ANATRIC
User Manual

Version: August 12, 2005
Mark Könnecke
Laboratory for Neutron Scattering
Paul Scherrer Institute
CH–5232 Villigen
Switzerland
Mark.Koennecke@psi.ch

1 Introduction

Welcome to the anatric user manual! Anatric is a program for the analysis of area detector data from a neutron four circle single crystal diffractometer. Currently the following diffractometer are supported:

- TRICS at SINQ
- D10 at ILL

Anatric's main purpose is the extraction of intensities for structure refinement from the data. The supported model of data collection is a PSD omega scan. This means that the instrument is set to some angles. Then a step scan in omega is performed for some range of omega. At each step in omega data is recorded for some time in the detector and stored. All other angles are fixed during the run. The result of such a measurement is a three dimensional dataset with the dimensions: detector-x, detector-z and omega. Anatric is not restricted to this measurement mode though. Hooks are present which allow for the implementation of further measurement modes.

Anatric has the following modes of operation:

- Adaptive peak detection mode.
- Adaptive Dynamic Mask Integration.

In **Adaptive peak detection** mode strong peaks are searched in a series of data files. Peaks are detected as local maxima fulfilling various criteria. A center of gravity is calculated for each maximum. The aim of this mode is to detect as many reflections as possible for UB matrix refinement.

The **Adaptive Dynamic Mask Integration** method integrates reflections from a given reflection list with the dynamic mask algorithm. The result is list of reflections and their intensities which then can be processed with a crystal structure analysis package.

Anatrics unique feature is hidden in the adaptive word in the description of the modes. Most common area detector data integration programs work with a fixed rectangular reflection box in which the reflection is located and integrated. The special design of the TRICS instrument now posed some problems which rendered this approach a bad choice.

- TRICS has three area sensitive detectors installed at 0, 45 and 90 degrees two-theta on a movable arm. This means that data is collected at low and high two theta values in the same measurement. With high two theta the resolution of the instrument decreases which causes reflections to shift their position on the detector by a large amount with each omega step. In order to cope with this, very large rectangular integration boxes would be required.
- On the other hand TRICS detectors are large, but not large enough. If a very large rectangular integration box would be choosen, to many well measured reflections would intersect the limits of the measured data and would be skipped in a border test.

- TRICS is a low resolution instrument. In order to prevent neighbouring reflections intruding into integration boxes and thereby skewing integration results, the smallest possible integration box should be choosen at all times.

In order to cope with these problems anatric does a couple of special things. In order to understand these two related concepts have to be understood. The first is the main axis of the reflection. This is just a line drawn through the center of the reflection in 3D. The next concept is the frame offset. This is the difference vector between the position of the center of a reflection between two consecutive frames in omega. A frame in this context is one detector picture collected at a defined omega value. In order to cope with the problems mentioned above, anatric implements two unique startegies:

- The frame offset is taken into account when integration reflections or determining their position. For integration purposes data is copied into a smaller rectangular box thereby taking into account the frame offset. This means that a oblique main axis of a reflection is aligned with the coordinate axis in this step. Anatric also has facilities to determine frame offsets in the peak detection phase.

- Anatric first tries to determine a minimum integration box size from projections of the data on all three coordinate axis. If this fails, the reflection is considered weak and a default minimum integration box is used.

The operation of anatric is controlled through a single control file in XML format. EXtended Markup Language (XML) is a standard for ASCII data files. This standard describes how to store information in a file. XML may look clumsy at first sight but has the advantage that it comes with ready made parsers for many programming languages. Also it will be possible to use standard XML editors and verification tools in order to facilitate the preparation of such files. The content of anatric's control file is described in more detail in a chapter below.

2 Starting Anatric

Running anatric is really easy. You only need the program and the control file, for example hamoglobin.xml. Then type:

```
anatric hamoglobin.xml
```

at the unix command prompt and you will be rewarded with error messages, warnings, normal program output and core dumps. Anatric is still experimental software, so the latter is really an option.

3 Algorithms Used

In order to integrate a reflection properly a couple of things are needed:

- A model of the background which has to be subtracted.

- A prediction about where the reflection is supposed to be on the detector. This is very important for reflections with very small intensities. It will almost always be possible to locate strong reflections in a dataset.

Some software components are common to both of the procedures which are currently implemented. First there is a configurable list of files to process. Then there is an abstraction for data reading and crystallographic transformations which are necessarily instrument dependent. More about this in the reference section for the configuration file. The other common component is the background treatment. The algorithm for this was lifted from Wolfgang Kabsch's[1] XDS program: Input is the dataset to analyze and a list of reflections discovered. Then for each detector pixel the value measured is summed along the omega axis, thereby keeping track of the number of pixels along omega contributing. Pixels which belong to the area of a reflection are excluded from this summation. In order to do this a box located around the reflections position is used. This box, also called window, should be large enough as to enclose the reflection. After the summation, averages and sigma's are calculated for each pixel, resulting in the background data for the detector.

3.1 Algorithm used in the Adaptive Peak Detection Procedure

The purpose of this algorithm is to provide a list of strong reflections with reasonably accurate detector positions and angle parameters for indexing and UB matrix refinement purposes. If a UB is already available, miller indeces can be assigned to reflections found by anatric as well. This step can happen without prior knowledge of crystallographic parameters. The algorithm is pretty straightforward:

- Search strong reflections through a local maximum search.
- Determine the center of gravity (COG) for each reflection found.

3.1.1 Local Maximum Search

In order to detect strong reflections a local maximum search is performed on the whole dataset. Each point in the dataset is checked in order to determine if it is a local maximum or not. A local maximum has the highest count rate in its surrounding. This is simple enough but a couple of enhancements to the method help to suppress multiple or wrong detections of maxima.

The strongest criterion is the size of the box of neighbouring pixels to test in order to verify a local maximum. This is an adjustable parameter in anatric. However large parameters for this parameter increase the processing time for the program.

In order to exclude local maxima arising from fluctuations in the background the steepness criterion is helpful. This condition is fulfilled if all pixels at the border of the test box have intensities less then the maximum intensity of the local maximum minus a parametrized value. In other words the intensity drops in all directions from the local maximum by a predefined amount.

Another obvious selection criterion is a intensity threshold for the local maximum.

With the spiky data observed at TRICS this still was not enough. Another citerion became necessary which surpresses maxima which are within a parametrized distance to a maximum already found.

3.1.2 Center Of Gravity Determination

Once a valid local maximum has been found its intensity and position on the detector are determined with more accuracy. This is done through a center of gravity (COG) calculation. For this calculation only pixels with an intensity above a certain threshold are considered in order to calculate th COG of the reflection and not of the background. This threshold is calculated as:
$threshold = bckAvg + 5 * bckSigma$
with bckAvg being the average background count and bckSigma being the sigma of the background. In order to cope with the strong reflection position shifts between frames at two theta the main axis vector is determined in this step from two consecutive frames and printed. The main axis vector thus determined is then used for offsetting frames in the center of gravity calculation.

3.2 The Adaptive Dynamic Integration Method

The purpose of this algorithm is the integration of the reflections. This algorithm copes with three problems: the first is the shift of the reflection center between frames which becomes so large at large two theta that integration within a rectangular box becomes is no longer feasible. The second problem is that the required size of the integration box varies with the reflection intensity. The third problem is that an overly large integration box will cause the data border test to fail for many observed reflections.

Possible reflections are choosen from a list of possible reflections specified by the user.

In order to cope with the shift of reflection centers between frames a main axis vector and an offset vector are introduced. The offset vector describes the shift of a reflection center from one frame to another. This offset vector is interpolated for each reflection from a table containing main axis vectors as a function of two–theta. This table has to be specified by the user. With this offset vector and a position for the reflection center the main axis vector is calculated. This is a vector of coordinates which holds the position of the reflection in each frame.

Using the main axis vector thus determined the necessary size of the integration box for the reflection is calculated. In order to do so a fairly large initial box is assumed. In a first step the limits in omega are determined. To this purpose rectangular areas the size of the inital windows x and y length are summed along the main axis vector of the reflection. The result is a histogram. Using the well known Lehman Larsen algorithm, the limits of the reflection in omega are determined from this histogram.

In a second step the position of the maximum in omega is determined. At this frame number the data is summed both in x and y. The sums go along the x and y length specified for the initial window. The center of the summed area is determined through the main axis vector. From the resulting x and y histograms the limits of the reflection are determined in x and y using the Lehman Larsen algorithm again.

To the reflection box size thus calculated a user definable border is added. As the Lehman Larsen algorithm fails for very weak reflection, this ensures that the reflection box has a certain user defined minimum size. If possible a center of gravity is calculated for the reflection and its position on the detector corrected. If this calculation fails, the predicted position of the reflection is used later on.

With the reflection position, the integration box size and a newly calculated main axis vector the raw data for the reflection is copied into a new rectangular box for further processing. The shift of the reflection center is thus corrected.

In a next step the exact peak shape is determined through a variation of the dynamic mask procedure as described by Wlodawer and Sjölin[2]. To this purpose the data within a box enclosing the reflection is smoothed using a median filter. Then an average background and a sigma background is calculated for the box enclosing the reflection. The average background is subtracted and the pixels in the reflection box are labelled according to their intensity. Pixels less then sigma background are labelled 0, those less then 2 sigma background 1, etc until three. The peak area is then determined as all the pixels with a label bigger then one which form a contiguous region around the peak center.

The mask ths determined allows to separate reflection data from background data. A new background average and sigma are then calculated from the background data in the integration box. Then the pixels belonging to the reflection are summed for integration. The formulas used are:
$I = \sum I_{pixel} - bckAvg$
$\sigma(I) = \sqrt{\sum}\sigma(I) + \sigma(bck)$
with the sums going over all pixels contributing to the reflection. bckAvg is the average background as determined above.

Summing it up the algorithm looks like this in pseudocode:

```
select reflections which could be found on the current detector and in
the current datafile.
FOREACH reflection in the possible list
interpolate offset vector
        refine reflection location and size of integration box
extract integration data from raw data thereby taking offset
             vector into account
        determine tha dynamic mask for the reflection
integrate the reflection
ENDFOR
```

4 The Anatric Configuration File

Anatrics operation is controlled through a XML configuration file. For more information about XML see the ample documentation available elsewhere. The basic framework of a anatric configuration file looks like this:

```
<?xml version="1.0" encoding="UTF-8"?>
```

```
<anatric>

</anatric>
```

All this is required by the XML mechanisms. The first line is a required header line. One fundamental building block of a XML file is a tag. A tag is some identifier enclosed in < and >. An example is anatric in the example above. Each tag must come with a closing tag. An example for a closing tag is the `</anatric>` line in the example. The area between an opening and closing tag can contain more tags or data. There is a shorthand for tags which do not have further data below them: `<blub/>` replaces `<blub> <\blub>`. Tags may have attributes associated with them. These are name=value pairs given behind the tag name within the < > pair. Please note that the value must be enclosed in quotes.

Between the anatric tags there must be some more tags. Available tags fall into three different groups:

- General tags
- Adaptive Peak Search specific tags
- Adaptive Dynamic Mask Integration specific tags.

The vast majority of tags are common to both algorithms and thus fall into the first category.

4.1 General Configuration Tags

4.1.1 Logfiles

By default anatric produces next to no output. In order to get some logging, log files can be configured. This is done with the logfile tag. This tag can have two attributes. The file attribute specifies to which file the log output will be written. The special filenames stdout and stderr redirect the output to the console. Another atrribute is verbosity which controls the amount of logging taking place. Higher values can be useful for debugging purposes. An example:

```
  <logfile file="stdout"> </logfile>
  <logfile file="snp1.log" verbosity="1"> </logfile>
```

configures the log output to go both to the console and to the file snp1.log. The verbosity is 1.

The verbosity levels form a hierarchy. This implies that if a high verbosity level is selected, the output from lower verbosity levels is printed as well. Most verbosity parameters apply to the integration algorithm only. The verbosity levels have the following meanings:

0 minimal output

5 prints reflection positions calculated from UB and detector parameters and prints pactive parameters for each file and detector.

10 print omega plot of reflection.

15 print mask and data in order to check the match.

¿30 debugging output.

4.1.2 File Lists

Typically anatric analyses a collection of related data files. A means is neded for configuring the files to process. This is done through the FileList tag. The FileList tag itself has a single required attribute, the format. This is the format string used for generating the file name from the run number. ANSI–C language formatting conventions apply. The FileList tag can have several subtags: The datapath tag gives the path to the directory where the data files are stored. The actual path is stored in the value attribute. Single files to process can be specified with the file tag. The value is then the run number of the file to process. A range of files can be specified with the range tag. The attributes start and end define the limits of the range of run numbers to process. An example:

```
<FileList format="trics%5.5d2001.hdf">
   <datapath value="/data/koenneck/src/dspreflex/data"/>
   <range start="12673" end="12703"/>
   <file value="12715"/>
</FileList>
```

This causes the TRICS files 12673 – 12703 and 12715 to be processed. Files are stored at /data/koenneck/src/dspreflex/data.

For use at SINQ there is another file list format: SinqFileList. This list format takes care of the SINQ sub directory structure which sorts data files into subdirectories holding 1000 files each. An example:

```
<SinqFileList format="%3.3d\trics%6.6dn2004.hdf">
   <datapath value="/afs/psi.ch/project/sinqdata/2004/trics"/>
   <range start="12673" end="12703"/>
   <file value="12715"/>
</FileList>
```

loads files 12673 – 12703 and 12715 from the central data file repository on AFS.

4.1.3 Crystal Parameters

Anatric needs some crystallographic information as well. While the library search phase can be run without a UB matrix, the integration phase needs a UB. Crystallographic information is specified within the crystal tag. Six sub tags are recognized in this section: The first is the sample tag with the attribute name. The second is the required tag lambda with the attribute value. This gives the wavelength to use. Then there is the UB tag which encloses the UB as character data. The other three tags recognized are the zero points for two theta, omega and chi as: zeroSTT, zeroOM and zeroCHI. Again an example:

```
  <crystal>
<sample name="Snpwhatever"/>
        <lambda value="1.179"/>
        <UB>
          -0.0178803 -0.0749231 0.0282804
          -0.0070082 -0.0368001 -0.057747
           0.1609116 -0.0099281 0.000627
        </UB>
<zeroOM value=''-1.2''/>
<zeroSTT value=''.0''/>
<zeroCHI value=''.3''/>
  </crystal>
```

4.1.4 Data File Reading

In order to analyse the data it has to be read first. This bit is also the most likely area where change will occur if anatric is ever used anywhere else outside PSI. Therefore various implementations are supported for this module as well. Moreover the data reading module is also responsible for creating an abstraction of the necessary crystallographic transformations for a given instrument. This is why many detector parameters are handled in this section as well. The general idea is that those parameters should be read from the data files as well but for TRICS a means is provided for overriding those parameters because TRICS is configuration hell. All this is handled in the DataFactory section. The DataFractory tag itself has an attribute, implementation, which selects the data reader to use.

For TRICS special tags for the configuration of detector parameters are available in the DataFactory section. As TRICS has three detectors a means is needed for distinguishing parameters specific for a detector. This is done by appending the detector number to the name of the parameter. Thus dist2 is the distance of detector 2 to the sample. The following detector parameters may be specified:

dist The distance between detector and sample.

xscale The conversion factor from detector pixels into mm for the x direction.

zscale The conversion factor from detector pixels into mm for the z direction.

xnull The x zero point of the detector in pixel.

znull The z zero point of the detector in pixel.

sttOffset The two theta offset of the detector to the value read from the motor.

frameScale Each detector has a different neutron sensitivity. Thus intensities measure on one detector will not match up with the same reflection measured on another detector. The frameScale compensates for the different detector responses.

Again an example for TRICS with detectors 2 and 3 installed:

```
<DataFactory implementation="trics">
   <dist2 value="616"/>
   <xscale2 value=".7149"/>
   <zscale2 value="1.42"/>
   <xnull2 value="128"/>
   <znull2 value="64"/>
   <sttOffset2 value="-45."/>
   <dist3 value="616"/>
   <xscale3 value=".7149"/>
   <zscale3 value="1.42"/>
   <xnull3 value="128"/>
   <znull3 value="64"/>
   <sttOffset3 value="-45."/>
   <frameScale value="2.0"/>
</DataFactory>
```

For the ILL-D10 diffractometer the implementation attribute must be d10. D10 has only one detector. The following special parameters are supported for D10:

distance The sample detector distance

xsize The size of detector pixels in the x direction

ysize The size of detector pixels in the y direction

An example of a configuration for D10:

```
<DataFactory implementation="d10">
   <distance value="616"/>
   <xsize value="2."/>
   <ysize value="2."/>
</DataFactory>
```

There are common parameters:

forcedChi Sometimes thieves steal the eulerian cradle. Then chi and phi are undefined. In such cases this tag enforces a particular chi value. The tag takes a value attribute which specifies the angle to use.

forcedPhi Same as above but for phi.

4.1.5 Background Processing

Anatric performs as a preliminary step a global estimation of the backgound for each detector and datafile. Again, the implementation of the algorithm used for this may change, therefore the algorithm for background processing can be selected in the BackgroundProcessor tag. The attribute implementation selects the algorithm to use. Currently only xds is supported which implements the background algorithm used in Wolfgang Kabsch's XDS[2] program. Example:

```
<BackgroundProcessor implementation="xds"/>
```

4.2 Detector Efficiency Coorection

The efficiency with which neutrons are detected on a detector can vary wildly. In order to compensate for this an efficiency correction needs to be performed. This is quite simple, it is just the normalisation of the counts to a standard value, but it requires another data file which holds the correction parameters. These files and their association are given within the DetectorEfficiency tags. The DetectorEfficiency tag has again an attribute implementation which specifies the format of the correction file. Currently only **trics** is supported for this parameter. Within the detector efficiency correction there are use tags which specify which correction file to use for which detector. The correction file is specified in the filename attribute, the detector for which the file is valid in the nDetector attribute. See an example below.

```
<DetectorEffficency implementation="trics">
   <use filename="shadenov4d2.dat" nDetector="2" />
   <use filename="shadenov4d3.dat" nDetector="3" />
</DetectorEfficency>
```

There is no detector efficiency correction implementation for D10. The philosophy is that the reflection is always at the centre of the detector and any scalings are taken care of by the overall intensity scaling factor during refinement.

4.2.1 Result Printing

At the end of the day a list of reflections must be printed in some format. The format and the name of the file to which to print this list is selected through the ReflectionPrinter tag. The attribute **file** determines the filename, the attribute **format** the format of the file to print. The following formats are currently supported:

rafin, rafini A format as understood by the ILL–RAFIN program for UB matrix refinement. A UB matrix is required for this. Miller indices are printed as integers.

rafinf A format as understood by the ILL–RAFIN program for UB matrix refinement. A UB matrix is required for this. This outputs miller indices in floating point format.

rafin2d A format as understood by an obscure version of the ILL RAFIN program for 2D detectors. A UB matrix is required for this.

rafin2di As above for RAFIN 2D but with miller indices rounded to the closes integer.

orient A list of four circle angles (two theta, omega, chi, phi) and intensity which will be understood by most common indexing aand UB refinement programs.

shelx A reflection list for the SHELX76 structure determination package.

jana2k A reflection list in the format understood by the Jana2000 structure refinement program. Miller indices are printed as integers.

jana2kf A reflection list in the format understood by the Jana2000 structure refinement program. Miller indices are printed as floating point numbers.

raw A reflection list with X,Y,Z,I, SIG(I), H, K, L all in floating point format. Some indexing programs take X,Y,Z lists as input (XYZ is Z4 is h*UB). This can also be useful for debugging.

oksana A reflection list holding: frame number, I, sig(I), detector number, temperature and filename. To be uesed when the same reflection has measured as a function of some physical parameter.

Again an example, resulting in RAFIN output:

```
<ReflectionPrinter format="rafin" file="rafin.dat"/>
```

A Lorenz correction is performed for all output modes. When converting miller indices to integer a threshold of .2 is observed. Reflections for which any miller indiex deviates by more then .2 from integer will not be printed into the reflection list. Instead an error message will appear in the log file.

Obviously anatric needs to be told which algorithm to use for processing. This is done through the implementation attribute of the Algorithm tag. The possible values are available for this tag:

adaptivemaxcog For the adaptive peak search mode.

adaptivedynamic For the adaptive dynamic mask integration mode.

Within the Algorithm section there are further algorithm specific tags for parameters.

4.3 Adaptive Peak Search

This algorithm is invoked by specifying **adaptivemaxcog** as value for the implementation attribute of the algorithm tag.

This algorithm is configured through the following parameters tags:

threshold The intensity threshold for accepting a reflection as strong.

shell The size of the box checked for a local maximum.

steepness The steepness required for a valid local maximum.

duplicateDistance The distance in pixels within which adjacent local maxima are considered to belong to a single peak.

maxEqual It happens sometimes that the reflection maximum is more like a ridge of equal values. MaxEqual is the maximum count of equal values accepted for a valid local maximum.

window The size of the box in which to search the boundaries of the reflection. This tag has the attributes x, y and z for the size in each direction.

An example input file for adaptive peak dectection is given below:

```
<?xml version="1.0" encoding="UTF-8"?>
<anatric>
<logfile file="oksanaub.log" verbosity="21"/>
<logfile file="stdout"/>
<FileList format="trics%5.5d2002.hdf">
  <datapath value="/data/lnslib/data/TRICS/2002"/>
  <range start="1819" end="1822"/>
</FileList>
<crystal>
<Sample name="CeB6"/>
<lambda value="1.179"/>
     <UB>
       0.2135097    -0.0798488    -0.0792820
       0.1125218     0.1506545     0.1512942
      -0.0005655    -0.1708077     0.1705059
     </UB>
     <zeroSTT value="0.7674"/>
     <zeroOM value="2.272"/>
</crystal>
<DataFactory implementation="trics">
   <dist1 value ="545.00"/>
   <dist2 value ="545.00"/>
   <dist3 value ="545.00"/>
   <xscale1 value ="-0.7408"/>
   <zscale1 value ="1.4864"/>
   <xscale2 value ="-0.7178"/>
   <zscale2 value ="1.4838"/>
   <xscale3 value ="-0.7851"/>
   <zscale3 value ="1.4864"/>
   <xnull1 value ="128.00"/>
   <znull1 value ="64.000"/>
   <xnull2 value ="128.00"/>
   <znull2 value ="64.000"/>
   <xnull3 value ="128.00"/>
   <znull3 value ="64.000"/>
   <sttOffset1 value="0.00"/>
   <sttOffset2 value="0.80"/>
   <sttOffset3 value="1.40"/>
   <framescale2 value=".5"/>
 </DataFactory>
<BackgroundProcessor implementation="xds"/>
<Algorithm implementation="adaptivemaxcog">
   <threshold value="20"/>
   <shell value="11"/>
   <maxequal value="3"/>
   <steepness value="3"/>
   <duplicateDistance value="20"/>
   <window x="40" y="40" z="30"/>
</Algorithm>
<ReflectionPrinter format="raw" file="ceb6.raw"/>
</anatric>
```

4.4 Adaptive Dynamic Integration Specific Parameters

This integration method is selected by choosing “adaptivedynamic” as a value for the implementation attribute to the Algorithm tag. Further tags include:

window The size of the box in which to integrate the reflection. This tag has the attributes x, y and z for the size in each direction.

border The width of the border to add to the determined reflection limits. This tag has the attributes x, y and z for the size in each direction.

minWindow This is the minimum box size to use for integration. Good values for this can be obtained by scanning the full log file output for weak reflection which still have been processed correctly.This tag has the attributes x, y and z for the size in each direction.

reflectionFile A file with a list of all possible reflections for this crystal. Can include non integral indices. The filename is specified in the filename attribute.

targetMonitor Anatric may be processing files wich have been measured with different monitor presets. The targetMonitor is the monitor value to which all integrated intensities will then be scaled. The targetMonitor can also be used to scale the integrated intensities to values compatible with a given reflection file format.

smoothSize Ths size of the box used for median smoothing in the dynamic mask algorithm. Set to the smallest value giving a reasonable smoothing.

loop The number of runs performed while searching for a contiguous area of pixels. Increase if parts of a reflection are not caught in the mask area.

minPeakCount The minimum amount of pixels contributing to an contiguous island required before the island is considered integration worthy.

displacementCurve Within this tag offset vectors can be specified for two theta values. This tag supports subtags, the map tag for specifying one offset value. This map tag has attributes twotheta, x and y. twotheta is th two–theta value for which the offset vector is defined, x and y are the offsets in x and y from one frame to another.

A complete example of a control file for an integration run with this method is given below:

```
<?xml version="1.0" encoding="UTF-8"?>
<anatric>
  <logfile file="stdout"> </logfile>
  <logfile file="intest2.log" verbosity="22"> </logfile>
  <FileList format="trics%5.5d2001.hdf">
     <datapath value="/data/koenneck/src/dspreflex/data"/>
     <file value="12681"/>
     <range start="12674" end="12703"/>
  </FileList>
  <crystal>
<lambda value="1.179"/>
        <sample name="Whatever Oxide"/>
        <UB>
          -0.0168477    -0.0751086     0.0280916
          -0.0074780    -0.0365193    -0.0578417
           0.1610018    -0.0095558     0.0002530
       </UB>
  </crystal>
  <DataFactory implementation="trics">
     <dist2 value="616"/>
     <xscale2 value=".7596"/>
     <zscale2 value="1.5099"/>
     <xnull2 value="128"/>
     <znull2 value="64"/>
     <sttOffset2 value="-45."/>
     <frameScale2 value="1.0"/>
     <dist3 value="616"/>
     <xscale3 value=".7293"/>
     <zscale3 value="1.4756"/>
     <xnull3 value="128"/>
     <znull3 value="64"/>
     <sttOffset3 value="-45."/>
     <frameScale3 value="1.65"/>
  </DataFactory>
  <BackgroundProcessor implementation="xds"/>
  <DetectorEfficency implementation="trics">
     <use filename="shadenov4d2.dat" nDetector="2" />
     <use filename="shadenov4d3.dat" nDetector="3" />
  </DetectorEfficency>
  <Algorithm implementation="adaptivedynamic">
     <window x="45" y="40" z ="40"/>
     <minWindow x="23" y="20" z ="17"/>
     <border x="6" y="4" z ="4"/>
     <reflectionFile filename="snbtest.hkl"/>
     <targetMonitor value="10000"/>
     <minPeakCount value="10"/>
     <maxBorderCount value="20"/>
```

```
      <smoothSize value="5"/>
      <loop value="1"/>
      <displacementCurve>
        <map twotheta="23.0" x="-1" y="0"/>
        <map twotheta="45.0" x="-.0" y="0"/>
        <map twotheta="68.0" x="-1.7" y="0"/>
        <map twotheta="107.0" x="-2.5" y="0"/>
      </displacementCurve>
  </Algorithm>
  <ReflectionPrinter format="raw" file="intest2.raw"/>
</anatric>
```

5 Usage Hints

Anatric is quite sensitive to all these parameters being right. Moreover, anatric has no chance to guess proper values for many of those parameters. Especially the crystallographic transformations can only work properly when all the detector geometry parameters, lambda and the UB are good. So triple check those parameters before complaining and be assured that these transformations have been verfied to work properly if accurate parameter values have been given.

In adaptive dynamic mask integration mode it is useful to set the logfile verbosity to > 15 and check the masks and positions found for the reflections very carefully. Again, the window size is the most crucial parameter.

6 Bugs

Anatric surely has bugs. May be the algorithms used are not even appropriate to the problem. If anatric dumps core on you please use the following procedure:

- First inspect the data file causing the problem visually. Problems due to horrible data may never be fixed.
- Try to exclude the offending file from processing and continue with the remaining data.
- If the data file is good forward the problematic file and your configuration file to anatrics programmers for further bug analysis.

7 References

1. Kabsch W. (1988) J. Appl. Cryst. 21, 916–924
2. Sjölin, L. & Wlodawer, A. (1981) Acta Crsyt. A37, 594–604