How to Process Laue Data Using **Precognition**

Zhong Ren

This tutorial shows a step-by-step procedure to process a Laue dataset. One of the first Laue dataset taken on 14-ID beamline after its upgrade is chosen as an example. This dataset consists of 31 images collected from a wild-type photoactive yellow protein (PYP) crystal at its ground state with 2° φ spacing between consecutive images, and φ spanning from -30° to 30°. A Mar165 CCD detector was used at a crystal-to-detector distance of 100 mm. Two collinear undulators U23 at a gap of 10.78 mm and U27 at 15.87 mm provided the X-ray source. The elapsed exposure time of each image is 2.13 s, but within this exposure time, there were only 8 X-ray pulses of 2 μs each. The synchrotron of APS was running in 324-bunch mode (APS fill patterns are described at http://www.aps.anl.gov/Facility/Storage_Ring_Parameters/node5.html).

PDF file of this document
Laue images in a tar-ball (99 MB)
Input scripts in a tar-ball
Entire processing directory in a tar-ball (103 MB)

0. Soft Limits

The particular sample used wasn't the greatest PYP crystal. The images were well exposed. The last image shows slightly decayed diffraction. Figure 1 shows the first image with some defect spot shape.

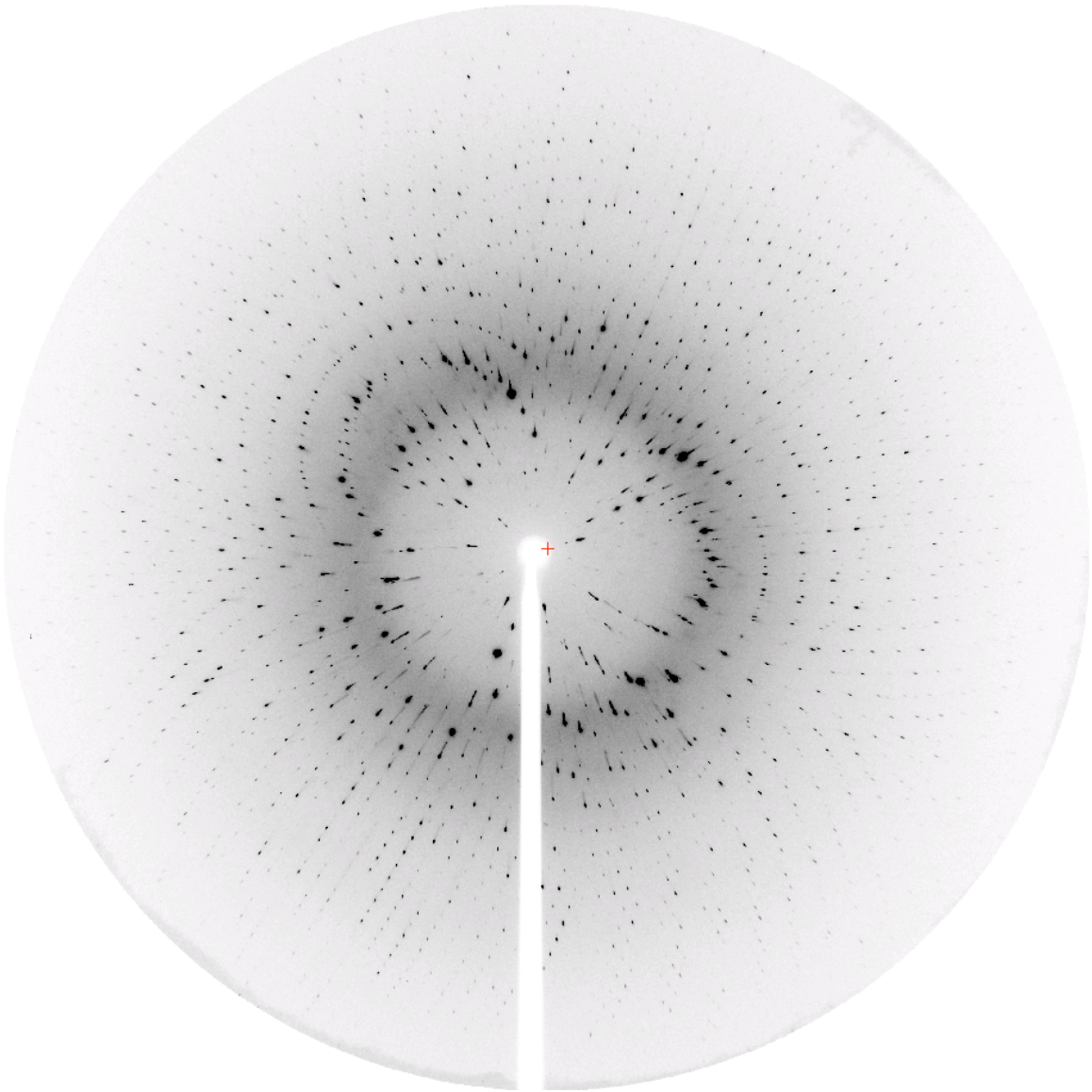

Figure 1. The first image in the dataset `pyp1_001.mccd`.

Before data processing, it is very useful to learn the soft limits of the dataset, such as a reasonable σ-cut (i.e., the minimum signal-to-noise ratio) and resolution limit d_{min}. They are called soft, since these limits are often associated with larger uncertainties than those of a monochromatic dataset. Nevertheless, choosing the appropriate values leads to trouble-free processing. A small script `limit1.inp` helps to identify these soft limits.

```
diagnostic    off
busy          on

Input
   Format     MarCCD
   Distance   100
   Center     1000.5 1030.5
   Pixel      0.0792 0.0792
```

```
   Image      images/pyp1_001.mccd
   Wavelength 1.05 1.4 1.09
   Quit

Spot 10 4 3.7 pyp1_001.spt
Profile
Limits
Quit
```

Listing 1. Input script `limit1.inp`.

The first two lines of the script `diagnostic` and `busy` are switches that controls message output. This script usually runs in foreground, so that busy messages are switched on. To run this script in foreground and save a copy of the log file, type:

```
% Precognition limit1.inp | tee limit1.log
```

The middle section provides a number of input data, all of which should be available from a beamline staff. The three values to the keyword `Wavelength` are λ_{min}, λ_{max}, and $\lambda_{reference}$. The wavelength where the source spectrum peaks makes a good reference wavelength.

The command `Spot` takes three values for spot length, width in pixel and σ-cut, which can be thought as the third dimension. At the beginning, one may simply choose 10, 10, and 3 as defaults to these values, if no better idea. We will see shortly that this procedure learns better values from the image specified. It performs a spot recognition throughout the image, and optionally saves the spots into a file, if a filename is given. It is highly recommended to plot the spots with your favorite plotting software. For example, `gnuplot` (`http://www.gnuplot.info`) is convenient to use for this purpose.

```
% gnuplot
gnuplot> plot 'pyp1_001.spt' using 4:5
gnuplot> set size ratio -1
gnuplot> set yrange [2000:0]
gnuplot> replot
```

Listing 2. Plotting a spot file using `gnuplot`.

It is important to make sure the plotted pattern (Figure 2) looks like the one in the original image (Figure 1) before proceeding further. If the original image is very over or underexposed, or if the crystal diffracts poorly, the first try of `limit1.inp` may result in something far from expectation.

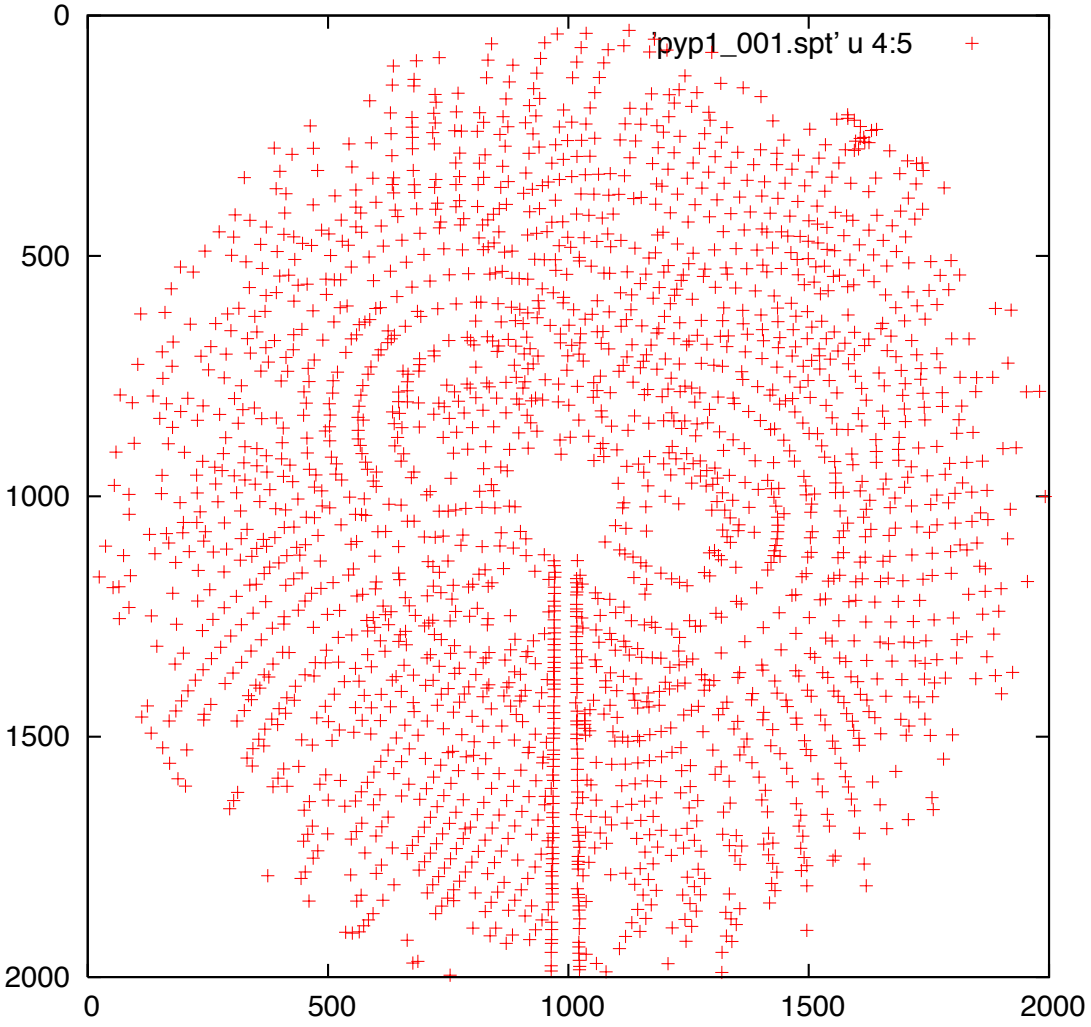

Figure 2. Scatter plot of the recognized spots saved in `pyp1_001.spt`.

`Precognition` prints some ASCII arts in its log file to signal important messages of different types. A few numbers near the end of the log file `limit1.log` are interesting. The first is about the mean spot profile, which is generated by the command `Profile`. The values of overall spot length and width shall replace the first two arguments to the command `Spot` above.

```
 ______
|      )_
| Report |
| ------ |
| ------ |
| ------ |
| ----   |
|________|

An overall mean profile is recognized.
Semi-major & -minor axes (pixel): 2.58865 1.05309
Non-elliptical correction:        0.000305247 0.0112322 -0.00527107 0.0050105
```

```
Non-Gaussian correction:          0.914063 0.780225

Overall spot length is set to 10 pixels.
Overall spot width  is set to 4 pixels.
Estimated crystal dimension is 0.192782 mm.
Estimated mosaic spread in FWHM is 0.110279 degree.
```

Listing 3. Log file section regarding a mean spot profile from `limit1.log`.

The command `Limits` generates more messages about the soft limits. The most important soft limit is σ-cut at various stages of processing. This procedure identified a σ-cut of 3.9 results in 10% spots that exceed an underlying spot distribution model. This value shall replace the third argument to the command `Spot` above. The spot distribution model implemented may not be very accurate, and may even fail completely under some circumstances, but the σ-cut values it identifies serve as guidelines during the following steps of processing. For example, during indexing and geometry refinement, the suggested range of σ-cut 4.5 to 5.8 will be helpful.

```
Best sigma-cut estimated at 4.45.
1803 real spots on this image.
Sigma-cut results in 10% noise is 3.87.
Suggested sigma-cut for indexing and geometry refinement is between 4.45 and
5.84.

Maximum spot density 9.6/mrad at Bragg angle of 14.3 degree.

Diffraction limit estimated at Bragg angle of 21.8 degree or 1.68 A resolution.

Suggested resolution for indexing and geometry refinement is 2.11 A.
```

Listing 4. Log file section regarding some soft limits from `limit1.log`.

1. Indexing

The first step of processing is indexing, that is, an assignment of Miller indices to all spots on the images. Indexing is required only for one image, often the first in a set, if all others have a known spatial relationship with the indexed one. Indexing is done by another small command script `index.inp`.

```
diagnostic    off
busy          off

Input
   @ pyp.inp
   Format     MarCCD
   Distance   100
   Center     997.5 1031.8
   Pixel      0.0792 0.0792
   Omega      -90 0
   Goniometer 0 0 -30
   Image      images/pyp1_001.mccd
   Resolution 2.2 100
   Wavelength 1 1.4 1.1
   Quit
```

```
Spot 10 4 5.8 pyp1_001.re.spt
Ellipse
Pattern     5 pyp1_001.pre.spt
Quit
```

Listing 5. Input script `index.inp`.

This script looks very similar, except that the `Input` section contains more information, and two different commands `Ellipse` and `Pattern` are used. In the `Input` section, the arguments to commands `Format`, `Distance`, `Center`, `Pixel`, `Omega`, and `Wavelength` should be known from a beamline staff. The command `Omega -90 0` is specific to 14-ID beamline of BioCARS.

The argument to `Image` specifies a filename. A relative path can be used as shown. All images are arranged in a subdirectory `images`.

The three arguments to `Goniometer` are ω, κ, and φ in degree.

A special command `@ pyp.inp` redirects the input stream to another script `pyp.inp`, which contains one line `Crystal 66.9 66.9 40.8 90 90 120 173`. These values are the unit cell constants and space group number. This line can optionally replace the `@` command.

Finally, the resolution limit often has significant impact to the processing. Suggestion from `limit1.log` (Listing 4) shall be followed. During the stages of indexing and geometry refinement, conservative resolution limit often works better.

The command `Spot` is exactly the same as before. It is recommended to use a σ-cut towards the higher end of the suggested range (Listing 4).

```
5 possible crystal orientations are recognized;
corresponding cell constants and detector parameters are refined.

Indexing 1
R.M.S. deviation (pixel):              2.44762
Number of spots matched:             570
Cell lengths (Angstrom):              66.9000    66.9000    40.8000
Cell angles (degree):                 90.0000    90.0000   120.0000
Euler angles (degree):                20.4120   111.4379   -84.9195
Euler angles (radian):                 0.3563     1.9450    -1.4821
Missetting matrix:                    -0.04397746   0.94481500   0.32463919
                                       0.37208200   0.31706507  -0.87236731
                                      -0.92715747   0.08242790  -0.36549237
Goniometer omega, chi, phi(degree):    0.0000     0.0000   -30.0000
Omega-axis polar orientation (deg):  -90.0000     0.0000
Detector type:                      flat
Crystal-to-detector distance (mm):   100.0000
Direct-beam center (pixel):          998.3930    1031.8162
Pixel size (mm):                       0.0792000    0.0792000
Detector swing angles (degree):        0.0000       0.0000
```

```
Detector tilt angles (degree):         0.0000       0.0000
Detector bulge corrections (10^-12):          0            0
```

Listing 6. Parameters of one orientation match in `index.log`.

The command `Pattern` accepts an integer and a filename. The integer specifies how many orientation matches should be found before the program exits. The orientation matrices and other parameters will be printed in the log file. The solutions will be sorted by their merits. The first two numbers are the most important: RMSD and matched spots. Obviously, the smaller the RMSD and the more the matched spots, the more reliable the solution. The filename is for another spot file that contains predicted spots under the best orientation match. The spot files `pyp1_001.re.spt` (*re*cognized) and `pyp1_001.pre.spt` (*pre*dicted) can be now plotted together to show the correctness of the indexing (Figure 3).

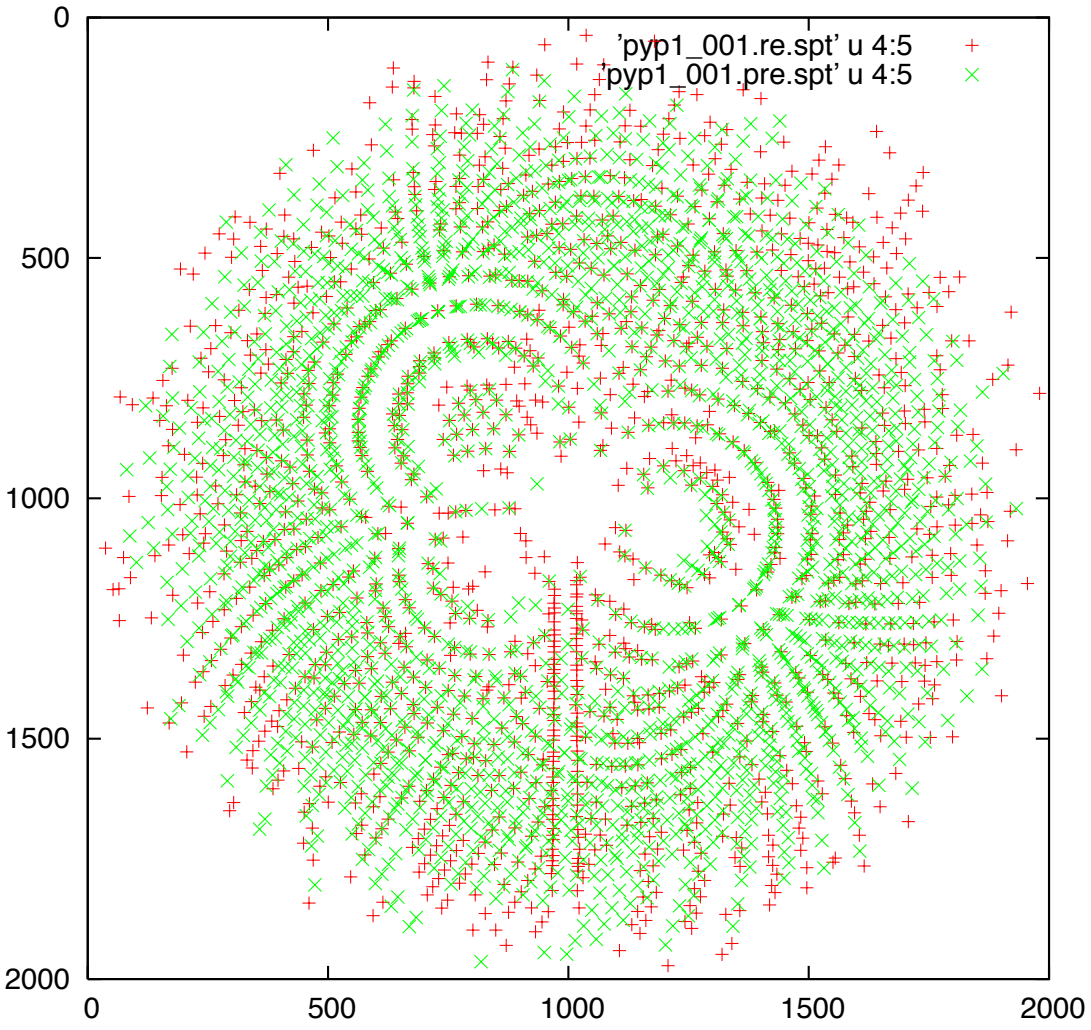

Figure 3. Superposition of the recognized (red +) and predicted (green ×) patterns.

A silent product of this script `index.inp` is another input script `pyp1_001.pre.spt.inp`, that is, the string argument to `Pattern` appended by `.inp`. This newly generated file essentially repeats all parameters listed in Listing 6, except that this file has the valid syntax to be loaded back in (see below).

2. Geometry Refinement

As shown in Figure 3, these two patterns obviously match with each other, but they look a bit displaced. Geometry refinement is a process to minimize the displacement while adjusting the geometric parameters, such as unit cell and detector parameters. On the other hand, this is also a process to distribute the crystal orientation found from one image by indexing to an entire dataset, provided that all images are related by a known φ-spacing.

```
diagnostic    off
busy          off

@ pyp1_001.pre.spt.inp

Input
   Crystal    0 0 0.2 0 0 0 free
   Format     MarCCD
   Distance   1.0    free
   Center     1.0    free
   Pixel      0.0001 free
   Tilt       0.2    free
   Bulge      fix
   Goniometer 0 0 -30 pyp1_001.mccd
   Goniometer 0 0 -28 pyp1_002.mccd
   …
   Goniometer 0 0  30 pyp1_031.mccd
   Resolution 2.0 100
   Wavelength 1 1.2 1.03
   Spot       10 4 4.5
   Quit

Dataset       progressive
   In         images
   Quit

Quit
```

Listing 7. Input script `refine.inp` (with omission).

Input script `refine.inp` first uses the `@` command we have seen before to load back the geometric parameters found by indexing. This set of parameters is the starting point of a refinement. Once again, an even longer `Input` section provides more control. Notice that the numerical argument turns into an allowed standard deviation when `fix` or `free` is the string argument. If a standard deviation is supplied to a parameter, the displacement of this parameter will be suppressed according to the standard deviation. This provides a means to restrain the refinement before some parameters drifting too far away from their physically meaningful ranges.

A sequence of the same command `Goniometer` adds frames to the dataset to be processed. From this point on, the word 'process' deviates to different meanings, but the command that triggers the 'process' is always `Dataset`. Now, 'process' stands for geometry refinement, and we choose one of the refinement modes called `progressive`.

As the log file reported, three files are generated for each image: two spot files and an input script `pyp1_001.mccd.inp`. These spot files can be again plotted together to check the goodness of refinement. Nevertheless, the log file reports a much better fit compared to the RMSD and matched spots after indexing. Now, the set of input scripts contains all geometric parameters needed for prediction of Miller indices, location, and wavelength of all spots in the dataset, visible or too weak to show.

```
File ./pyp1_001.mccd.re.spt is overwritten.

…

Cell constants, lattice orientation, goniometer setting,
and detector parameters: after geometric refinement

Title:                              pyp1_001.mccd
Cell lengths (Angstrom):              66.9000    66.9000    41.0262
Cell angles (degree):                 90.0000    90.0000   120.0000
Euler angles (degree):                20.3627   111.5005   -84.9477
Euler angles (radian):                 0.3554     1.9461    -1.4826
Missetting matrix:                    -0.04447376   0.94509715   0.32374908
                                       0.37291472   0.31635045  -0.87227118
                                      -0.92679917   0.08193762  -0.36650993
Goniometer omega, chi, phi(degree):    0.0000     0.0000   -30.0000
Omega-axis polar orientation (deg):  -90.0000     0.0000
Detector type:                      flat
Crystal-to-detector distance (mm):   100.2936
Direct-beam center (pixel):          998.1205    1031.8685
Pixel size (mm):                       0.0792047    0.0792000
Detector swing angles (degree):        0.0000       0.0000
Detector tilt angles (degree):        -0.1633      -0.2998
Detector bulge corrections (10^-12):           0            0
R.M.S.D. in pixel & matched spots:     0.3647 1093

File ./pyp1_001.mccd.pre.spt is overwritten.

File ./pyp1_001.mccd.inp is overwritten.
```

Listing 8. Result from geometry refinement in `refine.log`.

Optionally, a different refinement mode called `final` may be used after `progressive` mode is done. `final` mode may result in better fit in some, but not all, cases. Notice that the `@` commands are used repeatedly in script `final.inp`. This is also a chance to loosen up the standard deviations, resolution, and wavelength ranges, use more spots, or even refine more parameters.

```
diagnostic    off
busy          off
```

```
prompt off
result off
@ pyp1_001.mccd.inp
@ pyp1_002.mccd.inp
…
@ pyp1_031.mccd.inp
prompt on
result on

Input
   Crystal    0 0 0.2 0 0 0 free
   Format     MarCCD
   Distance   1      free
   Center     1      free
   Pixel      0.0001 free
   Tilt       0.2    free
   Bulge      fix
   Resolution 1.9 100
   Wavelength 1.02 1.2 1.04
   Spot       10 4 4.5
   Quit

Dataset       final
   In         images
   Quit

Quit
```

Listing 9. Input script `final.inp` (with omission).

Once a pattern is well refined, additional soft limits can be found out, that is, an estimated source spectrum, even before integration. The well-refined geometric parameters need to be loaded first. The command `Limits` now requires a numerical argument as λ_{max} and a filename for the estimated spectrum. The spectrum is plotted in Figure 4.

```
diagnostic    off
busy          off

@ pyp1_001.mccd.inp

Input
   Omega      -90 0
   Goniometer 0 0 -30
   Format     MarCCD
   Image      images/pyp1_001.mccd
   Quit

Spot 10 4 3.7 pyp1_001.spt
Limits    1.3 estimate.lam
Quit
```

Listing 10. Input script `limit2.inp`.

3. Integration

Integration is another meaning of 'process'; it also has various modes, but we choose `nonlinearAnalytical`, whatever it means, it is the most aggressive mode of integration (not necessarily the best for all cases). The `@` command `@ pyp1.inp` is essentially the same as loading a sequence of geometric input scripts in `final.inp`. They are now saved into another file `pyp1.inp` to keep the conciseness.

```
diagnostic    off
busy          off
warning       off

@ pyp1.inp

Input
   Image      start.lam
   Spot       10 4 4.5
   Quit

Dataset       nonlinearAnalytical
   In         images
   Resolution 1.5 1000
   Wavelength 1 1.3 1.04
   Quit

Quit
```

Listing 11. Input script `integrate.inp`.

A new item has appeared in `Input` section: `start.lam`, which is a small file that contains the source spectrum. The best spectrum can be used here, but if not, a few points on the spectrum are sufficient (Figure 4). The following file has wavelength in Å in the first column and the relative intensity in the second. The estimated spectrum `estimate.lam` from `limit2.inp` is another good option.

```
1.000 0.0
1.034 1.0
1.100 0.3
1.200 0.1
1.300 0.0
```

Listing 12. The starting source spectrum `start.lam`.

The spot parameters are the same, length, width, and height in the form of σ-cut, but the σ-cut is particularly important here, since it is one of the criteria to select sample spots for profile fitting. Needless to say, this has no effect if a non-profile-fitting integration mode is chosen (see `Precognition` manual for other integration modes). Follow suggestion in `limit1.log` and check `.re.spt` by plotting.

Resolution and wavelength ranges are now specified in `Dataset` section, since they are very specific to the integration process. All other input parameters are removed, since they reside in the set of geometric parameter scripts now.

Integration produces a set of `.ii` files that contain *i*ntegrated *i*ntensities. Distributions of all kinds can be plotted from these files for diagnostic purposes.

4. Scaling and Wavelength Normalization

This procedure is a large-scale, multi-parameter, nonlinear minimization. In most cases, a single pass of `scale.inp` solves the problem nicely, but sometime two or more passes may further improve the result.

```
diagnostic      off
busy            off
warning         off

@ pyp1.inp
# @ restore.inp

Input
   Image        start.lam
   Resolution   1.5 1000
   Wavelength   1 1.3 1.034
   Anomalous    off
   Quit

Scale
   Restore      restore.inp
   Sigma        2
   Mosaicity  0 fix
   Isotropy   0 scale
   Isotropy   0 temperature
   Expansion    fix
   Lambda-shift free
   Chebyshev    64
   Mapping      nonlinear
   Mode         global
   Quit

Quit
```

Listing 13. Input script `scale.inp`.

This script seems to have the same syntax and style, but is for another program `Epinorm`, one in `Precognition` package. Feeding this script to the program `Precognition` will cause an error.

```
% Epinorm scale.inp | tee scale.log
```

A # command appears in this script, which effectively comments out the line (see below for more on this line).

Once again, the starting source spectrum `start.lam` (Figure 4) is not required to be very accurate. In the worst case, if it is omitted, the program starts from a straight line.

The resolution and wavelength ranges should copy from the integration, unless specific ranges are desired.

`Scale` section listed in Listing 13 is the most common use of the program. The subcommands in the section are controls on data, parameters, and model selections. Please see `Precognition` manual for details. The first subcommand `Restore` specifies a filename for a parameter script, which can be loaded back in later. This is equivalent to the script for geometric parameters in indexing and refinement, except that the scaling parameters are stored here, and there is only one script needed for a dataset.

If a second pass of scaling is needed, the line `@ restore.inp` should be uncommented. Obviously, the line `Image start.lam` is then no longer needed.

The log file first reports about loading of the integrated intensities, from which one may get an idea of overall redundancy and data distribution across all images in the set.

```
73559 integrated intensities representing 12671 unique reflections from 31
frames loaded.

Integrated intensities from each frame
pyp1_001.mccd.ii 2430
pyp1_002.mccd.ii 2447
…
pyp1_031.mccd.ii 2319
```

Listing 14. Report on integrated intensities loaded for scaling.

Before scaling starts, there is one section in `scale.log` reports the starting statistics. The total, accepted, and rejected measurements are a few numbers to watch, where a measurement is an integrated intensity. R_{model}, R_{merge}, and signal-to-noise ratio at the beginning are going to be poor. The procedure runs through several cycles, within each certain parameters are scheduled to be refined. The same statistics are reported after every cycle. The final report should be checked.

```
Total measurements: 74899
  Accepted        : 73302 97.8678%
  Rejected        : 1597 2.1322%
Data-to-parameter : inf
Maximum iteration : 32
Tolerance         : 0.0001
Chi-square        :  4.75783e+06             -             -             -
R.M.S.D.          :      9130.93             -             -             -
(Current and previous values, absolute and relative changes)

Error distribution      Gaussian        Laplace     Lorentzian
Minimization target  4.75783e+06         147822         0.6428
Mean deviation           9130.93         2290.7        279.233
R-factor (%)              47.323        52976.1        33.0927
```

```
R-model          = 0.455773
Weighted R-model = 0.390213
R-models calculated from
73302 accepted integrated intensities.
These R-factors indicate how well the integrated
intensities are modeled by the current parameter set.

         R-merge on F^2 = 0.479852
Weighted R-merge on F^2 = 0.380871
         R-merge on F   = 0.238428
Weighted R-merge on F   = 0.195736
R-merges calculated from
73302 accepted integrated intensities of
12624 unique reflections with redundant measurements.
These R-factors indicate how well the symmetry-related
reflections agree with each other.

Mean F^2 / sigma(F^2)   = 5.83362
Mean F   / sigma(F)     = 11.0027
Signal-to-noise ratio calculated from
9306 unique reflections with highly redundant measurements.
```

Resolution range (A)	Unique refl.	Mean F^2/sigma(F^2)	Mean F/sigma(F)
1000.0000 - 3.8846	519	5.22	10.30
3.8846 - 3.0832	803	4.87	9.26
3.0832 - 2.6934	815	5.17	9.50
2.6934 - 2.4471	798	5.44	10.08
2.4471 - 2.2717	812	5.64	10.33
2.2717 - 2.1378	825	5.67	10.42
2.1378 - 2.0307	801	5.80	10.71
2.0307 - 1.9423	761	6.07	11.32
1.9423 - 1.8675	748	6.21	11.75
1.8675 - 1.8031	687	6.24	11.89
1.8031 - 1.7467	642	6.56	12.62
1.7467 - 1.6968	531	6.62	12.89
1.6968 - 1.6521	349	6.63	13.07
1.6521 - 1.6118	174	6.62	13.06
1.6118 - 1.5751	33	12.20	24.40
1.5751 - 1.5416	8	6.63	13.12

Listing 15. Report on scaling statistics in `scale.log`.

The most important result from wavelength normalization, as a part of scaling, is the wavelength normalization curve, also known as λ-curve. It is mostly the source spectrum, but with other effects mixed in, for example, the influences of energy responses of the detector, and absorption of all components along the beam, including the sample. Running a little script `lambda.inp` can save a λ-curve. The script loads back the scaling parameters, and the command `Lambda` accepts a filename.

```
prompt        off
result        off
@ restore.inp
prompt        on
result        on

Input
   Chebyshev  arbitrary
   Quit
```

```
Lambda          U23@10.78mm+U27@15.87mm.lam
Quit
```

Listing 16. Input script `lambda.inp`.

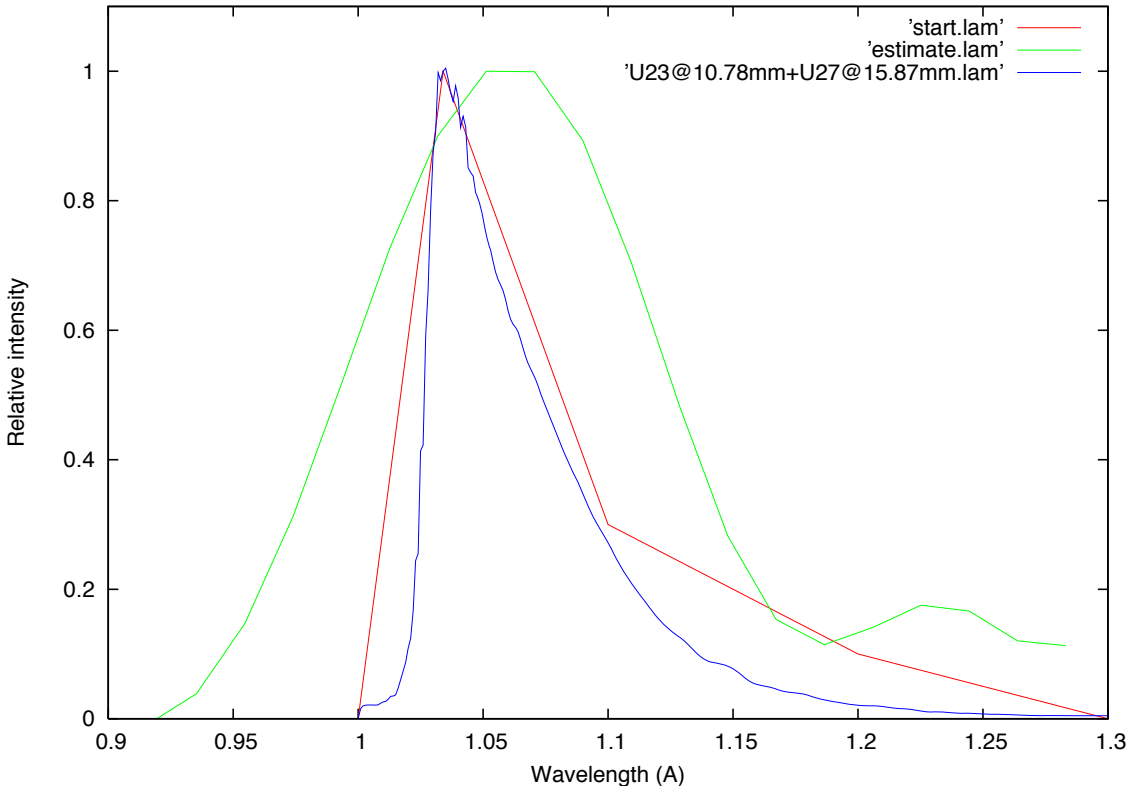

Figure 4. λ-curves.

5. Data Merging

If the λ-curve and other scaling statistics look acceptable, we are ready to take the last step to apply the wavelength normalization and all other scale factors, and to merge symmetry-related measurements. As shown in the script `apply.inp`, all geometric and scaling parameters must be loaded back in first; resolution and wavelength ranges are then specified in `Input` section. The overall spot length and width are again necessary, since they will be used during a final effort to deconvolute extremely close spatial overlaps. The `Scale` section seems to be out of place. It is optional to apply a local scaling algorithm, which is capable to reduce residual errors even further. The numerical argument to the command `Scale` specifies a σ-cut value for all data entering the local scaling procedure. Finally, the command `Apply` carries out data merging. The first numerical argument is again the σ-cut value, which should usually be the same as the one given to local scaling. The second numerical argument signals a mode of data merging. For general purpose, 3 is the default for harmonic and spatial overlap deconvolutions.

The filename is obviously for the merged structure factor amplitudes with *h*, *k*, *l*, *F*, and $\sigma(F)$ in the columns.

```
diagnostic    off
busy          off
warning       off

@ pyp1.inp

prompt off
result off
@ restore.inp
prompt on
result on

Input
   Resolution 1.5 100
   Wavelength 1 1.3 1.034
   Spot       10 4
   Anomalous  off
   Quit

Scale     1.5 local
   Quit

Apply   1.5 3 pyp1.1.5.hkl
Quit
```

Listing 17. Input script apply.inp.

Same as the normal scaling, local scaling is scheduled in several cycles too for different localities. Statistics are reported before and after each cycle. Comparison of R_{model}, R_{merge}, and signal-to-noise ratio before and after the procedure helps to decide whether local scaling should be used.

The command `Apply` will once again generate these statistics, and run through deconvolutions and calculate completeness tables along the way. The binning in the completeness table is designed in the way that the first line shows the completeness up to $2d_{\text{min}}$, a critical number in Laue diffraction.

Resolution (A)	Unique	Observed	Completeness (%)
100.00 - 3.00	2115	24236	98.23
3.00 - 2.38	2071	24112	97.81
2.38 - 2.08	2058	24098	97.64
2.08 - 1.89	2053	24068	97.21
1.89 - 1.75	2027	23846	97.05
1.75 - 1.65	1987	23426	95.06
1.65 - 1.57	1562	18518	74.90
1.57 - 1.50	144	1716	6.95
100.00 - 1.50	14017	164020	83.09

Listing 18. Completeness table in apply.log.