following shells are simply defined by

$$r_1 = \frac{r_{\max}}{\sqrt{N}}$$

$$r_i = r_1 \cdot \sqrt{i}$$

N is the maximum number of shells you want to analyze (the proof that all shells have the same area is easy by substituting $r_i - r_{i-1}$ into πr^2). r_i is shell number i. The wav/res for shell i is calculated with the following formula:

$$\left(\frac{\lambda}{\vartheta}\right)_i = 2 \sin \vartheta_i = 2 \cdot \sin \frac{\arctan \frac{r_i}{d}}{2}$$

after the shells are defined reject data outside the shells. Determine scale factors (scale.mtf) for the individual shells. Reject data to improve scaling to a sufficient level (R=14% is o.k.). Produce a λ -curve file by reading the scale factors into LaueView (<file> <scale> ... read the right *.sca file). And write out λ -curve by <file><x-ray spectrum> Inspect all resulting λ -curves graphically with xmgr. To produce numbers you can calculate the linear correlation coefficient with a (fortran) program CorLam (by M.S.). This program takes up to 10 λ -curves and calculates the correlation coefficient between all those curves (fills an N x N/2 matrix).

XtalView

1. Set up crystal file:

Directory: Environment variable \$CRYSTALDATA

start 'xtalmgr', edit the <crystal> field by entering the type of crystal (Mb, SOD, PYP etc.), the cell parameters and spacegroup. Press <update>. A new entry is created in \$CRYSTALDATA/crystals and a new file 'crystal' (e.g. SOD or PYP) is created. By setting one of the existent crystal names in the .login file, you can run XtalView with this particular crystal parameters.

2. If you want to use user supplied weights use xfft, because xfit cannot produce maps with user supplied weight (newer versions may). Use option F*FOM for the amplitudes. In order to produce a file with 'H K L Fobs Sigma Phase' you have to follow the following steps:

a: create a *.phs file from a hkl file. The hkl file must have the format 'H, K, L, Fobs, weight, 6th column' (see above). The phase (*.phs) file unfortunately has the columns 'H K L Fobs Fcalc Phase', so the sigma (or your supplied weight) is missing. In order to fix this goto step b.

b: join the phase file with the *.hkl file and replace the Fcalcs by your weight. The weight can be generated in multiple ways. One way is to generate it from sigma and the amplitude by:

$$w = \frac{1}{1 + \frac{\sigma^2}{F^2}} \quad (1)$$

try also:

$$w = \frac{1}{1 + \sigma^2} \quad (2)$$

You can use motif file join4weight.mtf for this purpose.

For available light and dark amplitudes Zhong Ren proposed a weighting scheme that seems to be superior over the simple weights shown above:

$$w = \frac{1}{1 + \frac{\Delta F^2}{\langle \Delta F^2 \rangle} + \frac{\sigma_{\Delta F}^2}{\langle \sigma_{\Delta F}^2 \rangle}} \quad (3)$$

$\sigma_{\Delta F}$ is the sigma of the difference amplitude ΔF derived from the dark and light amplitudes. Error propagation shows that $\sigma_{\Delta F}$ is simply the sum of sigmas of the light and dark amplitudes. $\langle \rangle$ is the mean value over the entire data sets. Note, in contrast to Zhongs paper the mean of the square is taken (and not the squared mean). A (fortran) program called Weight_Z (by M.S.) can be used to calculate the weight (3) together with the difference amplitudes. You do not need to perform steps 2a and 2b but you have to supply phases. Hkl, F1 and sigmaF will be loaded from file1 and file2, a present 6th column is ignored. File 2 must have been scaled properly to file1. File3 only supplies phases, the amplitudes are ignored and useless for our purpose, since they are scaled to Fc anyway. The program takes the file1 as a reference and searches for matching indices hkl in the second file2. The difference amplitude $F_1 - F_2$ together with the above weight (3) and the phase is written out. The sort order of HKLs in the output file is the same as for input file1. Therefore take care for correct sort in file1.

c: binned scaling of for example LIGHT and DARK together with “xmerge”. First create two additional lines in each .hkl file. Run “xmerge” which takes as an input the two files. Choose common to output only common reflections and about 20 bins for the binned scaling. Output file format .df (double). Then use awk to separate column 3,4 and 8,9 from output file. 8,9 contain the scaled Fs and sigmaFs (or Is and sigmaIs). 3 and 4 are not changed. You can use Weight_Z to calculate weight and other useful stuff.

d: calculate F*FOM map in xfft with structure factor file Fo, FOM, PHI in degrees.

d: Load map into xfit (press right mouse button on <Load Map> and select *.map format). In case of PYP cell constants will sometimes be set wrong. Check cell constants in Xfit.

d. estimate electron density found in a xfft (fsfour) map on an absolute scale:

1. calculate phases with stfact and a model as complete as possible. Program will output a value to put the structure factors on an absolute scale: $\Phi 1$

2. do scaling and weighting of difference structure factors as usual, it is not necessary to put the structure factors on absolute scale.

3. calculate weighted map with xfft. Xfft outputs a value the map is multiplied with in order to achieve that one (map) sigma equals 50 electrons/A³. This factor is called $\Phi 2$ in the following.

Structure factors and map of step1:

$$\begin{aligned} F^{abs} &= F^o \cdot \Phi 1 \\ \rho^{abs} &= \sum F^{abs} \dots = \sum F^o \cdot \Phi 1 \dots = \Phi 1 \cdot \sum F^o \dots = \Phi 1 \cdot \rho^o \end{aligned} \quad (4)$$

The map of step2:

$$\rho^{xfft} = \sum F^o \cdot \Phi 2 \dots = \Phi 2 \cdot \sum F^o \dots = \Phi 2 \cdot \rho^o \quad (5)$$

combining 4 and 5:

$$\rho^{abs} = \frac{\rho^{xfft} \cdot \Phi 1}{\Phi 2}$$

The same reasoning holds for difference structure factors and difference electron density.