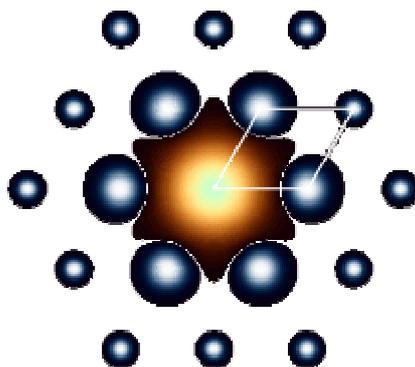


PrecognitionTM

(Release 5.0)

User Guide **with Reference and Tutorials**

John Zhong Ren



Renz Research, Inc.

All rights reserved
5/3/2006 11:15 AM

Precognition™ is a trademark of Renz Research, Inc.

John Zhong Ren

Renz Research, Inc.
P. O. Box 605
Westmont, IL 60559
U.S.A.

E-mail renz@renzresearch.com
WWW <http://renzresearch.com>
Tel. 630-230-0272
Fax 435-514-2645

What's New?

5/5/2006

Precognition 5.0.0

- Cylindrical and arbitrarily inclined flat detectors are supported.
- Spot recognition algorithm is improved.
- Geometry refinement is more robust.
- Neutron diffraction images can be processed.

6/1/2005

Epinorm 4.3.0

- An isotropic correction for geometric errors is introduced to wavelength normalization and scaling. This correction includes frame-specific wavelength shift and isotropic expansion correction. Errors due to geometric refinement become more tolerable.
- Nonlinear λ -mapping allows more parameters to describe sharply varying and peak regions of a λ -curve, and fewer parameters in flat regions and two wings. It is far better than the traditional linear mapping, since sharply varying details in a λ -curve can be described by a reduced number of total parameters, and at the same time, noise spikes at two ends of the spectrum can be suppressed.
- A new algorithm analogous to local scaling is available after wavelength normalization and normal scaling. The locality varies from wavelength, integrated intensity, to location on detector, etc. Local scaling corrects residual errors in an empirical fashion. It improves not only data merging of singles, but also harmonic and spatial-overlap deconvolution.
- A submenu is added to the top-level command `Scale`, which provides full control of this process. This will affect scaling command scripts written before this release.
- In the later cycles of scaling, the minimization target is switched to better suit Lorentzian error distribution instead of Gaussian.
- Minor changes in automatic data rejection.

1/4/2005

Precognition 4.2.1

- Faster integration.
- Bug fixing.

12/1/2004

Precognition 4.2.0

- Fixing and freeing geometric parameters become more convenient.
- Calibration of distance and cell constants during geometric refinement.
- A new integration mode `nonlinearAnalytical` is available.
- Analytical profile fitting works with a numerical compensation, which improves the accuracy of integration for strong spots.
- Negative integrated intensity becomes possible. Dealing with negative integrated intensities in Laue diffraction is complex. It is not recommended for general users.

Epiform 4.2.0

- Negative integrated intensity may contribute to scaling and be included in final results. Dealing with negative integrated intensities in Laue diffraction is complex. It is not recommended for general users.
- Frame-specific λ -curves are possible.

7/12/2004

Precognition 4.1.0

- Connection to the older program `LaueView` is no longer the default, but still possible by explicit command option, which signals the independence of Precognition system.

Epiform 4.1.0

- Extremely-close spatial overlaps can be deconvoluted by the same algorithm as harmonic deconvolution.
- All effects of energy partial are completely modeled in deconvolution of extremely-close spatial overlaps.
- Friedel pairs can be separated during data scaling and/or merging if anomalous scattering should be considered.

5/5/2004

Epiform 4.0.0

- Temperature factors can be fixed without refinement during scaling.
- Harmonic deconvolution is available, which marks the completion of all major functionalities in this software system.
- Crystal mosaicity, energy resolution and energy partial of harmonic reflection are completely modeled in harmonic deconvolution.

4/9/2004

Precognition 3.1.1

- Bug fixing.

3/26/2004

Precognition 3.1.0

- Soft limits and initial source spectrum can be recognized before and after indexing of a pattern.
- A new refinement mode `drunk` is added for solution of frequent crystal jumping.
- Random walk-around of crystal orientation is improved. Number of steps and radius can be automatically adjusted.
- Uncertainties of detector parameters can be input by user, so that freedom of these parameters are limited.
- User may choose to fix some parameters during geometry refinement.
- Manual geometry refinement is possible.

Epinorm 3.1.0

- Crystal mosaicity is refined.
- Partials in Laue diffraction due to a finite energy bandpass, called energy partial, are reduced.
- λ -curve becomes λ -surface. Wavelength normalization factor not only depends on wavelength, but also mosaicity and Bragg angle.
- Spike removal in λ -curve.
- Data rejection in the last release was based on absolute residual of fit which resulted in near constant final R -factors. It is changed in this release so that rejection is based on root-mean-squares of fitting residuals.
- Setting initial values of scaling parameters.
- Save and restore intermediate results.

11/6/2003

Epinorm 3.0.0

- The major addition to this release is a new program `Epinorm` (Energy Partial Improved NORMalization). `Epinorm` reduces Laue integrated intensities to structure factor amplitudes. Wavelength normalization, scaling, temperature factors, and beam polarization are determined. Compared to `LaueView`, `Epinorm` requires no multiple sessions, no interactive data rejection, no limitation to number of images, and runs faster. `Epinorm` makes anisotropic scaling possible.

9/23/2003

Precognition 2.2.0

- Indexing consistency check is available for re-indexing in middle of a set.
- Sorting of filenames in .gon file is removed. Frames are ordered as the sequence of loading.
- A major addition is residual pattern recognition (RPR) in geometry refinement. Residual pattern consists of all residual vectors from predicted spot locations to observed ones. This pattern varies according to the remaining errors in the parameters, and shrinks to a small Gaussian distribution when refinement is done well. Precognition automatically recognizes residual pattern using the newly developed RPR technique, makes geometry refinement more robust.

Precognition.py 2.2.0

- This is the first released trial GUI, including image display, zooming, contrast, brightness, marking recognized spots, etc.

Contents

	What's New Contents	iii vii
Chapter 1	Overview	1
1.1	Introduction	1
1.2	Edition, Version, Release, and Patch	2
1.3	Major Functionalities and Features	2
1.3.1	Precognition, a pattern recognition software for Laue diffraction	2
1.3.2	Epiform, a wavelength normalization and data scaling software for Laue diffraction	3
1.3.3	Features	3
1.4	A Tour of This Book	4
1.5	Conventions Used	6
1.5.1	Notations	6
1.6	Related Software	6
1.6.1	GCC	7
1.6.2	Python	7
1.6.3	Numeric/Numarray	7
1.6.4	PIL	8
1.6.5	Pmw	9
1.6.6	SWIG	9
1.6.7	FFTW	9
1.6.8	TNT	9
1.6.9	LAPACK and BLAS	10
1.6.10	GRACE	10
1.6.11	Gnuplot	11
1.7	Installation and Execution	11
1.7.1	Program Installation	11
1.7.2	License Installation	11
1.7.3	Execution	12
Chapter 2	Indexing of Laue Pattern	15
2.1	Required Input	15
2.1.1	Crystal information	17
2.1.2	Crystal-to-detector distance	17
2.1.3	Direct-beam center	18
2.1.4	Pixel size	19
2.1.5	Image format	20
2.1.6	Image file	20
2.1.7	Resolution and wavelength ranges	20
2.1.8	Exit from Input menu	20
2.2	First Indexing	20
2.2.1	Spot recognition	21

2.2.2	Auto-recognition of diffraction limit	25
2.2.3	Learning spot profile	26
2.2.4	Ellipse and nodal recognition	26
2.2.5	Indexing	26
2.3	Validation of Indexing	28
2.4	Routine Indexing	29
2.5	Mis-indexing	30
2.5.1	Check all possible matching	31
2.5.2	Use more nodal spots	33
2.5.3	Manual indexing	33
2.5.4	Other options	34
2.6	Quick Indexing	34
2.7	Goniometer Setting	34
Chapter 3	Geometric Refinement	35
3.1	Estimates of Soft Limits and Source Spectrum	35
3.1.1	Before indexing	35
3.1.2	After indexing	37
3.2	Goniometer Setting	39
3.3	Indexing and Refinement of a Set of Patterns	41
3.4	Crystal Slipping	44
3.5	Repairing an Individual Frame	46
3.6	Fixing Parameters and Limited Refinement	46
3.7	Calibration of Crystal-to-detector Distance and Cell Constants	47
3.8	Re-index	48
3.9	Manual Refinement	49
3.10	Final Refinement	56
Chapter 4	Spot Integration	59
4.1	General Issues for Integration	59
4.2	Integration On-the-fly, <code>box</code> Mode	62
4.3	Elliptical Peak Area, <code>fixedElliptical</code> Mode	63
4.4	Learned Elliptical Peak Area, <code>variableElliptical</code> Mode	63
4.5	Numerical Profile, <code>numeric</code> Mode	63
4.6	Analytical Profile, <code>linearAnalytical</code> Mode	64
4.7	Simultaneous Profile Fitting and Spatial-overlap Deconvolution, <code>nonlinearAnalytical</code> Mode	65
4.8	Numerical Compensation to Analytical Profiles	66
4.9	Hybrid Mode	67
Chapter 5	Data Reduction	69
5.1	Input Data and Control Parameters	69
5.2	Data Selection and Parameter Fitting	71
5.2.1	Data selection	72
5.2.2	Data isotropy	73
5.2.3	λ -curve modeling	74

5.2.4	Beam polarization	77
5.2.5	Crystal mosaicity	78
5.2.6	Absorption correction	78
5.3	Scaling Modes and Statistics	80
5.3.1	Scaling modes	80
5.3.2	Minimization cycle and statistical report	81
5.4	Saving and Restoring Scaling Results	85
5.4.1	λ -curves	85
5.4.2	Scaling parameter file	86
5.5	Local Scaling and Applying Scale Factors	90
5.5.1	Local scaling	90
5.5.2	Applying scale factors to singles	91
5.5.3	Data merging from a subset	93
5.6	Harmonic Deconvolution	94
5.7	Extremely Close Spatial Overlap	96
Chapter 11	Reference	99
11.1	Precognition_E*.x.*_P*.x precognition_E*.x.*_P*.x Precognition precognition	99
11.1.1	Precognition:Input (I i)	100
11.1.1.1	Precognition:Input:Format (F f)	100
	Precognition:Input:Format:Bas2000 (B b)	101
	Precognition:Input:Format:ESRF (E e)	101
	Precognition:Input:Format:Mar345 (M m)	101
	Precognition:Input:Format:MarCCD (C c)	101
	Precognition:Input:Format:Quantum315bin2x2 (9)	101
	Precognition:Input:Format:Quantum4 (4)	101
	Precognition:Input:Format: RigakuRaxisIPDSCQ (R r)	101
11.1.1.2	Precognition:Input:Image (I i)	101
11.1.1.3	Precognition:Input:Crystal (X x)	101
11.1.1.4	Precognition:Input:Omega (O o)	102
11.1.1.5	Precognition:Input:Goniometer (G g)	102
11.1.1.6	Precognition:Input:Matrix (M m)	102
11.1.1.7	Precognition:Input:Spot (L l)	103
11.1.1.8	Precognition:Input:Distance (D d)	103
11.1.1.9	Precognition:Input:Radius (D d)	103
11.1.1.10	Precognition:Input:Center (C c)	104
11.1.1.11	Precognition:Input:Pixel (P p)	104
11.1.1.12	Precognition:Input:Swing (S s)	105
11.1.1.13	Precognition:Input:Tilt (T t)	105
11.1.1.14	Precognition:Input:Bulge (B b)	105

11.1.1.15	Precognition:Input:Shift (S s)	106
11.1.1.16	Precognition:Input:Resolution (R r)	107
11.1.1.17	Precognition:Input:Wavelength (W w)	107
11.1.1.18	Precognition:Input:Chebyshev (V v)	107
11.1.1.19	Precognition:Input:Anomalous (A a)	108
11.1.1.20	Precognition:Input:Quit (Q q)	108
11.1.2	Precognition:Spot (S s)	108
11.1.3	Precognition:Profile (F f)	108
11.1.4	Precognition:Ellipse (E e)	108
11.1.5	Precognition:Pattern (P p)	109
11.1.6	Precognition:Limits (L l)	109
11.1.7	Precognition:Dataset (D d)	110
11.1.7.1	Mode of processing	110
	normal, Normal, or NORMAL	110
	progressive, Progressive, or PROGRESSIVE	110
	drunk, Drunk, or DRUNK	111
	calibration, Calibration, or CALIBRATION	111
	final, Final, or FINAL	111
	integration, Integration, or INTEGRATION	111
	box, Box, or BOX	111
	fixedElliptical, fixedelliptical, FixedElliptical, Fixedelliptical, or FIXEDELLEPTICAL	112
	variableElliptical, variableelliptical, VariableElliptical, Variableelliptical, or VARIABLEELLIPTICAL	112
	numeric, Numeric, or NUMERIC	112
	linearAnalytical, linearanalytical, LinearAnalytical, Linearanalytical, or LINEARANALYTICAL	113
	nonlinearAnalytical, nonlinearanalytical, NonlinearAnalytical, Nonlinearanalytical, or NONLINEARANALYTICAL	113
	hybrid, Hybrid, or HYBRID	113
11.1.7.2	Precognition:Dataset:In (I i)	113
11.1.7.3	Precognition:Dataset:Out (O o)	113
11.1.7.4	Precognition:Dataset:Path (P p)	113
11.1.7.5	Precognition:Dataset:OK (K k)	114
11.1.7.6	Precognition:Dataset:Resolution (R r)	114
11.1.7.7	Precognition:Dataset:Wavelength (W w)	114
11.1.7.8	Precognition:Dataset:Quit (Q q)	114
11.1.7.9	Connection to LaueView	114
11.1.8	Precognition:Quit (Q q)	114

11.2	Epinorm_Ex.x.x_Px.x epinorm_Ex.x.x_Px.x Epinorm epinorm	115
11.2.1	Epinorm:Input (I i)	115
11.2.2	Epinorm:Restore (R r)	115
11.2.2.1	Saving intermediate parameters	115
11.2.2.2	Epinorm:Restore:Polarization (P p)	115
11.2.2.3	Epinorm:Restore:Mosaicity (M m)	115
11.2.2.4	Epinorm:Restore:Scale (S s)	116
11.2.2.5	Epinorm:Restore:Temperature (T t)	116
11.2.2.6	Epinorm:Restore:Expansion (E e)	116
11.2.2.7	Epinorm:Restore:Lambda-shift (L l)	116
11.2.2.8	Epinorm:Restore:AnisoScale (A a)	116
11.2.2.9	Epinorm:Restore:AnisoTemperature (N n)	116
11.2.2.10	Epinorm:Restore:Wavelength (W w)	117
11.2.2.11	Epinorm:Restore:Normal (O o)	117
11.2.2.12	Epinorm:Restore:Coefficient (C c)	117
11.2.2.13	Epinorm:Restore:Quit (Q q)	117
11.2.3	Epinorm:Scale (S s)	118
11.2.3.1	Epinorm:Scale:Sigma (S s)	118
11.2.3.2	Epinorm:Scale:Data (D d)	119
11.2.3.3	Epinorm:Scale:Reference (F f)	119
11.2.3.4	Epinorm:Scale:Polarization (P p)	119
11.2.3.5	Epinorm:Scale:Isotropy (I i)	119
11.2.3.6	Epinorm:Scale:Mosaicity (M m)	120
11.2.3.7	Epinorm:Scale:Wavelength (W w)	120
11.2.3.8	Epinorm:Scale:Chebyshev (C c)	120
11.2.3.9	Epinorm:Scale:Mapping (N n)	120
11.2.3.10	Epinorm:Scale:Lambda-shift (L l)	120
11.2.3.11	Epinorm:Scale:Expansion (E e)	121
11.2.3.12	Epinorm:Scale:Restore (R r)	121
11.2.3.13	Epinorm:Scale:Mode (O o) initial, Initial or INITIAL global, Global or GLOBAL frame, Frame or FRAME local, Local or LOCAL test, Test or TEST	121 121 121 121 121
11.2.3.14	Epinorm:Scale:Quit (Q q)	122
11.2.4	Epinorm:Lambda (L l)	122
11.2.5	Epinorm:Apply (A a)	122
11.2.5.1	Data selection	122
11.2.5.2	Apply scaling parameters to single reflections	122
11.2.5.3	Merging single reflections	122

11.2.5.4	Harmonic deconvolution	123
11.2.5.5	Deconvolution of extremely-close spatial overlaps	123
11.2.6	Epinorm:Quit (Q q)	123
11.3	TBA	123
11.4	Precognition.py	123
11.5	Utilities and Diagnostic Tools	123
11.5.1	Spot recognition with pick.py	124
11.5.2	Superposition of two patterns with su.py	124
11.5.3	Curve of scatter plot with plot.py	125
11.5.4	Byte swapping with swap1.py	125
11.6	cpl.Precognition Module	126
11.7	Coordinates Systems	126
11.8	Files	126
11.8.1	File selection	127
11.8.1.1	File selection:Filename (F f)	127
11.8.1.2	File selection:Path (P p)	127
11.8.1.3	File selection:Name (N n)	127
11.8.1.4	File selection:Middle (M m)	127
11.8.1.5	File selection:Extension (E e)	127
11.8.1.6	File selection:Cancel (C c)	127
11.8.1.7	File selection:OK (O o)	127
11.8.1.8	Previously-selected filename	128
11.8.2	Crystal information file .xtl	128
11.8.3	Goniometer file .gon	128
11.8.4	λ -curve .lam	129
11.8.5	Spot file .spt	129
11.8.6	Integrated intensity file .ii	129
11.8.7	Energy-dispersive intensity file .edi	129
11.8.8	Structure factor amplitude file .hkl	129
Chapter 12	Tutorials	131
12.1	An Undulator Laue Dataset from a Protein Crystal	121
12.1.1	Visual evaluation of images and estimates of soft limits	121
12.1.2	Indexing	135
12.1.2.1	Indexing	136
12.1.2.2	Estimation of more soft limits	139
12.1.2.3	Indexing problems	141
12.1.3	Geometric refinement	142
12.1.4	Integration	144
12.1.5	Wavelength normalization and scaling	145
12.1.6	Data merging, harmonic and spatial deconvolution	147
12.1.7	Structure refinement	148
12.2	A 1/10 Dataset from a Small Molecule Crystal	149
Appendix 1	README File	157

CHAPTER 1

Overview

1.1 Introduction

Precognition is an end-user software system that processes single crystal diffraction images and yields experimentally-observed structure factor amplitudes. It is also an API of a small set of modules that can be incorporated into other software systems. Precognition is suitable for processing diffraction images from small molecule, protein and virus crystals. Each type of image processing requires very different implementation. The current release of Precognition focuses on white-beam Laue diffraction from X-ray and neutron. Precognition is also capable to process diffraction images from sources of much narrower spectrum, for example, ‘pink’ beam or quasi-Laue.

Diffraction image processing is the first step in a rather complex computational process leading to a crystal structure. It deals with the greatest portion of data in the computational pipeline. Automation level of an image processing software is important to the throughput of the entire process. Precognition utilizes some *pattern recognition* techniques to reduce user intervention as much as possible, and therefore enhance the automation level.

Success of crystal structure analysis and clarity of the resulting structure critically depend on the quality of the observed structure factor amplitudes. The result of diffraction image processing is a *precognition* of the outcome of the entire crystal structure analysis. Various new theories are applied and new techniques are implemented to improve the quality of diffraction image processing and data reduction.

Precognition heavily relies on CCL™/CPL™ (Crystallographic Concept Library™ and Crystallographic Protocol Library™) of Renz Research, Inc., and utilizes a wide range of public-domain software components (see 1.6 for details). Precognition is mainly written in object-oriented C++ and Python languages with small portion of the code in C and FORTRAN. All these languages are highly integrated.

Precognition can be interactively command-driven, or run through preprogrammed command scripts or via a graphical user interface. It can also be used as a small set of application program interface in forms of Python modules and classes that can be incorporated into other software systems. These styles of user interface may also be mixed in one program, and the choice of style depends on the functionality of a specific program.

1.2 Edition, Version, Release, and Patch

The name of a command-driven executable is `Program_Ex.x.x_Px.x`, where `Program` can be `Precognition` or `Epinorm` or others. The letter `E` indicates an edition for small molecule, protein, virus crystals, etc.; three numbers `x` are version, release, and patch numbers, respectively. The letter `P` indicates platform/operating system, and the other numbers show its version. `Precognition.py` is `Precognition` under a graphical user interface, which includes many additional functionalities that rely on graphics. Version number of `Precognition.py` can be found from its `Help/About` menu.



Figure 1.2.0.0.1 About `Precognition.py`.

A module of CPL called `cpl.Precognition` contains the major functions of the system. This module can be incorporated into other software systems. Version number of CPL is independent of `Precognition` and `Precognition.py`. It is shown when CPL is imported.

1.3 Major Functionalities and Features

1.3.1 `Precognition`, a pattern recognition software for Laue diffraction

- Auto-indexing of Laue diffraction pattern

- Geometric refinement of Laue pattern
- Profile fitting and integration of diffraction spots
- Deconvolution of spatial-overlapping spots

1.3.2 Epinorm, a wavelength normalization and data scaling software for Laue diffraction

- Wavelength normalization
- Data scaling
- Harmonic and extremely-close spatial overlap deconvolution
- Data reduction of partial reflections due to finite bandwidth
- Absorption correction
- Data merging

1.3.3 Features

- Soft limits and initial source spectrum can be recognized before and after indexing of a pattern.
- Indexing consistency check is available for re-indexing in middle of a set.
- Cylindrical and arbitrarily inclined flat detectors are supported in addition to normal flat detector.
- Residual pattern recognition (RPR) in geometry refinement. Residual pattern consists of all residual vectors from predicted spot locations to observed ones. This pattern varies according to the remaining errors in the parameters, and shrinks to a small Gaussian distribution when refinement is done well. Precognition automatically recognizes residual pattern using the newly developed RPR technique, makes geometry refinement more robust.
- A new refinement mode is added to solve the frequent crystal jumping problem.
- Random walk-around of crystal orientation is available during geometric refinement. Number of steps and radius can be automatically adjusted.
- Uncertainties of detector parameters can be input by user, so that freedom of these parameters are limited.
- Manual geometry refinement is possible.
- Geometric parameters can be fixed or freed during refinement.
- Calibration of distance and cell constants after geometric refinement is possible.
- Multiple integration modes are available.
- Analytical profile fitting works with a numerical compensation, which improves the accuracy of integration for strong spots.
- Negative integrated intensity becomes possible.
- Connection to the older program LaueView is no longer the default, but still possible by explicit command option.
- Initial values of scaling parameters can be set.

- Save and restore intermediate results of wavelength normalization and scaling.
- Isotropic and anisotropic scaling are available.
- Temperature factors can be fixed without refinement during scaling.
- Nonlinear λ -mapping allows more parameters to describe sharply varying and peak regions of a λ -curve, and few parameters in flat regions and two wings. It is far better than the traditional linear mapping, since sharply varying details in a λ -curve can be described by a reduced number of total parameters, and at the same time, noise spikes at two ends of the spectrum can be suppressed.
- Spike removal in λ -curve.
- A new algorithm analogous to local scaling is available after wavelength normalization and normal scaling. The locality varies from wavelength, integrated intensity, to location on detector, etc. Local scaling corrects residual errors in an empirical fashion. It improves not only data merging of singles, but also harmonic and spatial-overlap deconvolution.
- In the later cycles of scaling, the minimization target is switched to better suit Lorentzian error distribution instead of Gaussian.
- λ -curve becomes λ -surface. Wavelength normalization factor depends not only on wavelength, but also on mosaicity and Bragg angle.
- Frame-specific λ -curves are possible.
- An isotropic correction for geometric errors is introduced to wavelength normalization and scaling. This correction includes frame-specific wavelength shift and isotropic expansion correction. Errors due to geometric refinement become more tolerable.
- Crystal mosaicity is refined.
- Partial in Laue diffraction due to a finite energy bandpass, called energy partial, are reduced.
- Negative integrated intensity may contribute to scaling and be included in final results.
- Harmonic deconvolution is available.
- Crystal mosaicity, energy resolution and energy partial of harmonic reflection are completely modeled in harmonic deconvolution.
- Extremely-close spatial overlaps can be deconvoluted by the same algorithm as harmonic deconvolution.
- All effects of energy partial are completely modeled in deconvolution of extremely-close spatial overlaps.
- Friedel pairs can be separated during data scaling and/or merging if anomalous scattering should be considered.

1.4 A Tour of This Book

This manual has three parts, a user guide, reference, and some tutorials. This very first chapter collects discussions on general issues independent of each specific topic, such as conventions used throughout the book and software installation.

The user guide is split into several chapters. It was written with average readers in mind. One should learn the basics of X-ray diffraction from other sources, and have some experience of diffraction image processing. I tried to insert some boxes of background information, and will continue to do so throughout future revisions, but by no means these theoretical background are complete.

Processing of diffraction images from single crystals is always an indexing-integration-scaling dance, although Laue diffraction, monochromatic oscillation, and Weissenberg have very different styles. A harmonic dance requires precision in each step and tolerance to errors from the previous step. Chapter 2 is very important, since it sets the scene for the rest. Many basic input commands and files are common to later chapters. Given the basic inputs, the central topic in this chapter is indexing of Laue diffraction pattern. Various different scenarios are discussed, some easy cases and others troublesome.

Chapter 3 continues to deal with single crystal diffraction geometry, except this time in a refined manner. Geometric refinement of diffraction pattern aims at precise prediction of diffraction spots on the active surface of detector. This goal is achieved by correction of crystal lattice and detector parameters, but all geometric. An additional quantity predicted in case of Laue diffraction is the central wavelength by which a reflection is stimulated. A large portion of this chapter discusses minimization of geometric errors, which are crucial to the next step.

Once a precise prediction of all diffraction spots on detector surface is available. Spot integration is very much independent of diffraction geometry. In Chapter 4, an array of integration algorithms is discussed, some faster, some more accurate. All of them fall into two categories, summation and profile fitting. Laue diffraction images demand more sophisticated algorithms to deal with the commonly occurring spatial overlaps.

Chapter 5 covers scaling, wavelength normalization, absorption correction, harmonic deconvolution, and deconvolution of extremely-close spatial overlaps. These apparent complex issues are essentially data reduction and merging from integrated intensities to structure factor amplitudes, the final form of data to be used in structure determination and refinement, also known as structure factor magnitudes.

Chapter numbers 6 through 10 are reserved for future use.

The reference part is in Chapter 11. This part is meant to be the definite description of each and every program, command, argument, and file format of the software. This part is inevitably harder to read, and may contain information intended for programmers rather than users. One should check the reference for complete usage of a specific item. However, this part does not include strategies of using an item.

The rest chapters are examples that show many details of data processing. I arranged these chapters from easy to hard. A beginner would like to follow these examples to get a jump start, however, it would be a good idea to check back the relevant sections in the

user guide and reference, since better explanations might be found there. This book, all testing data, command scripts, and log files are available at <http://renzresearch.com/Precognition>.

1.5 Conventions Used

1.5.1 Notations

s, S	scalars
f, F	scalar functions
$\underline{c}, \underline{C}$	complexes; $\underline{c} = a + ib$
$c = \underline{c} $	amplitude of a complex
$\underline{c}^c, \underline{C}^c$	complex conjugates, if $\underline{c} = a + ib$, $\underline{c}^c = a - ib$
$\underline{v}, \underline{V}$	vectors; $\underline{v} = (a, b, c)$
$\overline{\underline{v}}, \overline{\underline{V}}$	vectors; $\overline{\underline{v}} = (-a, -b, -c)$ if $\underline{v} = (a, b, c)$
\mathbf{m}, \mathbf{M}	matrices
p, P	geometric points
(C, r)	circle with center C and radius r
$ OP $	distance from point O to point P
$\angle ABC$	angle
Arial	title
Times	general text
<i>Italic</i>	temporary note
Courier	code, program, command, argument, filename, path, URL, etc.

Table 1.5.1.0.1 Notations used.

1.6 Related Software

Precognition is an end-user application software written with CCL™/CPL™ (Crystallographic Concept/Protocol Libraries™), which in turn rely on many public domain software components. This section describes where to obtain these software and how to install them. This section provides a reference to these libraries and tools. General users do not have to install all of them.

Box 1.6.1.0.1 describes a common standard of GNU software installation. A specific package may require deviated procedure as noted in the subsections below. It is recommended to install all supporting software packages in the directory `/usr/local` or `/usr/local/rri/pub`. This section assumes that they are installed in `/usr/local`, but if other directory is chosen, it should replace all `/usr/local` below.

1.6.1 GCC

CCL, CPL and Precognition are compiled by GCC or GNU Compiler Collection (<http://gcc.gnu.org>). The current release has not yet been tested by other compilers.

`gcc-3.4.3.tar.gz` can be obtained from <ftp://ftp.gnu.org/gnu/gcc/gcc-3.4.3>. Please follow the standard procedure in Box 1.6.1.0.1 for installation of GCC, except that one more option to `./configure` can be used to install only relevant languages:

```
./configure [--prefix=/usr/local]
            [--enable-languages=c,c++,f77]
```

1.6.2 Python

Both CCL and CPL depend on the Python interpreter (<http://python.org>). Although CCL is written in C++, it embeds some CPython components. Python distribution `Python-2.4.tar.gz` can be downloaded from <http://python.org/2.4>. The installation is exactly same as the GNU standard in Box 1.6.1.0.1. If it is desired to replace the existing Python version on your system, use `/usr` in the `prefix` option. The documents are at <http://python.org/ftp/python/doc/2.4>.

Box 1.6.1.0.1 GNU Software Installation

It seems to be a GNU standard that a software package is always distributed in a compressed tarball, for example, `package-x.x.x.tar.gz`, where `package` is the software name and `x.x.x` is the release number. This file can be saved in a directory with its path `path`. The command

```
tar xzvf path/package-x.x.x.tar.gz
```

in a recommended directory `/usr/local` will create a new directory `/usr/local/package-x.x.x` or overwrite its contents if it already exists. A second command

```
./configure [--prefix=/usr/local]
```

in the top directory `package-x.x.x` will test the software environment and prepare for installation. The option specifies the location where to install, and can be omitted if `/usr/local` is the target. The next two steps are

```
make
make install
```

Specific software packages may deviate from this standard.

1.6.3 Numeric/Numarray

`Numeric` package provides multidimensional arrays. This package is in transition to a new generation of package `Numarray`. The latest distribution can

be found at <http://www.numpy.org>. Download Numeric-23.8.tar.gz and unpack it in a directory, say, /usr/local/Python-2.4/Extensions. If the directory Extensions does not exist, make such directory. In this top level directory Numeric-23.8, execute:

```
python setup.py install
```

Make sure that the python command is the very version Numeric is intended to be installed into. Installation to one Python version will not be available to other versions on the same system.

1.6.4 PIL

Python Imaging Library (PIL; <http://www.pythonware.com/products/pil>) adds image processing capability to Python. The latest distribution of PIL Imaging-1.1.4.tar.gz can be downloaded from <http://effbot.org/downloads>. Installation of PIL must be done after those of GCC and Python. Change your working directory to where Python is installed, e.g., /usr/local/Python-2.4. Make a directory Extensions, if there is not yet one. In the directory Extensions, unpack PIL distribution. Move into Imaging-1.1.4/libImaging, and run the following configuration and make commands:

```
./configure  
make
```

After these are done, move back to Imaging-1.1.4, and run:

```
python setup.py build  
python setup.py install
```

Before the last command of installation, make sure the python command is indeed what you intend to use. PIL installed by one python command will not be available for other Python releases on the same machine.

Note: error occurred during python setup.py build. To solve the problem, two lines were inserted into Imaging-1.1.4/_imagingft.c as suggested in freetype.h:

```
#include <ft2build.h> // insert  
#include FT_FREETYPE_H // insert  
#include <freetype/freetype.h>
```

1.6.5 Pmw

Pmw, Python Megawidgets (<http://pmw.sourceforge.net>), is used to create GUI components. Pmw.1.2.tar.gz can be obtained from Source Forge. Unpacking of this file in a directory, say, /usr/local/Python-2.4/Extensions, creates a new directory Pmw. The parent directory of Pmw, i.e., /usr/local/Python-2.4/Extensions should be added to the environment variable PYTHONPATH before using it. An alternative is to make a symbolic link of Pmw to /usr/local.

1.6.6 SWIG

SWIG, Simplified Wrapper and Interface Generator (<http://www.swig.org>), is used to wrap CCL into several modules of CPL. Several SWIG shared libraries are required at runtime, even the user does not rebuild CPL. swig-1.3.27.tar.gz can be obtained from <http://www.swig.org/download.html>. Installation of SWIG is as same as the GNU standard in Box 1.6.1.0.1.

1.6.7 FFTW

Fast Fourier Transform in the West (FFTW, <http://fftw.org>) package is used by CCL and in turn by CPL. Its shared libraries are required at runtime. fftw-3.0.1.tar.gz can be downloaded from <http://fftw.org/download.html>. Its installation procedure is as described in Box 1.6.1.0.1.

1.6.8 TNT

Template Numerical Toolkit (TNT, <http://math.nist.gov/tnt>) is a library contains only header files. It is required only when CCL and CPL are built. First, download tnt094.zip from <http://math.nist.gov/tnt/download.html>, then type command

```
unzip [path/]tnt094.zip
```

in the directory where TNT will be installed. /usr/local/include is recommended. A subdirectory tnt will be created, which contains a set of header files.

1.6.9 LAPACK and BLAS

Linear Algebra PACKage (LAPACK, <http://www.netlib.org/lapack>) and Basic Linear Algebra Subprograms (BLAS, <http://www.netlib.org/blas>) are low-level libraries used by CCL and CPL. It is usually unnecessary to install these packages, since they normally come with the operating system. But if it becomes necessary, rpms are available from <http://www.netlib.org/lapack/rpms>. First, download the following rpm files:

```
lapack-3_0-2_src.rpm
lapack-3_0-2_i386.rpm
lapack-man-3_0-2_i386.rpm
blas-3_0-2_i386.rpm
blas-man-3_0-2_i386.rpm
```

then use the command `rpm -i [--prefix=path] package.rpm` to install package in the directory path. `/usr/local` is recommended. By default, it installs to `/usr`.

1.6.10 GRACE

GRACE is a popular 2-dimantional plotting software that can be obtained from <http://plasma-gate.weizmann.ac.il/Grace>. It is used to generate graphs to show some results from Precognition. `grace-5.99.0.tar.gz` can be found at <ftp://plasma-gate.weizmann.ac.il/pub/grace/src/grace6>. Unpack and configure like this:

```
./configure --prefix=/usr/local
```

Here the option `--prefix` must be used even if the target is `/usr/local`. Then `make` and `make install`, but still have to link `/usr/local/grace/bin/xmgrace-5.99.0` to `/usr/local/bin`.

GRACE manual can be found at <http://plasma-gate.weizmann.ac.il/Grace/doc/UsersGuide.html>.

I am moving from GRACE to Gnuplot, but currently some utilities still depend on GRACE.

1.6.11 Gnuplot

Gnuplot (<http://www.gnuplot.info>) is currently optional. Future releases of our GUI programs may use `gnuplot.py` as a graphic module. Download and unpack `gnuplot-4.0.0.tar.gz`. To install `gnuplot`, follow the procedure described in Box 1.6.1.0.1. To install `gnuplot.py`, download and unpack `gnuplot-py-1.7.tar.gz` in directory `/usr/local/Python-2.4/Extensions`, move into the new directory `gnuplot-py-1.7`, and run command

```
python setup.py install
```

<http://t16web.lanl.gov/Kawano/gnuplot/index-e.html> is a great source of help.

1.7 Installation and Execution

1.7.1 Program Installation

Pre-compiled executables and shared libraries are distributed. Uncompress and unpack the distribution tar-ball `rriyyyymmdd.tgz` at a desired location, e.g., `/usr/local`. A new directory `rriyyyymmdd` will be created. Make a symbolic link of this directory to `rri`. Several environment variables must be set before appropriate execution. Assume that the software system is installed in `/usr/local/rri` or it is a symbolic link to the top directory, the following lines shall be added to the `.tcshrc` file or equivalent:

```
setenv RRILICENSE      /home/renz/license.rri
setenv RRI             /usr/local/rri
setenv CCLHOME        /usr/local/rri/ccl
setenv CPLHOME        /usr/local/rri/cpl
setenv PRECOGNITION   /usr/local/rri/gpr
setenv LD_LIBRARY_PATH /usr/local/rri/pub/lib
setenv PYTHONPATH
"${RRI}/pub:${CPLHOME}/.:.${CPLHOME}:${CPLHOME}/ptr:${CPLHOME}/std:${CPLHOME}/m
ac:${CPLHOME}/ccl"
setenv PATH            "${RRI}/bin:${RRI}/pub/bin:${PATH}"
```

Listing 1.7.1.0.1 Environment variables to be set.

For more detail of installation, see README file that comes with a distribution.

1.7.2 License Installation

In the list above, `RRILICENSE` shall point to a license file in each user's home directory. The user's account must have read and write permission on the license

file. If one prefers a centralized license file for multiple user accounts to access, the license file must be made readable and writable to all user accounts. To obtain a new license file, start a Python session after installation and setup of the environment variables, type the following commands in Python:

```
% python
Python 2.2.2 (#1, Feb 10 2003, 00:23:53)
[GCC 3.2 20020903 (Red Hat Linux 8.0 3.2-7)] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> import cpl

| | | | ) Welcome to CPL M0.12.0
| | | | \ Renz Research, Inc.

>>> cpl.license.getinfo()

SOURCE: cpl.license.getinfo
TYPE:   result
~~~~~
To obtain a license on this host, please send file info.rri to Renz Research.
~~~~~
```

Listing 1.7.2.0.1 Obtaining a host information file.

A file `info.rri` containing some information of the host is created. Send this file to us to request a license file (renz@renzresearch.com).

If you have multiple computers, you may supply a filename other than the default to the function `getinfo()`, such as `getinfo('myComputer.rri')`. Please note that the filename must be a quoted string.

1.7.3 Execution

If you can generate the information file as described above, CPL is installed appropriately. Most frequently, problems arise due to that some related software installed are inconsistent with the pre-compiled distribution. See 1.6 Related Software for details.

If you see the copyright notice and RRI logo when launching a program, installation is correct and complete. Again, frequent problems may be missing shared libraries or invalid license file. Please note that the name of a specific executable program is subject to change, when the system evolves. The following is an example of such executable program:

```
% precognition_S1.0.6
```

```
  |_)
  | recognition

  Release S1.0.6
  all rights reserved
  * * *
  John Zhong Ren
  Renz Research, Inc.
  P. O. Box 605, Westmont, IL 60559, U. S. A.
  e-mail renz@renzresearch.com
  tel (630) 230-0272
  fax (435) 514-2645
```

```
||||) Welcome to CPL M0.12.0
||||\ Renz Research, Inc.
```

```
DPR: Input   I: File & keyboard input (e.g., image & parameters)
      Output  O: Save result in files
      Quit    Q: Exit DPR (default)
DPR:
```

Listing 1.7.3.0.1 Launching Precognition.

Each program in the system may take a set of command-line options. These options will be discussed later.

A command-driven program in the system may support some global commands that can be executed at any level throughout the program. The global commands are usually used to alter the general behavior of the program, for example, to screen various types of messages.

A sequence of commands for a command-driven program may be collected into an input file and fed to the program at once. The input filename can be an argument when the program is started:

```
% precognition_S1.0.6 dpr.inp
```

where `dpr.inp` contains a script of input commands. See below for more detail. An alternative is to load the command script at an appropriate level of the command prompt, as shown in the following Listing.

```
DPR: Input   I: File & keyboard input (e.g., image & parameters)
      Output  O: Save result in files
      Quit    Q: Exit DPR (default)
DPR: @dpr.inp
```

Listing 1.7.3.0.2 Loading a command script after Precognition is launched.

If a log file is desired, the following two means are recommended:

```
% precognition_S1.0.6 dpr.inp | tee dpr.log
```

saves a copy of all printout on screen into the file `dpr.log`, and overwrites it without prompt if it already exists. All printout still show up on screen. This method is recommended for short jobs that you want to monitor its progress.

```
% precognition_S1.0.6 dpr.inp > dpr.log &
```

redirects printout on screen (via `stdout` to be exact) to the file `dpr.log`, and returns to the shell prompt immediately. No printout will show on screen anymore (except `stderr`). The file `dpr.log` will be overwritten without warning if it exists. Use `>>` if you want to append to the file instead of overwriting it. Use `>&` or `>>&` if you want to redirect `stderr` to the log file as well. The job will run in background. Use the command `jobs` to check its status. This method is recommended for long jobs or those you would not monitor.

If you can go so far up to this point, your CCL, CPL, Precognition, and their license installations are correct and functioning. You are ready to process data.

CHAPTER 2

Indexing of Laue Pattern

The very first release of Precognition was dedicated to a more robust auto-indexing algorithm of Laue diffraction patterns. Compared to `LaueView`, another Laue software written by the same author, a comprehensive test shows that over 90% of sparse images in a data set from a small molecule crystal can be independently indexed and refined with absolutely no user intervention; another 5% can be indexed and refined with some user assistance; while none of the images in the same data set was successfully indexed by myself and others using `LaueView` after days of trials, due to insufficient number of good nodal spots identified by eyes. The key improvement is automatic pattern recognition of diffraction spots, their common profile, direct-beam center, Laue ellipses, and significant nodal spots.

This chapter explains the usage of this indexing program, and leaves the underlying methodology elsewhere. This chapter is a starting point for a new user, since it describes many basic input and control parameters not only for indexing but also for the other tasks of the system.

2.1 Required Input

`Input` is a main level command to the program `Precognition` (Listing 2.1.0.0.1). It enters into a submenu that takes all input from files or from keyboard (11.1.1). In order to index a Laue pattern, the following input is required:

```
Input
  Format      MarCCD
  Crystal     /home/renz/xtal_info/bm.xtl
  Distance   40.9 0.1
  Center     1087 1368 1
  Pixel      0.0792 0.0792
  Image      /home/renz/work/bm1/images/BM1_1_100.img
  Resolution 0.7 100
  Wavelength 0.7 1.3
  Quit
```

Listing 2.1.0.0.1 Required input.

Starting from version 5, `Input` submenu is significantly changed. Command script written for previous releases may not function correctly when running version 5. `Input` submenu is a centralized location for parameter and file input for tasks other than indexing as well. The input parameters include crystal parameters or those control crystal orientation, detector parameters, and source parameters. They can also be divided into two categories: some are specific to a frame of diffraction image; others are overall to an entire dataset. Listing 2.1.0.0.2 shows `Input` menu when all items are activated and Table 2.1.0.0.1 summarizes the categories.

```

Input: Format      F: Image format          ( f [esrf])
      Goniometer  G: Goniometer omega, chi, phi (deg g 0 0 0)
      Omega       O: Omega polar orientation (deg o -90 0)
      Crystal     X: Crystal info file or cell & SG No
      Matrix      M: Missetting matrix      (by row)
      Distance    D: Crystal-detector distance (mm d 100)
      Radius      D: Cylindrical radius      (mm d 100)
      Center      C: Direct-beam            (pxl c 1000 1000)
      Center      C: normal projection      (pxl c 50 50 normal)
      Pixel       P: Pixel size             (mm p 0.1 0.1)
      Shift       S: Detector off-centering (mm s 1 0.1)
      Swing       S: Detector swing angles  (deg s 30 0)
      Tilt        T: Detector tilt angle(s) (deg t 0 0)
      Bulge       B: Detector bulge correction ( b 0 0)
      Image       I: Image or dataset files
      Spot        L: Length, width, sigma & file (pxl l 8 6 2 [f])
      Resolution  R: User defaults          (A r 3 20)
      Wavelength  W: User defaults          (A w 0.5 2)
      Chebyshev   V: Maximum degree of spectrum ( v 64)
      Anomalous   A: Anomalous scattering   ( a [on/off])
      Quit        Q: Exit this menu (default)
    
```

Listing 2.1.0.0.2 Input submenu.

Parameter/File	Dataset	Frame	Crystal	Detector		Source	Other
				Flat	Cylindrical		
Format	•			•	•		
Goniometer		•	•				
Omega		•	•				
Crystal		•	•				
Matrix		•	•				
Distance		•		•			
Radius		•			•		
Center		•		•	•		
Pixel		•		•	•		
Shift		•			•		
Swing		•		•			
Tilt		•		•	•		
Bulge		•		•			
Image		•		•	•		
Spot	•						•
Resolution	•		•				
Wavelength	•					•	
Chebyshev	•					•	
Anomalous	•		•				

Table 2.1.0.0.1 Input parameters in categories.

Among all commands to Input submenu, Format is the leading command, since it specifies image format and detector type. Irrelevant commands to the selected detector

type will be deactivated. This command usually should be given before other detector-related commands. However, flat detector is the default, which fits most needs. The program assumes flat detector even before `Format` is given. `Image` is another special command. It concludes a session of parameter input that is specific to the image loaded by this command, therefore it must be given after other frame-specific commands.

2.1.1 Crystal information

`Crystal` is a command in `Input` menu that takes 7 numerical arguments as cell constants $a, b, c, \alpha, \beta, \gamma$ in Å and degree, and an integer of space group number. If only one numerical argument is given, it is taken as space group number. Out of range values will cause error. See 11.1.1.3 for detail.

The command `Crystal` may also be followed by a crystal information file, which contains the name of the crystal, unit cell constants, and space group symbol. If no argument is given, the program will prompt a filename for crystal information (11.1.1.3). This is the crystal information file of CPMV, a virus particle crystal:

```
cpmv
317 317 317 90 90 90
spgp i23 197 24 12 cubic
```

Listing 2.1.1.0.1 Crystal information file.

The first line of the crystal information file (11.8.2) must only contain a single string without space. However, underscore (`_`) can be used, such as, `cowpea_mosaic_virus`.

There is so far no effort made by us to index a Laue pattern without knowing cell constants, however, determination of cell constants and assignment of space group may be added in later release.

2.1.2 Crystal-to-detector distance

`Distance` takes one numerical argument in mm as the crystal-to-detector distance and another optional value as its uncertainty (11.1.1.8). The default uncertainty of distance is 0, since distance is usually very insensitive to refinement. Error in distance contributes to the error in calculated angular parameters of diffraction rays, also known as reflected beams. At a Bragg angle of 22.5° , this contribution is just as large as that from the uncertainty of a spot location on detector surface. Normally, the uncertainty of distance shall be smaller than 1 mm to avoid mis-indexing (see 2.5). The uncertainty value will be used in geometry refinement (See Chapter 3).

In case of cylindrical detectors, this command evolves to `Radius`, since cylindrical radius is essentially crystal-to-detector distance. `Radius` behaves the same as `Distance`.

2.1.3 Direct-beam center

`Center` takes two numerical arguments in pixel that mark the location of direct-beam center (11.1.1.10). The first number measures from the left edge of an image to the center when viewing along the X-ray beam, and the second does from the top edge. See Figure 2.1.3.0.1 for detail. The command may also take one or two more optional values as the uncertainty of the beam center. If only one additional value is given, it is assumed that the uncertainty of beam center is isotropic. If two are given, they are the uncertainties of both coordinates, respectively. The default values are 0.2 pixel. Error in center affects the accuracy of diffraction ray at a much greater extent than error in distance does. This is the primary reason for the cause of failed or mis-indexing. `Precognition` improves the tolerance of error in center position compared to other programs, since center is automatically refined during the process of ellipse refinement. However, it is recommended to control the error to an extent as small as possible and no greater than half of the narrow dimension of a typical Laue spot on the image. For example, if a typical spot is 10-pixel long and 6-pixel wide, the tolerance of uncertainty in direct-beam center is not much greater than 3 pixels or 300 μm . However, you may recycle the refined values and rerun the program, if indexing fails. Future releases may include automatic cycling and a GUI that will accept visual assistance from user.

Direct-beam center may not be always available in case of inclined and other abnormal geometry. The command `Center` may also take a string argument `normal`, `Normal`, or `NORMAL`. This argument signals that the input center is the normal projection on the detector from the origin, where the crystal locates, instead of the direct-beam center. This option is obviously meaningless to cylindrical detector.

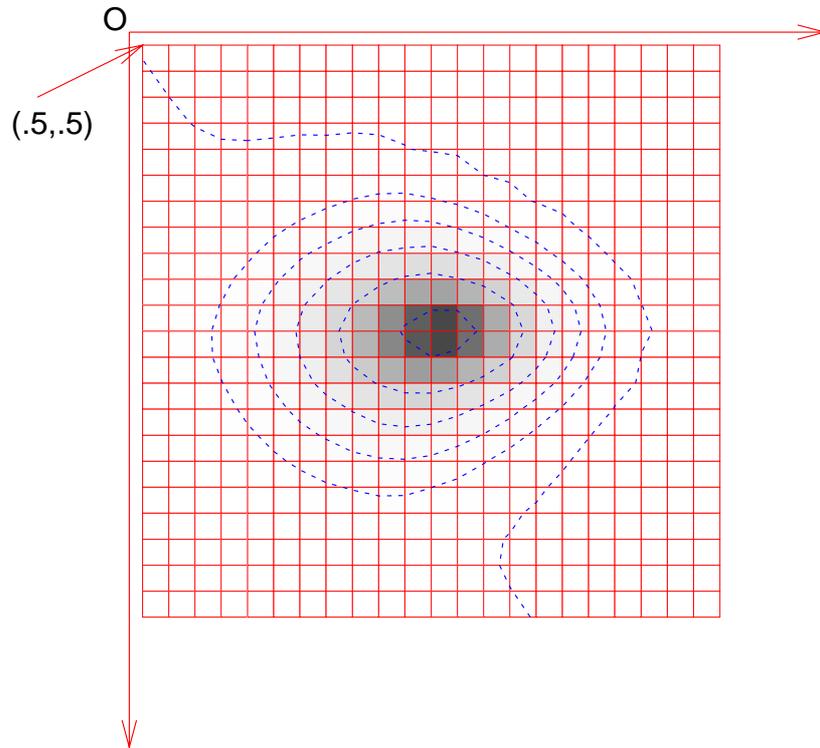


Figure 2.1.3.0.1 Coordinate system of an image file. The horizontal axis across is x-axis. The vertical axis down is y-axis. Each square is a pixel. The top leftmost corner of an image has coordinates of (0.5, 0.5). The top leftmost pixel centers at (1, 1). The origin O(0, 0) is outside of the image.

2.1.4 Pixel size

`Pixel` takes two numerical arguments in mm that measure the size of an averaged raster in horizontal and vertical direction, respectively (11.1.1.11). These numbers usually come from your detector manufacture or your own calibration experiment. Error in pixel size is directly related to error in distance. Both error sources affect the angular parameters of the diffraction rays. The command may take one or two more additional numbers as uncertainties, but they are rarely set by users. The default uncertainty for horizontal pixel size is 10 nm, and that in vertical direction is 0.

These required input do not have to be in any specific order, but they must be consistent with each other. Obviously, erroneous information in the input data will result in mis-indexing or failure to index. Minor error in the input data may prolong the process.

2.1.5 Image format

`Format` is a command at `Input` level. It is usually placed before detector-related commands. `MarCCD` is the code for supported detector format. This code is case-sensitive. The supported detector/image formats are listed in Table 11.1.1.1.1. This list will grow if users request other formats to be supported. Check the updated reference (11.1.1.1) for the latest list of format codes.

2.1.6 Image file

`Image` expects a string (do not quote) that specifies an image filename with full path or relative path (11.1.1.2). If a relative path is given, it is relative to where the program was launched. Obviously, a given image must be consistent to all other input. This command `Image` concludes a frame-specific session, and shall be placed after all frame-related commands.

2.1.7 Resolution and wavelength ranges

`Resolution` (11.1.1.16) and `Wavelength` (11.1.1.17) are followed by ranges of resolution and wavelength in Å, respectively. The order of the minimum and maximum values is arbitrary. `Wavelength` may take an additional value as a reference wavelength, but this must be in the last place. The wavelength where the spectral peak occurs is usually selected as a reference. These ranges need not to be so accurate and realistic, rather, underestimated limits work better during the stage of indexing. For example, it would be better to use only modest resolution limit, and half to 30% maximum in bandpass. Prior knowledge about how far the crystal diffracts and the X-ray spectrum will be helpful. See 2.2.2 and 3.1 for automatic estimates of soft limits.

2.1.8 Exit from Input menu

`Quit` closes this section and exits from `Input` menu (11.1.1.20). Obviously, this command must be the last.

2.2 First Indexing

If it is the very first time when you are working on a new kind of crystal, `Precognition` needs to learn something about the typical spot profile in order to estimate the scale of realistic error level associated with your images. A command `Profile` is used to just do that. This command may take a few minutes to run, however, it is not always necessary for the subsequent runs, if you are working on the same kind of crystal with the same experimental setup. See 2.4 for more.

```

diagnostic off
Input
...
Quit
Spot      re.spt
Profile
Ellipse
Pattern   pre.spt
Quit
    
```

Listing 2.2.0.0.1 Command script with profile learning.

This is an input script with a spot profile learning process prior to the ellipse and nodal recognition. The section omitted is a repetition of what we discussed in the previous section. The first line `diagnostic off` turns off diagnostic messages, which may be annoying to general users. This command is the so-called global command. It is accepted at any level.

2.2.1 Spot recognition

`Spot` is a top level command that performs spot recognition (11.1.2). It is optional to add a filename to be created or overwritten to store the recognized spots (11.8.5). The filename can be absolute or relative, if you would like to direct the file to a specific location. The recognized spots contained in the file are sorted according to their signal-to-noise ratio.

```

0  0  0  1159.24  1880.96  48678.0  2365.1
0  0  0   998.11  1098.16  85323.8  4161.8
0  0  0   863.66  1322.68  26602.8  1375.3
0  0  0  1012.06  1595.98  99479.5  6198.8
...
    
```

Listing 2.2.1.0.1 CPL reflection file that stores recognized spots.

The first three columns are h , k , l . They are all 0 since the pattern is not indexed yet. The next two columns are the location of recognized spots in pixel. The top-leftmost pixel of an image has coordinates (1, 1) when viewed along the X-ray beam (Figure 2.1.3.0.1). The last two columns are integrated intensities and associated uncertainties, respectively. This file is needed later when validating the indexing.

This command may also accept some optional numerical arguments. If one numerical argument is given, it is taken as the minimum acceptable $I/\sigma(I)$ value for the recognized spots, also known as σ -cut. The default σ -cut is 3. In case of weak diffraction pattern, one may set this value lower. If you have very strong pattern, use a larger value to reduce the number of spots. If two numerical arguments are given, they specify spot length and width of a typical spot in pixel. If three are given, they must be ordered as length, width and σ -cut. See 2.2.3 for spot profile.

A Python utility program `pick.py` wraps this command, and can be used independently. Type `pick.py -h` for its usage or see 11.5.1.

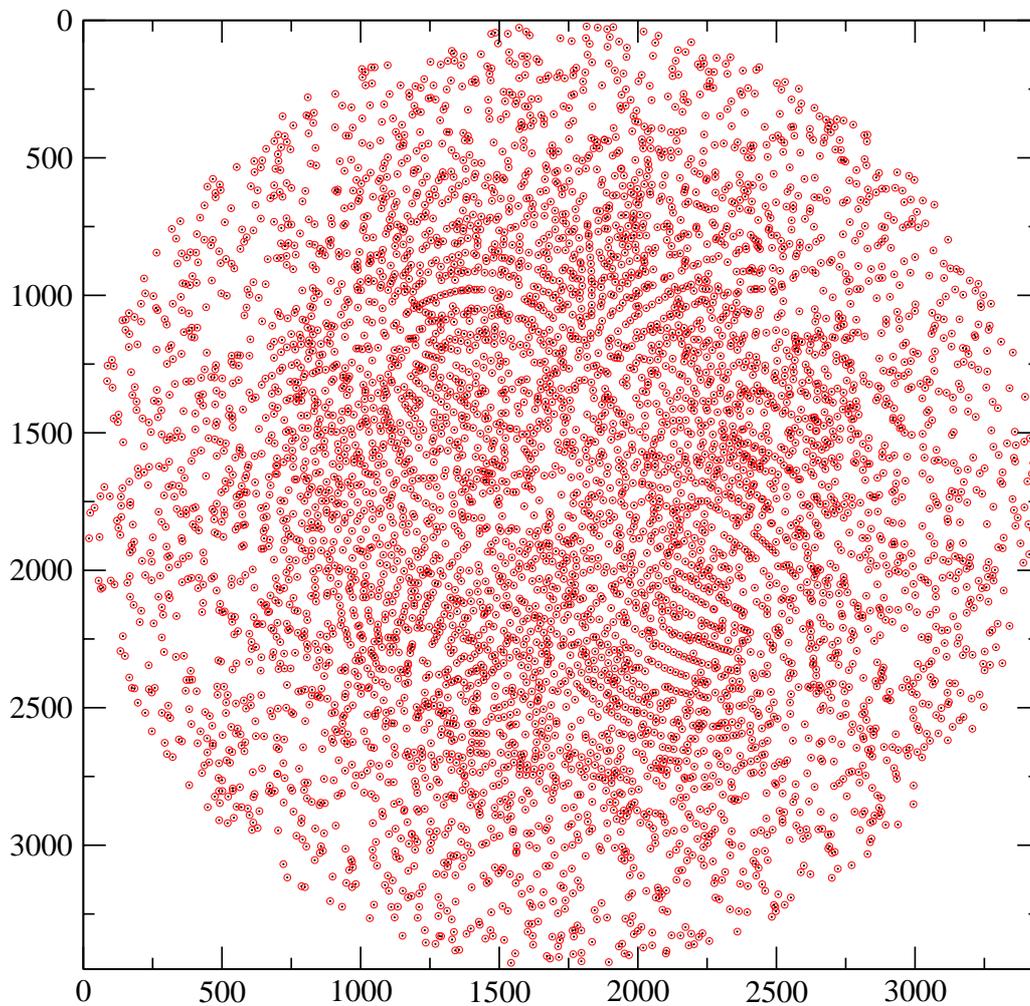


Figure 2.2.1.0.1 Spots recognized at σ -cut of 2.0. This is clearly over-picked, since the spot arrangement at peripheral does not appear to be a diffraction pattern. The real pattern is submerged in noise.

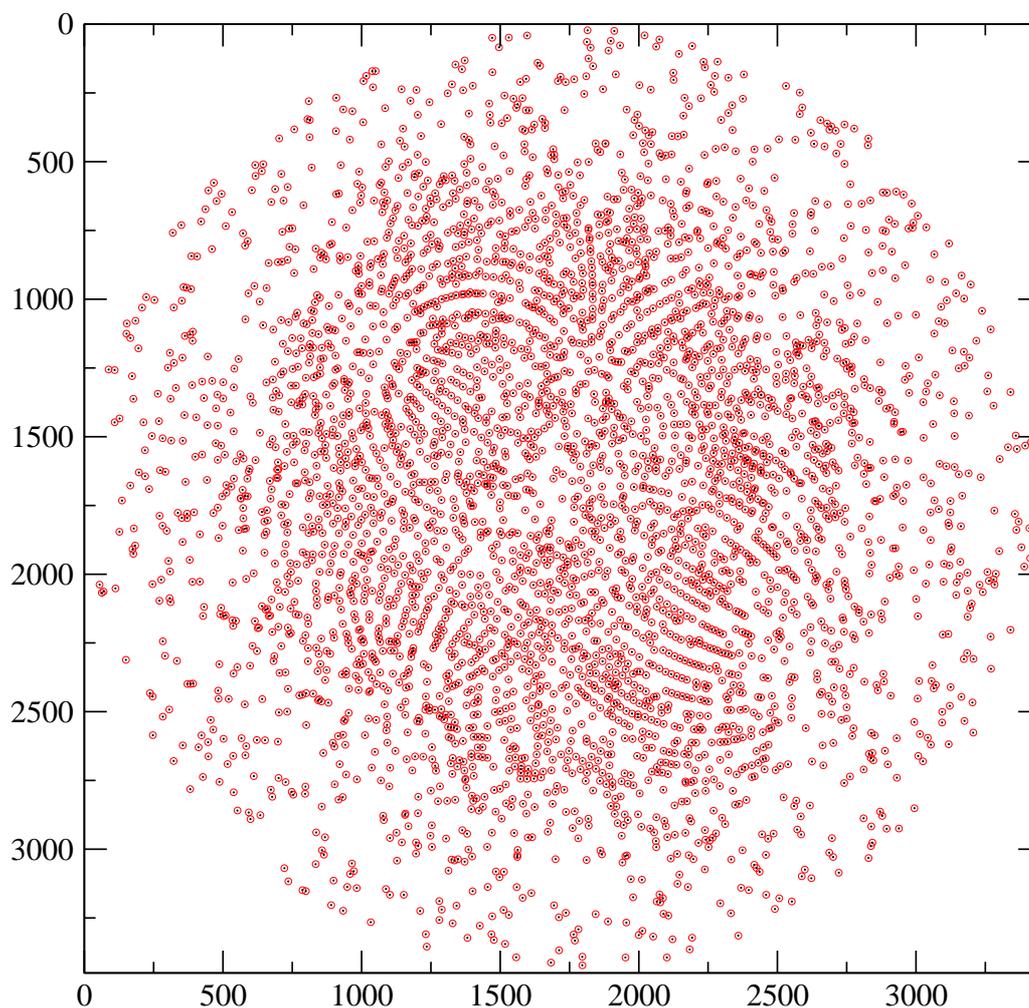


Figure 2.2.1.0.2 Spots recognized at σ -cut of 2.3. Diffraction pattern obviously stands out from noise, but this is arguably still over-picked.

It would be a good idea to use this utility to find an appropriate σ -cut value before indexing. Figure 2.2.1.0.1 through 2.2.1.0.4 are a few examples of σ -cut.

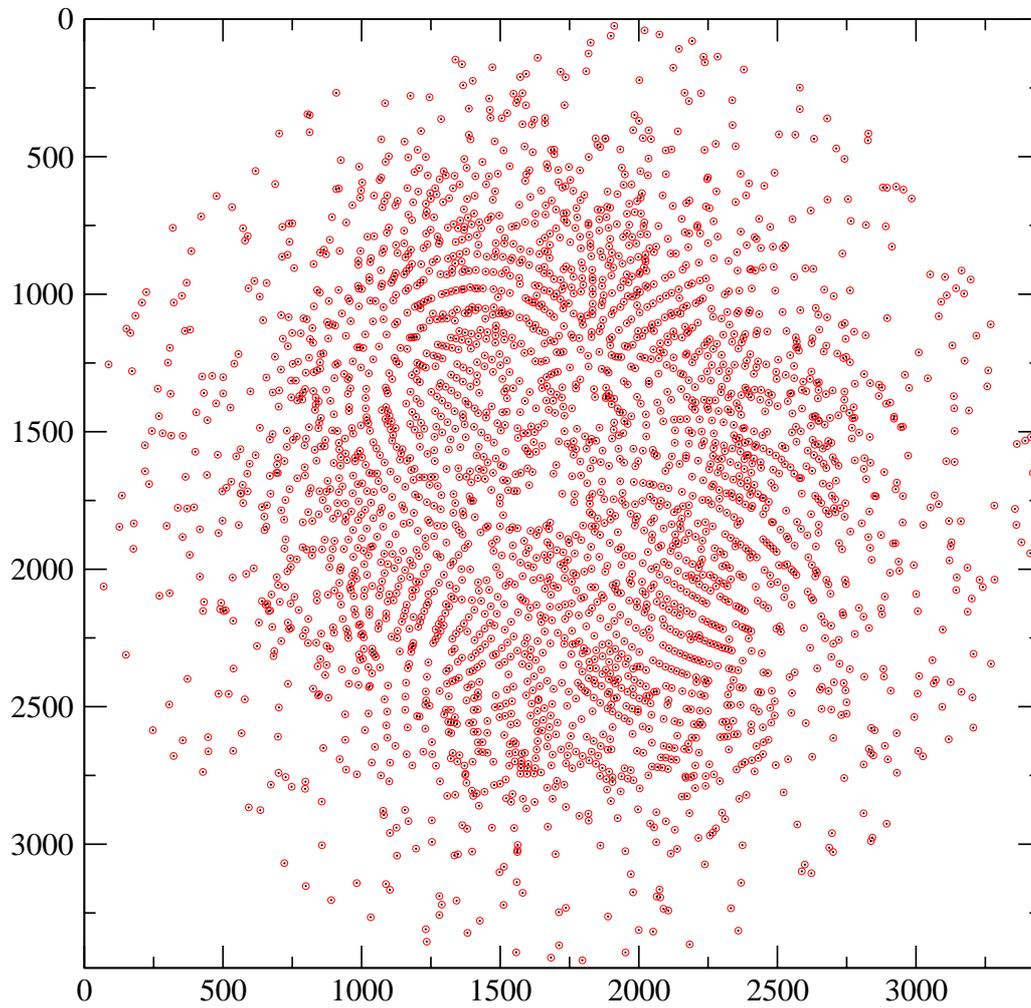


Figure 2.2.1.0.3 Spots recognized at σ -cut of 2.6. It seems to be just right, since not much noise can be removed any more without removing spots from the real pattern.

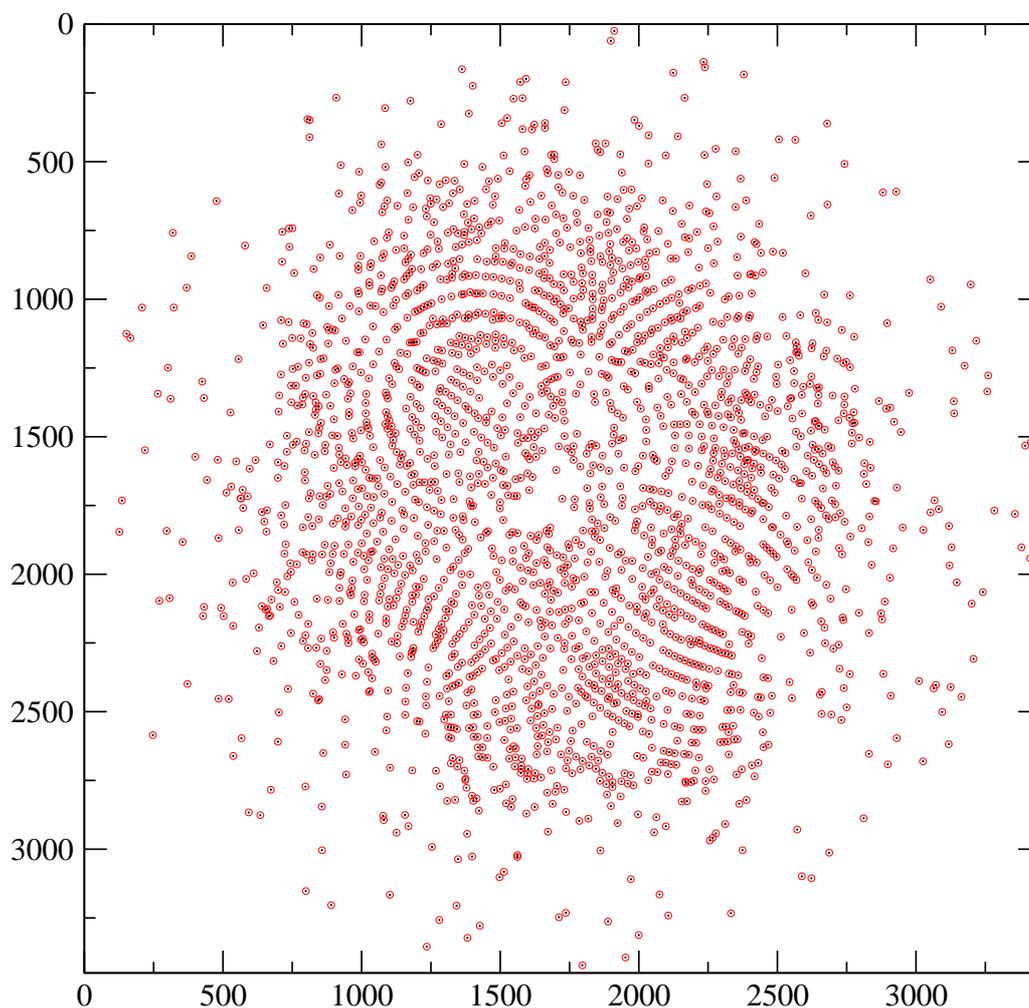


Figure 2.2.1.0.4 Spots recognized at σ -cut of 2.9. It seems to be under-picked. Compared to the last Figure, we start to lose some real spots.

2.2.2 Auto-recognition of diffraction limit

I cannot over emphasize the importance of accurate and consistent estimates of soft limits, such as crystal diffraction limit, source bandwidth, and a reliable minimum of signal-to-noise ratio, as early in the process as possible. Effective algorithm is implemented in Precognition to recognize highest diffraction resolution, maximum Bragg angle, and best σ -cut level before a pattern is even indexed and as soon as spot recognition is done. Recognition of other soft limits will have to wait until a pattern is indexed. However, this topic falls outside of the main flow of this section, first indexing. I would delay the discussion until the first part of Chapter 3. As it will be noted later, some soft limit recognition can be

carried out at this point already, and the result may better assist the task of first indexing.

2.2.3 Learning spot profile

`Profile` is a top-level command for learning the typical spot profile (11.1.3). This command helps set up the typical spot length and width. It generates a report like this:

```
An overall mean profile is recognized.
Semi-major & -minor axes (pixel): 2.73176    1.59699
Non-elliptical correction:      -0.021812  0.00429671
                                -0.0552877 -0.0120681
Non-Gaussian correction:       0.803568  0.636582
R.m.s.d. (detector count):     0.00303301
```

```
Overall spot length is set to 8 pixels.
Overall spot width is set to 5 pixels.
Estimated crystal dimension is 0.561176 mm.
Estimated mosaic spread in FWHM is 0.0606017 degree.
```

Listing 2.2.3.0.1 Report generated by command `Profile`.

The newly recognized overall spot length and width may improve the last command `Spot` for spot recognition slightly. They can also be used in later runs, if working with similar patterns, in which case the command `Profile` is no longer necessary in subsequent runs. See 2.4 for more detail.

The estimates of crystal dimension and mosaic spread is based on a model of spherical crystal and isotropic mosaicity.

2.2.4 Ellipse and nodal recognition

`Ellipse` is a top-level command for ellipse recognition (11.1.4). Ellipse recognition must be carried out after spot recognition. This command also refines the direct-beam center while the ellipses are being refined. The refined center may be used in future runs. The recognized ellipses are then used to solve nodal spots, those spots locate at the intersections of ellipses. The nodal spots are ranked by their significance.

Releases prior to version 5 permit an additional command `Nodal`. This is no longer available, and its functionality is combined into the command `Ellipse`.

2.2.5 Indexing

`Pattern` does indexing, a key step in recognition of every detail of a Laue pattern (11.1.5). If some nodal spots are automatically recognized or manually

given, this command tries to index the pattern and to refine the experimental geometry and cell parameters. This command takes some optional arguments, one of which is a filename for predicted spot locations, here `pre.spt`. This file can be used to compare with the observed spots for validation of indexing (See 2.3 for detail). This command generates a summarized report like this:

```
1 possible crystal orientation is recognized;
corresponding cell constants and detector parameters are refined.

Indexing 1
Cell lengths (Angstrom):   6.4232      7.3423      15.7870
Cell angles (degree):     95.8000     96.5907     102.5697
Missetting matrix:        -0.34351176  0.30330904  0.88882130
                          0.48259076  -0.75489410  0.44411829
                          0.80567105  0.58149680  0.11294086
Goniometer omega, chi, phi(degree): 0.0000  0.0000  0.0000
Omega-axis polar orientation (deg):  -90.0000      0.0000
Crystal-to-detector distance (mm):  41.7302
Direct-beam center (pixel):        1087.7794  1367.5079
Pixel size (mm):                0.0792287  0.0792000
Detector swing angles (degree):     0.0000      0.0000
Detector tilt angles (degree):       0.2309      0.3280
Detector bulge corrections:         -1.477e-05  1.113e-08

Missetting matrix compatible with LaueView:
-0.1414,0.3708,0.9179
0.3662,-0.8418,0.3965
0.9197,0.3922,-0.01677
```

Listing 2.2.5.0.1 Report from command `Pattern`.

Please note that the missetting matrix used in `Precognition` and `LaueView` are different. This program offers a conversion.

The refined parameters in Listing 2.2.5.0.1 will be saved into a file, if a string argument follows the command `Pattern`. This filename is the string suffixed by `.inp`, here `pre.spt.inp`. This file has the correct format of a command script for loading back into `Precognition`. None of the parameters included in this file is specific to a particular frame, although they were obtained from refinement against one frame. These parameters are specific to a fixed crystal and detector mounting during a dataset. Presumably, all of them can fit another pattern after slight refinement. This parameter file is therefore called non-frame-specific, and could be used on any other frame in the dataset later. See Chapter 3 for frame-specific parameter file. The naming convention of this file may be changed later.

```
Input
  Crystal   93.669 44.000 83.600 90.000 122.016 90.000 5
  Matrix    0.048116 -0.055959 0.997273 0.072781 0.995972 0.052375 -0.996187
0.070062 0.051995
  Omega     90.000 0.000
  Format     MarCCD
  Distance  101.988 0.250
  Center    1146.45 1143.35 1.00 1.00
```

```
Pixel      0.081717 0.081600 0.001000 0.001000
Swing      0.000 0.000 0.000 0.000
Tilt       -0.636992 -0.194799 0.100000 0.100000
Bulge      -0.000063569100 0.000000058691 0.000001000000 0.000000001000

Resolution 2.20 100.00
Wavelength 0.70 1.35
Quit
```

Listing 2.2.5.0.2 A Precognition generated input file that contains parameters that are not specific to a frame.

See the next chapter for usage of this file and the new commands `Matrix`, `Omega`, `Swing`, `Tilt`, and `Bulge` appeared in this file.

2.3 Validation of Indexing

How do I know whether it is mis-indexed? You can tell from the refinement residual, and it is immediately obvious from the observed and predicted patterns superimposed together. To plot such superimposed patterns, a utility program written in Python `su.py` is available. Type `su.py -h` for a message of usage or see 11.5.2. This utility prepares a project file for GRACE and launches it. A fully functional GRACE session will appear in a new window. The default graph will look like this in Figure 2.3.0.0.1.

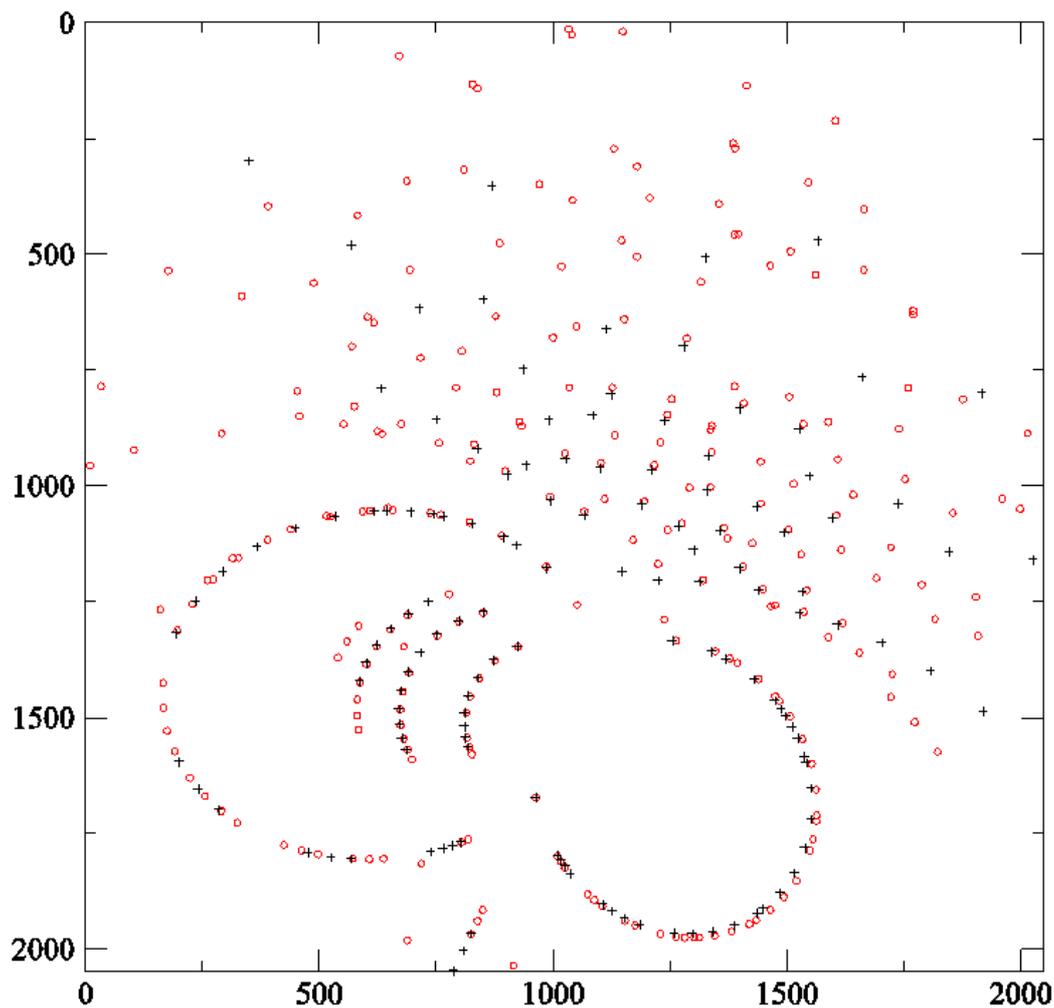


Figure 2.3.0.01. A GRACE-generated graph showing two Laue patterns superimposed together.

2.4 Routine Indexing

If you always work on a same type of crystals under a fixed experimental setup, you need to index a lots of patterns of a similar quality again and again. You must then have very good idea about the spot profile, direct-beam center, crystal-to-detector distance, etc. It would be quicker to use this script for such routine indexing:

```
diagnostic off
Input
  ...
  Spot      10 4
  Quit
Spot       re.spt
Ellipse
Pattern    pre.spt
Quit
```

Listing 2.4.0.0.1 Command script for routine indexing.

The only difference here is to set a typical spot length and width instead of running the command `Profile`. You may use the suggested values from a previous run or those yield success routinely.

Here, a new command `Spot` at `Input` level (11.1.1.7) may accept up to three numerical arguments. The rule is as same as that for `Spot` command at the main level, but this `Spot` command at `Input` level does not perform spot recognition. Once again, if one numerical argument is given, it is taken as the minimum acceptable $I/\sigma(I)$ value for the recognized spots for future use. If two numerical arguments are given, they specify spot length and width of a typical spot in pixel. If three are given, they must be ordered as length, width and σ -cut.

2.5 Mis-indexing

Everything designed and built into this program is to guard from mis-indexing, however this will still be the most frequent topic as it has always been. Most of the time, mis-indexing is obvious from superimposed patterns generated by the utility `su.py`, however, there are cases that mis-indexing is less obvious (Figure 2.5.0.0.1). A harder look will save you from going too far down a wrong track.

If your pattern is mis-indexed, the first thing you should check is whether all input information are correct and consistent with each other, cell constants, distance, pixel size, and most importantly, direct-beam center, and whether you are indexing the image file you really intend. If one has explored all options of the program, the pattern is still mis-indexed, a common mistake is often that wrong or inconsistent information are given. After all, if you do think it is the program's fault, this section explains what other options you have.

The process of Laue indexing is essentially a pattern matching game of diffraction rays in an angular space. All error sources will influence the outcome of the matching. Accidental mismatch does happen. The strategy to identify mis-indexing in this program is to rank all successful matching and to hope the top or near top matching is indeed the correct indexing. Mis-indexing at the final outcome of the program means that no successful matching near the top of the ranking list is a correct one. This can virtually never happen, if all input are error free. There must be a correct matching near the top of

the ranked list, if it is not the very top one. In order to avoid mis-indexing, we must first minimize the errors, and second alter the length of the ranked list.

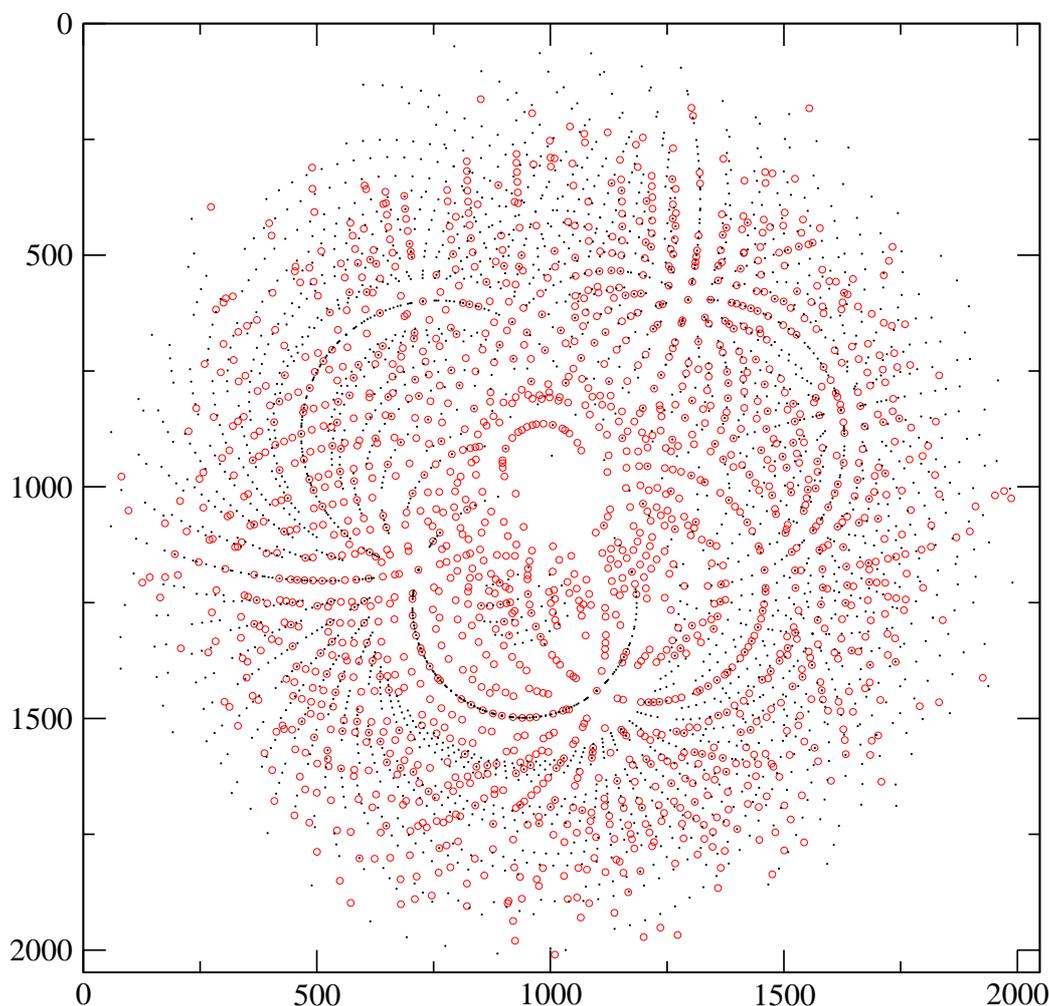


Figure 2.5.0.0.1 A mis-indexed pattern seems to fit to some extent.

2.5.1 Check all possible matching

A single numerical argument 0 added to the command `Pattern` will change the program's behavior: `Pattern 0 pre.spt`

This command will run much longer to search for all possible crystal orientations and the top 10 of them will be saved for your inspection. The very top solution is saved in `pre.spt`. The other 9 will be `1_pre.spt`, `2_pre.spt`, ...

If this numerical argument is positive, the program will keep searching until this many matching solutions are found. If this argument is 0 or negative, all possible

solutions will be examined before the program exits. This option may cause very long running time. Success rate will be dramatically increased if one allows to search the first a few solutions instead of the very first one. I suggest to use a small number, say 3 or 4 before trying larger ones, 10 or 20 for this option.

Figure 2.5.1.0.1 shows an example of mis-indexing in an initial trial, and the argument 0 worked successfully. This pattern happens to display two major ellipses nearly tangential to each other, that is, the diagonal of two major axes falls on the X-ray beam. Patterns of this kind are usually harder to index, and shall be avoided if possible. Compare to Figure 2.3.0.0.1 for difference.

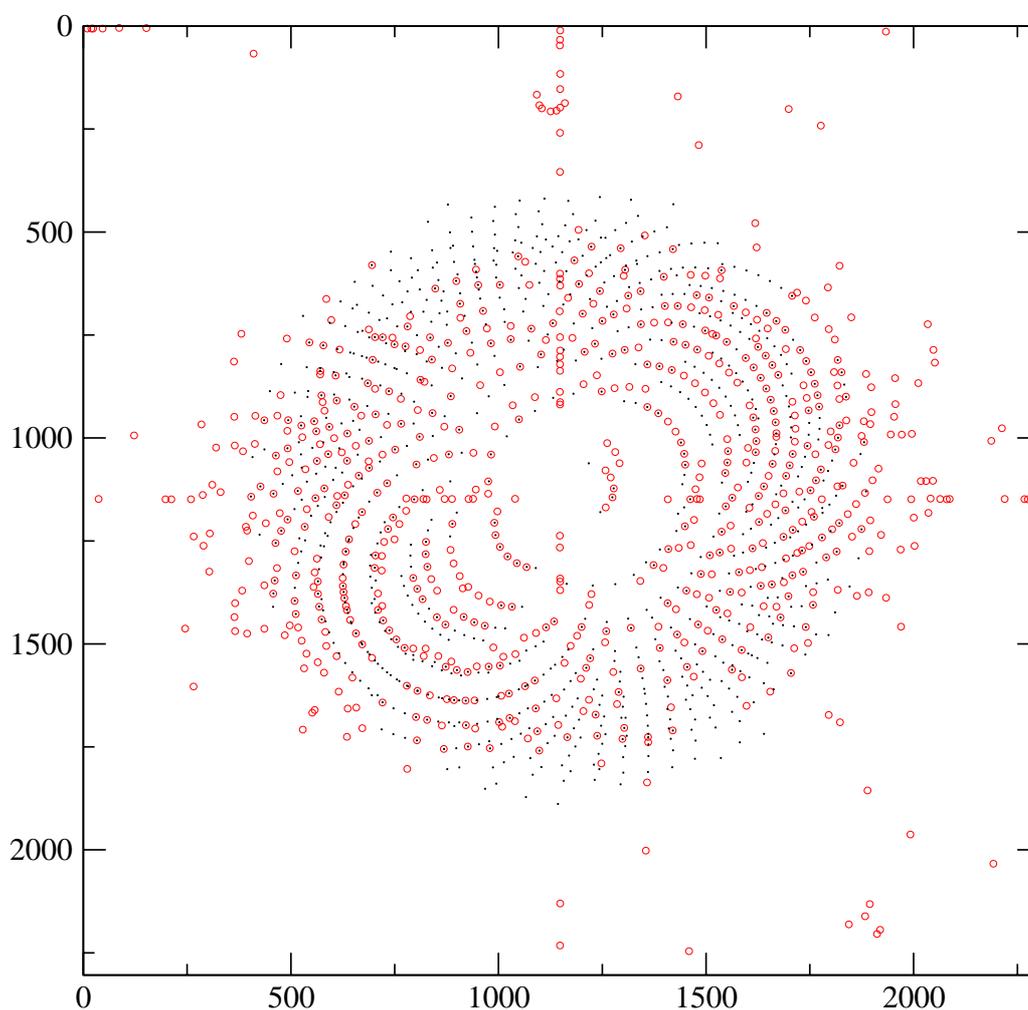


Figure 2.5.1.0.1 A Laue pattern mis-indexed with the default value of the numerical argument to `Pattern`, but correctly indexed with command `Pattern 0`.

2.5.2 Use more nodal spots

If none of the possible solutions identified are correct indexing, another option you may want to play with is the second numerical argument ranging from 8 to 16. 8 is the default, so that there is no need to repeat if you have run the default. This argument specifies the number of top quality nodal spots used in the indexing process. The greater this value is, the longer the program will run. If this second argument is greater than 10, it is recommended to combine with 1 or a small first argument. Otherwise, the program may run intolerably long.

2.5.3 Manual indexing

All above options insist on no user assistance on nodal selection. In fact, you may win the game much more easily if you are willing to help a little bit with your visual power. If you can personally inspect a pattern for a minute, pick a few major nodal spots, put them in a spot file (2.2.1 and 11.8.5), or if you can screen the program-generated spot file, select some major nodal spots, the program will have much better chance to get the right indexing from your hand-edited spot file. You can change the script slightly by specifying a spot file in `Input` section, and remove the command `Ellipse` for ellipse and nodal recognition. If the command `Ellipse` is used together with a nodal file, it performs ellipse recognition only, and the input nodal spots will be used in indexing.

```
diagnostic off
Input
...
  Spot    nodal.spt
  Quit
Spot 10 4 re.spt
Ellipse
Pattern  pre.spt
Quit
```

Listing 2.5.3.0.1 Command script to take user supplied nodal spots.

Although it is possible without the GUI version of Precognition, the task of manual picking of nodal spots becomes so much easier when using the graphical interface `Precognition.py`. If you have the GUI, launch it by running `Precognition.py`. Choose an image type from `File` menu, browse and select an image file. The image will be displayed in a graphic window. From `Tool` menu of this graphic window, choose `Spot` and then `Recognition`. Adjust `Spot Size` in the newly appeared dialog window, and use left mouse button to pick desired nodal spots in the graphic window. Each selected spot will be marked by a symbol. Click on a selected spot will delete this spot. Click `OK` button when done. From `File` menu, choose `Save` and then `Spots`. Browse and select a filename for saving the selected nodal spots. Please refer to 11.4 for more details on the graphical user interface.

2.5.4 Other options

If your patterns are from a small molecule crystal or collected using a narrow bandpass source, they become so sparse that few ellipses are sensitive to human vision. You will be surprised that Precognition does far better job than your eyes.

If you cannot index one pattern, can you choose some other patterns? This usually solves the difficulty easily, if your crystal happened to be mounted at an odd orientation. Nevertheless, there are still something else one can try after all these options failed. For example, were the data collected in room temperature, but you never know the RT cell of your crystal, and you simply assumed it is as same as the low temperature cell? This happens quite often these days. Are you sure that the detector is indeed normal to the X-ray beam as people always assume so.

Finally, call me, if you give up. I do want to learn from your case and to make Precognition more powerful.

2.6 Quick Indexing

If some nodal spots are known, either from a previous run or from hand picking, and an accurate direct-beam center is available, the following script carries out a quick indexing without searching nodal spots and refinement of center. The program will load nodal spots from the given file `nodal.spt`.

```
diagnostic off
Input
...
  Center  1087 1368
  Spot    nodal.spt
  Quit
Spot     7 4 re.spt
Pattern  pre.spt
Quit
```

Listing 2.6.0.0.1 Command script for quick indexing.

2.7 Goniometer Setting

It is not an issue, if you only work on a single image. If you have a set of images to process, their relative geometric relationship must be known. You need somehow to tell Precognition how these images were spatially arranged respect to each other. Goniometer setting is therefore needed. We delay this topic until the next chapter, when multiple images are concerned.

CHAPTER 3

Geometric Refinement

Once the orientation of crystal lattice is found by indexing, a set of Miller indices can be assigned to each diffraction spot, so that resolution and wavelength of each spot become known subsequently. Cell parameters of the crystal and other parameters of the experimental setup can be put together in order to negotiate the responsibility of apparent errors. This process is called geometric refinement since all parameters to be adjusted are geometric. The purpose of this process is to model the observed diffraction pattern as precise as possible, since it is crucial to the next process, integration of each spot. The assignment and refinement of these geometric parameters are not only local to a single image, but also global to a set of them with known relative geometric relationship.

Before we can discuss geometric refinement without interruption, two other issues, soft limits and goniometer setting, must be dealt with first in Sections 3.1 and 3.2, respectively.

3.1 Estimates of Soft Limits and Source Spectrum

Knowing of source spectrum and diffraction limit of a pattern makes the rest of the refinement process much better guided. It is also likely that estimates from one pattern in a set can be transferred to others in the same set. Some soft limits can be recognized before a pattern is indexed and as soon as spot recognition is done. Others can be analyzed after a pattern is indexed and refined successfully.

Strictly speaking, the topics of soft limits exceed the scope of Chapters 2 and 3, but it is so important to get these right as early as possible. I find it necessary to discuss soft limits before we have gone too far into a whole set of diffraction patterns.

3.1.1 Before indexing

Before indexing of a pattern, it is possible to estimate the best σ -cut level, the maximum Bragg angle, and therefore the highest diffraction limit by knowing of a reference wavelength. A reference wavelength should be selected as the most popular wavelength in the spectrum of the incident X-rays. The following script uses a new command `Limits` to perform this task (11.1.6).

```
diagnostic    off
Input
  Format      Mar345
  Distance    180
  Center      1704 1711
  Pixel       0.1 0.1
  Image       129w_25a_001.mar3450
  Wavelength 0.7 1.5 1.1
```

```
Quit
Spot      12 6 2.3
Limits
Quit
```

Listing 3.1.1.0.1 Command script that estimates soft limits before indexing.

Notice that this task does not require any crystal information. Only detector information are necessary. Wavelength range is required, but this procedure can tolerate large errors in these value. In fact, λ_{\min} and λ_{\max} are only place holders here. Only λ_{ref} is the functional value. Error in λ_{ref} can be corrected after next session of soft limit recognition. See below for more. Parameters controlling spot recognition are discussed in 2.2.1 and 2.2.3. Slightly over-picked spots are recommended such as those shown in Figure 2.2.1.0.2.

```
|_____)
| Report |
| -----|
| -----|
| -----|
| -----|
|_____)
```

```
Best sigma cut estimated at 2.42.
Sigma cut results in 10% noise is 2.32.
```

```
Maximum spot density
16.1/mrad at Bragg angle of
12.6 degree.
```

```
Diffraction limit estimated at Bragg angle of
19.5 degree or
1.65 A resolution.
```

```
Suggested resolution for indexing and geometry refinement is 2.08 A.
```

Listing 3.1.1.0.2 Results from the command `Limits` before indexing.

Results of this procedure are listed above. The σ -cut that would result in 10% noise is recommended for the purpose of this procedure. The best σ -cut is for indexing and geometry refinement. If the input σ -cut is much higher than the result, a warning message will suggest lowering of the input and retry. If the input is much lower than the best value, all results may appear slightly more optimistic. You may choose to rerun the procedure using the recommended σ -cut.

Running this procedure before indexing (2.2.2) may help you select a better σ -cut and resolution limit. As suggested, indexing and geometry refinement require an underestimated resolution range. It should also be noted that σ -cut ought to be lowered from the suggested value for smaller unit cell cases, and elevated somewhat for larger unit cells. Especially for ellipse recognition on large unit cell

patterns, it is only necessary to find far fewer spots than a normal σ -cut would find. Ellipse recognition would be prolonged significantly under the suggested, normal σ -cut.

Any report on λ -curve before indexing is arbitrary. Please discard the message and see below.

3.1.2 After indexing

If the command `Limits` is placed after indexing, that is, the command `Pattern` in Listing 2.2.0.0.1, the command `Limits` performs differently. Alternatively, a frame-specific parameter file can be loaded before the command `Limits`. A rough λ -curve can be estimated quickly when an indexed and refined pattern is available. Better reference wavelength can be read from the curve.

```
Input
...
Wavelength 0.7 1.3 1.1
Quit
...
Pattern      pre.spt
Limits      1.5 estimate.lam
...
```

Listing 3.1.2.0.1 Command script that estimates soft limits after indexing command `Pattern`.

This command script is as same as the one for indexing listed in Listing 2.2.0.0.1 except with the additional command `Limits` following `Pattern`. This command `Limits` will not work fully if the pattern is mis-indexed. When it is followed by a string argument as a filename, a rough λ -curve can be saved into this file from 0.25 to 2 Å. This λ -curve is a Chebyshev approximation of some automatically chosen degree. Users have no control of the degree at this point. This simple ASCII file has `.lam` format as described later in 4.1 and 11.8.4, essentially with wavelength in Å in the first column and relative intensity in the second column. The command `Limits` can also accept one or two numerical arguments, but these numerical arguments must work with a string argument. If one number is given, it specifies λ_{\max} of the saved wavelength range. If two numbers are given, the first specifies λ_{\max} , and the second will overwrite d_{\min} found by the command `Limits`. This option is used when the software limit d_{\min} is unreliable.

```
diagnostic  off
@ 129w_25a_001.mar3450.inp
Input
Format      Mar345
Image       129w_25a_001.mar3450
Wavelength  0.7 1.5 1.1
Quit
```

```
Spot      12 6 2.3
Limits    1.5 estimate.lam
Quit
```

Listing 3.1.2.0.2 Command script that estimates soft limits after loading a frame-specific parameter file.

Another way, perhaps more commonly, of checking soft limits is to load a parameter file generated by a previous indexing (Listing 3.1.2.0.2) rather than performing indexing on-the-fly. The Image command in Input section is necessary, since the parameter file does not really load the image file, while the image must be loaded for spot recognition. If a non-frame-specific parameter file is loaded, please also specify goniometer setting and image format before loading an image.

The resulted λ -curve in Figure 3.1.2.0.1 is already quite good except that it seems to include an incorrect baseline. It takes seconds to obtain the curve from data on a single frame. This curve may serve as a very good starting point in later wavelength normalization. This procedure also suggests a reference wavelength.

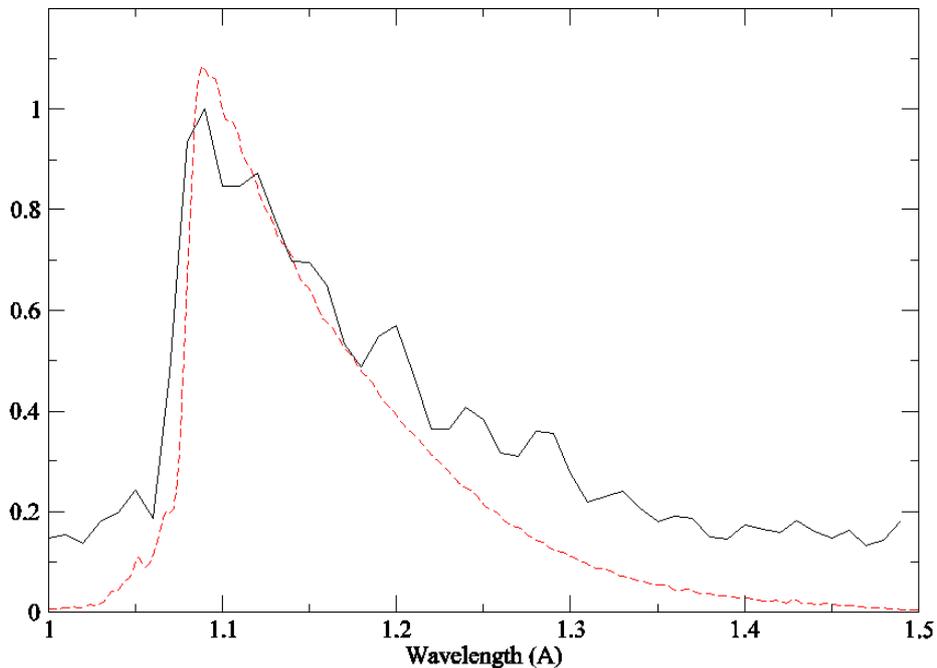


Figure 3.1.2.0.1 An estimated λ -curve in black solid line and final λ -curve from wavelength normalization in red dash line.

This procedure could fail if unit cell is too small or resolution is too low, since it requires a certain number of observed reflections to estimate a λ -curve. The procedure may not perform as well if $\lambda_{\min} < \lambda_{\max}/2$ due to the interference of energy overlap.

3.2 Goniometer Setting

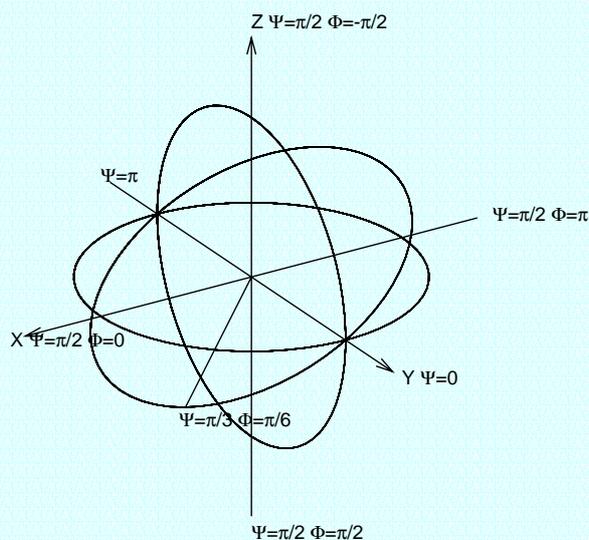
Before we can process a set of images, we need a means to specify their spatial arrangement. A virtual goniometer is simulated in Precognition in order to achieve this. More precisely, such simulation is carried out by a class of CCL, `ccl::goniometer`. Please note that the simulated goniometer may differ from the one you use in your lab or synchrotron, but the virtual goniometer must be universal enough to simulate all possible motion of a real goniometer by changing its setting.

Precognition chooses its coordinate system as the following: X- and Y-axes are horizontal to the right and vertical down, respectively. They are the same as those in an image file and a computer display. Z-axis is along the X-ray beam, if detector surface is normal to the X-ray beam (Figure 3.2.0.0.1 and 3.2.0.0.2).

The virtual goniometer has three circles ω , χ , and ϕ . The orientation of ω -axis is given by polar angles Ψ and Φ . Note that the polar angle Φ differs from the goniometer circle ϕ , and they shall not replace each other. See Box 3.2.0.0.1 and CCL/CPL reference for definition of polar angles.

Box 3.2.0.0.1 3D Orientation Defined by Polar Angles

There are several ways to define an orientation in 3-dimensional space, for example, orientation cosines, but polar angles Ψ and Φ are often used in crystallography to define an orientation. The angle Ψ can be thought equivalent to the geographical latitude, but starting from one pole as 0, and ending at the other pole as π . The angle Φ is equivalent to the longitude ranging from 0 to 2π . There are different conventions to match the latitude-longitude system with the orthogonal axes. This is the origin of the most frequent errors. Caution must be taken. MAC/CCL, the libraries that support Precognition, sets Y-axis at one pole where $\Psi = 0$, and X-axis at $\Phi = 0$. Z-axis may then be determined by the right-hand rule of a coordinates system. The positive direction of Φ is defined by the right-hand rule about the Y-axis.



ω -axis is therefore along Y-axis, vertical down, when both polar angles are 0. χ -circle is riding on ω -circle, χ -axis is always normal to ω -axis. χ -axis is along Z-axis, the X-ray beam, when ω -circle is at its home position, $\omega = 0$. χ -axis turns in a horizontal plane when ω is other than 0. ϕ -circle is riding on χ -circle, ϕ -axis is always normal to χ -axis. ϕ -circle is coaxial with ω -circle, when χ -circle is at its home position, $\chi = 0$ (Figure 3.2.0.0.1).

If $\Psi = \pi/2$, and $\Phi = 0$, ω -axis is along X-axis. χ -axis turns in a vertical plane parallel to X-ray beam (Figure 3.2.0.0.2). Under this definition, the goniometers of BioCARS stations at the APS 14-ID-B, 14-BM-C, and 14-BM-D have ω -circle set at $\Psi = \pi/2$, and $\Phi = 0$. If the κ -block is set at 0, both ω or ϕ can be used to specify crystal orientation. If the κ -block is not 0, use ω -circle only.

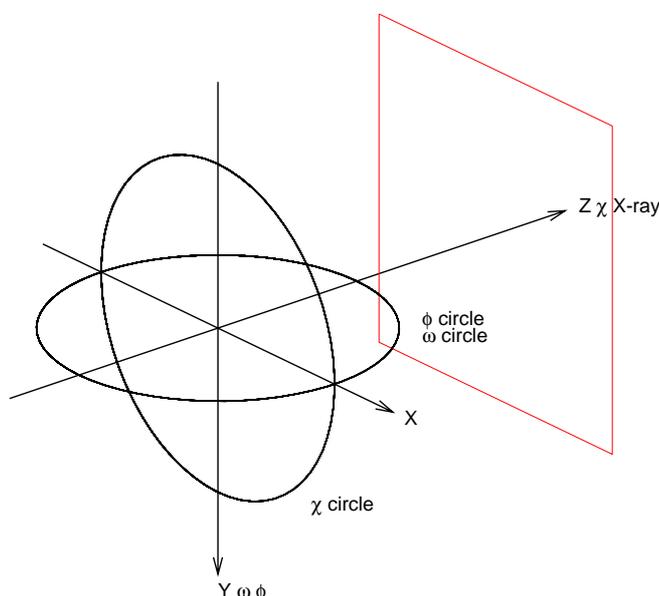


Figure 3.2.0.0.1 Three circles of the virtual goniometer at their home position. The orientation of ω -axis is $\Psi = \Phi = 0$.

Two commands `Omega` (11.1.1.4) and `Goniometer` (11.1.1.5) at Input level are available for input of these angular values. `Omega` takes two numerical arguments Ψ and Φ in degree. The default values are 90 and 0. `Goniometer` takes one to three numerical arguments ω , χ , and ϕ in degree. The latter angles are assumed to be 0, if they are missing.

With these two commands added to Input section, the command script for indexing looks like this:

```

diagnostic off
Input
...
Omega      90 0
Goniometer 3 0 0
...
Quit
Spot       re.spt
Pattern    pre.spt
Quit
    
```

Listing 3.2.0.0.1 Indexing of the first pattern with goniometer setting.

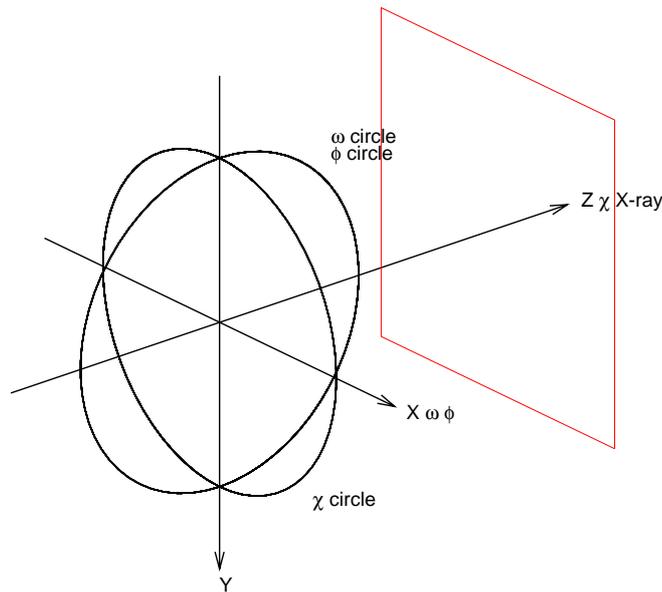


Figure 3.2.0.0.2 ω -axis of the virtual goniometer is set at $\Psi = \pi/2$ and $\Phi = 0$, that is, the X-axis. This is the setting of the goniometers at BioCARS beamlines.

3.3 Indexing and Refinement of a Set of Patterns

Once the first or any pattern in a set is indexed and refined with the correct goniometer setting, you are ready to refine all patterns in the set. Repeated use of a command `Goniometer` can provide a list of goniometer settings in a dataset. The command `Goniometer` accepts three numerical arguments as ω , χ , and ϕ angles in degree, and a string argument as the filenames of the images. Please note that the string argument should always be the last in the command. It is not possible to change orientation of ω -axis from image to image, it can only be set by the command `Omega`. All images to be processed in one session are of a same detector type. If they are different, they may still be in a dataset, but it is necessary to separate them in different sessions to process.

```
diagnostic      off
@ pre.spt.inp
Input
  Format          MAR3450Packed
  Goniometer 0 0 0 m37v3a_8us_002.mar3450
  Goniometer 0 0 3 m37v3a_8us_004.mar3450
# Goniometer 0 0 6 m37v3a_8us_006.mar3450
...
  Goniometer 0 0 180 m37v3a_8us_062.mar3450
  Resolution 2.1 100
  Wavelength 1 1.4
  Spot          20 6 0.7
  Quit
Dataset
  In             /mnt/jaz/m37v3_8us
  Out            first_try/
  Quit
Quit
```

Listing 3.3.0.0.1 A command script that refines a dataset.

The command script above performs geometric refinement of a set defined by a list of commands of `Goniometer`. This script first loads an input file generated by previous run of indexing. This input file can be any non-frame-specific one, like the one listed in Listing 2.2.6.0.2. That is to say, crystal and detector parameters from one frame can be transferred to another. The `Input` section specifies image type, loads goniometer settings, limits resolution and wavelength ranges, and gives spot size and σ -cut.

If a frame-specific parameter file is given, this frame is added to the frame list given by the repeated `Goniometer`.

A new top level command `Dataset` enters a submenu of several optional choices (11.1.7). The program starts to process all images in the `.gon` file by exiting from the submenu. The subcommand `In` takes a string argument as a directory where image files are located. `Out` specifies a directory where results can be saved. This directory must exist already. It is not going to be created by the program. Obviously, `Quit` exits from the submenu. If no `In` nor `Out` commands are given, the current directory is the default. In this case, the command `Quit` is still needed.

Previous releases require four subcommands following `Dataset`: `Path-OK-Path-OK`. The first `Path` specifies a directory where results can be saved. The second gives where image files are located. This release has backward compatibility, but it is recommended to stop using the old convention.

This process generates a set of `.inp` files similar to, but not exactly the same as that listed in Listing 2.2.6.0.2. The newly-generated input file includes some additional parameters that are specific to a frame, such as the goniometer setting and filename. Therefore, this type is called frame-specific. The filename of this new parameter file is chosen automatically as the image filename appended by `.inp`.

```

Input
  Crystal   91.870 91.870 45.983 90.000 90.000 120.000 168
  Matrix    0.615638 -0.057006 -0.785965 -0.758598 -0.312911 -0.571505 -
0.213356 0.948069 -0.235885
  Omega     90.000 0.000
  Goniometer 0.000 0.000 0.000

  Format     MAR3450Packed
  Distance   180.053 0.250
  Center     1704.59 1711.25 1.00 1.00
  Pixel      0.099995 0.100000 0.001000 0.000000
  Swing      0.000 0.000 0.000 0.000
  Tilt       0.103947 0.026568 0.100000 0.100000
  Bulge      0.000000598502 0.000000000226 0.000001000000 0.000000001000

  Image 0    ./129w_25a_001.mar3450
  Resolution 2.10 100.00
  Wavelength 1.00 1.30
  Quit

```

Listing 3.3.0.0.2 A frame-specific parameter file.

To shorten the script file, it is recommended to gather the long list of `Goniometer` commands in a separate file, and to name it by the dataset name or pass number. This file can be loaded into the other.

```

# m37v3a_8us pass 1
  prompt on
  result on
  Goniometer 0 0 0 m37v3a_8us_002.mar3450
  Goniometer 0 0 3 m37v3a_8us_004.mar3450
# Goniometer 0 0 6 m37v3a_8us_006.mar3450
...
  Goniometer 0 0 180 m37v3a_8us_062.mar3450
  prompt on
  result on

```

Listing 3.3.0.0.3 Command script `m37v3a_8us_pass1.inp` contains only the goniometer settings.

```

diagnostic off
@ pre.spt.inp
Input
  Format     MAR3450Packed
  @ m37v3a_8us_pass1.inp
  Resolution 2.1 100
  Wavelength 1 1.4
  Spot       20 6 0.7
  Quit
Dataset
  In         /mnt/jaz/m37v3_8us
  Out        first_try/
  Quit
Quit

```

Listing 3.3.0.0.4 A shortened command script that refines a dataset.

An alternative way of listing all goniometer settings is to use a small goniometer file (11.8.3). This method was the standard for the early releases of the software, and is being deprecated. The first string of each record is the filename. One to three numbers follow. They are ω , χ , and ϕ angles in degree, respectively. The missing values are assumed to be 0. All images listed in a goniometer file must be from a same type of detector. Goniometer file is loaded by the same command `Image` as an image file is loaded, thus it must use `.gon` as its file extension in order to distinguish from a diffraction image. It is not possible to change orientation of ω -axis from image to image, it can only be set by the command `Omega`. Once again, if the `.inp` file at the beginning is a frame-specific one, this frame is included regardless whether it is in the goniometer file.

```
...
m37v3b_8us_056.mar3450 165
m37v3b_8us_058.mar3450 171 0
m37v3b_8us_060.mar3450 177 0 0
...
```

Listing 3.3.0.0.5 Goniometer settings in a file.

```
diagnostic    off
@ pre.spt.inp
Input
  Format       MAR3450Packed
  Image       m37v3a_8us.gon
  Resolution  2.1 100
  Wavelength  1 1.4
  Spot        20 6 0.7
  Quit
Dataset
  In          /mnt/jaz/m37v3_8us
  Out         first_try/
  Quit
Quit
```

Listing 3.3.0.0.6 Command script refines a dataset.

3.4 Crystal Slipping

When working with protein crystals, especially when working on time-resolved projects, a number of matters are often bothersome. A protein crystal may quickly turn mosaic in white beam. Its diffraction may decay significantly after a few exposures. Laser shots for pumping the reaction in crystal may also contribute to these phenomenon. One of the worst is crystal slippage in a room temperature capillary mounting. For whatever reason, excessive mother liquid around the crystal and/or high power laser illumination causing temperature jump or sound wave, it is quite common that the recorded Laue patterns do not always fit predicted ones very well. They may be off by degrees. Losing orientation in middle of a run may happen, if your crystal experiences these problems. One option to the command `Dataset` helps greatly in this situation: `progressive`.

By default, the command `Dataset` will run under `normal` mode. The initial geometric parameters will be taken from the first pattern, that is, the user supplied `.inp` file. `progressive` mode forces the initial parameters be inherited from the last refined pattern, instead of the first. Therefore, the crystal orientation may slowly drift away from the original one. If your dataset requires using of `progressive` mode, the user supplied parameter file at the beginning of the script should be from the first frame, or the one that is chronologically close to the first in the set.

In addition, if the program is able to detect losing grip on crystal orientation, it first launches a small walk-around search in order to get the orientation back. If this effort fails, the program will re-index the pattern using some newly recognized nodal spots. These spots will be found near the predicted low-indices reflections, since we assume that we are not off too much at this point. Finally, if none of these efforts work, we desperately restart the pattern recognition sequence from the very beginning. Understandably, `progressive` mode runs slower than `normal` mode, but it saves a lot of your trouble.

`progressive` mode is good only for occasional slippage. It starts random search *after* detection of losing orientation. In some cases, crystal jumping is so frequent that the input goniometer settings do not predict the crystal orientations very well throughout the dataset. Another refinement mode named `drunk` is available. In `drunk` mode, random orientation search is performed *before* refinement of each frame. The number of steps and radius of the random walk depends on whether and how soon a better fit is found. The best fit resulted from the random walk serves the initial orientation for the refinement. As `progressive` mode, the other initial parameters come from the last refined pattern. This mode runs even slower.

The refinement modes `normal`, `progressive` and `drunk` are either-or. They are not usually combined.

```
diagnostic    off
@ pre.spt.inp
Input
  Format      MAR3450Packed
  @ m37v3a_8us_pass1.inp
  Resolution  2.1 100
  Wavelength  1 1.4
  Spot        20 6 0.7
  Quit
Dataset      progressive
  Quit
Quit
```

Listing 3.4.0.0.1 Command script for `progressive` mode.

3.5 Repairing an Individual Frame

Sometimes all frames in a data set are refined well except one or two. The refinement mode `drunk` has a special usage that can repair the bad frames.

```
diagnostic    off
@ m37v3b_8us_004.mar3450.inp
Input
  Resolution  2.1 100
  Wavelength  1 1.4
  Spot        20 6 0.7
  Quit
Dataset      drunk
  Quit
Quit
```

Listing 3.5.0.0.1 Command script to repair a bad frame by using `drunk` mode.

This example loads an input file with some errors in its parameters, and refines them by `drunk` mode. This mode is more powerful than any other modes when a frame is missteered previously. However, it is not a good idea to load several input files and refine them by `drunk` mode in one job hoping that all of them will be repaired. The program may not work as desired, instead some parameters of a earlier frame will be transferred to a latter one. This usage is meant to repair one frame at a time. See 3.4 for details.

3.6 Fixing Parameters and Limited Refinement

You may sometimes find that certain parameters are changed too much by the refinement. You may fix a parameter by reset its uncertainty to 0 before refinement. This can be done by editing either a non-frame-specific or a frame-specific input file.

```
Input
...
  Distance    180.053 0
...
Quit
```

Listing 3.6.0.0.1 Uncertainty of distance reset to 0.

It is highly recommended that you use this feature to fix crystal-to-detector distance during geometric refinement for all data sets except for some special-purpose small molecule work. A variable distance within a data set will introduce systematic error to wavelength assignment.

You may also set an uncertainty to a small value in order to limit its freedom of change. An attempted change greater than twice as large as the given uncertainty will be suppressed to a smaller value.

However, editing an entire set of `.inp` files can be tedious. It is good that the uncertainty values in the first input file before refinement of a whole set will be transferred to other frames when the refinement is done. For example, in Listing 3.3.0.0.1, uncertainties in file `pre.spt.inp` will be copied into other `.inp` files created during the refinement. Therefore, it is a good idea to set your desired uncertainty values by editing the first input file.

Two string arguments `free` and `fix` can be used with these commands in Input section: `Crystal`, `Distance`, `Center`, `Pixel`, `Tilt`, and `Bulge`. Command `Distance fix` will override all uncertainty values in `.inp` files, and prevent distance from being refined. Command `Distance 180 fix` will not only reset all uncertainty values to 0, but also take a fixed distance of 180 mm for the refinement mode of calibration. See next section for calibration mode. Command `Distance 0.5 free` will override all uncertainty values in `.inp` files by the value 0.5, and allow distance being refined under limited range. If no value is given with the string `free`, the command is not understood to what extent distance is freed, and ignored except a warning message.

`Center`, `Pixel`, and `Tilt` may take two uncertainty values for both directions, when argument `free` follows. If only one is given, the second value is assumed to be the same as the first. `Bulge` may take two uncertainties too, but if one is given, the second is assumed to be 0. `Crystal` must take 6 values for a , b , c in Å and α , β , γ in degree, respectively. These values are either the cell constants if `fix` follows, or uncertainties if `free` does.

See 11.1.1 for details.

`Matrix` cannot take `free` and `fix` options. This may be added in later releases.

3.7 Calibration of Crystal-to-detector Distance and Cell Constants

Wavelength of each Laue reflection is assigned after crystal-to-detector distance and cell constants are refined. These parameters are highly inter-dependent on, but not very orthogonal to, each other. For example, distance and cell lengths may be refined to greater values at the same time and still keep nice prediction pattern. These parameters may easily run away during refinement. Due to this possibility of systematic bias in refined values of distance and cell constants, wavelength assignment may also be biased. Such bias is hard to be corrected, since in Laue diffraction, there is seldom a precisely known characteristic wavelength in the source spectrum. A new refinement mode `calibration` may be used to calibrate distance and cell constants to known values. In order to calibrate a parameter, one must first fix it so that no more refinement is allowed, and specify a known, accurate value to which calibration will be done (see the last section).

```
diagnostic      off
@ m37v3b_8us_002.mar3450.inp
@ m37v3b_8us_004.mar3450.inp
...
@ m37v3b_8us_060.mar3450.inp
Input
  Crystal      93.22 44.00 83.56 90.00 121.95 90.00 fix
  Distance     180 fix
  Resolution   2.1 100
  Wavelength   1 1.4
  Spot         20 6 0.7
  Quit
Dataset        calibration
  Quit
Quit
```

Listing 3.7.0.0.1 Command script performs calibration.

One may calibrate cell lengths but not cell angles by combined use of `fix` and `free` arguments. The next example sets a nonzero uncertainty for cell angle β so that it will not be calibrated.

```
Input
...
  Crystal      56.2 29.7 40.9 90.0 114.7 90.0 fix
  # set cell constants and reset all uncertainties to 0
  Crystal      0 0 0 0 0 0.2 0 free
  # free beta only
...
Quit
```

Listing 3.7.0.0.2 Combined use of `fix` and `free` arguments.

Other parameters such as orientation matrix, direct-beam center, detector tilt angles and bulge corrections do not need calibration, since they cannot run away in refinement. The only other parameters may run away are pixel sizes. No calibration is provided because the vertical pixel size is never allowed to be refined.

This calibration can be avoided, if distance and cell were never refined.

3.8 Re-index

If none of the refinement modes can carry geometry refinement to the end, or due to whatever reason, you must re-index a pattern in middle of a set, a common concern is consistency of indexing. For example, consistent indexing is demanded when anomalous scattering signal from a heavy element is under investigation (See 5.1 and 5.4). Due to symmetry of a space group, an orientation matrix usually has a set of equivalent matrices. The first indexing carried out by command script in Listing 2.2.0.0.1 chooses one of the equivalent matrix with least missetting angles. To ensure a consistent re-indexing with a previous indexing, the previous missetting matrix shall be given. A report of discrepancy between the new and old indexing will be printed. Precognition automatically

selects the least discrepancy. This angular value shall be small enough, say less than a few degrees, to safely believe indexing consistency. The following script shows how a previous missetting matrix can be given to the indexing program. The numbers are arranged by row, such as

Matrix r_{11} r_{12} r_{13} r_{21} r_{22} r_{23} r_{31} r_{32} r_{33}

if the matrix is $\begin{pmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{pmatrix}$. The given matrix must be orthogonal. See 11.1.1.6.

```

diagnostic      off
Input
  Crystal       /home/renz/xtal_info/pypwt.xtl
  Matrix        -0.6630 -0.7483 0.0177 0.0794 -0.0468 0.9957 -0.7443 0.6616 0.0905
  Format        Mar345
  Distance      250
  Center        1704.5 1711.0
  Pixel         0.1 0.1
  Goniometer    16
  Image         /mnt/cdrom/wlnd_2a_05.mar3450
  Resolution    2.3 100
  Wavelength    1 1.4
  Spot          10 5 2.8
  Quit
Spot            wlnd_2a_05.re.spt
Ellipse
Pattern        0 wlnd_2a_05.pre.spt
Quit

```

Listing 3.8.0.0.1 Command script performs re-indexing.

3.9 Manual Refinement

What else can be done if none of the refinement modes works and re-indexing is unsuccessful? I urge you to report such cases to me. This should never happen if your dataset is of reasonable quality and if the program is truly bug-free. Nevertheless, before you give up, try to offer some manual helps to the program.

Since manual refinement deals with one frame at a time, always use a frame-specific parameter file. First, edit the parameter file to set all parameters to the values you can trust. For example in Listing 3.9.0.0.1, a previously refined value of horizontal pixel size 0.100348 remains. Tilt and bulge corrections are reset to 0. Distance fluctuates too much in refined results, which makes the experimental logged value 180 mm most reliable. All uncertainties are then reset to 0, which means no refinement allowed. Try to guess a better goniometer setting, here ϕ is set to 6.5° instead of the logged value of 6°.

```
Input
  Crystal   91.870 91.870 45.891 90.000 90.000 120.000 168
  Matrix    0.619239 -0.062438 -0.782716 -0.756918 -0.312611 -0.573892 -
0.208853 0.947827 -0.240842
  Omega     90 0
  Goniometer 0 0 6.5

  Format     MAR3450Packed
  Distance   180 0
  Center     1702.17 1710.69 0 0
  Pixel      0.100348 0.1 0 0
  Swing      0 0 0 0
  Tilt       0 0 0 0
  Bulge      0 0 0 0

  Image 0    ./129w_25a_003.mar3450
  Resolution 2.10 100.00
  Wavelength 1.00 1.30
  Quit
```

Listing 3.9.0.0.1 Parameter file specific to a certain frame for manual refinement.

```
diagnostic    off
@ 129w_25a_003.mar3450.inp
Input
  Format       MAR345
  Resolution   2.1 100
  Wavelength   1.0 1.3
  Spot        12 6 3
  Quit
Dataset
  Quit
Quit
```

Listing 3.9.0.0.2 Command script for manual refinement.

Since we are going to refine one frame only, no `.gon` file nor `Goniometer` command is needed. Simply load one parameter file and refine under `normal` mode. This is a trial-and-error approach. Compare the result versus your guess of goniometer setting. The following example illustrates how you may steer a Laue pattern at all directions. We start with a nicely refined pattern, then offset some goniometer angles so that the predicted pattern in black + is offset from the observed one in red o. This example shows how and how much you should make your educated guess in order to quickly capture a losing orientation.

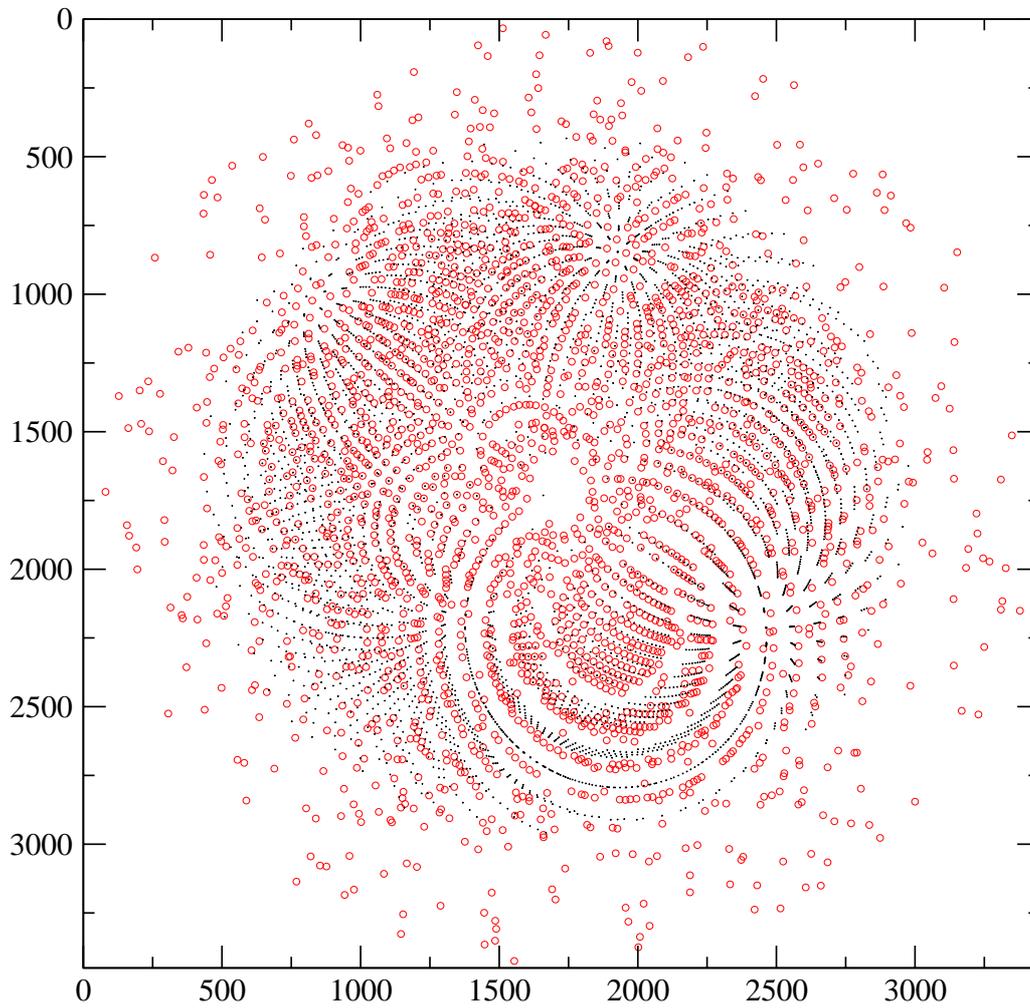


Figure 3.9.0.0.1 Offset predicted pattern in black + towards up by commands:

```
Goniometer 0 0 6.5  
Omega      90 0
```

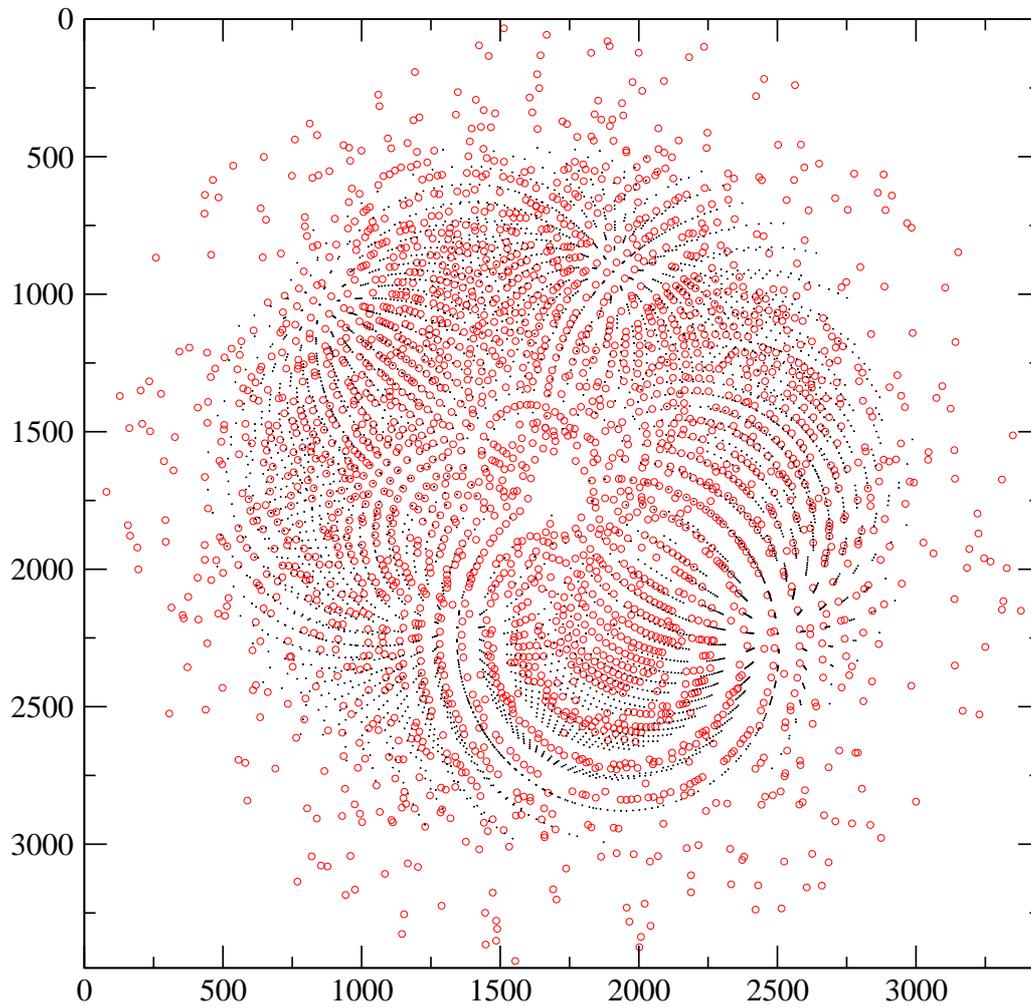


Figure 3.9.0.0.2 Offset predicted pattern in black + towards down by commands:

```
Goniometer 0 0 5.5  
Omega      90 0
```

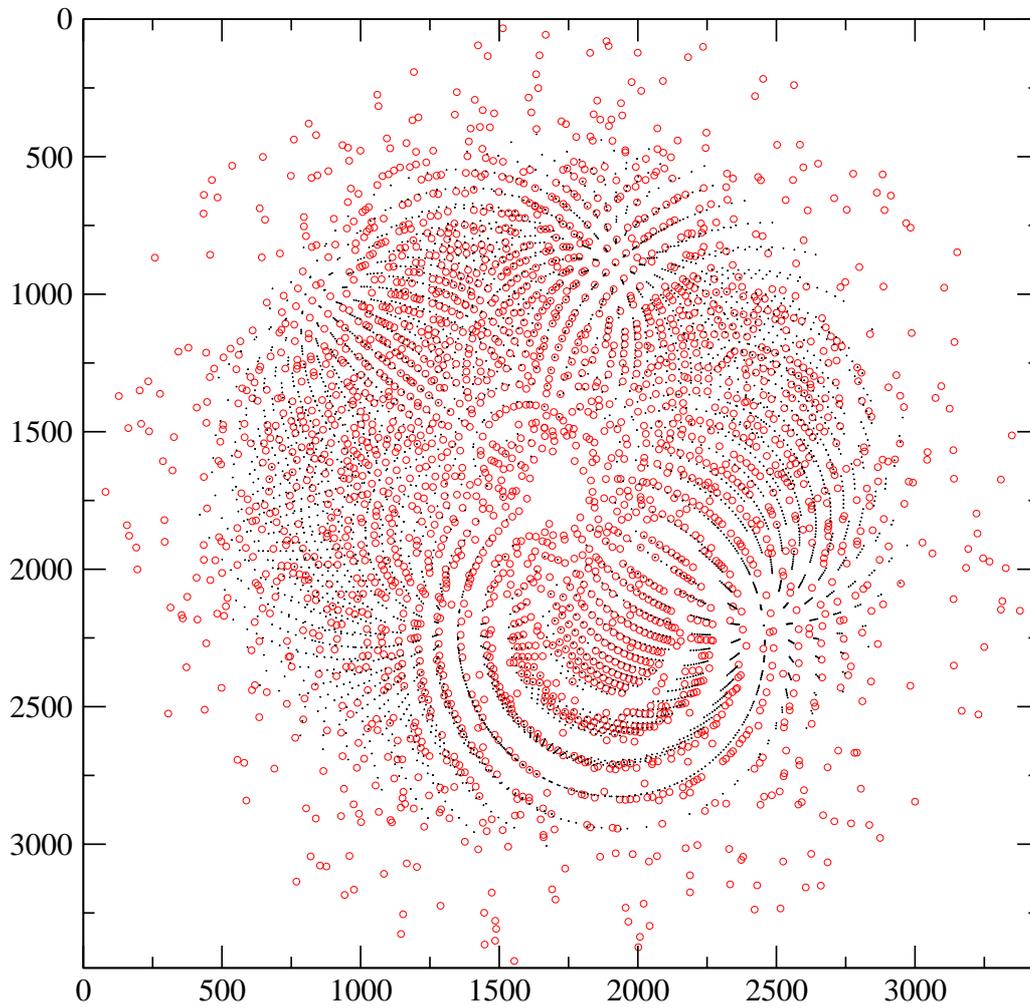


Figure 3.9.0.0.3 Offset predicted pattern in black + towards left by commands:

```
Goniometer 0 0 6  
Omega      90 -0.5
```

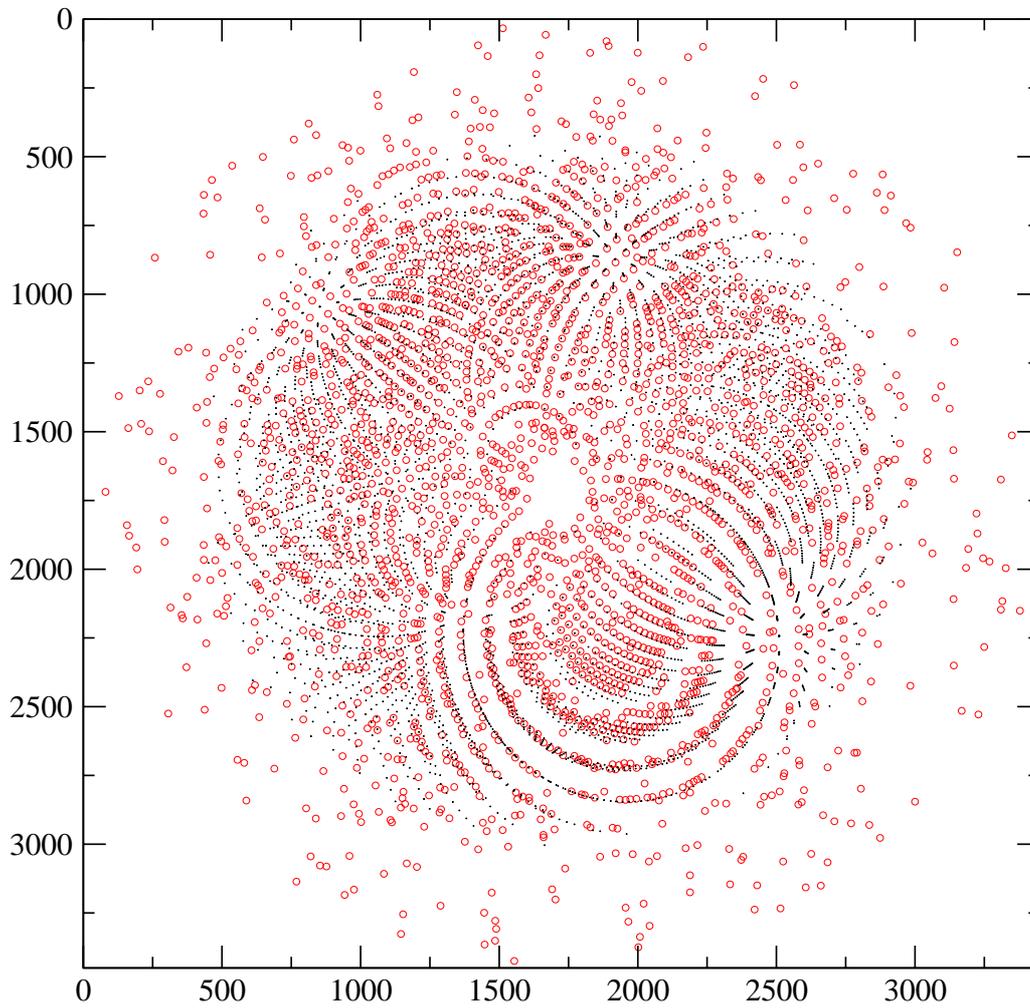


Figure 3.9.0.0.4 Offset predicted pattern in black + towards right by commands:

```
Goniometer 0 0 6  
Omega      90 0.5
```

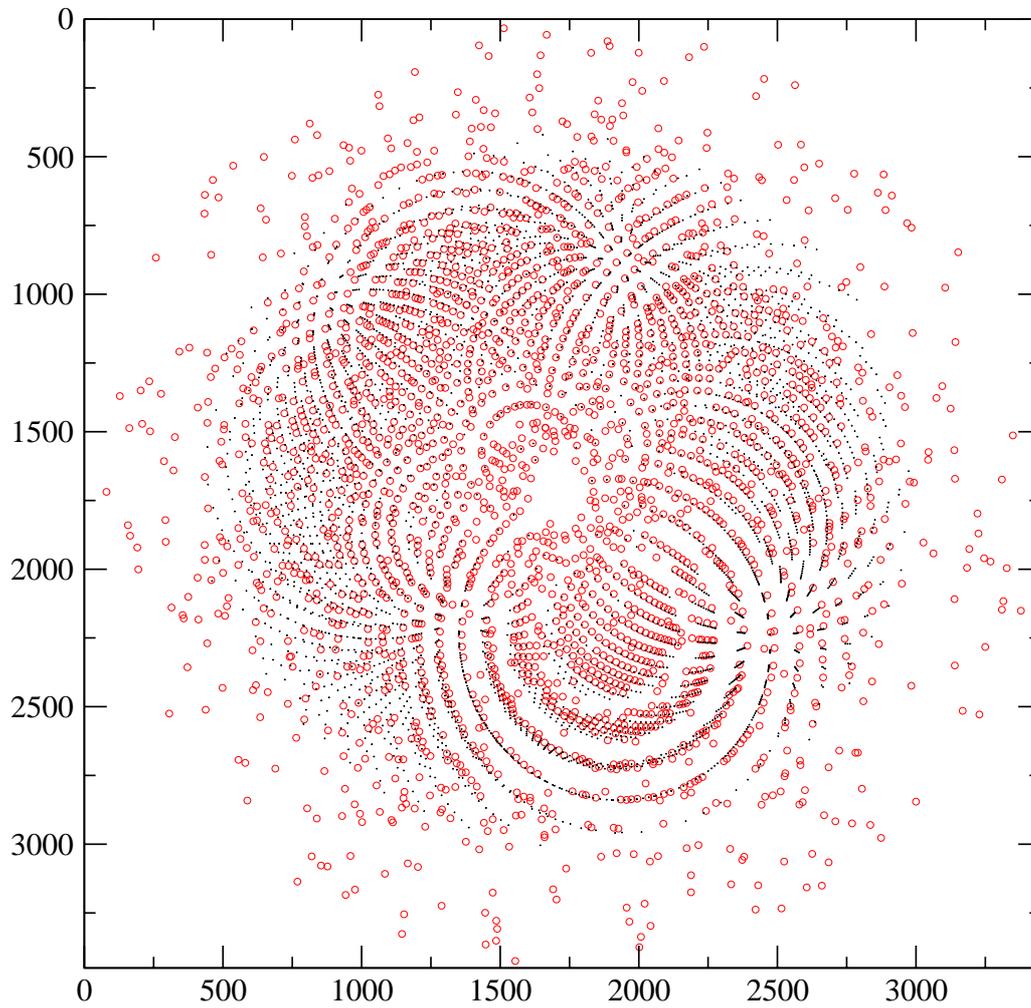


Figure 3.9.0.0.5 Offset predicted pattern in black + clockwise by commands:

```
Goniometer 0 1 6  
Omega      90 0
```

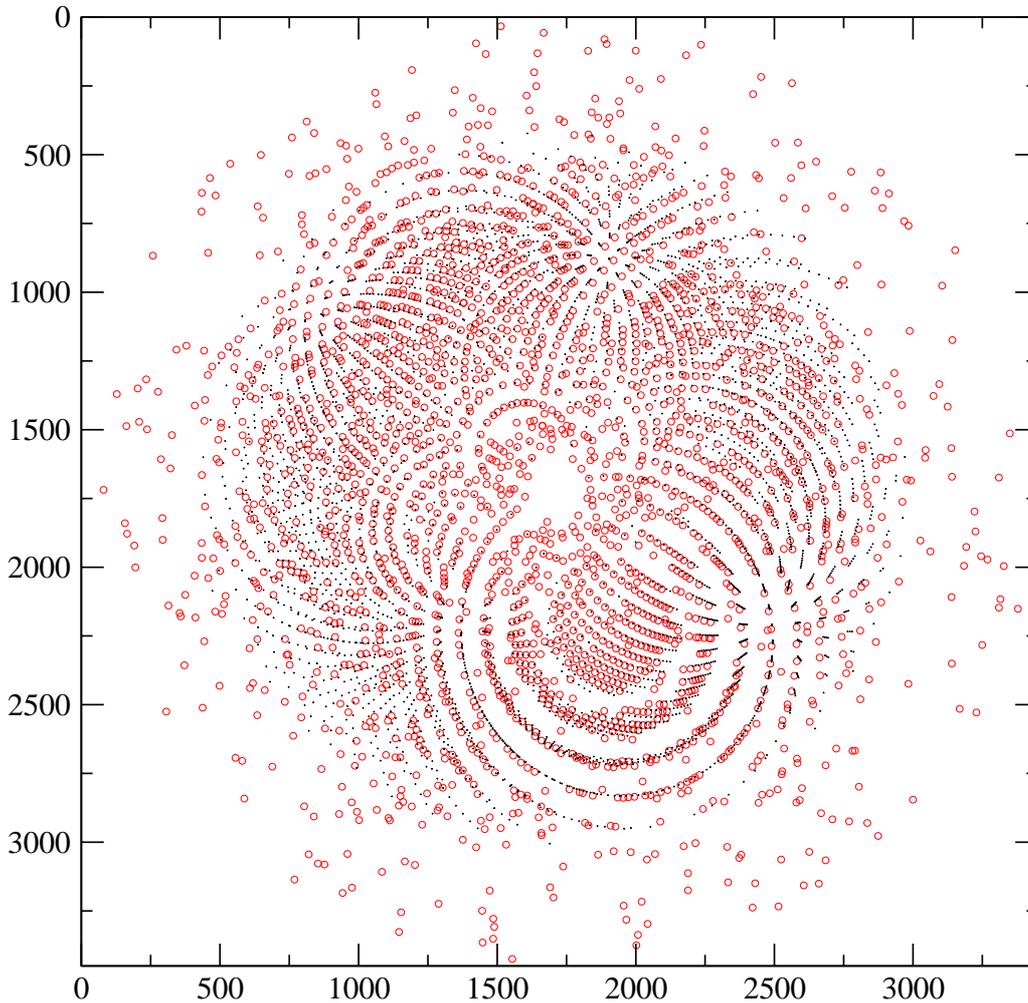


Figure 3.9.0.0.6 Offset predicted pattern in black + counterclockwise by commands:

```
Goniometer 0 -1 6
Omega      90 0
```

3.10 Final Refinement

Another refinement mode called `final` may be helpful after either `normal` or `progressive` or `drunk` is done, however, it is not always necessary.

```
diagnostic off
@ m37v3b_8us_002.mar3450.inp
@ m37v3b_8us_004.mar3450.inp
...
@ m37v3b_8us_060.mar3450.inp
Input
Resolution 2.1 100
Wavelength 1 1.4
```

```
Spot      20 6 0.7
Quit
Dataset   final
Quit
Quit
```

Listing 3.10.0.0.1 Command script performs final refinement.

This command script for `final` mode is slightly different from those for `normal` and `progressive` modes. First, a set of frame-specific input files are necessary, not just one. Second, no image type and `.gon` file are needed anymore, since all information are loaded from the frame-specific input files. This process overwrites the set of input files by new ones.

You may also choose to gather the long list of filenames in a separate file, and name the file properly to avoid confusion with the file that contains goniometer settings (3.3 and Listing 3.3.0.0.5). There is no enforced rule on whether the scripts are separated and how these file should be named. I leave the style to the users, but a hint is that the global printing switches `prompt` and `result help` to make the log file more concise.

```
# m37v3b_8us pass 1
prompt off
result off
@ m37v3b_8us_002.mar3450.inp
@ m37v3b_8us_004.mar3450.inp
...
@ m37v3b_8us_060.mar3450.inp
prompt on
result on
```

Listing 3.10.0.0.2 A script named `m37v3b_8us_pass1` defines a dataset or a pass.

```
diagnostic off
@ m37v3b_8us_pass1
Input
Resolution 2.1 100
Wavelength 1 1.4
Spot      20 6 0.7
Quit
Dataset   final
Quit
Quit
```

Listing 3.10.0.0.3 Final refinement by a shortened script.

CHAPTER 4

Spot Integration

The last two chapters have dealt with diffraction geometry almost exclusively. Given a prediction of the location of each reflection as precise as it can be on detector surface, the procedure of spot integration is very much independent of diffraction geometry. This procedure is responsible for finding diffraction power of each reflection. Three major issues are critical to the accuracy, completeness, and speed of integration. First, an accurate evaluation of a local background scattering is needed. Background scattering is modeled by a tilting plane for each spot in Precognition. Second, closely spaced spots need to be resolved in order to split their diffraction powers. Third, a variety of irregular spot shapes shall be modeled to obtain accurate summation of intensity, or more often known as integrated intensity. Precognition offers a spectrum of integration modes to selectively perform these functions at different costs of time.

4.1 General Issues for Integration

A command script for integration is shown below.

```
diagnostic    off
busy          off
warning       off
@ m37v3_8us_002.mar3450.inp
@ m37v3_8us_004.mar3450.inp
...
@ m37v3_8us_062.mar3450.inp
Input
  Image       lambda.lam
  Spot        20 6
  Quit
Dataset       hybrid
  Resolution  1.6 100
  Wavelength  0.98 1.6
  Quit
Quit
```

Listing 4.1.0.0.1 Command script for spot integration.

First, a list of input files for all images to be integrated is loaded in. On a dual-processor machine, split this list into two half and make both processors work. More concisely, the long list can be put in a separate file as shown in Listing 3.10.0.0.2, and loaded in by this script:

```
diagnostic    off
busy          off
warning       off
@ m37v3_8us_pass1
Input
  Image       lambda.lam
  Spot        20 6
  Quit
```

```
Dataset      hybrid
Resolution  1.6 100
Wavelength  0.98 1.6
Quit
Quit
```

Listing 4.1.0.0.2 A more concise command script for spot integration.

The `Input` section specifies a λ -curve and an average spot length and width in pixel. The `.lam` file is a two-column ASCII file with wavelength in Å in the first column, and relative intensity in the second (11.8.4). If the spectrum of your beamline is unknown, a very rough box is sufficient.

```
0.7 0
1.0 1
1.2 1
1.6 0
```

Listing 4.1.0.0.3 A rough λ -curve indicating the peak of the spectrum between 1 and 1.2 Å.

The purpose of λ -curve here is to enable resolution-dependent bandwidth. In contrast to monochromatic source, incident X-rays in a white or pink beam have a certain energy distribution. When reflections stimulated by X-ray photons at the most popular energy reach a saturation level of the detector, reflections stimulated by other energies may be far from saturated, even hardly observable, for example, those reflections in the highest resolution bin and stimulated by X-rays at the two ends of the spectrum. As a result, the effective bandwidth is a function of resolution. Simulation shows the method of resolution-dependent bandwidth predicts observable Laue spots far better than a constant bandwidth. That is to say, the volume in reciprocal space where Bragg diffraction condition might be met is bounded by not only three spheres on Ewald construction, $1/\lambda_{\min}$, $1/\lambda_{\max}$ and $1/d_{\min}$, but also a strange contour surface derived from the input λ -curve (Figure 4.1.0.0.1).

The `Spot` command in `Input` section may also accept a third number as a σ -cut. In case of profile fitting, the σ -cut plays a role in selection of sample profiles. A low level of σ -cut may allow many weak spots to contribute to profiling, therefore the quality of the profile may be sacrificed. A large σ -cut value may results in insufficient number of sample profiles, and prolonged execution time. The default σ -cut is 3, which is good for most cases. You may adjust this value according to your image quality, and check the saved `.spt` file for evaluation. In general, σ -cut should not be lower than 2. Obviously, σ -cut affects those integration modes involving profile fitting only. See 4.5 through 4.7 for details of those integration modes.

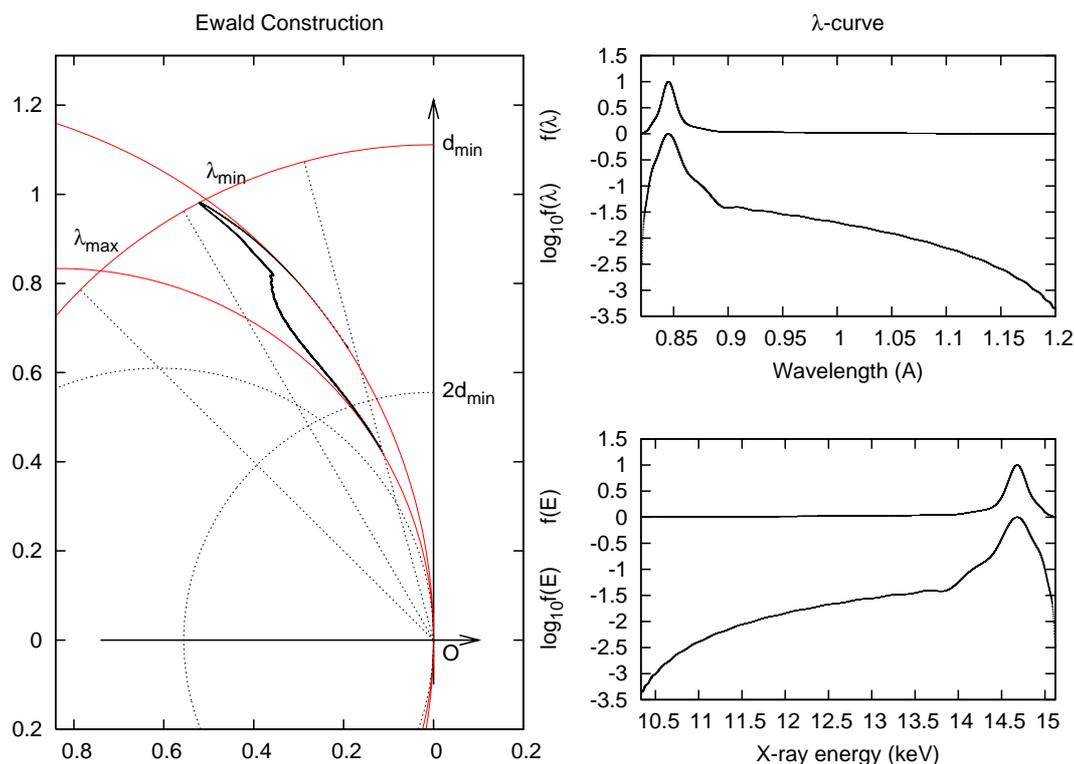


Figure 4.1.0.0.1 Ewald construction that shows resolution-dependent bandwidth.

The `Dataset` command discussed in the previous chapter can also accept integration modes (11.1.7). Once again, this command enters into a submenu. If an integration mode is given, resolution and wavelength ranges can be given here in addition to directory names. Despite what ranges is set in `Input` section, these ranges in `Dataset` section will determine the integration ranges. If no new ranges are given, the old ranges set in `Input` section will take effect. The directories where images should be loaded and where results should be saved can be specified in the same manner as described in the last chapter (3.3). The result of integration is a set of `.ii` (integrated intensities) files (11.8.6) to be loaded by the scaling program `Epiform` (Chapter 5).

The command `Dataset` with an integration mode may also accept an optional numerical argument that specifies a starting 'pattern number' of `LaueView`. `.sht` files will also be saved, so that scaling can be performed by `LaueView`. By default, `.sht` files will not be generated starting from release 4.1.0.

An integrated intensity file (11.8.6) is a point of entry to the system of Precognition. It is a free format ASCII file with arithmetic values only. Each record has a same format and represents an observed reflection. Harmonic or spatially overlapping reflections must be arranged consecutively. General users shall avoid any modification to these files.

The first three columns are Miller indices h , k , and l . The fourth integer is multiplicity of harmonics. The fifth and sixth columns are center coordinates of the reflection spot on detector in pixel values. See 2.1.3 and 2.2.1 for definition of pixel coordinates. The content of these two columns may be changed in future releases. Column 7 and 8 are resolution and wavelength of the reflection in Å, respectively. The last two columns are integrated intensity and its standard deviation error, respectively, in $\text{pixel}^2 \times \text{detector unit}$.

```

...
  5   7  -22  1 2076.7 2427.9  3.0961 1.257307463      328.44    54.93
  6   8  -27  1 2050.0 2431.1  2.5606 1.029589599       24.93   123.97
  4   4  -20  2 1923.3 2422.6  3.6535 1.380610908    16129.92  121.86
  5   5  -25  2 1923.3 2422.6  2.9228 1.104488727    16129.92  121.86
  6   6  -28  1 1956.5 2463.7  2.6009 1.040187484     4279.40   99.42
  5   5  -23  1 1964.0 2472.5  3.1638 1.280254859     1055.61   93.11
  4   4  -21  1 1902.4 2394.7  3.4859 1.265264294     8475.77  144.94
...

```

Listing 4.1.0.0.4 A section of integrated intensity file.

Algorithms of spot integration can be divided into two distinctive categories: summation and profile fitting. Summation methods are certainly faster. They are also very good for evaluation of strong spots. But summation methods have no ability of identifying and filtering noise, which make their results from weak spots poor. They also experience great difficulty when spots are spatially overlapping. Profile fitting methods are learning processes. They make connection between strong and weak reflections by learning profile from strong ones and applying it to weak ones. Such process identifies and filters noise, so that they are in general better methods for evaluation of weak spots. Knowing spot profile also greatly facilitates deconvolution of spatially overlapping spots. However, noise filtering can be a risky business. This is because that the base of profile learning can be too wide or too narrow, as we discussed above. When a learned profile does not fit individual spots very well, its noise filtering and therefore its integration become biased. This happens in more significant way to stronger spots. Between summation and profile fitting, and more toward the latter, numerical profile has the advantage of profile fitting, but it is not so powerful in terms of noise filtering, since it inherits some numerical features of summation methods.

Precognition offers the full spectrum of integration algorithms. More importantly, each implementation tries to extend its advantage and to eliminate its shortcomings as much as possible. For example, some summation methods may do its best to isolate two closely spaced spots. The analytical profile fitting methods are combined with numerical compensation. All these efforts aim at evaluation of diffraction power of as many reflections as accurate as possible.

4.2 Integration On-the-fly, box Mode

A fast integration mode named `box` is available for quick checking of data quality while data are being collected. The algorithm used here is the fairly common square box method. A circular partition inside the box is designated to be the peak area. The size of

the square box is specified by the average length in `Input` section. The limitation of this algorithm is rather severe when used for integration of Laue spots, since no resolution of spatial overlap is attempted and background is only modeled by a non-tilting level. Many weak spots are not observable by this method.

4.3 Elliptical Peak Area, `fixedElliptical` Mode

A better method called `fixedElliptical` is to use an elliptical peak area in integration, since Laue spots are often streaking in radial directions. A rectangle box of size specified in `Input` section is reoriented for each spot being integrated. An elliptical peak area is partitioned in each rectangle box. The size of the box is specified by user. A typical size shall be used, that is, the input size shall represent the greatest population of spots. This algorithm can also resolve some spatial overlaps as long as no pixel is involved in more than two peak areas. Thirdly, background near each spot is modeled by a tilting plane, which makes summation of peak much more accurate.

4.4 Learned Elliptical Peak Area, `variableElliptical` Mode

This integration mode called `variableElliptical` offers another level of automation. The integration box for each image is determined by the program itself. The user input is only an initial hint. The major problem of `fixedElliptical` mode is therefore solved. This mode is more necessary when spot streakiness varies significantly in a data set.

4.5 Numerical Profile, `numeric` Mode

Spot profile is a typical example of pattern recognition. Weaker spots are recognized by learned profile from stronger ones. This integration mode `numeric` learns an averaged numerical spot profile within a small area on detector surface from some strong and well separated spots, which are sometimes called samples. See above for selection of sample profiles. This profile is then applied to all predicted spots. This algorithm makes resolution of spatial-overlaps possible. A lot of weak spots also become observable.

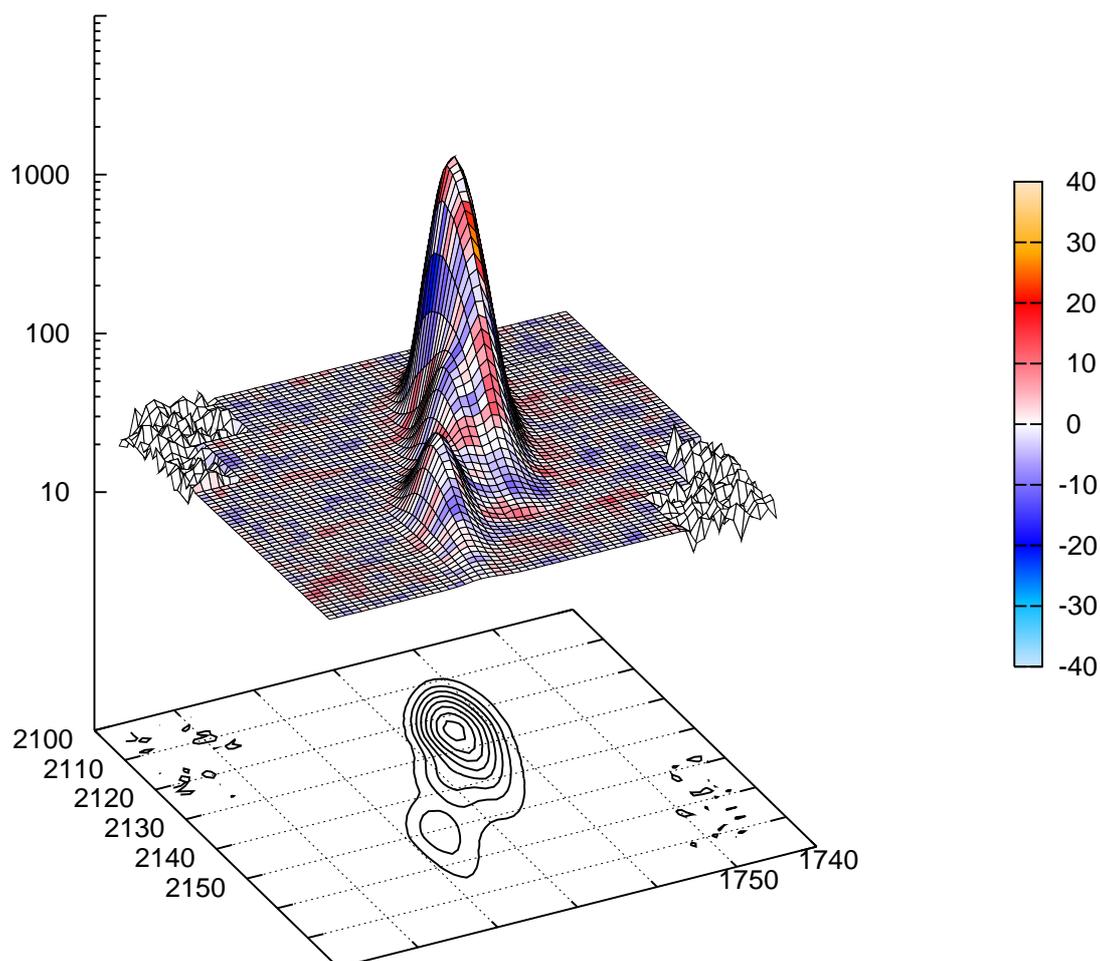


Figure 4.6.0.0.1 Three spatially-overlapping spots fitted by analytical profiles. One strong spot and one weak spot are obvious. A third spot between the other two is shown as a bulge on the contour plot and a shoulder of the main peak on the surface plot. The surface plot is on a logarithmic scale. The colors show residual of fitting.

4.6 Analytical Profile, `linearAnalytical` Mode

Ren et al. (*J. Appl. Cryst.* **28**, 461-81, 1995) have shown previously that a multi-parameter analytical function can be used as spot profile for Laue diffraction. This analytical profile has its flexibility and stability, and therefore advantage over a numerical profile. Even more weak spots can be integrated by analytical profile than by numerical one. This is because analytical profile naturally suppresses additional noise contained in numerical profile. This mode is called `linearAnalytical` since spatial overlap resolution is carried out by a linear model. This mode is the most similar one to the algorithm used in `LaueView`, and so far remains the best integration mode. The linear model of spatial overlap does very good job isolating spots from each other, if they are reasonably spaced. See Figure 4.6.0.0.1 for an example. However, deconvolution of

spatial overlaps by profile fitting will eventually reach a limit when spots are getting extremely close to each other respect to their own size and the amplitude of prediction error. A second attempt to deconvolute those extremely-close spatial overlaps is implemented with the same strategy of harmonic deconvolution, that is, to take advantage of redundant and symmetry-related measurements. See 5.4 through 5.6 for detail.

4.7 Simultaneous Profile Fitting and Spatial-overlap Deconvolution, nonlinearAnalytical Mode

This mode of integration pushes the envelope to an extreme. In case of large unit cell, it may be difficult to find enough well separated spots for profile fitting. Or in case of small unit cell, it may also be difficult to find enough sample spots, since there are not many spots to start with. This may be more true in some area of a frame than in others, for example, where major ellipses are passing through. As described above, the last mode `linearAnalytical` performs profile fitting only on selected samples and deconvolutes spatial overlaps when applying the previously obtained profile. This integration mode merges these two procedures, so that the profile is being constantly adjusted. Figure 4.7.0.0.1 gives one example that `nonlinearAnalytical` mode does a better job to minimize prediction error than `linearAnalytical` mode does.

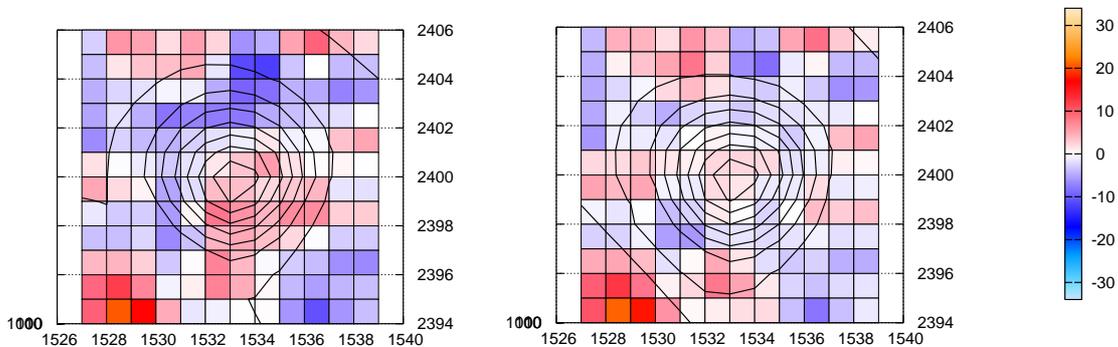


Figure 4.7.0.0.1 Profile fitting of one spot by `linearAnalytical` (left) and `nonlinearAnalytical` (right) modes. Color shows the residual of fit on each pixel. The left panel shows positive errors on one side of the peak and negative on the other, which indicates that positional error is not completely corrected. The right panel shows slightly different position and shape of the profile, and random error distribution.

4.8 Numerical Compensation to Analytical Profiles

Although profile fitting is so powerful for integration of weak spots, no single profile, numerical or analytical, can fit all spots at a same time. It is more obvious that strong spots are harder to be fitted perfectly. An algorithm of numerical compensation corrects such misfit to a large extent, while keeps loyalty to the analytical profile found. Both analytical modes `linearAnalytical` and `nonlinearAnalytical` feature this compensation.

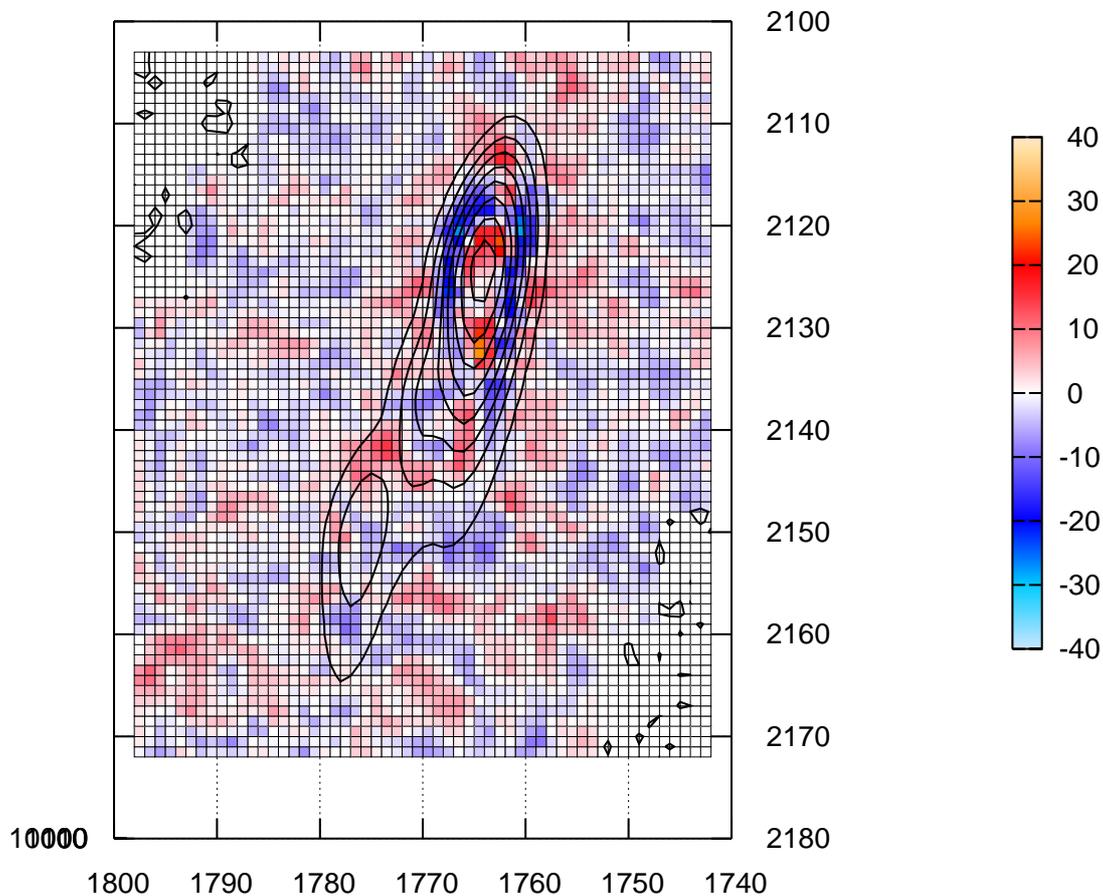


Figure 4.8.0.0.1 The contour view of the same three spots shown in Figure 4.6.0.0.1. The main peak height is over 1000 detector counts. The color-coded fitting residual ranges $\pm 2\%$ after numerical compensation.

4.9 hybrid Mode

This integration mode `hybrid` is trying to find an optimal balance between summation and profile methods. This mode is a hybrid between `variableElliptical`, the best summation method and `linearAnalytical`, the best profile fitting method. This mode may be changed in future release.

CHAPTER 5

Data Reduction

This part of data processing is often called scaling in monochromatic methods, which refers to frame-to-frame scaling and determination of relative temperature factors. In Laue diffraction, a far more important and complex task, wavelength normalization, plays the major role. Wavelength normalization results in a wavelength-dependent function often called λ -curve. This curve looks mostly like the X-ray spectrum of the incident beam, when converted to an energy-dependent function. This curve is in fact an overall correction of wavelength dependency, including absorption correction (see 5.2.4 for details).

Each Laue spot is usually associated with a wavelength, called a single, or several discrete wavelengths, called a multiple, until Ren et al. (*J. Synchrotron Rad.* **6**, 891-917, 1999) pointed out that this is a gross oversimplification of real Laue diffraction.

$$\frac{\Delta\lambda}{\lambda} = \frac{\delta}{\tan\theta}.$$

This equation reformulated from Ren et al. essentially states that relative bandwidth of a Laue reflection is proportional to crystal mosaicity and inversely proportional to tangent of Bragg angle, and approximately the angle itself in nearly all protein cases. A Laue spot is stimulated by X-rays within an energy band of several hundreds of eV, and can often become a partial reflection. This is particularly true at low Bragg angles. A partial reflection in monochromatic oscillation happens since the oscillation range is limited to a small angle of 1° or so. Such partial is called angular partial. A Laue partial happens due to insufficient energy range, therefore Laue partial is called energy partial. Consideration of energy band for each Laue reflection and energy partials improves data merging from different Bragg angles.

An individual program `Epiform` (*energy partial improved normalization*) carries out all the functionalities described above. What `Epiform` really does is data reduction from integrated intensities to structure factor amplitudes or magnitudes.

5.1 Input Data and Control Parameters

An example of command script for scaling is shown below.

```
diagnostic      off
busy            off
warning        off

@ m37v3_8us

Input
  Image          initial.lam
  Resolution    2.1 100
```

```
Wavelength 1 1.5 1.1
Chebyshev 64 unimodal
Spot 8 6
Quit

Scale 3 2 1 m37v3_8us_002.mar3450.ii
Quit
Lambda refined.lam
Apply 1 m37v3_8us.hkl
Quit
```

Listing 5.1.0.0.1 Command script to run `Epinorm`.

The command script of `Epinorm` shown in Listing 5.1.0.0.1 is very similar to that of `Precognition`. Misusing a script will generate error message. Just like the one for integration, this script first loads in a set of `.inp` files, but indirectly from a separate file. Bad frames should be excluded from this list.

The `Input` section is optional. An initial λ -curve can be loaded here. See Chapter 4 and 11.8.4 for its format. If no λ -curve is loaded, a straight line is assumed. In this case, you should specify a wavelength range. If not, 0.5 to 1 Å is the default. If an initial λ -curve is loaded, you may still change the wavelength range by using the `Wavelength` command. If part of the new wavelength range falls out of the λ -curve, 0 intensity is assumed. The `Wavelength` command can take a third, optional number as the reference wavelength. However, please note that the explicit input of wavelength range and reference will take effect only if this command is after the initial λ -curve. If no reference is given, it is automatically determined from the initial λ -curve. The shortest wavelength corresponding to the greatest intensity values in the initial λ -curve will be chosen as a reference.

The command `Resolution` specifies a resolution range within which data are loaded for scaling. In most cases, this command is unnecessary, so that data at all resolutions are loaded. If you explicitly specify a resolution range, use the integration range. This command is only useful in special cases, and will be explained elsewhere.

The command `Chebyshev` is as same as the one in `Scale` section. See 5.2.3 for description.

The command `Spot` with two numerical arguments initializes crystal mosaicity. Obviously, more streaky the spots, larger the mosaicity it would be. If no `Spot` is given, the default mosaicity is 0. See 5.2.5 for more.

If there are some heavy atoms present in your crystal, and if you desire to examine the anomalous scattering signal from them, an additional control command `Anomalous` in `Input` section can be used (11.1.1.19, not shown in Listing 5.1.0.0.1). This command toggles a flag that signals whether anomalous scattering should be considered during data reduction. The default state is off, which is suitable for the most cases. This is the very first point in the entire process where an explicit option can be given, if anomalous

scattering should become a concern. However, implicitly but clearly, at the very beginning of the data processing, consistent indexing of all frames in a dataset is crucial to extraction of anomalous signal. One must take great care of such consistency by using all possible means provided in Chapters 2 and 3. If re-indexing cannot be avoided, specify a desired orientation matrix prior to re-indexing as described in 3.8. Switching on the anomalous flag would signal the program to separate Friedel pairs, so that each member of a Friedel pair is considered independent of, instead of equivalent to, the other. R_{merge} 's calculated later will not include discrepancy between Friedel pairs (See 5.2.6). It is also possible to delay the switch after scaling and before merging of redundant and equivalent data. See 5.4 for detail. It must be noted that these two alternatives reflect different strategies of handling anomalous signal. The former preserves the maximum amount of anomalous signal, however, may misidentify some systematic errors as anomalous signal. The latter guards from possible systematic errors, but may unknowingly attenuate some real anomalous signal. I left this as a user's choice.

The command `Anomalous` in `Input` section may take an optional string argument `on` or `off`. If no argument or no recognizable one is given, the command negates the current state.

5.2 Data Selection and Parameter Fitting

The main command is `Scale`. It may take three numeric arguments and a string argument. All these arguments are optional. Starting from release 4.3.0, this command enters a submenu, which must be closed by a command `Quit`. Please caution that command scripts written prior to this release will not work properly. All optional arguments to the command `Scale` still function, but they are taken as initial values when entering to the submenu. The submenu shown below has the full control of the scaling process.

```
Scale
  Sigma      3 and
  Data       1000
  Isotropy   0 scale
  Isotropy   0 temperature
  Mosaicity  8 6 fix
  Wavelength 1 1.5 1.1
  Chebyshev 64 unimodal
  Mapping    nonlinear
  Lambda-shift free
  Restore    m37v3_8us.inp
  Mode       global
  Quit
```

Listing 5.2.0.0.1 `Scale` section that gives full control of scaling process.

The entire submenu can be roughly divided into several topics: data selection criteria, data isotropy, λ -curve modeling, crystal mosaicity, absorption correction, scaling mode, and a few minor points such as beam polarization, reference frame, and parameter restoring. They are discussed in the next a few sections.

5.2.1 Data selection

The first numerical argument to the command `Scale` specifies a σ -cut. 3 is the default. If $I/\sigma(I)$ less than this value, this integrated intensity will not be used in scaling, however, this does not mean that this data point will be lost forever. This minimization process does not require all the data points available. You may watch the reported data-to-parameter ratio during scaling. If this ratio reaches a few hundreds, there should be enough data points to over-determine the parameters. σ -cut must be a value greater than or equal to 0. 0 σ -cut means that all positive, but not 0, integrated intensities will join scaling. The σ -cut is the only control where user can intervene the data rejection. Other data rejection criteria are automatic. See Listing 5.3.2.0.1 and text below.

Another way to control data selection is to specify number of data points loaded from each frame. If the first numeric argument is equal to or greater than 100, it is no longer considered as σ -cut, rather the number of data points per frame. If you have tens of frames to scale at once, a few hundreds data points per frame would be sufficient. If you only scale a few images, you may need more. Controlling data points per frame usually makes the program run faster, however, it opens a possibility of insufficient data. It should be understood that data-to-parameter ratio is not the only thing to consider here. Data population as functions of resolution and wavelength is more important. Using only the strongest, and therefore insufficient data may results in no representation at high resolution and two wings of the spectrum. This could cause arbitrary temperature factors and noisy λ -curve.

The subcommands `Sigma` and `Data` also perform data selection, and may function in a combined fashion. `Sigma` takes a numerical argument as σ -cut, and an optional string argument of `and` or `or`. `Data` also takes a numerical argument but as the number of data points per frame to be selected, and an optional string argument of the same choices. Consider any given frame, there will be a certain number N_σ of integrated intensities, as I call them data points, qualify the σ -cut given by the first argument to the command `Scale` or (modified) by the subcommand `Sigma`. We denote the number per frame given by the subcommand `Data` N . Comparing N_σ and N , and combining the key word `and` or `or`, there will be 4 situations as tabulated in Table 5.2.1.0.1.

N_σ/N	Logic	Data selection	Usage
$N_\sigma > N$	and	Randomly select N data points from N_σ qualified ones.	Use for strong patterns. Evenly distribute data.
$N_\sigma > N$	or	The default. Use all N_σ qualified data points.	Use for fair quality patterns. Maybe slow when N_σ is very large.
$N_\sigma < N$	and	Use N_σ qualified data points.	Use for weak patterns. Limit the minimum data quality.
$N_\sigma < N$	or	Use N best data points even some do not qualify the σ -cut.	Use for weak patterns. Limit the minimum number of data.

Table 5.2.1.0.1 σ -cut versus number of data points.

5.2.2 Data isotropy

The second argument to the command `Scale` can be -1, 0, 1, or 2. 0 is the default, which indicates isotropic scale factors and temperature factors only. -1 indicates isotropic scale factors only. All temperature factors will be kept as initialized. 1 indicates anisotropic but linear scale factors and temperature factors can be used.

$$f_a = (a_0 + a_{11}h + a_{12}k + a_{13}l)^2, \text{ and}$$

$$f_b = \exp[b_0(\sin \theta / \lambda)^2 + b_{11}h + b_{12}k + b_{13}l],$$

are scale factor and temperature factor, respectively.

2 indicates nonlinear anisotropic scale factors and temperature factors are allowed.

$$f_a = (a_0 + a_{11}h + a_{12}k + a_{13}l + a_{21}\frac{h^3}{|h|} + a_{22}\frac{k^3}{|k|} + a_{23}\frac{l^3}{|l|} + a_{31}hk + a_{32}kl + a_{33}lh)^2, \text{ and}$$

$$f_b = \exp[b_0(\sin \theta / \lambda)^2 + b_{11}h + b_{12}k + b_{13}l + b_{21}\frac{h^3}{|h|} + b_{22}\frac{k^3}{|k|} + b_{23}\frac{l^3}{|l|} + b_{31}hk + b_{32}kl + b_{33}lh]$$

Anisotropic factors in general help minimize local errors, but they may be refined to some unreasonably large values, if there are not enough data to restrain them. Use them judiciously.

The string argument to the command `Scale` specifies a reference frame. The isotropic scale factor a_0 of this frame is fixed as initialized, which has the default value of 1. All other factors a 's and b 's of this frame will be fixed as initialized. All defaults are 0. If no reference is specified, the first frame is assumed to be the reference.

The program initialized a 's and b 's are 0 except that a_0 's are 1. The second numerical argument and the string argument to the command `Scale` function as selectors to the initialized values, but these arguments do not reset these factors. Therefore, these factors can also be initialized to other user-specified values, and the arguments to the command `Scale` choose to fix some of them and to free others. See 5.3 on user-initialized factors.

The subcommand `Isotropy` takes a numerical argument of the same choices -1, 0, 1, or 2, but specific to either scale or temperature factor only as a string argument of `scale` or `temperature` indicates. The subcommand `Isotropy` can be repeated. Another subcommand `Reference` specifies a reference frame (not in Listing 5.2.0.0.1).

5.2.3 λ -curve modeling

There are several subcommands related to λ -curve. `Wavelength` behaves as same as the one in `Input` section. If one numerical argument is given, it is the reference wavelength. If two are given, they are λ_{\min} and λ_{\max} without a specific order. If three are given, the third one is the reference wavelength. The value on the λ -curve equals to 1 at the reference wavelength. If a new wavelength range is given that is different from the existing spectrum, the existing λ -mapping (see below) is invalidated, and a linear mapping takes effect. This should be avoided in general unless

Box 5.2.3.0.1 Chebyshev Polynomial and Approximation

It is a popular way to model a 1-dimensional continuous function within a known interval by Chebyshev approximation. A *Chebyshev polynomial of n degree* is denoted $T_n(x)$. In trigonometric form,

$$T_n(x) = \cos(n \arccos x), -1 \leq x \leq 1$$

Chebyshev polynomials of various degrees can also be written in polynomial form:

$$\begin{aligned} T_0(x) &= 1, \\ T_1(x) &= x, \\ T_2(x) &= 2x^2 - 1, \\ &\dots \\ T_{n+1}(x) &= 2xT_n(x) - T_{n-1}(x), n > 0. \end{aligned}$$

These Chebyshev polynomials are very useful in approximation of functions. Let a function be $f(x)$. It can be shown that

$$f(x) \approx -\frac{c_0}{2} + \sum_{i=0}^{N-1} c_i T_i(x)$$

is exact where N x values satisfy $T_n(x) = 0$, therefore the summation of these polynomials is a *Chebyshev approximation* to the function $f(x)$, if the *Chebyshev coefficients* and *abscissas* are

$$\begin{aligned} c_i &= \frac{2}{N} \sum_{j=0}^{N-1} f(x_j) T_i(x_j), \text{ and} \\ x_j &= \cos \frac{\pi(j+1/2)}{N}, \text{ respectively,} \end{aligned}$$

where N is said to be the *order* of the Chebyshev approximation.

If the function $f(x)$ is defined on interval $[a, b]$ instead of $[-1, 1]$, a mapping function is needed:

$$y \equiv \frac{2x - b - a}{b - a}.$$

Other nonlinear mapping functions may also be used as long as the interval $[a, b]$ is uniquely mapped to $[-1, 1]$.

it is intended. The command `Wavelength` is only needed for the first time before a λ -curve is available. The subsequent scaling often inherits from the previous results. This command may be used to change the reference wavelength, but it should be rare to change wavelength range.

The command `Chebyshev` can be used to specify a maximum order of Chebyshev polynomials (11.1.1.18). λ -curve is modeled by a Chebyshev approximation of a chosen order (Box 5.2.3.0.1). If no maximum order is specified, the program may set a default depending on the complexity of the initial λ -curve. If a second integer less than or equal to the first one is given, this many Chebyshev terms at higher degree are allowed to have frame specific values, that is, each frame may have its own λ -curve. If the second integer is missing, its default is 0, that is, all frames share a single λ -curve. The second integer cannot be greater than the first; or it will be ignored except a warning message.

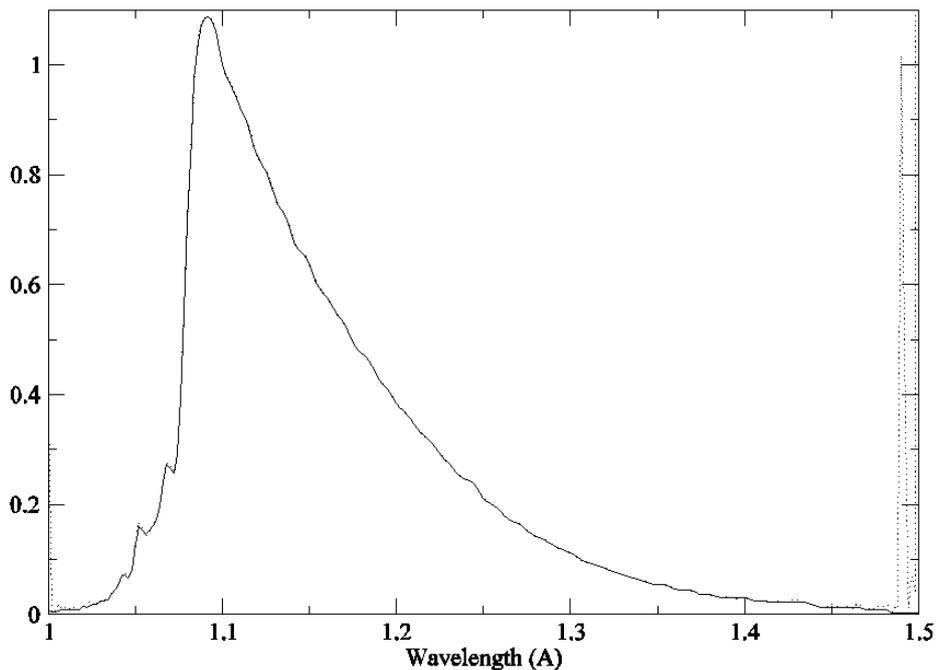


Figure 5.2.3.0.1 λ -curves of 128-term Chebyshev approximation derived by `Epinorm`. The dotted and solid lines are before and after spike removal and both ends, respectively.

An optional string argument of choices `unimodal`, `bimodal`, `arbitrary`, `free`, or `fix` can be given. The first three choices hint the program to find a unimodal, bimodal or arbitrary spectrum, respectively. The program will try to

remove some spiky features at both ends of the derived spectrum, if no string argument is given. An explicit argument `arbitrary` forces the program to leave all spikes unmodified. See Figure 5.2.3.0.1 for an example of spike removal. Option `fix` signals the program not to refine the spectrum, and `free` reverses.

Mapping is a command that takes a string argument of `fix`, `linear`, or `nonlinear` to specify the method of λ -mapping. The linear mapping function shown in Box 5.2.3.0.1 maps λ_{\min} to -1 and λ_{\max} to 1. A strategically designed nonlinear mapping function shown in Figure 5.2.3.0.2 will perform better since larger interval is mapped to those wavelengths where the λ -curve varies sharply or has strong intensity, and only small intervals are mapped to two wings of the λ -curve. For example in Figure 5.2.3.0.2, a large interval (-0.95, 0.42) is mapped to a small wavelength range (0.81, 0.85) where the peak and its rising edge locate. On the other hand, only a very small interval (0.96, 1) maps to half of the entire wavelength range (1, 1.2) where the leveled tail is at long wavelengths. The effect of such nonlinear mapping is to use most of Chebyshev polynomial coefficients to describe the important, central region of the λ -curve, and to leave few coefficients to model flat and potentially noisy wings. The λ -curve shown in Figure 5.2.3.0.1 is before spike removal, that is, not enough parameters are available at the tail to form any spikes, if nonlinear λ -mapping is used.

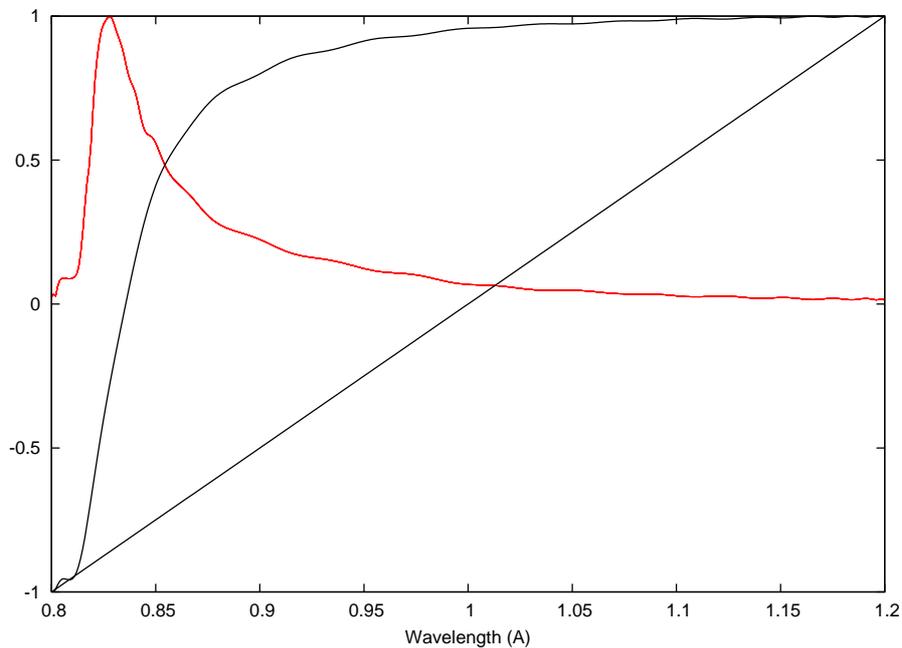


Figure 5.2.3.0.2 λ -curve in red and its λ -mapping function in black. The straight line represents the linear mapping function.

For the first scaling pass, nonlinear λ -mapping is suggested for normally behaved λ -curves. Starting from the second pass of scaling, if more than one pass are needed, fix λ -mapping by specifying `Mapping fix` or deleting the command `Mapping`. The default is `fix`. If the key word `linear` or `nonlinear` is used for each pass of scaling, remapping will take place and destabilize the λ -curve. The program requires no user intervention in decision of the details of a nonlinear mapping function. The key word `nonlinear` is the only user command needed to enter nonlinear mapping.

Two more commands `Lambda-shift` and `Expansion` are related to λ -curve modeling. Both of them take a string argument of `fix` or `free` to signal the program to fix or free two sets of frame-specific parameters from refinement. These parameters correct errors due to geometric refinement (Chapter 3). The outcome of geometric refinement splits into two parts, first, prediction of spot locations on detector that serves integration (Chapter 4), and second, wavelength assignment to all spots and their integrated intensities. The second part is consumed here in wavelength normalization. Errors in the first part are rather obvious, since they may be displayed graphically (see Figures 3.9.0.0.1 through 3.9.0.0.6 for examples), however, errors in wavelength are much harder to detect. Consider a wavelength error modeled by $\mu\lambda + \nu$, the corrected wavelength is:

$$\lambda_{\text{corrected}} = (1 + \mu)\lambda + \nu,$$

where ν is frame-specific λ -shift, a constant within a frame. It can be shown that such constant mis-assignment in wavelength can originate from inconsistently refined geometric parameters from frame to frame, for example, crystal-to-detector distance and pixel size. μ is an isotropic expansion factor of the unit cell. Both correction factors should be small. The reference pattern (5.2.2) has both factors fixed as initialized. The program defaults to all these factors are 0. The λ -shift ν is the most commonly occurring error, therefore the default argument to `Lambda-shift` is `free`. Isotropic expansion factor μ is less likely to be significant. The default argument to `Expansion` is `fix`.

5.2.4 Beam polarization

Command `Polarization` accepts a numerical argument that initializes beam polarization factor bounded between -1 and 1, and a string argument of `fix` or `free`. Both arguments are optional. The default polarization factor is 0.9 and free to refine. This command may only be used to initialize polarization factor, and cannot change the factor if it is restored from a previously saved parameter file (5.4).

5.2.5 Crystal mosaicity

The third numerical argument to the command `Scale` can be 0, 1, or 2. 0 is the default, which indicates that crystal mosaicity will be fixed as initialized. 1 indicates that an overall mosaicity can be refined, and 2 means frame-by-frame mosaicity.

The subcommand `Mosaicity` is easier to use. It may accept one numerical argument as full-width at half-maximum of mosaic spread in degree. It may also take two numerical arguments as spot length and width in pixel, as same as the `Spot` command in `Input` section (see 5.1). This command may only be used to initialize overall mosaicity, and cannot change mosaicity if it is restored from a previously saved parameter file (5.4). However, there is a subtle difference between 1- and 2-argument commands. If only one numerical argument is given, this value can take effect when no other mosaicity value is restored from a previous parameter file (5.4). That is to say, a restored value from previous scaling has the highest priority. It cannot be overridden by a command input. This is to prevent accidental change by users, but users can set mosaicity value if no previous scaling result is loaded, not even 0. If two numerical arguments to the command `Mosaicity` are given, they will be translated into mosaicity only if no scaling parameter file is loaded, that is, a fresh scaling process starting from scratch.

`Mosaicity` may also take a string argument of `fix`, `free`, or `frame` to indicate fixed, free overall, or free frame-specific mosaicity, respectively.

5.2.6 Absorption correction

It is obvious that a λ -curve obtained from the process of wavelength normalization accounts for the total effects of the source spectrum and all absorption by optical elements in the incident beam prior to the sample crystal, including obstacles like air and front wall of sample capillary. It is very appropriate to call this wavelength-dependent correction λ -curve, instead of spectrum. It is less obvious that a λ -curve also corrects an overall effect of absorption by elements around the crystal environment, for example, sample crystal itself, surrounding liquid, flow-cell, cryoloop, capillary, diamond anvil cell, gasket, air, front layer of detector, etc. Why would a λ -curve without special consideration be already capable of correcting a large portion of the seemingly complex absorption? Consider absorption by only one element for the simplicity of the argument. Absorption correction factor

$$f_A = e^{-\mu(\lambda)p(t)},$$

where $\mu(\lambda)$ is linear absorption coefficient as function of wavelength, and $p(\mathbf{t})$ is path length through the absorbing element as function of the orientation of a reflected beam \mathbf{t} . Path length can be rewritten as a constant mean path length and a deviation from the mean as function of orientation:

$$p(\mathbf{t}) = p_0 + \Delta p(\mathbf{t}).$$

Absorption correction factor then becomes a product of two parts:

$$f_A = e^{-\mu(\lambda)p_0} e^{-\mu(\lambda)\Delta p(\mathbf{t})},$$

where the first part is wavelength-dependent only. This part will be automatically corrected by λ -curve. When the range of $\Delta p(\mathbf{t})$ is smaller than p_0 , which is often the case, the most of absorption effect has already been taken care of by λ -curve. What is left uncorrected is the second, orientation-dependent portion. Therefore, absorption correction factor can be redefined as:

$$f_A = e^{-\mu(\lambda)\Delta p(\mathbf{t})},$$

in one element case.

In general, if a reflected beam at orientation \mathbf{t} passes through n types of materials, absorption correction factor can be written as:

$$f_A = \exp\left[-\sum_{i=1}^n \mu_i(\lambda) \Delta p_i(\mathbf{t})\right].$$

In X-ray wavelength range, mass absorption coefficients are roughly proportional to squared wavelength, so that a generalized path length $P(\mathbf{t})$ can be defined independent of wavelength:

$$f_A = \exp[-\lambda^2 P(\mathbf{t})].$$

The generalized path length $P(\mathbf{t})$ is a spherical function or simply a 2-dimensional function in detector space that includes variation of path lengths, densities, and steepness of mass absorption coefficients of all materials involved. Contrasted to

Box 5.2.4.0.1 Absorption Coefficients

Light intensity after an absorbing material is reduced to a fraction of the original intensity before the material.

$$I = I_0 e^{-\mu p},$$

where μ is *linear absorption coefficient*, a function of the light wavelength, and p is the path length through the absorbing material. Path through a longer material is certainly equivalent to path through a denser one. More often, *mass absorption coefficient* μ/d is used.

$$I = I_0 e^{-(\mu/d)p},$$

where d is density of the material. Mass absorption coefficients of various elements and materials can be found at <http://physics.nist.gov/PhysRefData/XrayMassCoef>.

wavelength normalization, absorption correction focuses on the unevenness across the detector space rather than wavelength dependency.

Absorption correction by analytical means as outlined in this section faces a major difficulty, too many parameters. A trial implementation is not yet released in the latest version of `Epinorm`. However, a local scaling procedure behaves as an empirical means of absorption correction. See 5.3 and 5.5 for detail.

5.3 Scaling Modes and Statistics

Laue scaling is a multitask, multi-parameter minimization process. The key is a schedule of cycles in which desired parameters are released to refinement. The schedule attempts to correct the major errors at each stage of the process. This is largely the burden of the program, but users have one choice of scaling modes: initial, global, or frame-by-frame scaling. The scaling cycles can be monitored by reports of statistics.

5.3.1 Scaling modes

If a set of integrated intensities has never been scaled, there is an option to initialize the process in a less error prone way, initial mode, but this is not always necessary. To use the initial mode, specify a string argument `initial` to the command `Scale`, or use the subcommand `Mode` followed by a string argument `initial` (Listing 5.3.1.0.1). Initial mode refines isotropic scale factor and overall λ -curve only. The scaling results can be saved into a file specified by the command `Restore`, and loaded back in for further scaling. See 5.4 for saving and restoring intermediate results.

```
diagnostic    off
busy          off
warning       off

@ m37v3_8us

Input
  Image       initial.lam
  Resolution  2.5 100      # Lower resolution
  Quit

Scale         5          # Use strong observations
  Wavelength 1 1.5 1.1    # Use same bandwidth as in subsequent scaling
  Chebyshev  16         # Lower Chebyshev order
  Restore     m37v3_8us.inp
  Mode        initial
  Quit

Quit
```

Listing 5.3.1.0.1 Command script for an initial scale.

Global mode is the most commonly used, but this mode does not work with frame-specific λ -curves, nor any kind of anisotropic scaling, because there will be too many free parameters to be refined at a same time. Listing 5.2.0.0.1 is a typical example of global scaling. The string argument to the command `Scale` can specify global mode as well. If frame-specific λ -curves and/or anisotropic scaling are specified, the program automatically switches to frame mode, a schedule for frame-by-frame scaling, despite `global` is given.

Frame mode is explicitly specified by command `Mode frame` or implicitly by requesting frame-specific λ -curves and/or any type of anisotropic scaling. This mode of scaling is an intermediate between global and local scaling (see below). It is suitable for large number of frames, say 100, and a lot of parameters. The program runs through several cycles of global scaling, and then switches mode to work on one frame at a time, finally switches back to global scaling. Frame-by-frame scaling releases only those parameters specific to one frame, so that avoids minimization of large number of parameters.

Other than initial, global, and frame modes, there is a test mode solely for diagnostic purpose, which general users shall avoid. Local scaling is a powerful, special mode, which will be discussed together with applying and merging (5.5).

5.3.2 Minimization cycle and statistical report

The minimization process is scheduled in many cycles. Each cycle generates a report like the one listed below. First, a title tells what parameters are refined in this cycle, followed by a section of basic statistics on the data. Data rejection is done automatically based on several criteria. A non-redundant measurement is rejected, since it cannot contribute to the refinement. Data points with large errors are also rejected automatically, but once again, rejected data during scaling may still be included in the final output.

```

=====
Scaling Cycle 4
  Isotropic scale factor
  Overall spectrum
=====
Total measurements: 123021
  Accepted          : 115161 93.6108%
  Rejected          : 7860 6.38915%
Data-to-parameter  : 1251.75
Maximum iteration  : 32
Tolerance          : 0.0001
Chi-square         : 2.8666e+07 3.32024e+07 -4.53632e+06 -13.6626%
R.M.S.D.           : 936.884 1008.29 -71.4082 -7.0821%
Quadratic R-factor: 14.2902%
(Current and previous values, absolute and relative changes)

```

```

|-----)
| Report |
|-----|
|-----|
|-----|
|-----|
|-----|
|-----|

```

```

R-model          = 0.125352
Weighted R-model = 0.117885
R-models calculated from
115161 accepted integrated intensities.
These R-factors indicate how well the integrated
intensities are modeled by the current parameter set.

```

```

          R-merge on F^2 = 0.168612
Weighted R-merge on F^2 = 0.127676
          R-merge on F   = 0.0993019
Weighted R-merge on F   = 0.0804078
R-merges calculated from
115131 accepted integrated intensities of
33883 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) = 10.7618
Mean F   / sigma(F)   = 21.4314
Signal-to-noise ratio calculated from
9333 unique reflections with highly redundant measurements.

```

Resolution range (Å)	Unique refl.	Mean F ² /sigma(F ²)	Mean F/sigma(F)
1000.0000 - 4.7877	235	15.97	31.64
4.7877 - 3.8000	527	19.34	38.30
3.8000 - 3.3196	615	17.18	34.07
3.3196 - 3.0161	601	14.53	28.93
3.0161 - 2.7999	625	12.49	24.85
2.7999 - 2.6348	588	11.54	23.11
2.6348 - 2.5028	590	10.65	21.33
2.5028 - 2.3938	538	9.58	19.16
2.3938 - 2.3017	580	9.95	19.88
2.3017 - 2.2223	569	8.51	17.00
2.2223 - 2.1528	626	8.93	17.83
2.1528 - 2.0912	652	8.12	16.15
2.0912 - 2.0362	657	7.85	15.65
2.0362 - 1.9865	659	7.75	15.41
1.9865 - 1.9413	653	7.62	15.16
1.9413 - 1.9000	618	7.18	14.28

File light.inp is overwritten.

File m37v_1a_004.mar3450.ii.lam is overwritten.

File m37v_1a_006.mar3450.ii.lam is overwritten.

...

File m37v_1b_062.mar3450.ii.lam is overwritten.

Listing 5.3.2.0.1 Statistics report from each cycle of scaling.

Box 5.3.2.0.1 Error Distributions and M-estimates

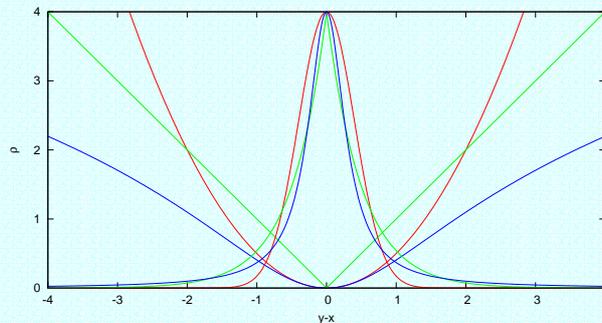
Let $\mathbf{X} = \{x_1, x_2, \dots, x_N\}$ be a set of experimental observations, and $\mathbf{W} = \{w_1, w_2, \dots, w_N\}$ be a set of weights associated with the experimental data. Theoretical calculations of these data points $\mathbf{Y} = \{y_1, y_2, \dots, y_N\}$ can be obtained from a model that contains a parameter set. When the deviation of theoretical calculations in \mathbf{Y} from the observations in \mathbf{X} is minimized, the parameter set is considered to be the best. According to maximum likelihood theory, the

minimization target is $\sum_{i=1}^N \rho(y_i - x_i)$, and the best parameters resulted from the minimization

are called *M-estimates*. ρ is a function of deviation $y_i - x_i$ ($i = 1, 2, \dots, N$) also called error. When the error distribution takes different forms, the function ρ varies. This table lists three commonly occurring error distributions. The figure shows the normalized distributions and the function ρ .

Error distribution	Normal or Gaussian (-)	Double exponential or Laplace (-)	Cauchy or Lorentzian (-)
Probability density	$\frac{1}{\sigma\sqrt{2\pi}} \exp[-\frac{(y-x)^2}{2\sigma^2}]$	$\frac{1}{2b} \exp(-\frac{ y-x }{b})$	$\frac{1}{\pi} \frac{\Gamma/2}{(y-x)^2 + (\Gamma/2)^2}$
Minimization target	$\sum_{i=1}^N w_i (y_i - x_i)^2$	$\sum_{i=1}^N \sqrt{w_i} y_i - x_i $	$\sum_{i=1}^N w_i \log[1 + (y_i - x_i)^2]$
Mean deviation	$\sqrt{\frac{\sum_{i=1}^N w_i (y_i - x_i)^2}{\sum_{i=1}^N w_i}}$	$\frac{\sum_{i=1}^N \sqrt{w_i} y_i - x_i }{\sum_{i=1}^N \sqrt{w_i}}$	$\sqrt{\frac{\sum_{i=1}^N w_i \log[1 + (y_i - x_i)^2]}{\sum_{i=1}^N w_i} - 1}$
R-factor (mean deviation/mean observation)	$\sqrt{\frac{\sum_{i=1}^N w_i (y_i - x_i)^2}{\sum_{i=1}^N w_i x_i^2}}$ (geometric mean deviation/geometric mean observation)	$\frac{\sum_{i=1}^N \sqrt{w_i} y_i - x_i }{\sum_{i=1}^N \sqrt{w_i} x_i }$ (arithmetic mean deviation/arithmetic mean observation)	$\sqrt{\frac{\sum_{i=1}^N w_i \log[1 + (y_i - x_i)^2]}{\sum_{i=1}^N w_i} - 1}$ $\sqrt{\frac{\sum_{i=1}^N w_i \log(1 + x_i^2)}{\sum_{i=1}^N w_i} - 1}$

In case of normal distribution, the minimization target is often called χ^2 , the mean deviation is R.M.S.D., and the process is called *least-squares minimization*. The definitions of mean deviation do not give a relative sense. R-factors, often expressed in percentage, are therefore defined in crystallography to show the error level relative to experimental observation, where R stands for reliability.



$$\chi^2 = \sum_{i=1}^N w_i (\text{obs}I_i - \text{mod}I_i)^2, \text{ and}$$

$$\text{R.M.S.D.} = \sqrt{\frac{\chi^2}{\sum_{i=1}^N w_i}},$$

where the summation is over N accepted data points, and $\text{obs}I$ and $\text{mod}I$ are observed and modeled integrated intensities, respectively. χ^2 and R.M.S.D. measure how well the observed integrated intensities are modeled, but they do not give a relative sense. $R_{\text{quadratic}}$ in least-squares sense and R_{model} in maximum likelihood sense defined below indicates such relative residual of fitting:

$$R_{\text{quadratic}} = \sqrt{\frac{\chi^2}{\sum_{i=1}^N w_i \text{obs}I_i^2}}, \text{ and}$$

$$R_{\text{model}} = \frac{\sum_{i=1}^N w_i |\text{obs}I_i - \text{mod}I_i|}{\sum_{i=1}^N w_i \text{obs}I_i}.$$

χ^2 and closely related R.M.S.D. and $R_{\text{quadratic}}$ are the least-squares minimization target. R_{model} is the maximum likelihood target when absolute deviation is minimized instead of squared deviation. `Epnorm` switches minimization target from early least-squares to the logarithmic target for Lorentzian error distribution at later cycles (Box 5.3.2.0.1).

The R_{merge} , also known as R_{symm} , measures how well the symmetry-related and redundant measurements agree with each other after applying the current correction factors. All these statistics shall improve cycle by cycle, if the refinement is going well. However, you may notice that some R factors may increase slightly. This is due to newly applied data rejection may accept more data point into the refinement while the process converges.

Mean $\langle F^2 \rangle / \sigma(F^2)$ and mean $\langle F \rangle / \sigma(F)$ are meant to be objective measures of signal-to-noise ratio. The sample standard deviation σ is calculated from redundant observations of at least 4 times, so that σ indicates how much the redundant measurements deviate from each other, and therefore is an objective lower bound of the real noise content. Each redundantly measured reflection has $\langle F^2 \rangle / \sigma(F^2)$ and $\langle F \rangle / \sigma(F)$, two measures of signal-to-noise level, where $\langle F^2 \rangle$ and $\langle F \rangle$ are averages within the reflection. The mean value of the entire dataset or a resolution bin is an overall signal-to-noise index.

5.4 Saving and Restoring Scaling Results

Results of the minimization process include isotropic and anisotropic scale and temperature factors, λ -curves, etc.

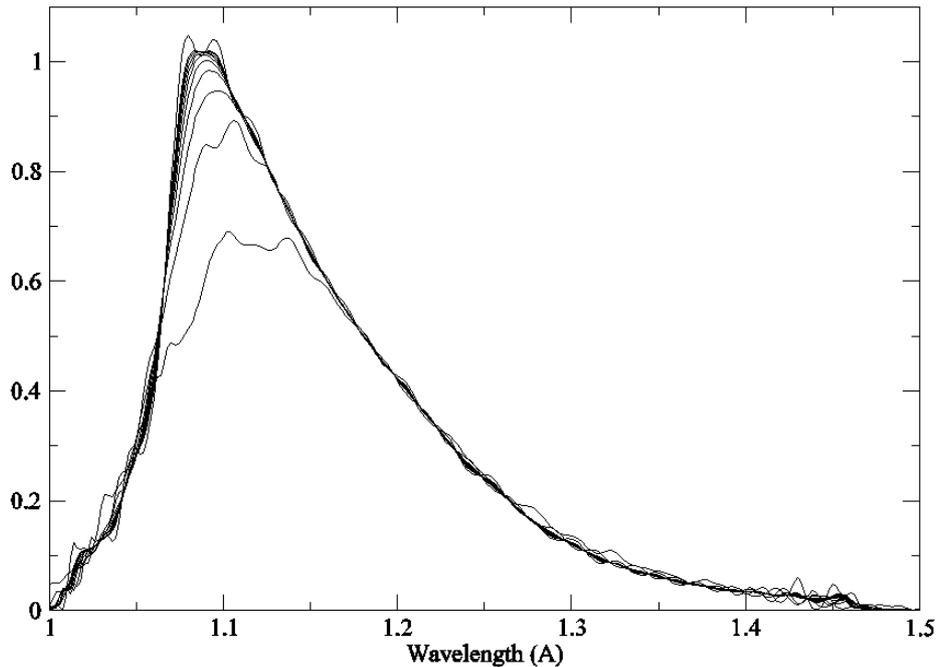


Figure 5.4.1.0.1 λ -curves of 64-term Chebyshev approximation. These curves from the lowest are at Bragg angles 1, 2, ..., 10 degrees, respectively. The top curve corresponds to 0 mosaicity.

5.4.1 λ -curves

If frame-specific λ -curves are refined, or if isotropic geometric error correction factors or frame-specific mosaicities are nonzero, frame-specific λ -curves are saved after each cycle of refinement that alters the λ -curves. A series of filenames are automatically selected as the `.ii` filenames suffixed with `.lam`. Frame-specific and overall λ -curves can be saved into files by using the explicit top level command `Lambda` after `Scale` section, which takes a string argument as the filename for the overall λ -curve. The first section of each saved file has two columns, wavelength in Å and relative intensity. When crystal mosaicity is nonzero, there will be other sections that have an additional column indicating the Bragg angle in degree (11.8.4). That is to say, the traditional ‘ λ -curve’ is in fact no longer a curve, instead, it is a surface. Reflections occur at a same wavelength

but different Bragg angles receive different normalization factors due to crystal mosaicity. An example of λ -curve is shown in Figure 5.4.1.0.1.

The command `Lambda` may also be used after restoring previously saved parameters (see below) without running refinement. To save a λ -curve defined by a parameter file, simply load the parameter file and run command `Lambda` followed by a filename.

5.4.2 Scaling parameter file

Intermediate results of the refined parameters can be saved into file and restored for further refinement. This feature is useful for exploring the effect of various details in λ -curve, crystal mosaicity and anisotropic scaling. One may start with less free parameters, save intermediate results, and carry on to more and more aggressive parameterization. A same command both at the top level and in `Scale` section `Restore` performs the function of saving and restoring.

`Restore` with a string argument specifies a filename to which intermediate results will be saved during iterations of scaling and after it is finished. Obviously, this command must be given before `Scale` command or in `Scale` section. The saved file is in the legal format ready to be loaded back in by `Epinorm`. It includes all scaling parameters that exactly define each and every correction.

The command `Restore` without argument enters a deeper level of the menu until command `Quit` exits back to the top level. In a `Restore` session, various commands are used to restore parameters as shown below.

```
Restore
Polarization      0.54397

Mosaicity         0.01856
Mosaicity         0.00000 m37v_1a_001.mar3450.ii
Mosaicity         0.00000 m37v_1a_003.mar3450.ii
...
Mosaicity         0.00000 m37v_1a_061.mar3450.ii

Scale             1.00000 m37v_1a_001.mar3450.ii
Scale             0.75081 m37v_1a_003.mar3450.ii
...
Scale             0.80385 m37v_1a_061.mar3450.ii

Temperature      0.00000 m37v_1a_001.mar3450.ii
Temperature      2.07864 m37v_1a_003.mar3450.ii
...
Temperature      1.81860 m37v_1a_061.mar3450.ii

Expansion        0.00000000 m37v_1a_001.mar3450.ii
Expansion        0.00000000 m37v_1a_003.mar3450.ii
...
Expansion        0.00000000 m37v_1a_061.mar3450.ii
```

```

Lambda-shift      0.00000000 m37v_1a_001.mar3450.ii
Lambda-shift      0.00033828 m37v_1a_003.mar3450.ii
...
Lambda-shift      -0.00012503 m37v_1a_061.mar3450.ii

AnisoScale        0.00000  0.00000  0.00000  0.00000  0.00000  0.00000
0.00000  0.00000  0.00000 m37v_1a_001.mar3450.ii
AnisoScale        0.00000  0.00000  0.00000  0.00000  0.00000  0.00000
0.00000  0.00000  0.00000 m37v_1a_003.mar3450.ii
...
AnisoScale        0.00000  0.00000  0.00000  0.00000  0.00000  0.00000
0.00000  0.00000  0.00000 m37v_1a_061.mar3450.ii

AnisoTemperature  0.00000  0.00000  0.00000  0.00000  0.00000  0.00000
0.00000  0.00000  0.00000 m37v_1a_001.mar3450.ii
AnisoTemperature  0.00000  0.00000  0.00000  0.00000  0.00000  0.00000
0.00000  0.00000  0.00000 m37v_1a_003.mar3450.ii
...
AnisoTemperature  0.00000  0.00000  0.00000  0.00000  0.00000  0.00000
0.00000  0.00000  0.00000 m37v_1a_061.mar3450.ii

Wavelength        1 1.5 1.1

Normal            0  1.13846
Normal            1  0.74865
...
Normal            63  0.00290

Coefficient        0  0.660381
Coefficient        1 -0.14187
Coefficient        2 -0.221711
...
Coefficient        63  0.0024312

Coefficient        0  0.660381 m37v_1a_001.mar3450.ii
Coefficient        1 -0.14187 m37v_1a_001.mar3450.ii
Coefficient        2 -0.221711 m37v_1a_001.mar3450.ii
...
Coefficient        63  0.0024312 m37v_1a_001.mar3450.ii

...

Quit

```

Listing 5.4.2.0.1 Refined parameters saved in a file.

The command `Polarization` takes a fractional argument as the beam polarization factor (11.2.2.2).

Command `Mosaicity` takes a numerical and an optional string argument. If no string is given, the number is taken as the isotropic mosaicity for all frames in full-width at half-maximum and in degree. This is called overall mosaicity. If a string of `.ii` filename is given, the number before the string is an additional value to be added to the overall mosaicity, which results in the mosaicity of this frame (11.2.2.3). If no overall mosaicity is restored, the value given by the single argument to subcommand `Mosaicity` in `Scale` section will take over (see 5.2.5).

Commands `Scale` (11.2.2.4) and `Temperature` (11.2.2.5) take a numerical argument and a string argument, which are scale and temperature factors for the given frame, respectively.

Commands `Expansion` (11.2.2.6) and `Lambda-shift` (11.2.2.7) take a numerical argument and a string argument, which are isotropic expansion factor and λ -shift for the given frame, respectively.

Commands `AnisoScale` (11.2.2.8) and `AnisoTemperature` (11.2.2.9) take 9 numerical and 1 string arguments. They are the anisotropic scale and temperature factors for the given frame.

The filenames of `.ii` files should be consistent. Any inconsistency would cause missing parameters for some frames. Missing parameters are filled with defaults.

Command `Wavelength` is as same as the one in `Input` section, which take three numerical values as λ_{\min} , λ_{\max} , and λ_{ref} in Å. See 2.1.7, 11.1.1.17 and 11.2.2.10 for detail.

Command `Normal` may take one, two or three numerical arguments (11.2.2.11). These values are defined exactly same as those numerical arguments to the command `Coefficient` described below, except that the wavelength-dependent function defined by the Chebyshev approximation in this case is a λ -mapping function, which normalizes the wavelength range given by the command `Wavelength` to the interval $[-1, 1]$. See Figure 5.2.3.0.2 for example of a λ -mapping function. `Epinorm` saves λ -mapping functions in 2-argument format. The other formats have higher priority, which offer an opportunity of user-supplied or user-modified mapping function. See below for more. If no command `Normal` is given, the default is linear mapping (Figure 5.2.3.0.2).

Command `Normal` may accept an optional string argument `remap`, `Remap`, or `REMAP`. If one of these strings is given, λ -curve will be remapped using the new mapping function defined by this command `Normal`. It is the string argument `remap` that distinguishes the old and new mapping functions, so it must be given to all `Normal` commands that specifies the new mapping function. Commands without `remap` argument define the old mapping function.

Command `Coefficient` may take one, two or three numerical arguments and an optional string argument. If two numbers are given, the first is taken as an integer, the second is a Chebyshev coefficient of the term specified by the integer. This command should usually repeat for 0 through $n - 1$, where n is the maximum degree of the Chebyshev approximation to the λ -curve. The greatest first argument defines the order of the approximation. Missing terms within the maximum degree will be assumed as 0 coefficient. The order of 2-argument

commands is not enforced, however it is recommended to use ascending order of the integer argument. `Epinorm` saves λ -curve in this format.

If only one number is given to the command `Coefficient`, this value is taken as a function value of λ -curve instead of a Chebyshev coefficient. This command should repeat as well. The number of repeats defines the order of the Chebyshev approximation. The function values locate at Chebyshev abscissas in ascending order (Box 5.2.3.0.1).

Commands with different number of arguments cannot be used together, that is to say, λ -curve is defined either by Chebyshev coefficients or function values at specific abscissas. The latter is very similar to splines. If more than one format are mixed, the 1-argument command `Coefficient` has higher priority, and all coefficients given by 2-argument command are discarded regardless of order.

If three numerical arguments are given to command `Coefficient`, the first is taken as an integer, which specifies the order of Chebyshev approximation. The second is wavelength in Å and the third is a function value at the wavelength. This command shall also repeat for different wavelengths in ascending order. If the first argument to a later command specifies a different value from the previous one, the later one overwrites the earlier ones. The 3-argument format is more convenient than the 2-argument one to supply a user-defined or -modified λ -curve. The 3-argument format has the highest priority in case it is mixed with other formats.

If a string argument is given, the coefficient or the function value given with the string will be used to construct a λ -curve specific to a frame named by the string. If no literal string is given, the λ -curve represents an overall spectrum.

This section of multiple commands of `Normal` and `Coefficient` should not be changed by users under normal circumstances unless one really knows what he/she is doing. Changing these values means altering λ -curve. Changing mapping-function may dramatically reshape λ -curve.

Loading back a previously saved file is as easy as loading a geometry `.inp` file, that is, to use an at sign (@) followed by the filename before or in `Scale` section. Using this file of scaling parameters, one may also set initial parameters different from the default values before the first scaling.

```
...
@ pass1.inp
Scale      1000
  Restore  pass2.inp
...
```

Listing 5.4.2.0.2 Restoring previous parameters and saving into another file.

Combined use of the different usages, it is possible to load from one file and save into another. If a same filename is used, it will be updated by the latest parameters.

5.5 Local Scaling and Applying Scale Factors

All scaling modes except one, local scaling, are discussed in 5.3.1, since local scaling is of very different nature. It is closely related to applying of correction factors and data merging. It must have a good foundation of wavelength normalization and normal scaling. Local scaling is designed to minimize residual errors that previous scaling is unable to correct for known or unknown reasons. Local scaling and applying of scale factors must be done simultaneously.

5.5.1 Local scaling

The scaling mode `local` can be given as a string argument to the command `Scale` or by using `Mode local` in `Scale` section. In case of local scaling, no other subcommands in `Scale` section are useful except `Sigma`. The following script gives a concise example, in which σ -cut is specified by the first numerical argument to command `Scale`, and the scaling mode is given as the string argument. It should be obvious that all geometric and previous scaling parameters are necessary.

```
diagnostic    off
busy          off
warning       off

@ pyp24a_dark # geometric parameters
@ dark.inp    # scaling parameters

Scale        1 local
Quit

Apply        1 pyp24a_dark.hkl
Quit
```

Listing 5.5.1.0.1 Command script for local scaling and applying.

Like the other scaling modes, local scaling is also scheduled in cycles. Users have no control on the scheduling. Statistics are reported after each cycle (Listing 5.3.2.0.1). Local scaling results in one scale factor for each observation of single reflection. Due to large number of local scaling factors, a design decision was made not to save local scaling factors into a file, except for diagnostics purpose. Therefore, unlike the other scaling modes, local scaling cannot be done in multiple passes, and must be performed on-the-fly. Local scaling factors must be consumed immediately in `Apply` session following local scaling.

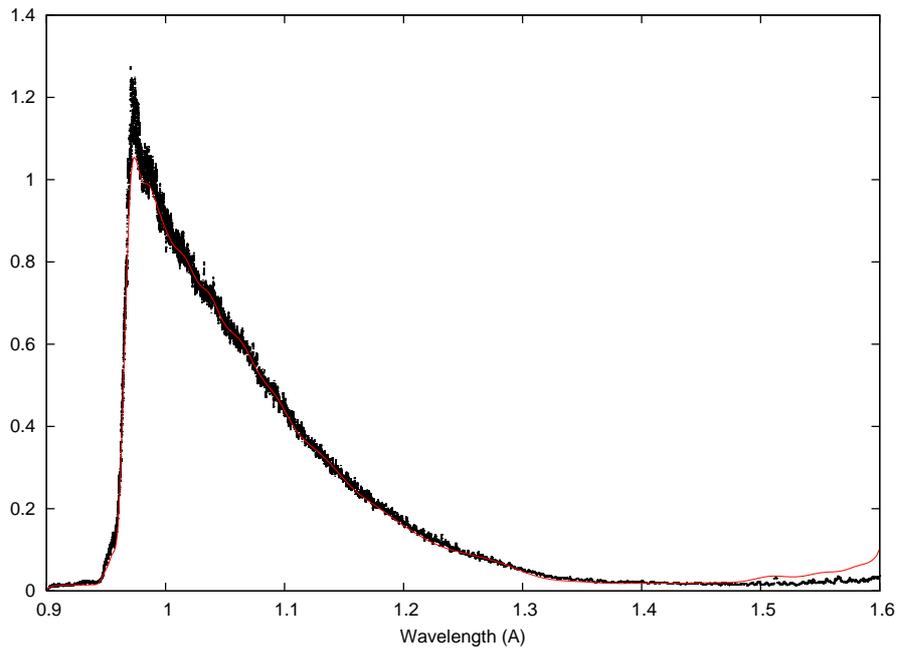


Figure 5.5.1.0.1 λ -curves before (red) and after (black) local scaling.

Local scaling further enhances λ -curve. Extremely sharp and detailed features of λ -curve may show after local scaling. Nevertheless, local scaling also corrects other errors related to resolution, Bragg angle, integrated intensity, location on detector, etc.

5.5.2 Applying scale factors to singles

Applying of scale factors can be done either with or without local scaling. All local scaling factors are reset to 1, if no local scaling is performed, so that only the previous scaling results are applied.

Applying may even follow the normal scaling session, so that normal, local scaling and data merging are carried out in a single job. However, it is recommended to scale the data once, and to apply the scale factors in many different ways afterwards. It should be obvious that all the frames loaded for applying of scale factors and data merging must be previously scaled, and their scaling parameters must be restored. A fatal error will occur otherwise.

Before applying correction factors and merging redundant data, new resolution and wavelength ranges can be specified so that noisy portions of the data set can be excluded. However, new ranges do not work with local scaling. They will be ignored if local scaling mode is selected.

The flag for anomalous scattering can be switched on, if it is desired to consider anomalous signal. The default state of anomalous scattering is off. If it is switched on, Friedel pairs will be separated in output. See the end of 5.1 for more discussion.

```
diagnostic    off
busy          off
warning       off
prompt        off
result        off
@ m37v3_8us_002.mar3450.inp
@ m37v3_8us_004.mar3450.inp
...
@ m37v3_8us_062.mar3450.inp
prompt        on
result        on

@ m37v3_8us.inp

Input
  Resolution 2.1 100
  Wavelength 1 1.5
  Anomalous  off
  Spot       8 6
  Quit

Apply      1  m37v3_8us.hkl
Quit
```

Listing 5.5.2.0.1 Command script for applying correction factors and data merging.

`Apply` is the command that applies all correction factors and merges symmetry-related and redundant data. Please note that the wavelength normalization factor is NOT directly derived from the top λ -curve in Figure 5.4.0.0.1, rather from one of the others depending on the Bragg angle of this reflection and mosaicity of the crystal. The command `Apply` takes three optional arguments, two numbers and one string. The first number specifies a σ -cut value used when data are merged. This value is usually smaller than or equal to that to the command `Scale`, so that more data can be accepted in output. The R factors and signal-to-noise ratios will be reported again. These statistics are calculated from all data passed the new σ -cut, and show the quality of the output. The report is similar to Listing 5.3.2.0.1.

The second number can be -1, 0, 1, 2, or 3, specifying the mode of applying. 0 signals the program to apply scale factors only without merging redundant and symmetry-related measurements. Therefore, the outcome is scaled intensities, which are proportional to the squared values of structure factor amplitudes. This operation can only apply to singles, since the scale factors for overlapping multiples remain unknown prior to deconvolution. The results will be saved into a 7-column, CPL reflection file. The columns are h , k , l , λ , $\sigma(\lambda)$, scaled intensity, and its uncertainty. This file is usually called `.edi` file for energy-dispersive intensity (11.8.7).

If the second numerical argument is -1, the program behaves the same except that the energy-dispersive intensities are saved into multiple files with the `.ii` filename suffixed by `.edi`. This helps to trace the source of each measurement.

If the second numerical argument is 1, only singles are scaled and merged. The result is a set of unique structure factor amplitudes in a 7-column, CPL reflection file format. The columns are h , k , l , F_+ , $\sigma(F_+)$, F_- , and $\sigma(F_-)$. Missing Friedel mate is filled by $F = 0$ and $\sigma(F) = 1$. If anomalous flag is off, Friedel mates are merged.

The other options involve harmonic and/or spatial deconvolution, which will be explained in the following sections. The default is 3, the maximum degree of deconvolution.

The string argument is a filename including an absolute or relative path, to which file the scaled and/or merged data will be saved. If no filename is given, only statistics are reported and the results will be discarded.

It is possible to repeat the command `Apply` with different σ -cuts and filenames to generate a series of output. You may then exam the effect of σ -cut in terms of completeness and signal level. For better completeness, use σ -cut of 1 or even less. For better data quality, use higher σ -cut. Unfortunately, repeating `Apply` commands cannot work with a single pass of local scaling. See 5.5.3 for more.

5.5.3 Data merging from a subset

Sometimes, it is desirable to merge a subset of all frames scaled. For example, in time-resolved work, dark and light data from a same crystal may be scaled together, but dark and light data must be separated during merging. It is first necessary to execute the command `Image` in `Input` section with a special argument `reset`. This command clears all previous frames loaded. Then reload the `.inp` files in a desired subset and apply scale factors. Repeat to merge other subsets, if any. If there are any frames not scaled, a fatal error will occur, since the corresponding scale factors are unknown.

```
...
Input
  Image      reset
  Quit
@ m37v_1a_001.mar3450.inp
@ m37v_1a_003.mar3450.inp
...
@ m37v_1a_061.mar3450.inp
Apply      1  dark.hkl

Input
  Image      reset
  Quit
@ m37v_1b_001.mar3450.inp
```

```
@ m37v_1b_003.mar3450.inp
...
@ m37v_1b_061.mar3450.inp
Apply      1    light.hkl
...
```

Listing 5.5.3.0.1 A section of command script to save two subsets of data.

Unfortunately, this style of applying cannot work with local scaling. It is recommend to combine normal scaling within an entire dataset, but to separate local scaling from subset to subset. The following example shows local scaling and data merging for a dark set, but using the scaling parameter file from a combined normal scaling of dark and light data. It must be repeated for light set.

```
diagnostic    off
busy          off
warning       off

@ pyp24a_dark # geometric parameters for dark only
@ pyp24a.inp  # scaling parameters for dark and light

Scale        1 local
Quit

Apply        1 pyp24a_dark.hkl
Quit
```

Listing 5.5.3.0.2 Command script for local scaling and applying of a subset.

If the command `Apply` is repeated for different σ -cut values, local scaling factors will only apply once. Due to the nature of local scaling, an exactly same body of data must be loaded for both local scaling and applying. Local scaling at one σ -cut value and applying at another, higher or lower, violates this rule and will result in reset of all local scale factors. Repeat the entire job for various σ -cut values, if needed.

5.6 Harmonic Deconvolution

This and next sections discuss two special aspects of Laue data reduction, deconvolution of harmonic overlap, also known as energy overlap, and deconvolution of spatial overlap.

Merging of redundant and symmetry-related observations is relatively straightforward. Harmonic deconvolution algorithm described by Ren et al. (*J. Appl. Cryst.* **28**, 482-93, 1995) is essentially a least-squares fitting of an equation system setup from redundant and symmetry-related observations. Harmonic deconvolution replaces the role of applying scale factors and merging redundant and symmetry-related observations, but in a more complex fashion. Therefore, there is no extra command for this process, only a different numerical option to the command `Apply`, 2. The result is a set of structure factor amplitudes from singles and deconvoluted multiples. All features discussed in

several previous sections function in the same way that they do for multiples as for singles.

There is no exception to local scaling. It can be combined with harmonic deconvolution. Local scaling results in m factors for each observation of a multiple spot with multiplicity m . The local scale factors are used in the equation system of deconvolution. Local scaling not only improves merging of singles, but also improves harmonic deconvolution.

```

|-----)
| Report |
| -----|
| -----|
| -----|
| -----|
|-----|

```

```

R-model = 0.166767
Weighted R-model = 0.190516
R-models calculated from
65145 accepted integrated intensities of
3905 unique prime reflections.
These R-factors indicate how well the energy overlapped
integrated intensities are deconvoluted into the current
set of harmonic reflections. Unlike singles, there is no
R-merge or R-symm can be defined for multiples.

35235 reflections in single and multiple merged amplitude set.
10209 harmonic reflections deconvoluted.
7792, 22.1144% accepted from harmonic deconvolution.
1948, 5.52859% never observed as singles.

```

Listing 5.6.0.0.1 A report generated by harmonic deconvolution.

This part of `Epiform` is not just a reimplementation of `LaueView`. Crystal mosaicity, energy resolution of multiple spots, partial reflection are also completely modeled in the process of harmonic deconvolution, in addition to local scaling. For example, if two harmonic reflections overlap at (d, λ) and $(2d, 2\lambda)$. They have the exact same Bragg angle. The reflection at low resolution $2d$ has twice the wavelength span as the one at high resolution d (see the equation at the beginning of this chapter). Their wavelength normalization factors are not simply derived from a λ -curve, but two different integrations over corresponding ranges. Further more, one of them could be a partial reflection, say λ is just beyond the short wavelength boundary, while the other, 2λ , could be well in middle of the wavelength range. Consideration of these factors caused by crystal mosaicity makes the equation system of deconvolution better established.

5.7 Extremely Close Spatial Overlap

Spatial overlaps are far more severe in Laue diffraction than in monochromatic oscillation. A first attempt of resolving these overlaps is made during spot integration (Chapter 4). The integration modes `fixedElliptical` and

variableElliptical are able to isolate some lightly overlapping spots. Profile fitting methods, especially the analytical profile fitting, make serious attempts to separate recorded intensity of each pixel into several overlapping spots. See Chapter 4 for detail. Nevertheless, deconvolution of extremely close spatial overlaps (See Figures 4.6.0.0.1 and 4.8.0.0.1 for example) will be inevitably influenced by detector noise and by accuracy of geometry refinement. The algorithm of harmonic deconvolution is implemented once again with modification to make second attempt on deconvolution of extremely close spatial overlaps. Harmonic overlap occurs on an exact radial line in reciprocal space, however, spatial overlap does within a small solid angle. A second, redistribution of integrated intensity in a local area on detector based on symmetry equivalency and redundancy greatly improves the quality of deconvolution of extremely close spatial overlaps. Unlike the first attempt during profile fitting, this technique of spatial overlap deconvolution does not have a limit on how close the spatial overlap occurs. This technique is proven capable to resolve exact overlap, such as the situation occurs in energy overlap. Once again, all special effects due to energy partial are fully considered in this deconvolution. Local scaling can also be combined into this deconvolution, and helps.

Same as harmonic deconvolution, no extra command is needed for spatial overlap deconvolution. Set the second numerical argument of command Apply to 3. Singles, multiples, and spatially overlapping measurements will be scaled, deconvoluted, and merged in the result. This maximum degree of deconvolution is the program default. An additional user control is the overall or typical spot size, which helps to define extremely close spatial overlaps. Check 2.4 for the command Spot in Input section.

```
|_____)
| Report |
| -----|
| -----|
| -----|
| -----|
| -----|
|_____|
```

```
Symmetry-related cases merged.
500 spatial overlap cases total.
15104 integrated intensities total.
18524 records total.
```

```
R-model          = 0.149931
Weighted R-model = 0.179691
R-models calculated from
12550 accepted integrated intensities of
499 cases of spatial overlap deconvolution.
These R-factors indicate how well the spatially
overlapping integrated intensities are deconvoluted.
Unlike singles, there is no R-merge or R-symm can be
defined for overlapping reflections.
```

```
35323 reflections in merged amplitude set.
1624 spatial overlapping reflections deconvoluted.
697, 1.97322% accepted from spatial overlap deconvolution.
88, 0.249129% never observed as isolated spots.
```

Listing 5.7.0.0.1 A report generated by spatial overlap deconvolution.

This report shows the basic statistics of spatial overlap deconvolution, such as how many cases occurred in the data set and how well they are deconvoluted. The R_{model} 's are usually higher than those in harmonic deconvolution, since harmonic overlaps occur largely at low resolution, but spatial overlaps populate heavily at high resolution.

CHAPTER 11

Reference**11.1 Precognition_E×.×.×_P×.×****precognition_E×.×.×_P×.×****Precognition****precognition**

This is the command-driven version of `Precognition`. It prints the release number and credits, imports CPL, the supporting library, and prints CPL version number. If appropriate license is found, a main menu is printed, otherwise, the program terminates with an error message. The main functionalities of this program are indexing, geometric refinement, spot profile fitting, integration, and spatial overlap deconvolution. `E×.×.×` indicates edition, version, release, and patch numbers. `P×.×` is platform or operating system and its version.

Each menu begins with a colon-separated (`:`) path that indicates the level of the menu. The same path is printed again at the end of the menu, and serves as a prompt. All activated menu items are listed in the second column. Disabled items are not shown, but they may be activated when conditions are met. The full string of a menu item is the command to execute the item. An all-lower-case command is also valid. An abbreviated single letter listed in the third column functions the same way. The single letter is not case sensitive, and is in parentheses following each section title in this chapter. The fourth column includes some brief instruction on the usage of the menu item.

```
Precognition: Input      I: File & keyboard input (image & parameters)
                Quit      Q: Exit Precognition
Precognition:
```

Listing 11.1.0.0.1 A menu.

At any level of the program, if the non-space leading character is an at sign (`@`) or a less than sign (`<`), the remaining command is considered as a filename of a preprogrammed command script. This script will be loaded in, if found. All commands in the script will be executed one after another. If the end-of-file is reached, the control will return to keyboard or previous control. If the program is told to terminate inside the script, the program will exit without execution of the rest of the commands. Any unknown command in the script will cause a fatal error and immediate termination of the program. A command script can be loaded and executed when the program is launched by giving the script filename as an argument to the program.

At any level of the program, if the non-space leading character is a left (() or right () parenthesis, an exclamation mark (!), a number sign (#), or a semicolon (;), this line is considered as a comment, and will cause no action.

At any level of the program, if the non-space leading character is a dollar sign (\$), a percent sign (%), or a greater than sign (>), the rest of the command will be passed to the shell where the program is launched. This feature is system-dependent.

11.1.1 Precognition:Input (I|i)

Enters Input submenu. This submenu is responsible for taking many data and control parameters.

11.1.1.1 Precognition:Input:Format (F|f)

Accepts a literal string that identifies image file format and detector type. The string is case sensitive. Unrecognized string or mismatching the string with an image file will cause error while loading an image, but not always a warning or error message. This command activates and deactivates other menu items that are related to the detector type, therefore, this command shall be placed prior to other detector-related commands.

The current version supports a number of formats listed in Table 11.1.1.1.1. Additional formats can be implemented upon request.

Format code	Detector	Type
Bas2000, FujiImagingPlate	Fuji imaging plate Bass2000 scanner 100 μm	Flat
ESRF, ESRFImageIntensifiedCCD	ESRF image-intensified CCD	Flat
LADI	LADI neutron imaging plate 200 μm	Cylindrical
Mar345, MAR3450Packed	Mar345 packed 100 μm	Flat
MarCCD, MARCCD165	Mar CCD 165 80 μm	Flat
Quantum315bin2x2	ADSC Quantum-315 2×2 binned	Flat
Quantum4	ADSC Quantum-4	Flat
RigakuRaxisIPDSCQ	Rigaku R-AXIS IP scanner DSC-Q	Cylindrical

Table 11.1.1.1.1. Supported detector formats.

If no string is given, this command enters a submenu for format selection.

Precognition:Input:Format:Bas2000 (B|b)
Precognition:Input:Format:ESRF (E|e)
Precognition:Input:Format:LADI (L|l)
Precognition:Input:Format:Mar345 (M|m)
Precognition:Input:Format:MarCCD (C|c)
Precognition:Input:Format:Quantum315bin2x2 (9)
Precognition:Input:Format:Quantum4 (4)
Precognition:Input:Format:RigakuRaxisIPDSCQ (R|r)
Selects the corresponding format.

11.1.1.2 **Precognition:Input:Image (I|i)**

Loads an image file. This command expects a string as a filename of the image to be loaded. The filename may include a relative or absolute path. This command must be executed after a detector/file format has been specified by command `Format`, or an error message of unknown format will be printed. The image to be loaded must be consistent with the format. If no string follows the command, the program enters file selection menu. An optional numerical argument of choices 0 or 1 can be given. If 0 is given, the image will not be loaded except that the filename will be added to a list. 1 is the default. If a string argument of `reset`, `Reset`, or `RESET` is given, the list of previously loaded images or filenames is cleared. This feature is used to merge a subset of data.

This command deactivates many crystal, detector, and goniometer related commands, so that it closes a session.

This command may also load a file of goniometer settings or λ -curve, if the filename has an extension of `.gon` or `.lam`, respectively. See goniometer file and `.lam` file for details. This feature and goniometer file are being deprecated.

11.1.1.3 **Precognition:Input:Crystal (X|x)**

Loads crystal information such as unit cell constants and space group. If a filename of crystal information follows this command, cell constants and space group can be loaded from the file. See crystal information file for detail. If no argument to this command, it enters the file selection menu for crystal information file. If one numerical argument follows, it is considered as a space group number. Valid numbers are integers 1 through 230. Invalid values will cause a fatal error. If six numerical arguments are given, they are taken as cell constants a , b , c in Å and α , β , γ in degree. If seven numerical arguments are given, they are cell constants and space group number. If the string argument is `free`, `Free`, `FREE`, `freed`, `Freed`, or `FREED`, six numerical arguments must also be given. They are considered as uncertainties of cell constants. If no or fewer than six numerical arguments, the string argument is considered as a

filename of crystal information. If the string argument is `fix`, `Fix`, `FIX`, `fixed`, `Fixed`, or `FIXED`, all uncertainties of cell constants will be set to 0. If six or seven numerical arguments are also given, they are cell constants and space group number. If one numerical argument is given, it is space group number.

11.1.1.4 `Precognition:Input:Omega (O|o)`

Takes two numerical arguments as polar angles Ψ and Φ in degree. See Box 3.2.0.0.1 for definition of polar angles. These polar angles define the orientation of ω -axis of the simulated goniometer `ccl::goniometer`. See 3.2 for more information. If no or fewer than two arguments is given, the program prompts for polar angles.

11.1.1.5 `Precognition:Input:Goniometer (G|g)`

Takes three numerical arguments as ω , χ , and ϕ settings in degree. See 3.2 for details on the simulated goniometer. If no argument is given, the program prompts for goniometer settings. If fewer than three arguments are given, the missing values are assumed to be 0.

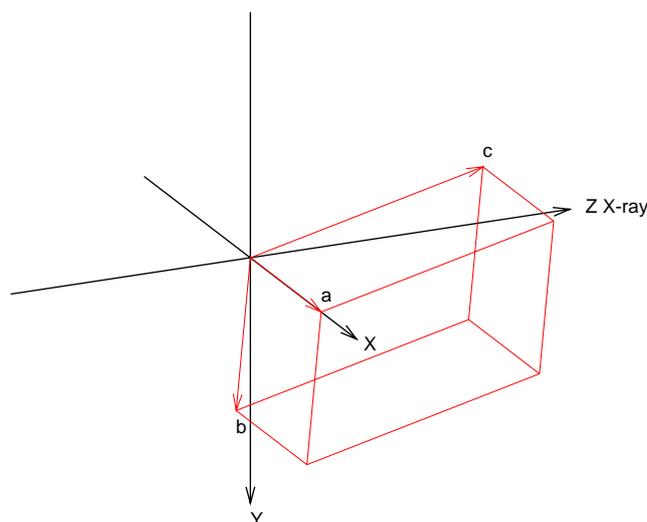


Figure 11.1.1.6.1 Aligned crystal orientation. $\mathbf{a} \parallel \mathbf{x}$ and $\mathbf{a} \times \mathbf{b} \parallel \mathbf{z}$.

11.1.1.6 `Precognition:Input:Matrix (M|m)`

Takes nine numerical arguments as missetting matrix by row. The matrix must be orthogonal. If not, the matrix will be rejected with an error message. If fewer than nine numerical arguments are given, this command is ignored. Missetting matrix defines the rotation from an aligned crystal orientation to any other orientation. The aligned crystal orientation is defined as the following: \mathbf{a} - \mathbf{b} plane is normal to X-ray beam,

that is, $a \times b$ or c^* is along X-ray beam or Z-axis. a is horizontal to the right when looking along the X-ray beam, that is, X-axis.

11.1.1.7 `Precognition:Input:Spot (L|l)`

Specifies overall spot size and signal-to-noise ratio. If no argument is given, the program prompts for overall spot length, width and σ -cut. If one numerical argument is given, it is the σ -cut. If two are given, they are spot length and width in pixel. If three are given, they are spot length, width, and σ -cut, respectively. Spot size and σ -cut are used to assist spot recognition and many other procedures where an estimate of error level is required. One shall use the typical spot size on an image or a set of images, where 'typical' means the greatest population.

The program default length and width are 10 pixels. The program default σ -cut is 3.

This command may also take a string argument as a filename of a spot file (see 11.8.5 for format). The spots may be loaded in to serve as nodal spots in indexing, to be used in ellipse recognition.

11.1.1.8 `Precognition:Input:Distance (D|d)`

Takes one numerical argument as the crystal-to-detector distance in mm. In case of inclined detector setting, this value is the normal distance. In case of normal detector in back reflection position, this value is still positive. 2θ angle is used to distinguish the difference between forward and backward position. This command is deactivated if cylindrical detector is selected.

If no argument is given, the program prompts for distance. If two numerical arguments are given, the first is distance and the second is its uncertainty in mm. If a string argument of `free`, `Free`, `FREE`, `freed`, `Freed`, or `FREED` is given, a numerical argument must also be available to specify an uncertainty of distance in mm. String argument `free` without a number is ignored. An uncertainty value must be non-negative, or it will be ignored. If a string argument of `fix`, `Fix`, `FIX`, `fixed`, `Fixed`, or `FIXED` is given, the uncertainty of distance is reset to 0. A numerical argument with a string `fix` is considered a distance value rather than uncertainty. The default uncertainty of distance is 0.

11.1.1.9 `Precognition:Input:Radius (D|d)`

This command is the same as `Distance` (11.1.1.8), except that it is only activated when a cylindrical detector is selected.

11.1.1.10 Precognition:Input:Center (C|c)

Takes two numerical arguments as the coordinates of direct-beam center in pixel unit. If no argument is given, the program will prompt for direct-beam center. If three numbers are given, the first two are the coordinates, and the third one is the uncertainty for both coordinates. If four number are given, they are coordinates and their uncertainties.

If a string argument of *free*, *Free*, *FREE*, *freed*, *Freed*, or *FREED* is given, at least one numerical argument is needed to specify the uncertainty for both coordinates. If two numerical arguments are given with the string *free*, they are the uncertainties for the coordinates, respectively. If no numerical argument is given with the string *free*, the command is ignored. An uncertainty value must be non-negative, or it will be ignored. The default uncertainty of direct-beam center is 0.2 pixel in both directions.

If a string argument of *fix*, *Fix*, *FIX*, *fixed*, *Fixed*, or *FIXED* is given, the uncertainties of direct-beam center are reset to 0. If there are two numerical arguments given with the string *fix*, they are considered as direct-beam center in pixel.

This command may also take a string argument of *normal*, *Normal* or *NORMAL*. If so, the two numerical values are the coordinates of normal projection of origin to a flat detector surface instead of direct-beam center. This option is useful when a detector is (nearly) parallel to the X-ray beam, when direct-beam center is way off the detector. Obviously, this option is meaningless to a cylindrical detector. An error will occur, if this option is used with a cylindrical detector.

11.1.1.11 Precognition:Input:Pixel (P|p)

Takes one numerical argument as the size of a pixel in mm, or two numbers as the size at horizontal and vertical directions. If no argument is given, the program prompts for pixel size. If three numbers are given, the first two are pixel sizes, and the third is the uncertainty of horizontal pixel size in mm. If four number are given, they are pixel sizes and their uncertainties.

If a string argument of *free*, *Free*, *FREE*, *freed*, *Freed*, or *FREED* is given, at least one numerical argument is needed to specify the uncertainty of horizontal pixel size. If two numbers are given with the string *free*, they are uncertainties of horizontal and vertical pixel sizes, respectively. If no number is given with the string *free*, this command is ignored. Any negative uncertainty value will be ignored.

Please note that the behavior of this command is slightly different from `Center`. The default uncertainty of vertical pixel size is 0. Although explicitly changing it to a positive value is legal, this shall virtually never be done. The default uncertainty of horizontal pixel size is 10 nm.

11.1.1.12 `Precognition:Input:Swing (S|s)`

Takes two numerical arguments as the swing angles of a flat detector in degree. This command is deactivated if a cylindrical detector is selected. The first swing angle is around X-axis (horizontal), and the second is around Y-axis (vertical). If one number is given, the second is assumed to be 0. The rotation center is the crystal. Therefore, one of them or both can be used as 2θ angle of a flat detector depending on specific setup.

If no argument is given, the program prompts for swing angles. If three numbers are given, the third value is the uncertainty for both swing angles. If four numbers are given, they are swing angles and their uncertainties, respectively.

Please note that swing angles cannot be refined. Their default uncertainties are 0.

11.1.1.13 `Precognition:Input:Tilt (T|t)`

Same as the command `Swing`, except that the rotation center of tilt angles is the direct-beam center on detector. Tilt angles are used to correct prediction errors.

If a string argument of `free`, `Free`, `FREE`, `freed`, `Freedy`, or `FREED` is given, at least one numerical argument is needed to specify the uncertainty of both angles. If two numerical arguments are given with the string `free`, they are uncertainties of the tilt angles, respectively. If no number is given with the string `free`, this command is ignored. The default uncertainties of tilt angles are 0.1 degree.

If a string argument of `fix`, `Fix`, `FIX`, `fixed`, `Fixed`, or `FIXED` is given, the uncertainty of tilt angles is reset to 0. One or two numerical arguments given with the string `fix` are considered tilt angles rather than uncertainties.

11.1.1.14 `Precognition:Input:Bulge (B|b)`

Takes two numerical arguments as bulge correction coefficients, quadratic and cubic, of a flat detector. This command is deactivated if a cylindrical detector is selected. These coefficients are dimensionless quantities. They are usually very small numbers, therefore the unit is chosen as 10^{-12} . These corrections are largely inherited from film and off-line imaging plate work from the past. ESRF image-intensified CCD detector also

requires such correction. They are very much fudge factors for other detectors.

If no argument is given, the program prompts for bulge correction coefficients. If one number is given, it is the quadratic coefficient. If two are given, they are quadratic and cubic coefficients. If three are given, the third one is the uncertainty of the quadratic coefficient. If four are given, they are bulge coefficients and their uncertainties.

If a string argument of *free*, *Free*, *FREE*, *freed*, *Freed*, or *FREED* is given along with a numerical argument, this value is the uncertainty of the quadratic coefficient. If two numbers are given with the string *free*, they are uncertainties of quadratic and cubic coefficients, respectively. If no numerical argument is given, this command is ignored. The default uncertainties are 0.

If a string argument of *fix*, *Fix*, *FIX*, *fixed*, *Fixed*, or *FIXED* is given, the uncertainties are reset to 0. One or two numerical arguments given with the string *fix* is considered as bulge coefficients in stead of uncertainties.

11.1.1.15 `Precognition:Input:Shift (S|s)`

Takes two numerical arguments as off-centering shifts in mm of a cylindrical detector. This command is deactivated if a flat detector is selected. The first shift is along the vertical Y-axis, that is, the first shift corrects mis-alignment error when the cylindrical axis is not crossing X-ray beam. The second shift is along Z-axis, i.e., the X-ray beam. It corrects error when the crystal is not perfectly centered on the cylindrical axis.

If no argument is given, the program prompts for off-centering shifts. If one number is given, it is the vertical shift. If two are given, they are shifts along Y- and Z-axis. If three are given, the third one is the uncertainty of the vertical shift. If four are given, they are both shifts and their uncertainties.

If a string argument of *free*, *Free*, *FREE*, *freed*, *Freed*, or *FREED* is given along with a numerical argument, this value is the uncertainty of the vertical shift. If two numbers are given with the string *free*, they are uncertainties of the shifts. If no numerical argument is given, this command is ignored. The default uncertainties are 0.

If a string argument of *fix*, *Fix*, *FIX*, *fixed*, *Fixed*, or *FIXED* is given, the uncertainties are reset to 0. One or two numerical arguments given with the string *fix* is considered as shifts in stead of uncertainties.

11.1.1.16 `Precognition:Input:Resolution (R|r)`

Specifies a resolution range. If no numerical argument is given, the program prompts for resolution range. If one number is given, it is taken as the highest resolution d_{\min} in Å. The default of lowest resolution d_{\max} is 1000 Å. If two numbers are given, they defines a resolution range. No specific order is expected. Only positive values are legal. Resolution range is used by many procedures throughout the system.

11.1.1.17 `Precognition:Input:Wavelength (W|w)`

Specifies a wavelength range. If two numerical arguments are given, they defines a wavelength range. No specific order is expected. If three numbers are given, the third one is a reference wavelength. If fewer than two numerical arguments is given, the program prompts for wavelength range. All values must be positive and less than 10, otherwise they will be ignored. Wavelength range is used in many procedures throughout the system.

11.1.1.18 `Precognition:Input:Chebyshev (V|v)`

Specifies the order of Chebyshev approximation to λ -curves (Box 5.2.3.0.1). All frame-specific λ -curves must have the same order as that of the overall λ -curve. However, some Chebyshev polynomials of lower degrees may have their coefficients fixed, while those of higher degrees have their coefficients free to refine.

If no numerical argument is given to this command, the program prompts for Chebyshev approximation order for overall spectrum and the free order at higher degrees of frame-specific spectra. If one number is given, it is taken as the order of the overall spectrum, and no free order for frame-specific λ -curves, that is, all frames share a single λ -curve. If two numbers are given, the second one specifies the number of free order at higher degrees of frame-specific λ -curves. Obviously, both numbers must be integers, and the second must not be greater than the first. Or, error message will be printed.

If a string argument of `unimodal`, `Unimodal`, `UNIMODAL`, `bimodal`, `Bimodal`, or `BIMODAL` is given, it hints the program to restrain the λ -curves to be unimodal or bimodal. If a string argument of `arbitrary`, `Arbitrary`, or `ARBITRARY` is given, λ -curves will be left as they are obtained without restrain or spike removal. If a string argument of `fix`, `Fix`, `FIX`, `fixed`, `Fixed`, or `FIXED` is given, λ -curves will not be refined. String argument of `free`, `Free`, `FREE`, `freed`, `Freed`, or `FREED` reverses it. By default, if no string argument is given, a spike removal procedure will try to remove sharp spikes that might present at both ends of the λ -curves.

11.1.1.19 `Precognition:Input:Anomalous (A|a)`

Sets or negates the flag of anomalous scattering. If a string argument of `on`, `On`, or `ON` is given, the flag is set to true. If a string argument of `off`, `Off`, or `OFF` is given, the flag is reset to false. If no argument or no recognizable argument is given, the flag is negated. Anomalous scattering flag affects the performance of the program when anomalous scattering signal should or should not be considered.

11.1.1.20 `Precognition:Input:Quit (Q|q)`

Exits from this menu.

11.1.2 `Precognition:Spot (S|s)`

This command is only activated when an image is loaded. This command has the same numerical arguments as the command `Input:Spot` (11.1.1.7), overall spot length, width in pixel and σ -cut. This command uses these values to recognize spots in the loaded image. If some or all numerical arguments are missing, the program uses the previous values given by `Input:Spot` (11.1.1.7). If no previous values are given, the program defaults take effect. The program defaults of spot length and width are both 10 pixels.

If a string argument is given, the recognized spots will be saved into a file with the string as its filename. See 11.8.5 Spot file `.spt` for its format. If the command is obtained from a keyboard input, existing file will not be overwritten until confirmed. If the command is from a preprogrammed script, existing file will be overwritten without warning, but a message of overwriting is printed.

11.1.3 `Precognition:Profile (F|f)`

This command is activated when spot recognition is successful. This command executes a procedure that learns a mean spot profile across the loaded image, and sets the overall spot length and width automatically. The learned profile parameters will be printed.

Diagnostic feature is available to save and display the learned profiles.

11.1.4 `Precognition:Ellipse (E|e)`

This command is activated when spot recognition is successful or spots are loaded. This command executes a procedure that recognizes Laue ellipses on flat detector, or other conics on cylindrical detector. An optional numerical argument from integers 1 through 10 specifies the resolution of image processing involved in the procedure. 4 is the default.

If ellipse recognition is successful, this command keeps on to recognize nodal spots and sorts them according to their significance. This command also refines the direct-beam center. If a string argument is given, the recognized nodal spots and other intermediate results will be saved into files for diagnostic purpose.

11.1.5 `Precognition:Pattern (P|p)`

This command is activated when some nodal spots are recognized or loaded. This command executes a procedure of indexing and refinement. A string argument of filename of a spot file is virtually always necessary, although it is legal to omit the filename. Predicted spots will be saved into this file in order to validate the indexing. See 11.8.5 for its format. A geometric parameter file will also be saved if a string argument is given. Its filename is chosen as the spot filename suffixed by `.inp`. This parameter file is a legal command script that can be loaded back into the program so that all parameters will be restored. See the beginning of 11.1 for loading a command script.

This command may take two integer arguments m and n . If the first integer m is positive, the indexing procedure will keep searching until m solutions are found or until its natural end. If m is 0 or negative, the procedure will search all possible solutions until its natural end. The default is 1. If $m > 1$, more spot files of prediction and parameter files, but no more than 9, will be saved. The extra filenames are prefixed by `1_`, `2_`, ..., `m-1_`.

The second integer n has its legal values from 8 through 16. The default is 8. The first n nodal spots will be used in indexing procedure. Nodal spots recognized by the command `Nodal` are sorted by significance. User supplied nodal spots shall be sorted by the user before loading. No sorting will be done by the program.

11.1.6 `Precognition:Limits (L|l)`

This command is activated when spot recognition is successful. This command detects a number of soft limits at very early stage of the data processing, such as resolution limit, reliable resolution for geometric refinement, Bragg angle limit, maximum spot density, best σ -cut, and an estimate of λ -curve. These quantities are called soft limits, since they often do not have definite values. At early stages of the data processing, they are even more uncertain. This command provides early estimates to these limits as accurate as they can, so that users do not have to rely on their experience too much. Good estimates to these soft limits may guide data processing away from possible troubles.

If this command is executed before an image is indexed. Any information related to λ -curve is arbitrary. Information of λ -curve becomes valid only when an image is indexed or its parameter file from previous indexing is loaded. This command may accept a string argument as a filename of the estimated λ -curve

from 0.25 to 2 Å if no numerical argument is given. If one numerical argument is given with a string argument, it specifies λ_{\max} for the estimated λ -curve. If two are given, the second is d_{\min} that overwrites the soft limit obtained by this procedure. This option is used when this procedure over or under estimate d_{\min} . A better d_{\min} can be specified by user during estimation of λ -curve.

11.1.7 Precognition:Dataset (D|d)

This command enters a submenu for choice of several control options. When exiting from the submenu, a procedure loops through a set of images and processes them. The methods of processing fall in two categories, either geometric refinement or spot integration. The dataset is defined by a user supplied parameter file and a goniometer setting file (11.8.3) or a series of repetition of `Input:Goniometer`. It may also defined by loading a series of parameter files.

Diagnostic options are available.

11.1.7.1 Mode of processing

This command `Dataset` accepts an optional string argument that specifies a method of processing, called mode. These modes are listed below:

normal, Normal, Or NORMAL

This is a mode of geometric refinement. This is the default when no string argument is given. Also when a non-recognizable string is given, this mode will take effect with a warning message. A set of images will go through geometric refinement. The initial geometric parameters of each pattern are taken from the first pattern via a user supplied parameter file. This mode does not tolerate any failure of refinement, not even a drop in quality of refinement. The procedure will be terminated if it happens. A message will report the point of quit. A set of parameter file will be generated or overwritten without confirmation. The filenames are automatically chosen as the image filename suffixed by `.inp`. This mode is capable of processing well behaved images.

progressive, Progressive, Or PROGRESSIVE

This is a refinement mode that is the same as `normal` mode except that the initial parameters are taken from the last refined pattern. Under this mode, some parameters may drift away, when the refinement goes on from frame to frame. If the procedure detects losing crystal orientation, or for whatever reason, a pattern is not refined as well as the previous ones. A random-walk algorithm will be launched to trace back the orientation. If this is not successful, the pattern will be re-indexed using spots close to the known nodal positions. It assumes that the correct orientation is not

very far away, although it is currently lost. If this second attempt is not successful, the pattern recognition procedure is desperately restarted from the very beginning. Ellipse and nodal recognition will be executed automatically followed by indexing and refinement. If any of the attempts succeed, the processing will go on, otherwise, it will be terminated. This mode is capable of processing images from decaying protein crystals and occasionally slipping crystals.

drunk, Drunk, Or DRUNK

This is a refinement mode that is the same as `progressive` mode except that the random-walk algorithm is invoked before each refinement. The step size and the total steps of the random-walk are automatically determined according to the success of better fits or lack of them. The parameters from the best fit will serve as the initials of the geometric refinement. This mode is more suitable to process images from frequently jumping crystals in time-resolved studies.

calibration, Calibration, Or CALIBRATION

This is a special refinement mode that can be used to calibrate crystal-to-detector distance and cell constants. No other parameter can be calibrated. Any parameter to be calibrated must be fixed first, that is, its uncertainty is reset to 0. Known values of these variables can be assigned. This mode of refinement will adjust the parameters to be calibrated from their initial values to the assigned values and fix them at the assigned values. In the mean time, any error caused by the calibration will be redistributed to other free parameters. This mode is useful when a parameter is changed too much beyond its trusted range by previous refinement, and one would like to trace it back to a known value.

final, Final, Or FINAL

This is a refinement mode. The initial values of each pattern under this mode come from its own parameters. This behavior differs from that under `normal` and `progressive` modes, but is the same as that of `calibration` mode. This mode is used at a final stage of geometric refinement as its name suggests, but this stage may not be always necessary. One may choose to use a higher resolution limit and a wider wavelength range for this mode to finalize the refinement.

integration, Integration, Or INTEGRATION

This is a generic mode of spot integration. This mode is not defined yet in the latest release.

box, Box, Or BOX

This is a fast integration mode. A common box algorithm is used by this mode. A square box is located at each predicted spot. All pixels inside and outside a central circular partition are summed up to obtain peak and

background intensities, respectively. There may be a certain partition that functions as a guard area between peak and background. The box size is related to the overall length of typical spots given by `Input:Spot`. This mode implementing a simple summation method offers a means of integration on-the-fly, but has severe limitations. This mode shall not be used to process images with very streaky spots. Also no attempt is given to resolve spatial overlaps. An integrated intensity file for each image will be created or overwritten without confirmation. The filenames are automatically chosen as the image filenames suffixed by `.ii`. See integrated intensity file for its format (11.8.6).

fixedElliptical, fixedelliptical, FixedElliptical, Fixedelliptical, OR FIXEDELLIPTICAL

This is an integration mode of a better summation method. A rectangle box of size given by `Input:Spot` is located at each predicted spot, and reoriented to a radial direction. A central elliptical partition serves as the peak area, the rest serves as the background area. There is also guard area between the peak and background areas. This algorithm attempts to resolve some spatial overlapping intensities as long as no pixel involves in three or more spots. Local background around a spot is modeled by a tilting plane. This integration mode is also fast and suitable to process strong, non-decaying images with isotropic streakiness and light spatial overlaps.

variableElliptical, variableelliptical, VariableElliptical, Variableelliptical, OR VARIABLEELLIPTICAL

This integration mode is the same as `fixedElliptical` mode except that the box size is determined automatically for each image rather than taken from a user input. This mode is better for processing strong images with developing spot streakiness and light spatial overlaps.

numeric, Numeric, OR NUMERIC

This is a mode of integration using a numeric profile fitting algorithm. Each spot profile is obtained from averaging of strong, well-separated spots within a small area of the image. These qualified spots are called samples. Selection of samples can be influenced by spot size and σ -cut (11.1.1.7). The profile is then applied to all spots of this area. Spatial overlaps are deconvoluted during the fitting. This mode is suitable for processing of strong images with spatial overlaps. It is also the best choice for slightly splitting spots or other odd shaped spots.

**linearAnalytical, linearanalytical,
LinearAnalytical, Linearanalytical, Or
LINEARANALYTICAL**

This is an integration mode that employs the analytical profile fitting algorithm (Ren et al., *J. Appl. Cryst.* **28**, 461-81, 1995). An analytical profile is obtained from fitting the sample spots of an area of the image under processing, and applied to all spots in this area. Spatial overlaps are deconvoluted. Analytical profile removes more noise than numeric profile does. This is the most accurate integration mode in most cases. It is a far better choice when processing weak images with lots of spatial overlaps.

**nonlinearAnalytical, nonlinearanalytical,
NonlinearAnalytical, Nonlinearanalytical, Or
NONLINEARANALYTICAL**

This is an integration mode that uses the same analytical profile fitting algorithm except that each profile obtained from samples will continue to evolve slightly during the fitting to all spots in its home area, however, the extent of additional adjustment to a profile depends on how strong each individual spot is. No adjustment is allowed during fitting to weak spots. This integration mode may have advantage in some cases.

hybrid, Hybrid, Or HYBRID

This is an integration mode of two or more algorithms. It is designated to an automatic choice of integration mode depending on type of image and spot is encountered. This mode is a hybrid of `variableElliptical` mode, the best summation method, and `linearAnalytical` mode, the best profile fitting method in the current release. This may be changed in future releases.

11.1.7.2 Precognition:Dataset:In (I|i)

Accepts a path from where diffraction images can be loaded. If a string argument is given, it is taken as a relative or absolute path. A relative path refers to the directory where the program is launched. The path may or may not end with a slash (/). If no string argument is given, the program prompts for a path for input directory. The default path is the current directory, if this command is not called.

11.1.7.3 Precognition:Dataset:Out (O|o)

Same as above except the path is a directory where results will be saved into.

11.1.7.4 Precognition:Dataset:Path (P|p)

Accepts a path for either input or output directory. This command is available for backward compatibility to deprecated features. Use commands `In` and `Out` instead.

11.1.7.5 Precognition:Dataset:OK (K|k)

A dummy command for backward compatibility to deprecated features.

11.1.7.6 Precognition:Dataset:Resolution (R|r)

This command is activated when an integration mode is specified. This command is the same as Input:Resolution. If a new resolution range is specified here, the integration procedure will use the new range. If this command is not called, previous range given by Input:Resolution will be used.

11.1.7.7 Precognition:Dataset:Wavelength (W|w)

This command is activated when an integration mode is specified. This command is the same as Input:Wavelength. New wavelength range given by this command has higher priority than previous range.

11.1.7.8 Precognition:Dataset:Quit (Q|q)

Exits from this menu and starts data processing depending on the mode specified (11.1.7.1).

11.1.7.9 Connection to LaueView

If an integration mode is given, the command Dataset may also accept an integer argument as a 'pattern number' used by LaueView, the old Laue data processing software (<http://cars.uchicago.edu/biocars/pages/lauemanual.html>). Integrated intensities will also be saved in LaueView format as .shf files. Each image will be assigned by a 'pattern number' starting from the integer given by this command.

11.1.8 Precognition:Quit (Q|q)

Attempts to exit from the program. A confirmation is prompted, if this command is input from a keyboard. If confirmed, the program terminates; if not, returns to the main menu. If this command is obtained from a script, the program terminates without confirmation.

11.2 `Epiform_Ex.x.x_Px.x` `epiform_Ex.x.x_Px.x` `Epiform` `epiform`

This is the command-driven version of `Epiform`. It prints the release number and credits, imports CPL, the supporting library, and prints CPL version number. If appropriate license is found, a main menu is printed, otherwise, the program terminates with an error message. The main functionalities of `Epiform` are wavelength normalization, data scaling, harmonic and spatial overlap deconvolution, and data merging. `Ex.x.x` indicates edition, version, release, and patch numbers. `Px.x` is platform or operating system and its version. See 11.1 for description of full command, abbreviated command, script loading, comment, and shell command.

11.2.1 `Epiform:Input (I|i)`

This command redirects to `Precognition:Input`. See 11.1.1.

11.2.2 `Epiform:Restore (R|r)`

This command specifies a filename for saving of intermediate scaling results, or restores scaling parameters from a previously saved file.

11.2.2.1 Saving intermediate parameters

If a string argument follows this command, it will be used to save intermediate results during scaling before the first cycle and after each cycle. The saved file is a legal command script to be restored. This feature is similar to geometric parameter file of `Precognition`. See 11.1 for loading of script file. If no argument follows, this command enters a submenu as described below.

11.2.2.2 `Epiform:Restore:Polarization (P|p)`

Accepts a numerical argument as a polarization factor of the X-ray beam. This value must be between and including -1 and 1. Invalid value will be ignored. If no argument is given, this command is also ignored without prompt. This is the general style of this submenu, since the commands in this submenu are usually issued from a command script rather than a keyboard input.

11.2.2.3 `Epiform:Restore:Mosaicity (M|m)`

Accepts a numerical argument as an overall mosaicity in full-width at half-maximum and in degree. If a string argument of `.ii` filename

follows, the value will be added to the overall mosaicity, and the sum is associated with this frame only.

11.2.2.4 `Epinorm:Restore:Scale (S|s)`

This command must be followed by a numerical and a string arguments. The number must be positive. It is taken as the isotropic scale factor of a frame specified by the string argument. Any command and arguments that do not fit this description will be ignored.

11.2.2.5 `Epinorm:Restore:Temperature (T|t)`

Accepts a numerical and a string arguments that specify the isotropic temperature factor of a frame.

11.2.2.6 `Epinorm:Restore:Expansion (E|e)`

Accepts a numerical and a string arguments that specify the isotropic expansion factor μ of a frame, so that the wavelength of a reflection on this frame is $(1 + \mu)\lambda + \nu$, where ν is λ -shift (11.2.2.7).

11.2.2.7 `Epinorm:Restore:Lambda-shift (L|l)`

Accepts a numerical and a string arguments that specify the λ -shift of a frame, so that the wavelength of a reflection on this frame is $(1 + \mu)\lambda + \nu$, where μ is isotropic expansion factor (11.2.2.6).

11.2.2.8 `Epinorm:Restore:AnisoScale (A|a)`

Accepts nine numerical arguments $a_{11}, a_{12}, a_{13}, a_{21}, a_{22}, a_{23}, a_{31}, a_{32}, a_{33}$ and a string argument as the `.ii` filename of a frame. An anisotropic scale factor f_a is defined as below:

$$f_a = (a_0 + a_{11}h + a_{12}k + a_{13}l + a_{21}\frac{h^3}{|h|} + a_{22}\frac{k^3}{|k|} + a_{23}\frac{l^3}{|l|} + a_{31}hk + a_{32}kl + a_{33}lh)^2$$

where a_0 is the isotropic scale factor (11.2.2.4).

11.2.2.9 `Epinorm:Restore:AnisoTemperature (N|n)`

Accepts nine numerical arguments $b_{11}, b_{12}, b_{13}, b_{21}, b_{22}, b_{23}, b_{31}, b_{32}, b_{33}$ and a string argument as the `.ii` filename of a frame. An anisotropic temperature factor f_b is defined as below:

$$f_b = \exp[b_0(\sin \theta / \lambda)^2 + b_{11}h + b_{12}k + b_{13}l + b_{21}\frac{h^3}{|h|} + b_{22}\frac{k^3}{|k|} + b_{23}\frac{l^3}{|l|} + b_{31}hk + b_{32}kl + b_{33}lh]$$

where b_0 is the isotropic temperature factor (11.2.2.5).

11.2.2.10 `Epinorm:Restore:Wavelength (W|w)`

This command is the same as `Input:Wavelength` that takes three numerical arguments as λ_{\min} , λ_{\max} , and λ_{ref} . The first two values have no specific order, but the reference wavelength must be the last. Availability of this command in this submenu makes an input script more concise.

11.2.2.11 `Epinorm:Restore:Normal (O|o)`

Specifies a coefficient of Chebyshev approximation to λ -mapping function. This command accepts one, two or three numerical arguments. If two numbers are given, the first must be an integer that specifies the degree of a Chebyshev polynomial. The second is the corresponding coefficient of the Chebyshev approximation. This command should be repeated from 0 through $n - 1$, recommended in ascending order, where n is the order of Chebyshev approximation.

If only one number is given, it is taken as the function value of the Chebyshev approximation instead of a coefficient. This command shall also repeat for n times, that is, the number of repeat determines the order of the Chebyshev approximation. These function values are assumed to locate at Chebyshev abscissas in ascending order of wavelength.

If three numbers are given, the first is an integer that specifies the order of Chebyshev approximation. The second is a wavelength in Å, and the third is the function value at the wavelength. This command shall repeat at different wavelengths in ascending order. The first arguments to the repeating command are likely the same. However, if they are different, the last value overwrites all earlier ones.

If different formats of this command are given in a same session, a 3-argument command has the highest priority, a 1-argument one has medium priority, and a 2-argument one has the lowest priority. Commands with lower priority will be ignored.

11.2.2.12 `Epinorm:Restore:Coefficient (C|c)`

Specifies a coefficient of Chebyshev approximation to λ -curves. If a string argument of `.ii` filename is given, the λ -curve is specific to this frame, otherwise, it is the overall λ -curve. The rule of numerical arguments is as same as that for command `Normal` (11.2.2.11).

11.2.2.13 `Epinorm:Restore:Quit (Q|q)`

Exits from this submenu.

11.2.3 `Epinorm:Scale (S|s)`

This command is activated when data are loaded. It enters a submenu. When exits from the submenu, this command starts a procedure that performs wavelength normalization, frame-to-frame scaling, determination of temperature, crystal mosaicity, etc.

The command `Scale` may take three numerical arguments and a string argument. The first numerical argument is a σ -cut, if it is between 0 and 100 and including 0. The default is 3. Data points with $I/\sigma(I)$ greater than this value will contribute to scaling. If σ -cut is 0, all positive integrated intensities but not 0 will be used. If this numerical argument is greater than or equal to 100, it is taken as the number of data points per frame to be loaded for scaling rather than σ -cut. The best data points will be used.

The second numerical argument specifies data isotropy, and has the choices of -1, 0, 1, or 2. The default is 0, which indicates isotropic scaling and temperature factors only. All anisotropic factors will be fixed at their initial values. If -1 is given, all temperature factors and anisotropic scale factors will be fixed at their initial values, and only isotropic scale factors are refined. 1 indicates anisotropic linear scale and temperature factors a_{11} , a_{12} , a_{13} , and b_{11} , b_{12} , b_{13} are refined. 2 indicates all anisotropic scale and temperature factors are free. See 11.2.2.8 and 11.2.2.9 for anisotropic factors.

The third numerical argument has the choices of 0, 1, or 2. The default is 0, which means that crystal mosaicity is not refined. If the third number is 1, an overall crystal mosaicity is refined. If it is 2, frame-specific mosaicities are refined.

If a string argument to the command `Scale` other than those can be recognized as a scaling mode (11.2.3.13) is given, it is considered a `.ii` filename of a reference frame. Some parameters of the reference frame are fixed. See 11.2.3.3 and 11.2.3.5 for detail. The string could also be a scaling mode if recognized as one of the legal choices.

All values given by the numerical and string arguments are initial values when entering the submenu. They may be modified by the subcommands.

11.2.3.1 `Epinorm:Scale:Sigma (S|s)`

Takes a numerical argument for σ -cut. It may also take an optional string argument of `and`, `And`, `AND`, `or`, `Or`, or `OR` for logic control between the two criteria: σ -cut and data points per frame. Consider any given frame, there will be a certain number N_σ of data points qualify the σ -cut. We denote the user-specified number of data points per frame N . Comparing N_σ and N , and combining the logic control, there will be 4 situations as

tabulated in Table 5.2.1.0.1, where the method of data selection and usage of each situation can be found.

11.2.3.2 `Epinorm:Scale:Data (D|d)`

Takes a numerical argument for number of data points per frame to be loaded for scaling. It may also take an optional string argument of `and`, `And`, `AND`, `or`, `Or`, or `OR` for logic control between the two criteria: σ -cut and data points per frame. This option is the same as the one of the previous command `Sigma`. The last entered logic overwrites the previous. Consider any given frame, there will be a certain number N_σ of data points qualify the σ -cut. We denote the user-specified number of data points per frame via this command N . Comparing N_σ and N , and combining the logic control, there will be 4 situations as tabulated in Table 5.2.1.0.1, where the method of data selection and usage of each situation can be found.

11.2.3.3 `Epinorm:Scale:Reference (F|f)`

Takes a string argument of a `.ii` filename as a reference frame. If this frame does not exist, is not loaded for scaling, or no reference frame is specified, the first frame loaded is the reference. The isotropic scale factor a_0 of the reference frame is fixed as initialized. 1 is its default value. All other factors a 's and b 's of the reference frame are fixed as initialized. Their defaults are all 0. See 11.2.2.8, 11.2.2.9, and 11.2.3.5 for more.

11.2.3.4 `Epinorm:Scale:Polarization (P|p)`

Takes an optional numerical argument between and including -1 and 1 as an initial value of beam polarization factor. However, this input is ignored, if beam polarization factor has been or will be restored in `Restore` section (11.2.2).

This command may also take an optional string argument of `free`, `Free`, `FREE`, `freed`, `Freed`, or `FREED`. Beam polarization factor will be refined. If the string argument is `fix`, `Fix`, `FIX`, `fixed`, `Fixed`, or `FIXED`, polarization will be fixed in refinement.

11.2.3.5 `Epinorm:Scale:Isotropy (I|i)`

Takes a numerical argument of choices -1, 0, 1, or 2 for the meanings of fixed, isotropic, linear anisotropic, and nonlinear anisotropic, respectively. It also takes a string argument of `scale`, `Scale`, `SCALE`, `temperature`, `Temperature`, or `TEMPERATURE` to fix or free scale or temperature factors for refinement. This command should be repeated for scale and temperature factors. Both scale and temperature factors are fixed for reference frame. If some factors are fixed, their initial values may not be 0 or 1. They may be restored from the previous scaling. 1

indicates anisotropic linear factors a_{11} , a_{12} , a_{13} , or b_{11} , b_{12} , b_{13} are refined. 2 indicates all anisotropic factors are free. See 11.2.2.8 and 11.2.2.9 for definitions of anisotropic factors. Anisotropic factors make the process much more expensive in time and space. They may help to minimize local errors, however they are in general unnecessary. Poor data may be over fit by these parameters. Use them judiciously.

11.2.3.6 `Epinorm:Scale:Mosaicity (M|m)`

If one numerical argument is given, it is an initial value of an isotropic crystal mosaic spread in full-width at half-maximum and in degree. If two numerical arguments are given, they are taken as overall spot length and width in pixel. The effect is as same as the first two numbers to `Input:Spot`. Isotropic mosaic spread will then be evaluated from the spot size. The initial value of crystal mosaicity will be ignored, if any value from previous scaling has been or will be restored in `Restore` section (11.2.2). This command may take an optional string argument. If the string is `fix`, `Fix`, `FIX`, `fixed`, `Fixed`, or `FIXED`, mosaicity will not be refined. That does not mean it is 0. Its value may be restored from the previous scaling or set by this command or command `Input:Spot`. If the string is `free`, `Free`, `FREE`, `freed`, `Freed`, or `FREED`, an overall mosaicity that applies to all frames will be refined. If the string is `frame`, `Frame`, or `FRAME`, frame-specific mosaicities will be refined. Crystal mosaicity, especially frame-specific ones, may cause over fitting. Use them as later alternatives cautiously.

11.2.3.7 `Epinorm:Scale:Wavelength (W|w)`

Same as `Input:Wavelength`. See 11.1.1.17.

11.2.3.8 `Epinorm:Scale:Chebyshev (C|c)`

Same as `Input:Chebyshev`. See 11.1.1.18.

11.2.3.9 `Epinorm:Scale:Mapping (N|n)`

Takes a string argument. If it is `fix`, `Fix`, `FIX`, `fixed`, `Fixed`, or `FIXED`, λ -mapping will be kept as it is. If the string is `linear`, `Linear`, or `LINEAR`, λ -mapping will be changed to linear mapping (Box 5.2.3.0.1 and Figure 5.2.3.0.2) regardless of the existing mapping function. If the string is `nonlinear`, `Nonlinear`, or `NONLINEAR`, λ -mapping will be changed to a nonlinear mapping function (Figure 5.2.3.0.2) according to the current λ -curve. The command is ignored if no argument is given.

11.2.3.10 `Epinorm:Scale:Lambda-shift (L|l)`

If a string argument of `fix`, `Fix`, `FIX`, `fixed`, `Fixed`, or `FIXED` is given, λ -shifts will not be refined. If it is `free`, `Free`, `FREE`, `freed`,

Freed, or FREED, they will be refined. The default is free. If no argument is given, this command is ignored.

11.2.3.11 `Epinorm:Scale:Expansion (E|e)`

If a string argument of fix, Fix, FIX, fixed, Fixed, or FIXED is given, isotropic expansion factors will not be refined. If it is free, Free, FREE, freed, Freed, or FREED, they will be refined. The default is fix. If no argument is given, this command is ignored.

11.2.3.12 `Epinorm:Scale:Restore (R|r)`

Same as `Epinorm:Restore`. See 11.2.2.

11.2.3.13 `Epinorm:Scale:Mode (O|o)`

Takes a string argument that specifies scaling mode. If no argument is given, this command is ignored. The default scaling mode is global. The available scaling modes are listed below:

initial, Initial, Or INITIAL

An initial scaling will be performed. Data rejection is more stringent during the initial scaling. Initial scaling is useful to get λ -curve and isotropic scale factors roughly right before a more relaxed data rejection. Initial scaling is not always necessary.

global, Global, Or GLOBAL

Global scaling mode is selected. This is the standard mode. Many frame-specific parameters will be refined at the same time, however, if frame-specific λ -curves, or anisotropic factors are to be refined, the program will automatically switch to frame mode.

frame, Frame, Or FRAME

Frame scaling mode is selected. In the beginning cycles, this mode is identical to global mode, then the program refines parameters specific to a single frame at a time. It ends with more global scaling cycles.

local, Local, Or LOCAL

A special local scaling mode is selected. None of the normal parameters are refined in this mode, but each single observation receives a local scale factor, and each harmonic observation of multiplicity m receives m local scale factors.

test, Test, Or TEST

A test mode for diagnostic purpose is selected. User shall ignore this mode.

11.2.3.14 `Epinorm:Scale:Quit (Q|q)`

Exits from this submenu and starts scaling process.

11.2.4 `Epinorm:Lambda (L|l)`

Saves λ -curves, overall and frame-specific. This command may accept a string argument as a filename for the overall λ -curve. If no argument is given, the overall λ -curve is printed on screen only. Frame-specific λ -curves have automatically chosen filenames. They are the `.ii` filenames suffixed by `.lam`.

11.2.5 `Epinorm:Apply (A|a)`

Applies scaling parameters and merges redundant and symmetry-related measurements. This command may accept a string argument as a filename into which the results will be saved. If no string argument is given, results will be discarded except printing statistics.

11.2.5.1 Data selection

Same as the command `Scale`, the first numerical argument specifies a σ -cut (see 11.2.3), however, this σ -cut can be negative, which means that some negative integrated intensities can be included in the results. This value can be smaller than that used in scaling, that is, more weak reflections are included in the results, although they do not contribute to scaling.

11.2.5.2 Apply scaling parameters to single reflections

If the second numerical argument is 0, scaling parameters are applied to integrated intensities of single reflections only. Redundant and symmetry-related measurements are not merged. This procedure is not applicable to harmonic overlaps since their correction factors are not known yet. The output file has the 7-column CPL reflection file format, and is usually called `.edi` (energy-dispersive intensity, 11.8.7) or `.si` (scaled intensity) file, but the file extensions are not enforced.

If the second numerical argument is -1, energy-dispersive intensities will be saved in multiple files with filenames automatically chosen as the `.ii` filename suffixed by `.edi`. This option helps to trace the source of each measurement.

11.2.5.3 Merging single reflections

If the second numerical argument is 1, scaling parameters are applied to integrated intensities of single reflections. Redundant and symmetry-related measurements are then merged. If anomalous scattering flag is unset (see 11.1.1.19), Friedel mates are also merged. The output file has

the 7-column CPL reflection file format, and is usually called `.hkl` file (11.8.8). The file extension is not enforced. The output contains structure factor amplitudes rather than intensities.

11.2.5.4 Harmonic deconvolution

If the second numerical argument is 2, scaling parameters are applied to integrated intensities of single reflections, harmonic overlaps are deconvoluted, and then redundant and symmetry-related measurements are merged. Anomalous scattering flag (11.1.1.19) defines whether Friedel mates are considered equivalent. The output is a CPL reflection file containing structure factor amplitudes (11.8.8).

11.2.5.5 Deconvolution of extremely-close spatial overlaps

If the second numerical argument is 3, extremely-close spatial overlaps are also deconvoluted in addition to harmonic overlaps. The overall spot size (11.1.1.7) defines extremely-close spatial overlaps. All redundant and symmetry-related measurements are merged. Anomalous scattering flag (11.1.1.19) defines whether Friedel mates should be merged as well. The output is a CPL reflection file containing structure factor amplitudes (11.8.8).

11.2.6 `Epnorm:Quit (Q|q)`

Attempts to exit from the program. A confirmation is prompted, if this command is input from a keyboard. If confirmed, the program terminates; if not, returns to the main menu. If this command is obtained from a script, the program terminates without confirmation.

11.3 TBA

11.4 `Precognition.py`

To be written.

11.5 Utilities and Diagnostic Tools

This section describes utilities and diagnostic tools that work with Precognition system. Most of these tools involve graphical display of data. GRACE (1.6.10 and <http://plasma-gate.weizmann.ac.il/Grace/doc/UsersGuide.html>) and gnuplot (1.6.11) are two packages used to present graphics.

11.5.1 Spot recognition with `pick.py`

A Python utility program `pick.py` wraps spot recognition of Precognition, and can be used independently. Type `pick.py -h` for its usage:

```
% pick.py -h

|)|)| Welcome to CPL M0.15.2
|)|)\ Renz Research, Inc.

SOURCE: /home/renz/code/adaptors/pick.py
TYPE: error
~~~~~
Usage: /home/renz/code/adaptors/pick.py [-h|help] image format [sigmacut [spot
length [spot width]]] [spotfile]
 Picks diffraction spots from a given image.
 image: An image filename.
 format: A string that specifies the image format. Recognized formats are:
 ['KodakStoragePhosphor',
 'Fuji',
 'Q4',
 'Quantum4',
 'ESRFImageIntensifiedCCD',
 'MarCCD',
 'Kodak',
 'ESRF',
 'MAR3450Packed',
 'MAR345',
 'FujiImagingPlate',
 'Mar345',
 'MARCCD',
 'MARCCD165']
 sigmacut: Displayed spots must have signal-to-noise ratio greater than this
 value. The default is 1.
 spot length and width: If specified, must follow sigmacut. The default is
 10 pixels.
 spotfile: Save recognized spots in this file, if specified.
 WARNING - Existing file will be overwritten.
 -h|help: Prints this usage.
~~~~~
```

Listing 11.5.1.0.1 Usage of spot picking utility `pick.py`.

This program launches a GRACE session that displays recognized spots initially. One may alter and save the graph in GRACE if needed. Consult GRACE manual for help (1.6.10).

11.5.2 Superposition of two patterns with `su.py`

To plot two superimposed patterns, a utility program written in Python `su.py` is available. Type `su.py -h` for a message of usage:

```
Usage: su.py [-h or -help] [background] [foreground] [dimension] [sigmacut]
Displays two patterns superimposed together.
background, foreground: Two spot files. The first one will be displayed
in red circles; the second in black dots. If omitted, the defaults
are re.spt and pre.spt.
dimension: Dimension of a square displaying area. The default is 2048.
sigmacut: Displayed spots must have signal-to-noise ratio greater than this
value. The default is 1.
```

Listing 11.5.2.0.1 Usage of `su.py`.

This utility prepares a project file for GRACE and launches it. A fully functional GRACE session will appear in a new window. Consult GRACE manual if you want to manipulate the graph.

11.5.3 Curve and scatter plot with `plot.py`

`plot.py` plots curves and scatter plots saved in CPL reflection file format. See CPL reference for details of the format. Type `plot.py -h` for a help message:

```
Usage: plot.py [-h|help] [-c|s|cc|ss|cs|sc] data [data2] [x [x0 [y [y0]]]]
Displays a curve and/or a scatter plot.
data, data2: cpl.io.reflection files that contain data in the 4th and 5th
columns.
x, x0, y, y0: If no numerical argument is given, the entire curve(s) will
be plotted. If some numbers are given, plots the curve(s) in the ranges
of [x0, x] and [y0, y].
-c|s|cc|ss|cs|sc: Specifies plotting styles of curve or scatter plot for
data and data2, respectively.
-h|help: Prints this message.
```

Listing 11.5.3.0.1 Usage of `plot.py`.

11.5.4 Byte swapping with `swap1.py`

`swap1.py` swaps bytes of a file. The original file will be overwritten. Type `swap1.py` without an argument for usage. This utility is useful when some images have swapped bytes (Figure 11.5.4.0.1).

```
Usage: swap1.py file
```

Listing 11.5.4.0.1 Usage of `swap1.py`.



Figure 11.5.4.0.1 A byte-swapped diffraction image.

11.6 cpl.Precognition Module

To be written.

11.7 Coordinates Systems

To be written.

11.8 Files

This section describes a common file selection menu and various file formats used by the system of Precognition.

11.8.1 File selection

When a filename is required for either input or output but no prior filename is specified, or when an I/O error occurs due to attempts to overwrite an existing file, to read a nonexistent file, or to write a write-protected file, a common file selection menu is entered in order to resolve the ambiguity before more serious errors occur.

A complete filename is constructed from four parts: path, name, middle name, and extension. A complete filename looks like this: path/name.middlename.extension. path is a string that may contain slashes (/). It may even start with a slash (/), that is, the root. If so, it is called an absolute path, otherwise it is a relative path. name, middlename, and extension may not contain slash (/), but they may contain dots (.).

11.8.1.1 File selection:Filename (F|f)

Specifies a complete filename by a string argument following the command or by entering a string at a prompt.

11.8.1.2 File selection:Path (P|p)

Specifies a path by a string argument following the command or by entering a string at a prompt.

11.8.1.3 File selection:Name (N|n)

Specifies a name by a string argument following the command or by entering a string at a prompt.

11.8.1.4 File selection:Middle (M|m)

Specifies a middle name by a string argument following the command or by entering a string at a prompt.

11.8.1.5 File selection:Extension (E|e)

Specifies an extension by a string argument following the command or by entering a string at a prompt.

11.8.1.6 File selection:Cancel (C|c)

Exits from this menu without altering the previous file selection.

11.8.1.7 File selection:OK (O|o)

Exits from this menu and returns the last filename selected. If path, name, middle name, and extension are selected separately, they will be used to construct a complete filename.

11.8.1.8 Previously-selected filename

Any filename selected or synthesized previously, including partial filename, is added to the menu items listed above. They may be reselected by their sequential number. For example in the file selection menu listed below, two filenames `work/my.image` and `work/my.music` were previously selected. They may be reselected by entering the number before them, and then confirmed by OK.

```
File selection: 0          0: path/name.middle.extension
                1          1: work/my.image
                2          2: work/my.music
Filename       F: Set full name (e.g., f /path/name.mid.ext)
Path           P: Set path (e.g., p /path)
Name           N: Set name (e.g., n name)
Middle         M: Set middle name (e.g., m mid)
Extension     E: Set extension (e.g., e ext)
Cancel        C: Cancel file selection
OK            O: Exit this menu (default)
```

Listing 11.8.1.8.1 A file selection menu with some previously-selected filenames.

11.8.2 Crystal information file `.xtl`

This small file contains cell constants and space group of a crystal. It can be loaded wherever it is needed so that these values do not have to be entered again and again.

The first line of this file must be a string of title without any space. The second line should contain six space-separated numbers as the cell constants a , b , c in Å and α , β , γ in degree. The third line must be a string `spgp` followed by space group symbol and number, where the space group symbol is only a note to the user, has no specific format. The space group number must be valid, that is, 1 through 230. The rest of the line are arbitrary.

This file format is being deprecated.

11.8.3 Goniometer file `.gon`

This file contains goniometer settings of a series of images. Each line is occupied by one image, starting with the image filename, followed by one to three numbers ω , χ , and ϕ in degree. The missing numbers are assumed to be 0. This file must have `.gon` as its extension.

This file format is being deprecated.

11.8.4 λ -curve .lam

This is a 2- or 3-column file. The first two columns are wavelength in Å and relative intensity, respectively. If a third column exists, it is Bragg angle in degree.

11.8.5 Spot file .spt

This file format complies with the 7-column CPL reflection file format. Each line represents a spot. The first three columns are h , k , and l . They may be all 0, if a pattern is not yet indexed. Column 4 and 5 are coordinates of spots in pixel. Column 6 and 7 are integrated intensity and its standard deviation in detector unit $\times \text{pixel}^2$.

11.8.6 Integrated intensity file .ii

This is a 10-column file format that does not comply with the CPL reflection file format. The first three columns are h , k , and l . The fourth column is the multiplicity of a harmonic reflection. Column 5 and 6 are coordinates of spots in pixel. Column 7 and 8 are resolution and wavelength in Å. Column 9 and 10 are integrated intensity and its standard deviation in detector unit $\times \text{pixel}^2$. In addition, harmonic and spatial overlaps must be arranged consecutively.

11.8.7 Energy-dispersive intensity file .edi

This file format complies with CPL reflection file format. This file stores corrected but unmerged intensities. The first three columns are h , k , and l . Column 4 and 5 are wavelength and its standard deviation in Å. Column 6 and 7 are scaled intensity and its standard deviation in detector unit $\times \text{pixel}^2$.

11.8.8 Structure factor amplitude file .hkl

This is a typical CPL reflection file. The first three columns are h , k , and l . The other columns are F_+ , $\sigma(F_+)$, F_- , and $\sigma(F_-)$. Missing data are filled with 0 and 1. If Friedel mates are merged, results are placed in column 4 and 5. Column 6 and 7 are filled with 0 and 1, respectively.

CHAPTER 12

Tutorials

This chapter presents several real data sets. Through these examples, I hope to show how Precognition system works in real life. Good data processing strategies are discussed. Common mistakes are also analyzed. All diffraction images, command scripts, and log files can be found at <http://renzresearch.com/Precognition>.

12.1 An Undulator Laue Dataset from a Protein Crystal

This dataset was collected at BioCARS 14-ID-B station of APS using an undulator with a gap of 25 mm. This dataset was collected from a crystal of the M37V mutant of CO-bound dimeric clam hemoglobin. This crystal is in monoclinic C2 space group. Its cell constants are 93.22, 44.00, 83.56 Å, and 90.00, 121.95, 90.00 degree. The diffraction images were recorded on a MAR345 image plate detector. The entire dataset was collected in two passes of 31 images each. Both passes have 6° angular spacing between consecutive images. Pass a and pass b are offset from each other by 3°. All image files are stored in a subdirectory `images` of the working directory. This dataset is a courtesy of Vukica Srajer, Reinhard Pahl of BioCARS, The University of Chicago, and James Knapp, William Royer of The University of Massachusetts.

12.1.1 Visual evaluation of images and estimates of soft limits

Visual evaluation of the diffraction images is the first step of data processing. By visual inspection, one shall have a main idea of the data quality and develop some strategies on how to process. There are many graphics programs for this purpose. `Precognition.py` has the capability to display images as well, but it is currently under development.

Figure 12.1.1.0.1 shows the first image in this dataset. It is a high quality image, that shall give good data quality, but the crystal orientation shown in the first image may not be ideal for indexing, since two major ellipses are nearly tangential to each other. Figure 12.1.1.0.2 shows a lower central portion of the first image. Spot shapes are nicely round. Some spatial overlaps are clearly visible. From this zoom-in image, one may find the typical spot size. It seems to be 8×8 pixels for this dataset, approximately. Several images into the dataset shows a very typical Laue pattern at a random crystal orientation, which is in general better for indexing (Figure 12.1.1.0.3). Images at the end of pass a and pass b show some disruption to the crystal, but they largely maintained the original diffraction quality.

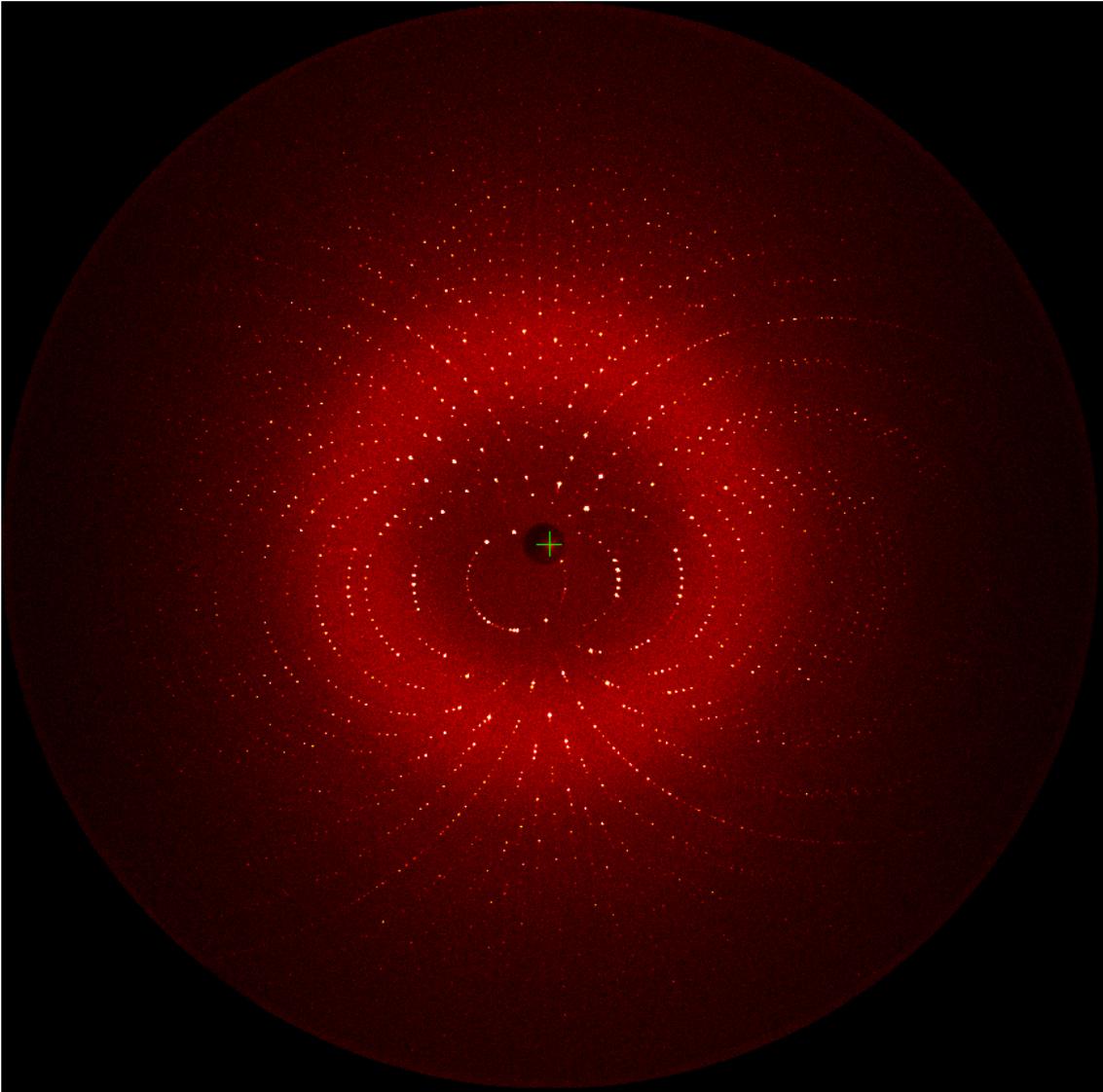


Figure 12.1.1.0.1 The first image in the dataset m37v_1a_001.mar3450.

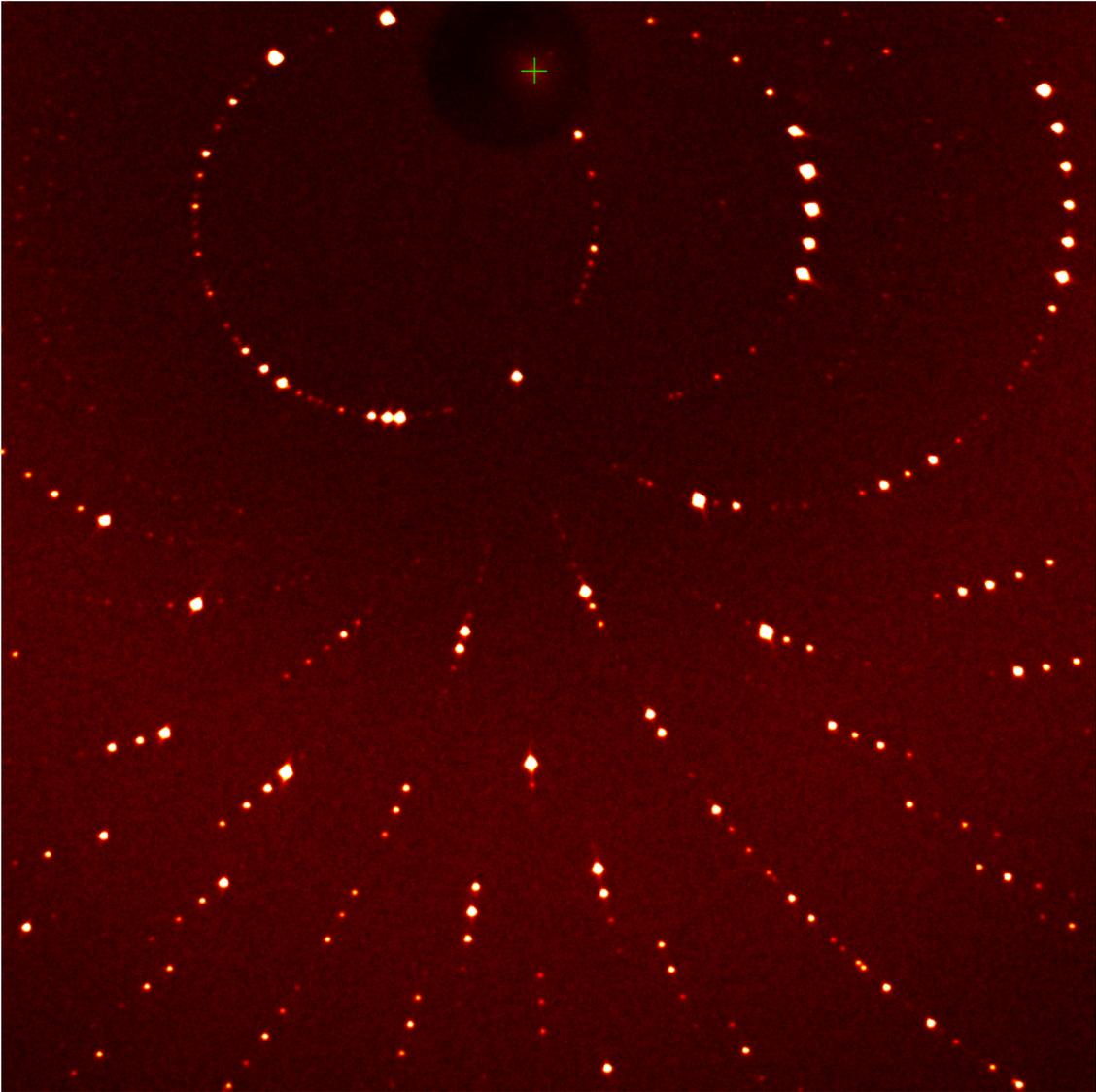


Figure 12.1.1.0.2 A portion of the first image in the dataset m37v_1a_001.mar3450.

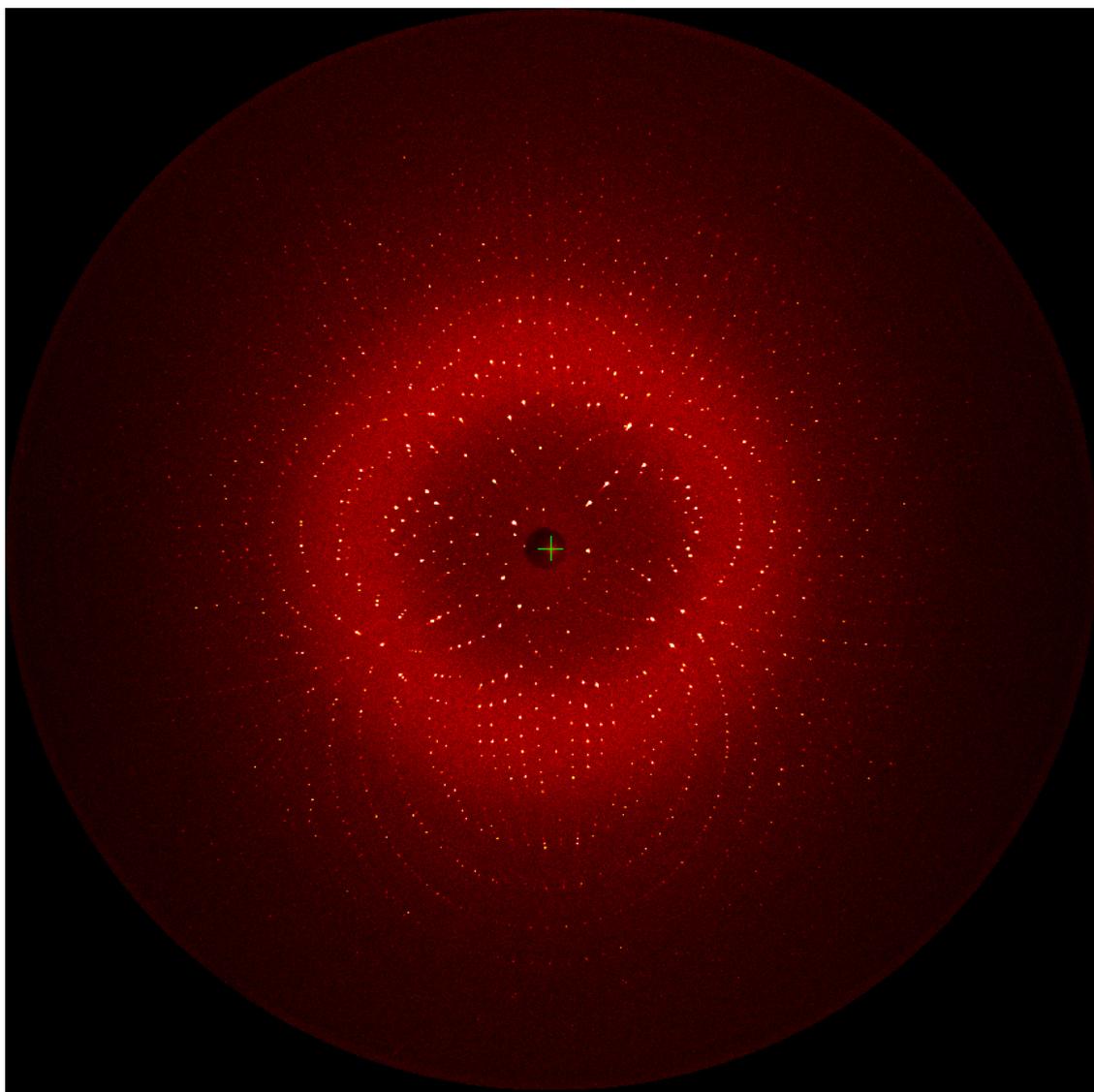


Figure 12.1.1.0.3 An image in the dataset `m37v_1a_015.mar3450`.

Unlike monochromatic diffraction images, it is not immediately obvious from a Laue image to what resolution the crystal diffracts. The feature of soft limit estimation can be used now, even before indexing. Listing 12.1.1.0.1 is the command script. Among the input, crystal-to-detector distance and detector pixel size should be precisely known. Direct-beam center, wavelength range and the peak wavelength should be roughly known. Spot size can be estimated from visual inspection of the images. One may start with the default σ -cut of 3. Running of this command script suggests a lower σ -cut. A part of the log file is in Listing 12.1.1.0.2. As the results of this job suggest, the subsequent indexing and geometric refinement shall use a resolution of 2.0 Å and σ -cut of 6. The diffraction limit of this crystal is estimated at 1.56 Å. We will see how good these estimates are.

```
diagnostic      off
Input
  Format        Mar345
  Distance     180
  Center       1705 1711
  Pixel        0.1 0.1
  Image        images/m37v_1a_015.mar3450
  Wavelength   1 1.5 1.1
  Quit
Spot           8 8 2.4
Limits
Quit
```

Listing 12.1.1.0.1 `limit1.inp`, estimation of soft limits before indexing.

```
|_____)
| Report |
| -----|
| -----|
| -----|
| -----|
| -----|
|_____|
```

```
Best sigma cut estimated at 2.56.
3321 real spots on this image.
Sigma cut results in 10% noise is 2.4.
Suggested sigma cut for indexing and geometry refinement is between
2.56 and 6.6.
```

```
Maximum spot density
14.8/mrad at Bragg angle of
16.0 degree.
```

```
Diffraction limit estimated at Bragg angle of
24.6 degree or
1.56 A resolution.
```

```
Suggested resolution for indexing and geometry refinement is 1.96 A.
```

Listing 12.1.1.0.2 Part of `limit1.log`, results from soft limit estimation.

12.1.2 Indexing

Under the guidance of these soft limits, we are now ready to index a pattern. As noted before, it is in general the easiest to index a random orientation pattern like `m37v_1a_015.mar3450` (Figure 12.1.1.0.3) rather than the first image (Figure 12.1.1.0.1). As a matter of fact, the indexing routine of Precognition is so powerful that I had hard time to find a pattern in this set to demonstrate mis-indexing. I had no trouble to index the first image at all.

12.1.2.1 Indexing

The indexing script is listed below. Once again, distance, pixel size, and goniometer setting are precisely known. It is a good idea to fix distance without refinement. Center and wavelength should be approximately known. See 2.1.3 for discussion on error of direct-beam center. Resolution and σ -cut are taken from the soft limit estimation. Here, a one-line script for crystal information is recommended, instead of the crystal information file (11.8.2), which is being deprecated. This script can be stored in a centralized special directory, say `~/xtal_info`, to be accessed by many data processing jobs. Two spot files `m37v_1a_015.re.spt` and `m37v_1a_015.pre.spt` will contain *recognized* and *predicted* spots after geometric refinement.

```
Crystal 93.22 44.00 83.56 90.00 121.95 90.00 5
```

Listing 12.1.2.1.1 `m37vHbI-CO.inp`, a one-line script of crystal information.

```
diagnostic      off
busy            off
Input
  @ m37vHbI-CO.inp
  Format        Mar345
  Distance      180 fix
  Center        1705 1711
  Pixel         0.1 0.1
  Goniometer    0 0 42
  Image         images/m37v_1a_015.mar3450
  Resolution    2.0 100
  Wavelength    1.0 1.5 1.1
  Quit
Spot           8 8 6 m37v_1a_015.re.spt
Profile
Ellipse
Pattern        m37v_1a_015.pre.spt
Quit
```

Listing 12.1.2.1.2 `index.inp`, command script for indexing.

The process carried out by `index.inp` has several other findings regardless success or failure of indexing (Listing 12.1.2.1.3). First, the command `Profile` learns an overall spot profile, therefore better spot size can be suggested. For example, this process suggests an overall spot size of 8×6 instead of 8×8 pixels. The newly suggested spot size shall be used by subsequent jobs. Second, by learning the spot profile, it estimates the crystal dimension and mosaic spread by a spherical crystal and isotropic mosaic model. It is harder to validate their accuracy, nevertheless, these values are useful in scaling. It would also be a good idea to insert a command `Profile` into the script `limit1.inp` (Listing 12.1.1.0.1) after the command `Spot`, so that these estimates can be

obtained even earlier. Third, direct-beam center is refined by the command `Ellipse` from (1705, 1711) to (1704.65, 1711.50). This center is subsequently refined to (1704.648, 1711.326) against more than 1000 spots, which demonstrates that the center refined by a few ellipses is already quite accurate. See 12.1.2.3 for more discussion on the error tolerance of initial direct-beam center. Finally, some nodal spots are found. Even if indexing is unsuccessful, these information may be useful in manual indexing.

This pattern was indexed without any struggle, and refined to a very low r.m.s.d. residual of 20 μm from 1238 spots. Figure 12.1.2.1.1 shows the predicted pattern after refinement. Accurate prediction lays a solid foundation for spot integration.



```
An overall mean profile is recognized.
Semi-major & -minor axes (pixel): 2.7398 1.93204
Non-elliptical correction:      0.00750808 0.0213153 0.0429673
0.00486854
Non-Gaussian correction:       0.918543 0.821306
R.m.s.d. (detector count):     0.000264097
```

```
Overall spot length is set to 8 pixels.
Overall spot width is set to 6 pixels.
Estimated crystal dimension is 0.579033 mm.
Estimated mosaic spread in FWHM is 0.00856099 degree.
```

...

```
Direct-beam center is set to 1704.65, 1711.5 in pixel.
```

...

```
72 Nodals Recognized (Ordered by Rank)
```

			Center (pixel)		Intensity	Sigma(I)
0	0	0	1839.44	1729.09	127480.8	628.4
0	0	0	1509.85	2002.07	10219.2	185.5
0	0	0	1190.44	2027.94	60649.0	434.3
0	0	0	1607.14	1641.84	353385.2	1035.3
0	0	0	1578.12	2209.16	34719.0	326.2
0	0	0	1264.19	1296.46	18101.8	249.0
0	0	0	1734.85	2321.03	213.5	88.0
0	0	0	2251.67	2511.87	5748.0	142.9

```

0    0    0    2307.05  1533.35    4859.0    147.5
0    0    0    1867.74  2155.45    90569.8    522.1
...

```

```

|_____)
| Report |
| -----|
| -----|
| -----|
| -----|
|_____|

```

1 possible crystal orientation is recognized;
corresponding cell constants and detector parameters are refined.

```

Indexing 1
R.M.S. deviation (pixel):          0.196704
Number of spots matched:          1238
Cell lengths (Angstrom):          93.2118    44.0000    83.5906
Cell angles (degree):             90.0000    122.0802    90.0000
Missetting matrix:               -0.06422445  0.02471138  -0.99762947
                                   -0.04430840  0.99863688   0.02758878
                                   0.99695135  0.04597524  -0.06304198
Goniometer omega, chi, phi(degree): 0.0000    0.0000    42.0000
Omega-axis polar orientation (deg): 90.0000    0.0000
Crystal-to-detector distance (mm): 180.0000
Direct-beam center (pixel):        1704.6480  1711.3262
Pixel size (mm):                   0.1000042  0.1000000
Detector swing angles (degree):     0.0000    0.0000
Detector tilt angles (degree):      0.1024    0.0627
Detector bulge corrections:         -1.212e-06  4.493e-10

```

Listing 12.1.2.1.3 Part of index.log.

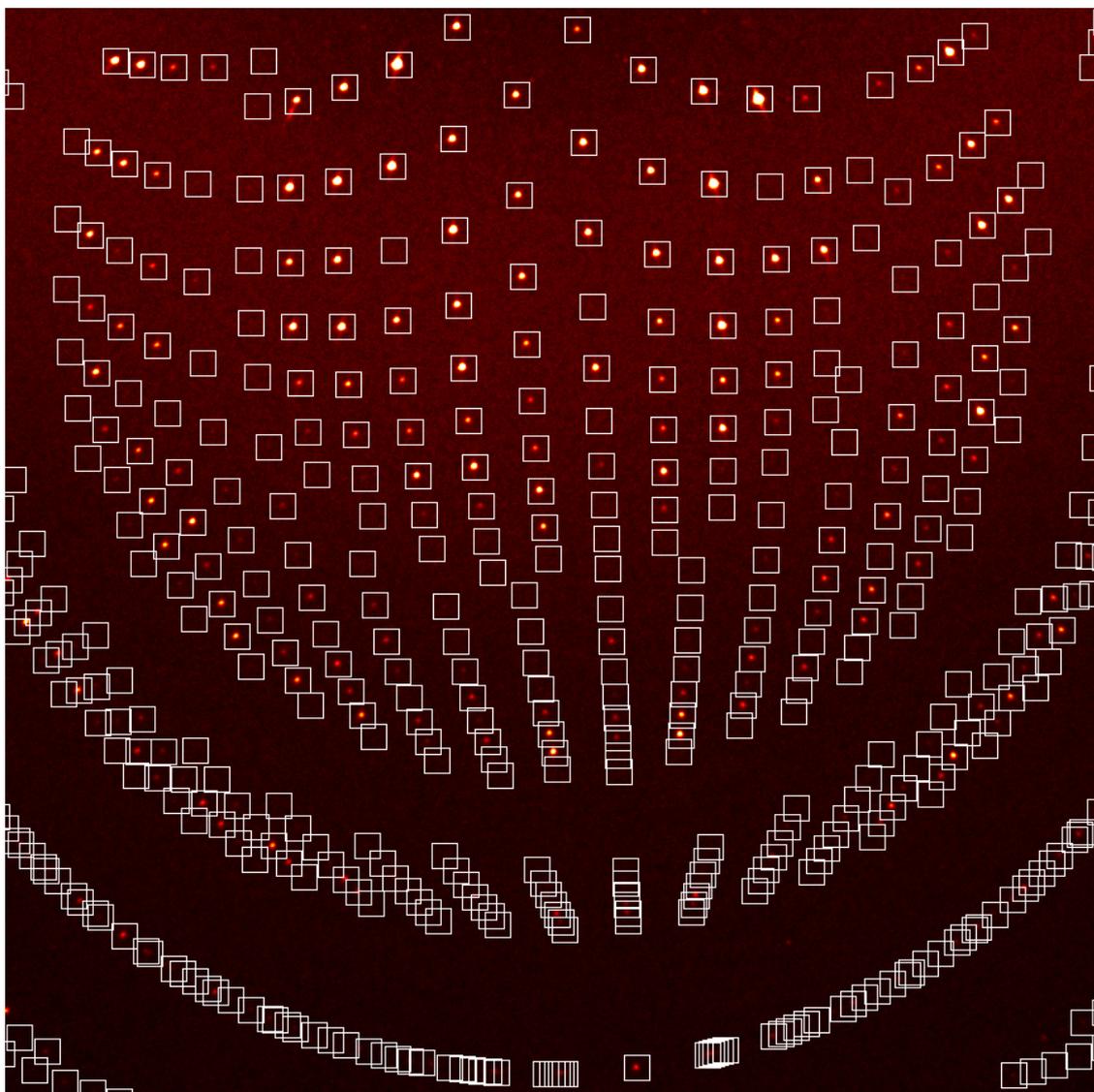


Figure 12.1.2.1.1 Predicted pattern displayed over real image.

12.1.2.2 Estimation of more soft limits

Once a pattern is indexed and refined, some more soft limits, even the λ -curve, can be estimated. This is done by the same command `Limits`, however, there are a number of differences in the command script. First of all, the non-frame-specific parameter file obtained from indexing shall be loaded, instead of the commands `Distance`, `Center`, and `Pixel`. Second, goniometer setting must be included, since the non-frame-specific parameter file does not have the frame-specific information. For the same reason, image must be loaded explicitly. Third, spot size and σ -cut shall be updated with the latest values. In addition, a filename can be supplied as a string argument to the command `Limits` for saving the estimated λ -

curve. If a frame-specific parameter file is given, goniometer setting is no longer needed.

```

diagnostic    off
@ m37v_1a_015.pre.spt.inp
Input
  Omega      90 0
  Goniometer 0 0 42
  Format     Mar345
  Image     images/m37v_1a_015.mar3450
  Wavelength 1 1.5 1.1
  Quit
Spot         8 6 2.16
Limits      1.5 estimate.lam
Quit
    
```

Listing 12.1.2.2.1 limit2.inp, estimation of soft limits after indexing.

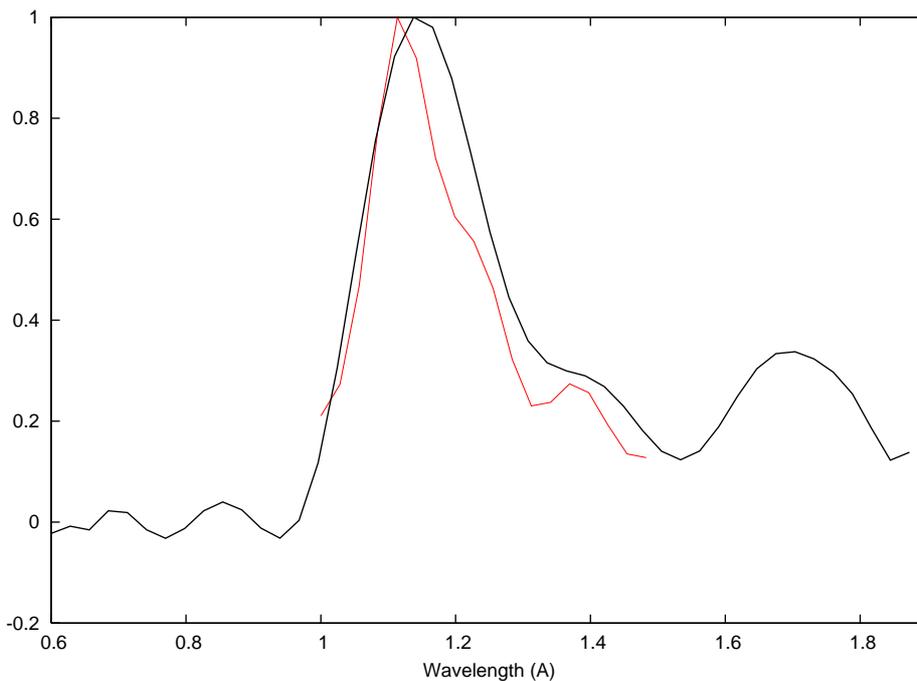


Figure 12.1.2.2.1 Estimated λ -curves.

The results are similar to those in Listing 12.1.1.1.2, except all values may be updated due to the newly specified spot size. The estimated λ -curves are shown in Figure 12.1.2.2.1. λ_{\max} can be specified by the numerical argument to the command `Limits`. The curve with narrower range

shown in red is more accurate. From the λ -curves, better reference wavelength may be read.

12.1.2.3 Indexing problems

We have a pattern in a set indexed and refined. We have very good idea about many soft limits. We are ready to refine each pattern in the set. However, let's deviate from the main flow a bit in this section, and look at some potential problems in indexing.

The most common source of error is uncertainty in direct-beam center. A test shows that the radius of convergence is 5 pixels around the correct center. See 2.5 for discussions on various options after a mis-indexing. I present below an example of user-supplied nodal spots that can be used to correct mis-indexing. m37v_1a_015.mar3450 (Figure 12.1.1.0.3) was mis-indexed when the initial center deviates too far from the correct coordinates, however, 70 nodal spots were found (Figure 12.1.2.3.1). The nodal spot file can be manually edited to select the most significant nodal spots. Selected spots shall be listed at the beginning of the file (Listing 12.1.2.3.1). This file can be loaded instead of running the auto-recognition. The rest of the indexed ran smoothly.

```

0 0 0 1264.19 1296.46 18101.8 249.0
0 0 0 2040.03 1329.59 21089.0 263.1
0 0 0 2138.23 1921.38 135509.0 640.3
0 0 0 1190.44 2027.94 60649.0 434.3
0 0 0 2251.67 2511.87 5748.0 142.9
0 0 0 2307.05 1533.35 4859.0 147.5
0 0 0 1668.85 921.73 6006.0 153.3
0 0 0 2827.23 1495.98 426.0 60.9
...

```

Listing 12.1.2.3.1 m37v_1a_015.ndl.spt, manually edited nodal spots.

```

diagnostic off
busy off
Input
@ m37vHbI-CO.inp
Format Mar345
Distance 180 fix
Center 1706 1716
# correct center is 1705 1711
Pixel 0.1 0.1
Goniometer 0 0 42
Image images/m37v_1a_015.mar3450
Resolution 2.0 100
Wavelength 1.0 1.5 1.1
Spot m37v_1a_015.ndl.spt
Quit

```

```
Spot      8 6 6 m37v_1a_015.re.spt
Ellipse
Pattern   m37v_1a_015.man.spt
Quit
```

Listing 12.1.2.3.2 `index_man.inp`, script for manual indexing.

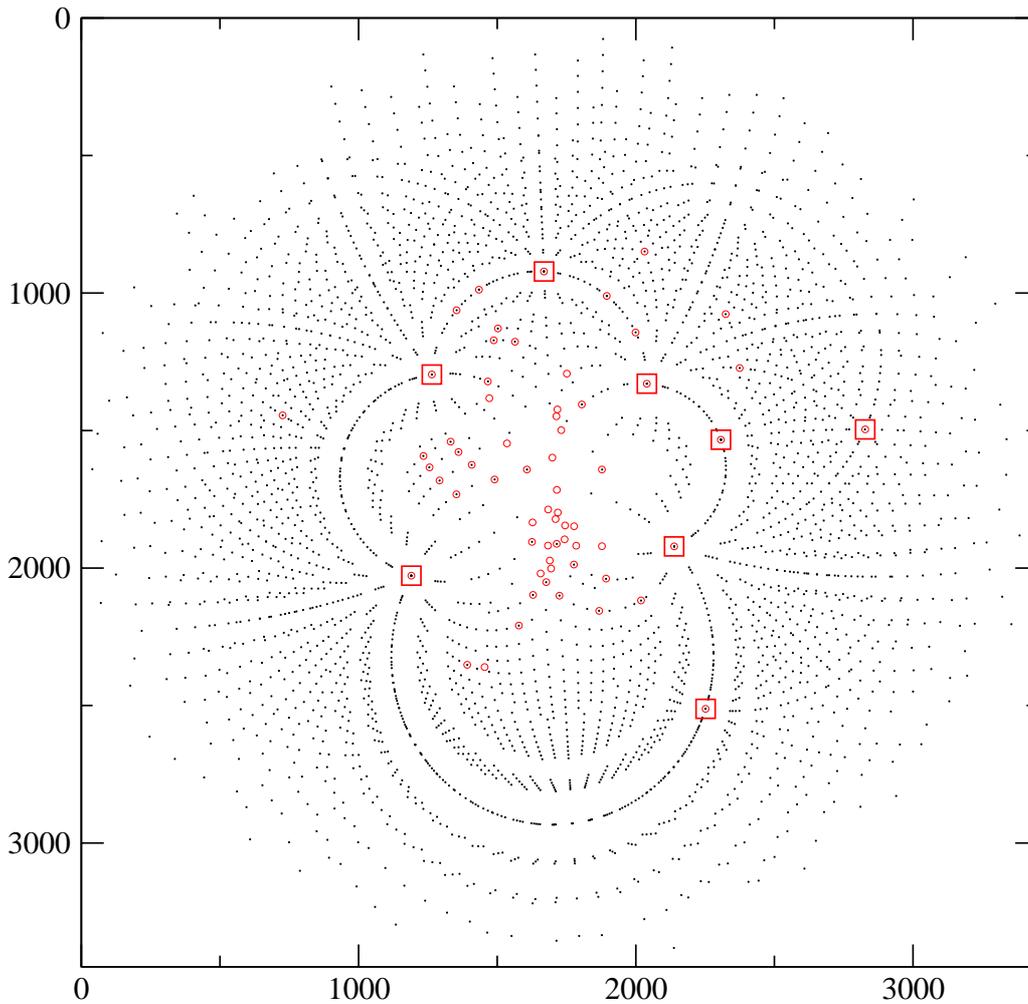


Figure 12.1.2.3.1 Auto-recognized nodal spots marked in red circles. Manually selected ones are marked in red squares.

12.1.3 Geometric refinement

Pass a and b can be refined together or separately. In case of slipping crystal, refinement of pass a can even be broken into two parts, one before the indexed

pattern in reverse order and another after. Listing 12.1.3.0.1 is an example of the script file.

```
diagnostic      off
busy            off
@ m37v_1a_015.pre.spt.inp
Input
  Crystal       0.05 0 0.05 0 0.1 0 free
  Format        MAR345
  Distance      fix
  prompt       off
  result       off
Goniometer 0 0 42 m37v_1a_015.mar3450
Goniometer 0 0 36 m37v_1a_013.mar3450
Goniometer 0 0 30 m37v_1a_011.mar3450
Goniometer 0 0 24 m37v_1a_009.mar3450
Goniometer 0 0 18 m37v_1a_007.mar3450
Goniometer 0 0 12 m37v_1a_005.mar3450
Goniometer 0 0 6 m37v_1a_003.mar3450
Goniometer 0 0 0 m37v_1a_001.mar3450
  prompt       on
  result       on
  Resolution 2 100
  Wavelength 1 1.5
  Spot         8 6 2.3
  Quit
Dataset        progressive
  In           images
  Quit
Quit
```

Listing 12.1.3.0.1 A command script that refines patterns before the indexed one in reverse order. Such order and progressive mode is necessary for refinement of slipping crystals.

```
diagnostic      off
busy            off
@ m37v_1a_015.pre.spt.inp
Input
  Crystal       0.05 0 0.05 0 0.1 0 free
  Format        MAR345
  Distance      fix
  @ m37v_1a.inp
  @ m37v_1b.inp
  Resolution 2 100
  Wavelength 1 1.5
  Spot         8 6 2.3
  Quit
Dataset        progressive
  In           images
  Quit
Quit
```

Listing 12.1.3.0.2 refine.inp, command script that refines pass a and b together.

It is recommended to refine cell constants under certain confinement, and not to refine distance at all. Two script files `m37v_1a.inp` and `m37v_1b.inp` define all goniometer settings of pass a and b, respectively. These files replace the deprecated `.gon` file. The results of the refinement include a set of parameter files for all frames.

A process of final refinement is optional. It updates each parameter file with the further refined values. One may choose to fix cell constants and perhaps use higher resolution limit for final refinement.

```
diagnostic      off
busy            off
prompt          off
result          off
# pass a
@ m37v_1a_001.mar3450.inp
@ m37v_1a_003.mar3450.inp
...
@ m37v_1a_061.mar3450.inp

# pass b
@ m37v_1b_001.mar3450.inp
@ m37v_1b_003.mar3450.inp
...
@ m37v_1b_061.mar3450.inp
prompt          on
result          on

Input
  Crystal       fix
  Distance      fix
  Resolution    1.8 100
  Wavelength   1 1.5
  Spot          8 6 2.3
  Quit
Dataset        final
  In           images
  Quit
Quit
```

Listing 12.1.3.0.3 `final.inp`, command script of final refinement.

The list of geometric `.inp` files loaded to the script above can also be isolated in a file. I name two files for both passes `m37v_1a` and `m37v_1b`. From here on, we use these files.

12.1.4 Integration

The command script for integration in Listing 12.1.4.0.1 is very similar to that of final refinement (Listing 12.1.3.0.3). An initial λ -curve `outline.lam` is

loaded by the command `Input:Image`. This λ -curve will be used to filter out high resolution reflections at both wings of the spectrum according to resolution-dependent bandwidth. The previously estimated λ -curve `estimate.lam` shall serve the same purpose. The command `Input:Spot` specifies a spot size that initializes crystal mosaicity. These values may be updated automatically under some integration modes. The σ -cut is used during selection of sample spots for profile fitting. Appropriate value of σ -cut ranges as the same as that for indexing and refinement (Listing 12.1.1.0.2), but usually toward the greater end and preferably greater than 3. The saved `.re.spt` files can be used to check the suitable σ -cut. This process saves a set `.ii` files that contain integrated intensities.

```
diagnostic      off
busy            off
warning        off

@ m37v_1a
@ m37v_1b

Input
  Image         outline.lam
  Spot          8 6 3.9
  Quit

Dataset         linearAnalytical
  In            images
  Out           linearAnalytical
  Resolution    1.6 100
  Wavelength    1 1.5
  Quit

Quit
```

Listing 12.1.4.0.1 `integrate.inp`, script for integration.

12.1.5 Wavelength normalization and scaling

From this step on, the rest of the tasks are carried out by another program `Epinorm` (11.2), but command script has the same style. Using the previously estimated λ -curve as an initial is a good idea, if it looks all right. Even an initial λ -curve is given, explicitly specifies wavelength range, reference wavelength, and order of Chebyshev approximation are usually necessary, although the program default values are available. The given spot size initializes crystal mosaicity. The command `Restore` specifies a filename for saving the parameter set of scaling. If further scaling is desired, this file can be loaded back. Listing 12.1.5.0.2 gives an example of such script. This script scales linear anisotropic factors, and overwrites the previous parameter file `m37v_ab.inp`. The λ -curves are shown in Figure 12.1.5.0.1.

```
diagnostic    off
busy          off
warning       off

@ m37v_1a
@ m37v_1b

Input
  Image       estimate.lam
  Wavelength  1 1.5 1.1
  Chebyshev   32
  Spot        8 6
  Quit
Restore      m37v_ab.inp
Scale        2
Lambda       m37v_ab.lam
Quit
```

Listing 12.1.5.0.1 scale.inp, script for scaling.

```
diagnostic    off
busy          off
warning       off

@ m37v_1a
@ m37v_1b

prompt        off
result        off
@ m37v_ab.inp
prompt        on
result        on

Restore      m37v_ab.inp
Scale        2 1
Quit
```

Listing 12.1.5.0.2 Command script for loading previous results and scaling again.

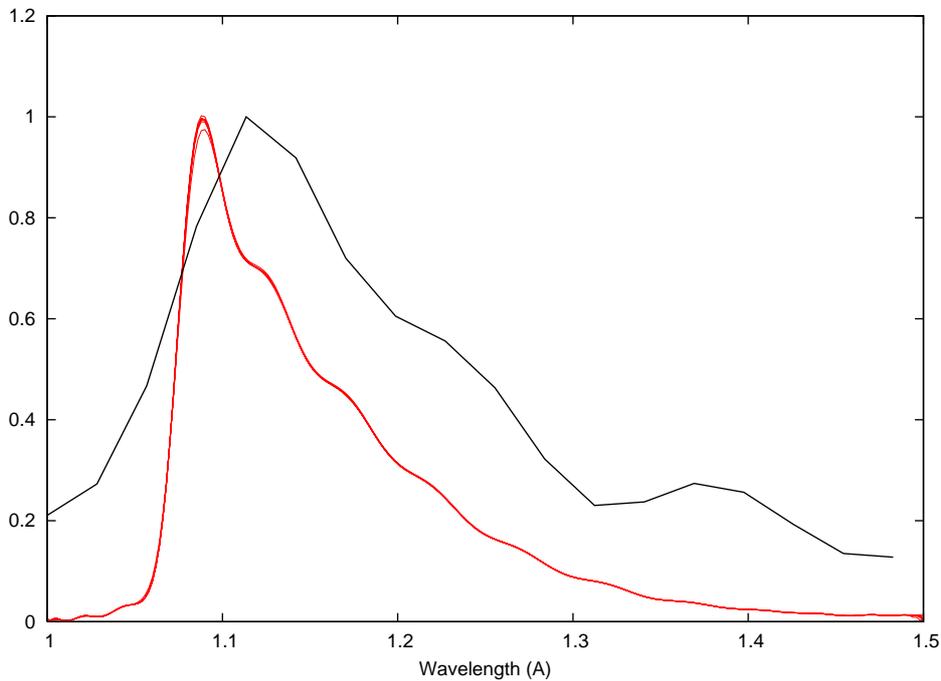


Figure 12.1.5.0.1 λ -curves derived from scaling in red and the initially estimated λ -curve in black.

12.1.6 Data merging, harmonic and spatial deconvolution

Since harmonic and spatial overlap deconvolution takes advantage of the intrinsic high redundancy of Laue dataset, these deconvolutions become special cases of data merging. This job is carried out by the script in Listing 12.1.6.0.1. Resolution and wavelength ranges of accepted data can be given here. Spot size influences spatial overlap deconvolution. A series of commands `Apply` accept data at difference σ -cuts. Merging statistics are shown in Table 12.1.6.0.1.

```

diagnostic      off
busy            off
warning         off

@ m37v_1a
@ m37v_1b

prompt         off
result         off
@ m37v_ab.inp
prompt         on
result         on
    
```

```

Input
  Resolution 1.6 100
  Wavelength 1.05 1.4
  Spot      8 6
  Quit
Apply      0.0 m37v_ab.0.0.hkl
Apply      0.5 m37v_ab.0.5.hkl
Apply      1.0 m37v_ab.1.0.hkl
Apply      1.5 m37v_ab.1.5.hkl
Apply      2.0 m37v_ab.2.0.hkl
Quit
    
```

Listing 12.1.6.0.1 apply.inp, script for data merging, harmonic, and spatial overlap deconvolution.

σ -cut	Single			Harmonic			Spatial overlap			S/N
	$R_{\text{model}}/R_{\text{merge}}$	II	UR	R_{model}	II	UR	R_{model}	II	Case	
0.0	0.069/0.068	185063	32166	0.061	49745	3672	0.149	907	58	36.8
0.5	0.062/0.060	175505	31567	0.061	49371	3639	0.146	901	58	40.3
1.0	0.053/0.044	150534	29913	0.061	48704	3600	0.143	782	49	49.7
1.5	0.048/0.036	132530	28281	0.061	47835	3567	0.144	749	49	57.5
2.0	0.044/0.031	119417	26831	0.061	46687	3507	0.140	704	47	64.2

Table 12.1.6.0.1 Merging statistics at different σ -cuts. II is integrated intensity. UR is unique reflection. S/N is signal-to-noise ratio.

12.1.7 Structure refinement

In order to test the data quality, the merged structure factor amplitude set at a completeness of 85.9% was used to refine the structure of the dimeric hemoglobin, and resulted in R_{cryst} of 0.147 and R_{free} of 0.167. The r.m.s.d. from standard bond lengths is 0.010 Å and from standard bond angles is 0.068°. Figure 12.1.7.0.1 shows the electron density map of a heme region of the protein. These statistics and map confirm that Laue data processed by Precognition system are suitable for protein structure analysis.

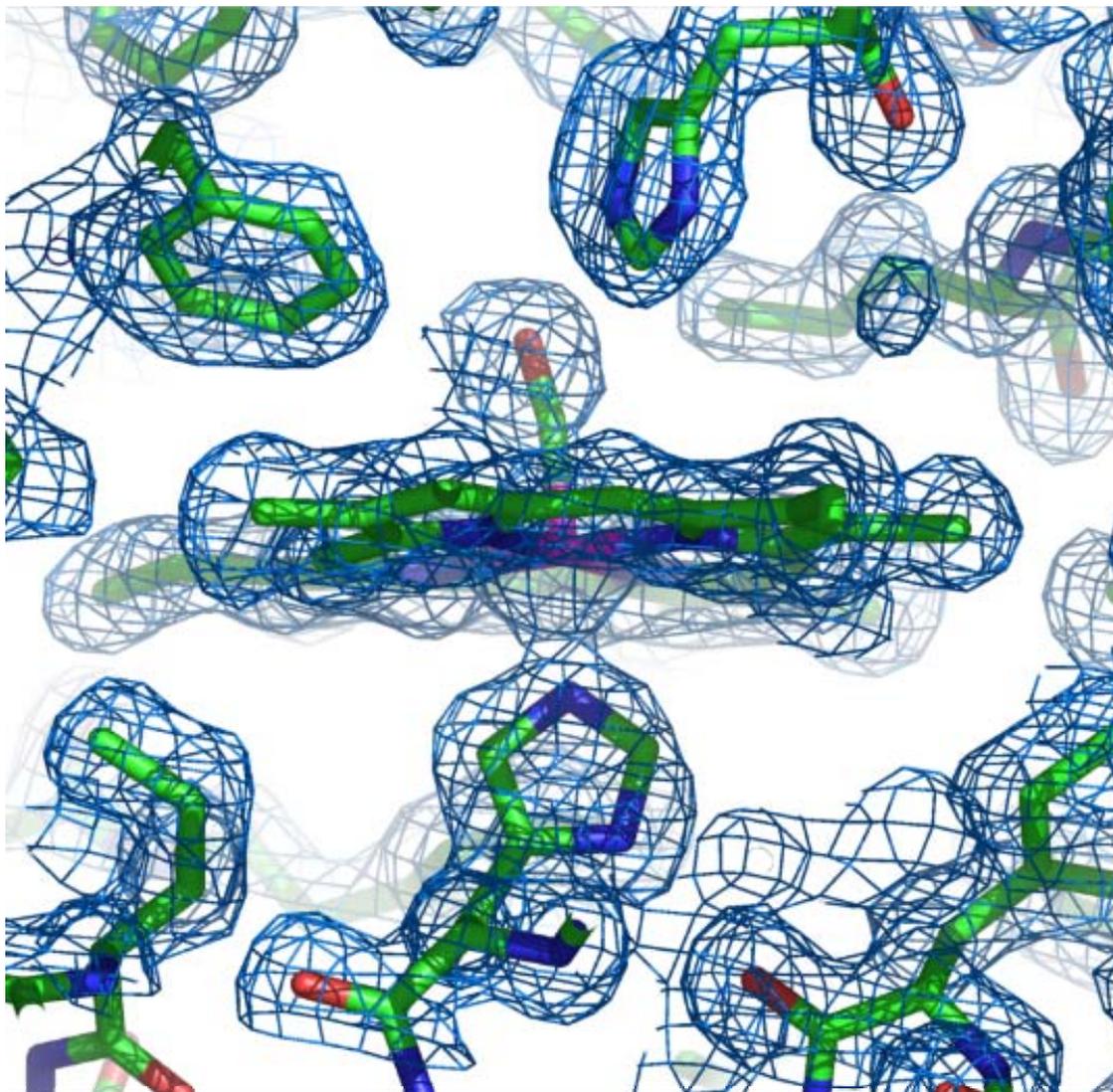


Figure 12.1.7.0.1 Electron density map calculated using $2F_o-F_c$.

12.2 A 1/10 Dataset from a Small Molecule Crystal

This dataset was collected at BL40XU beamline of SPring-8 using a helical undulator source (<http://www.spring8.or.jp/e/bl/BL40XU>). The small molecule crystal is in triclinic P1 space group. The diffraction images were recorded on a MarCCD165 detector. The entire dataset has 280 images with 1 degree angular spacing. This dataset is a courtesy of Dr. Shinichi Adachi of KEK, Japan. I demonstrate briefly the processing of 1/10 of all images available. The selected images are 10 degrees apart.

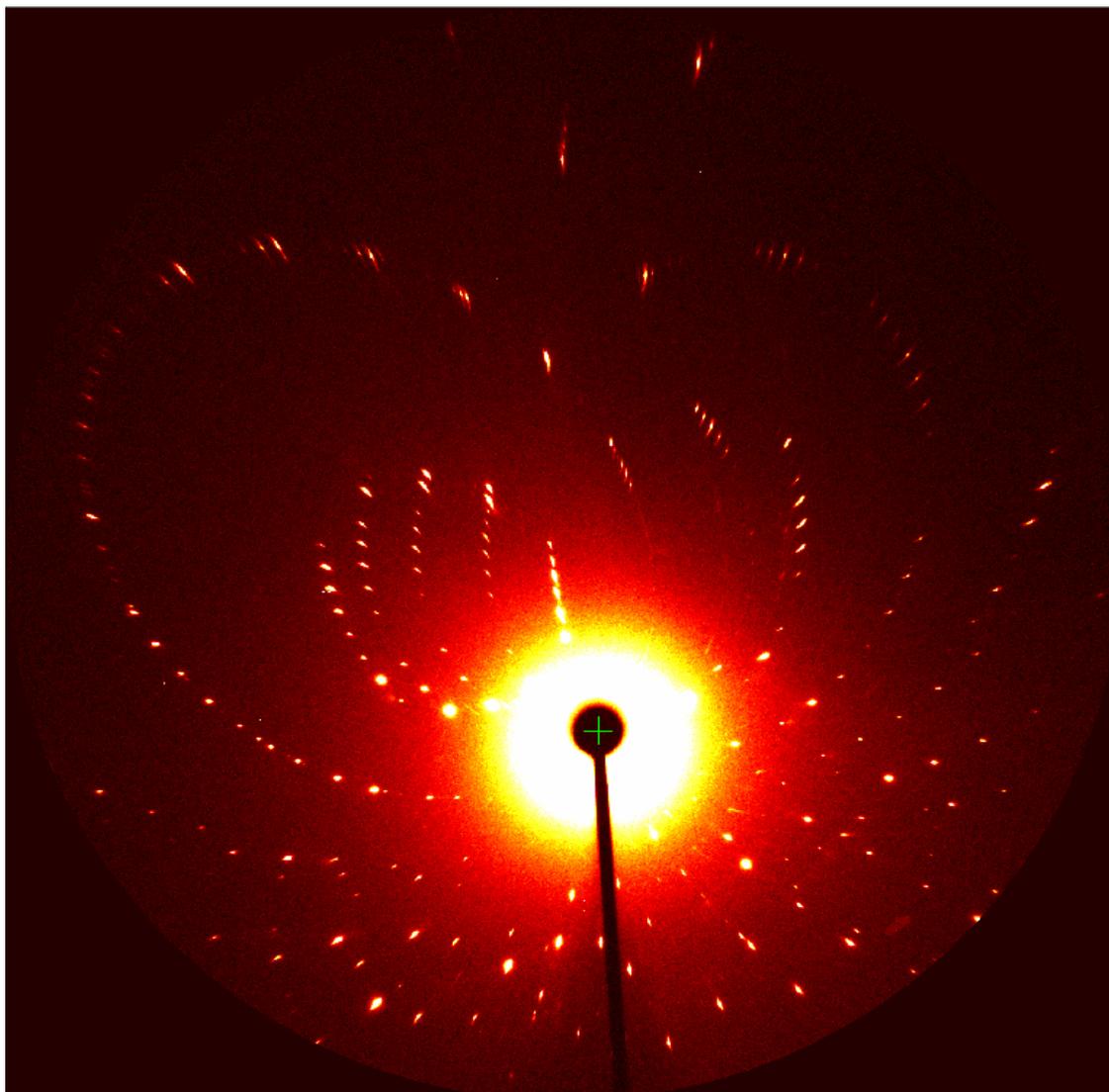


Figure 12.2.0.0.1 One of the image in the dataset.

The image quality is not the best, since some of them show split and very elongated spots. The programs for detecting soft limits usually do not work well in small molecule case, because too few spots are available on each image. Also under normal detector geometry, angular coverage is often not sufficient to record the natural diffraction limit of a small molecule crystal. The condition of the statistical model used to detect soft limits is not met to yield reliable estimates. Visual inspection is therefore required to help make choice of some initial parameters. I chose 2.5 to 3.0 as a reliable σ -cut. The command `Profile` is able to find an average spot profile of 10×5 pixels. Indexing was easy by the script listed below, where resolution and wavelength ranges are initial guesses.

```
diagnostic    off
busy          off

Input
  @ bm.inp
  Format       MarCCD
  Distance    40.9
  Center      1087 1368
  Pixel       0.0792 0.0792
  Omega       180 0
  Goniometer  10
  Image       images/BM1_1_010.img
  Resolution  1 100
  Wavelength  0.75 1.5
  Quit

Spot 10 5 2.5 BM1_1_010.re.spt
Ellipse
Pattern       BM1_1_010.pre.spt
Quit
```

Listing 12.2.0.0.1 Indexing script.

Geometry refinement was done in two passes with `refine.inp` (Listing 12.2.0.0.2) and `final.inp` (Listing 12.2.0.0.3). Both scripts limit the freedom of the parameters to be refined by setting their uncertainty values, but the first pass sets more stringent limits, and the second pass moves to a higher resolution and a lower σ -cut. Distance was always fixed.

```
diagnostic    off
busy          off

@ BM1_1_010.pre.spt.inp

Input
  Crystal      0.05 0 0.05 0.1 0.1 0.1 free
  Format       MarCCD
  Distance    fix
  Center      0.5 free
  Pixel       0.0001 free
  Tilt        fix
  Bulge       0.0000001 0.0000000001 free

  @ bm1_10.goniometer.inp

  Resolution  1 100
  Wavelength  0.75 1.5 0.8
  Spot        10 5 2.5
  Quit

Dataset       progressive
In            images
Quit

Quit
```

Listing 12.2.0.0.2 First pass of geometry refinement.

```
diagnostic      off
busy            off

@ bm1_10.inp

Input
  Crystal      0.05 0 0.05 0.1 0.1 0.1 free
  Distance     fix
  Center       1      free
  Pixel        0.0001 free
  Tilt         0.05   free
  Resolution   0.9 100
  Wavelength   0.75 1.5 0.8
  Spot         10 5 2
  Quit
Dataset        final
  In           images
  Quit
Quit
```

Listing 12.2.0.0.3 Final geometry refinement.

In the command scripts above, the file `BM1_1_010.pre.spt.inp` was generated by indexing. `bm1_10.goniometer.inp` and `bm1_10.inp` are lists of goniometer settings and frame-specific parameter files, respectively.

```
# 1/10 of the dataset
prompt off
result off
Goniometer 10 BM1_1_010.img
Goniometer 20 BM1_1_020.img
...
Goniometer 280 BM1_1_280.img
prompt on
result on
```

Listing 12.2.0.0.4 List of goniometer settings.

```
# 1/10 of the dataset
prompt off
result off
@ BM1_1_010.img.inp
@ BM1_1_020.img.inp
...
@ BM1_1_280.img.inp
prompt on
result on
```

Listing 12.2.0.0.5 List of frame-specific geometric parameter files.

Spot integration with both numerical and analytical profile fitting resulted in comparable qualities. Analytical profile, especially the nonlinear version, performs better. In contrast to protein cases, resolution-dependent bandwidth is not always required, and sometime even harmful in small molecule data processing, because diffraction limit of a small molecule crystal is often very high, and not accurately estimated at the time of integration. I suggest not to load any λ -curve in integration command script.

```

diagnostic    off
busy          off
warning       off

@ bm1_10.inp

Input
# Image       integrate.lam # Disable resolution-dependent bandwidth
  Spot       10 5 3
  Quit
Dataset      nonlinearAnalytical
  In         images
  Resolution  0.7 1000
  Wavelength 0.82 1.2 0.845
  Quit
Quit

```

Listing 12.2.0.0.6 Integration without resolution-dependent bandwidth.

The first pass scaling starts with a rough initial λ -curve. A 40-term Chebyshev approximation with nonlinear λ -mapping was used. Temperature factors for all frames are usually kept at 1 in small molecule cases. The results were saved into a file `global.inp`. The second pass loads back from this file. Notice that the second pass does not load the initial λ -curve anymore, and λ -mapping is fixed.

```

diagnostic    off
busy          off
warning       off

@ bm1_10.inp

Input
Image         initial.lam
Resolution    0.7 1000
Anomalous     off
Quit

Scale
Sigma         1
Mosaicity     0 fix
Isotropy      0 scale
Isotropy      -1 temperature
Expansion     fix
Lambda-shift  free
Wavelength    0.82 1.2 0.845
Chebyshev     40
Mapping       nonlinear
Restore       global.inp
Mode          global
Quit
Quit

```

Listing 12.2.0.0.7 First pass scaling.

```
diagnostic      off
busy            off
warning         off

@ bm1_10.inp
@ global.inp

Scale
  Sigma         1
  Mosaicity     0 fix
  Isotropy      0 scale
  Isotropy      -1 temperature
  Expansion     fix
  Lambda-shift  free
  Wavelength    0.82 1.2 0.845
  Chebyshev     40
  Mapping       fix
  Restore       global.inp
  Mode          global
  Quit
```

Listing 12.2.0.0.8 Second pass scaling.

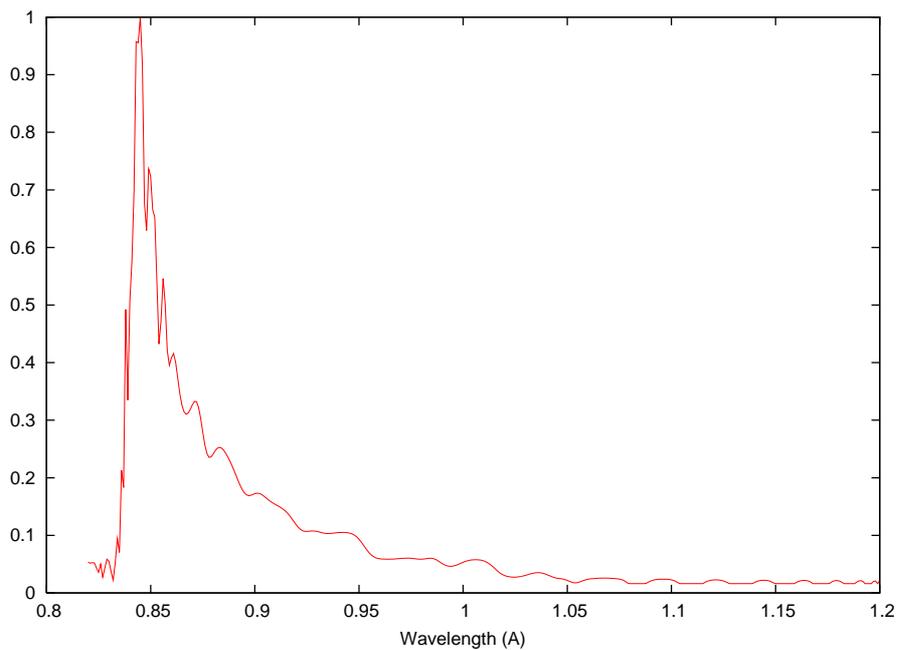


Figure 12.2.0.0.2 λ -curve derived from the 1/10 dataset.

The λ -curve is slightly noisy, but shows that very sharp peak can be modeled by nonlinear λ -mapping method. Lowering Chebyshev order to 32 may help reduce the noise, or using more images in the set would also improve the quality of λ -curve. The spectrum is about 1.6% bandwidth at half-maximum.

```

diagnostic    off
busy          off
warning       off

@ bm1_10.inp
@ global.inp

Input
  Resolution 0.7 100
  Wavelength 0.82 1.2 0.845
  Spot       10 5
  Anomalous  off
  Quit

Scale      1.0 local
  Quit

Apply     1.0 3 bm1_10+local.hkl
  Quit

```

Listing 12.2.0.0.9 Local scaling, applying scale factors and data merging.

Local scaling was done on-the-fly with data merging. Usually local scaling helps reduce the remaining noise whatever the reasons might be. The table below shows the improved statistics by local scaling. Finally, the 1/10 dataset is not very complete yet, but it nearly reaches 90% to 1 Å.

		R_{model}	R_{merge}	II	UR
w/o local scaling	Singles	12.4%	13.2%	5606	1692
	Multiples	20.7%	-	1568	255
w/ local scaling	Singles	10.0%	10.1%	5591	1684
	Multiples	18.6%	-	1570	255

Table 12.2.0.0.1 Merging statistics on F^2 .

Resolution (Å)	Unique	Observed	Completeness (%)
100.00 - 1.40	473	946	85.23
1.40 - 1.11	503	1006	93.15
1.11 - 0.97	497	994	88.91
0.97 - 0.88	391	782	72.54
0.88 - 0.82	271	542	50.65
0.82 - 0.77	160	320	28.07
0.77 - 0.73	47	94	8.88
0.73 - 0.70	5	10	0.88
100.00 - 0.70	2347	4694	53.41

Listing 12.2.0.0.10 Completeness as reported by `Epinorm`.

APPENDIX 1

README File

A README file can be found in the top directory of the distribution.

This README file is in /usr/local/rriyyyyymmdd. If not, you need to replace /usr/local with other appropriate path.

```
0) As root,
% cd /usr/local
% tar xzvf [path/]rriyyyyymmdd.tgz
% cd rriyyyyymmdd
```

You must have done these, if you are reading this file.

```
1) As root,
% cd /usr/local
% ln -s rriyyyyymmdd rri
```

```
2) As a user,
% cd
Insert these lines into your .tcshrc file:
```

```
setenv RRILICENSE      "your_home_directory/license.rri"
setenv RRI              /usr/local/rri
setenv PATH             "${RRI}/bin:${RRI}/pub/bin:${PATH}"
setenv CCLHOME         "${RRI}/ccl"
setenv CPLHOME         "${RRI}/cpl"
setenv PRECOGNITION    "${RRI}/gpr"
setenv LD_LIBRARY_PATH "${RRI}/pub/lib"
setenv PYTHONPATH      "${RRI}/pub:${CPLHOME}/..:${CPLHOME}:${CPLHOME}/ptr:${CPLHOME}/std:${CPLHOME}/mac:${CPLHOME}/ccl"
```

```
% source .tcshrc
```

3) If you have a license file, skip to 4).

```
As a user,
% cd
% python
Python 2.3.4 (#1, Oct 21 2004, 21:26:45)
[GCC 3.4.1] on linux2
Type "help", "copyright", "credits" or "license" for more
information.
```

```
>>> import cpl
```

```
|||) Welcome to CPL M0.16.0
|||\ Renz Research, Inc.
```

```
>>> cpl.license.getinfo()
```

```
SOURCE: cpl.license.getinfo
```

Ren: Precognition™ User Guide with Reference and Tutorials

TYPE: result

~~~~~  
~~~~~

To obtain a license on this host, please send file info.rri to Renz Research.

~~~~~  
~~~~~

E-mail the file info.rri to renz@renzresearch.com to request a license file.

4) Put the license file license.rri in your home directory.

```
% printenv RRILICENSE
```

Check if this points to your license file. If not, change in .tcshrc. See

2).

5) % Precognition and

```
% Epinorm
```

Refer to </usr/local/rri/doc/userGuide.pdf> or the updated manual at <http://renzresearch.com/pdf/userGuide.pdf>.

Enjoy.

You may reach me by any means:

Zhong Ren

Renz Research, Inc.

P. O. Box 605

Westmont, IL 60559

U. S. A.

Tel 630-230-0272

Fax 435-514-2645

E-mail renz@renzresearch.com

WWW <http://renzresearch.com>

Listing A1.0.0.0.1 README file.