Version: `$Id$`

1 Introduction

Normal crystallographic conventions are used for all crystal variables and parameters, *e.g.* unit cell, space group, etc.

The coordinate frame used in the DNA project is that defined for Mosflm (see the Mosflm User Guide, Appendix IV, reproduced here as Appendix I). There are simple one-to-one conversions between this frame and those used by other data processing programs.

In addition to this, there are variables and parameters which need to be passed between Mosflm and the Server, and the Server and its client programs (e.g. the Mosflm GUI or the ProDC or PXGen++ expert systems). Broadly speaking, these fall into two distinct categories, *viz.* those concerned with automatic processing of images and those used where automatic processing has failed and some intervention is required.

The different types of input parameters will be marked thus;

- • required input
- ◦ required input (ProDC only, assuming that all information comes from a database, not from the image headers)
- ⋆ recommended extra input which may be omitted
- ▷ optional input, usually not required
- ◇ optional input, usually not recommended

For the sake of brevity, "expert system" is used to mean either or both the ProDC and PXGen++ expert systems as and where appropriate.

The units of measurement are as follows:

1. Macroscopic and microscopic linear dimensions (i.e. > 0.001mm) are quoted in mm
2. sub-microscopic linear dimensions are quoted in Ångströms (Å).
3. All angles are quoted in degrees (°).

2 Crystal parameters

2.1 Unit cell dimensions

Crystallographic unit cell dimensions are given as a right-handed set as conventionally written (see International Tables for Crystallography Volume A, p. 723):

- a (Å) length of the unit cell dimension along the crystallographic x-axis
- b (Å) length of the unit cell dimension along the crystallographic y-axis
- c (Å) length of the unit cell dimension along the crystallographic z-axis
- α (°) the angle between the y- and z- crystallographic axes

- β (°) the angle between the x- and z- crystallographic axes
- γ (°) the angle between the x- and y- crystallographic axes
- $a^\star$ (Å^{-1}) length of the reciprocal space unit cell dimension perpendicular to and forming a right-handed set with b and c.
- $b^\star$ (Å^{-1}) length of the reciprocal space unit cell dimension perpendicular to and forming a right-handed set with a and c.
- $c^\star$ (Å^{-1}) length of the reciprocal space unit cell dimension perpendicular to and forming a right-handed set with a and b.
- $\alpha^\star$ (°) the angle between $b^\star$ and $c^\star$
- $\beta^\star$ (°) the angle between $a^\star$ and $c^\star$
- $\gamma^\star$ (°) the angle between $a^\star$ and $b^\star$

2.1.1 Markup of the Cell

In the Mosflm server interface, the unit cell is decribed with the real-space cell dimensions a, b and c, and the angles α, β and γ as in the following document fragment (using `$a` to refer to the value of a etc.)

```
<cell>
  <a>$a</a>
  <b>$b</b>
  <c>$c</c>
  <alpha>$alpha</alpha>
  <beta>$beta</beta>
  <gamma>$gamma</gamma>
</cell>
```

- The A matrix (dimensionless) is the wavelength dependent combined orientation matrix calculated by UB and is in the Mosflm frame. The elements of the first column give the components of the vector $\lambda a^\star$ axis along the X, Y and Z- axes, respectively where λ is the wavelength of radiation. Similarly, the second and third columns give the components of $\lambda b^\star$ and $\lambda c^\star$ respectively.
- The crystal orientation can also be described as the product of a rotation matrix U and an orthogonalization matrix B. The orhtogonalization matrix defines a set of orthogonal axes X_0 Y_0 Z_0 based on the crystal axes. In Mosflm, this orthogonalization set is defined with X_0 parallel to $a^\star$, Z_0 parallel to c and Y_0 forming a righthanded set. The definition of the B matrix is given below. The U matrix then describes the orientation of these orthogonal crystal axes relative to the laboratory frame, so that columns of U are the direction cosines of X_0 Y_0 and Z_0 relative to the laboratory axes X, Y and Z respectively.

$$B_{11} = a^\star$$
$$B_{12} = b^\star \cos\gamma^\star$$
$$B_{13} = c^\star \cos\beta^\star$$
$$B_{21} = 0$$
$$B_{22} = b^\star \sin\gamma^\star$$
$$B_{23} = -c^\star \sin\beta^\star \cos\alpha$$
$$B_{31} = 0$$
$$B_{32} = 0$$
$$B_{33} = \lambda/c$$

2.1.2 Markup for the Matrices

The A and U matrices, as described above, are described in an element by element fashion, thus:

```
<a_matrix>
  <e11>$e11</e11>
  <e12>$e12</e12>
         .
         .
         .
  <e33>$e33</e33>
</a_matrix>
```

With a similar description for the U matrix, as `u_matrix`.

2.2 Space Group

Space Group information may be given *either* as the space group number as given in International Tables for Crystallography, Vol. A, *or* as the simplified Hermann-Mauguin symbol, *e.g.* space group #19, $P2_12_12_1$ is represented by "P212121"

2.2.1 Markup

The markup for the spacegroup is simple, as there are no internal structure considerations. The `symmetry` forms a part of the indexing solution, thus:

```
<solution>
  <symmetry>p212121</symmetry>  <!-- this is a fragment - there is more
                                     information in the solution element -->
</solution>
```

3 Detector

The default image types for Mosflm are from the Mar image plate scanner and the Crystallographic Binary File. For all other image types, the Mosflm server must be given the information explicitly.

3.1 Type

The detector manufacturer is according to the Mosflm convention, *i.e.*

- MAR - Mar image plate
- MARCCD - Mar CCD
- ADSC - Area Detector Systems Corporation CCD (Quantum 4 or Quantum 210)
- RAXIS - Rigaku R-Axis II
- RAXIS4 *or* RAXISIV - Rigaku R-Axis IV
- RAXIS5 *or* RAXISV - Rigaku R-Axis V
- JUPITER - MSC Jupiter CCD
- DIP2000 - Mac Science DIP 2000
- DIP2030 - Mac Science DIP 2030
- DIP2040 - Mac Science DIP 2040
- LIPS - ESRF Large image plate scanner
- SBC1 - Ed Westbrook 3x3 CCD
- LMB - LMB large Mar
- MD - Molecular Dynamics offline scanner
- BRUKER - Bruker/Siemens CCD detectors (not fully supported)
- CBF - Crystallographic Binary File - preferred format, cross detector, generic

3.1.1 Markup

The detector type and parameters are described in the `detector` element, with `$type` one of the above list, as:

```
<detector>
  <type>$type</type>  <!-- at this stage, this is complete -->
</detector>
```

4 Diffraction image

Many of the details regarding the diffraction images are stored either in the image itself (as header or tailer information) or in an external database. In the former case, it is largely unnecessary for the DNA expert system to supply the information separately (the image header information may be assumed to be correct, at least in the first instance); for the latter case, however, the information must be supplied by the expert system to the Mosflm server.

Conventionally, diffraction images are stored in arrays. The first pixel in the image is defined as that image pixel with the lowest array index. The X co-ordinate is that along the slowly changing direction on the detector with respect to the first pixel in the image, while the Y co-ordinate is along the quickly changing direction.

- The direct beam co-ordinates (mm, mm) are given relative to an origin at the first pixel in the image, with X and Y in the slow and fast changing directions respectively.
- The resolution (Å) of the image is defined as the range of resolutions within which diffraction is to be processed on the image. The high resolution limit has a smaller numeric value, and the low resolution limit has a larger numeric value. The resolution may be isotropic or anisotropic (in which case three values may be provided describing effective resolution along the $a^\star$, $b^\star$ and $c^\star$ respectively).

  Currently, the resolution is *only* used in data integration and post-refinement. It is not used in spot-finding.
- The pixel size (mm) is the linear dimension of the pixel in the slowly changing direction. In the absence of other information, the pixel size in the fast changing direction is assumed to be the same, but where it is not is calculated by (pixel size)/YSCALE (dimensionless).
- Phi(start) (°) is defined as the starting rotation angle about the oscillation axis for the image. Increasing phi is clockwise when viewed from the crystal to the goniometer mount.
- Phi(range) (°), which is always positive, is defined as the difference between the starting rotation angle and finishing rotation angle about the oscillation axis for the image.
- Phi(end) (°) is defined as the finishing rotation angle about the oscillation axis for the image.

Note that only two of Phi(start), Phi(range) and Phi(end) need be given as any two define the third; the Mosflm server uses only the first two of these explicitly.

4.0.2 Markup

- The direct beam position:

```
<beam>
  <x>$x</x>  <!-- these are in mm -->
  <y>$y</y>
</beam>
```

- The resolution, both isotropic and anisotropic:

```
<resolution>
  <lower>
    <a_star>$x lower limit</a_star>  <!-- these are given in terms of the -->
    <b_star>$y lower limit</b_star>  <!-- reciprocal lattice axis -->
    <c_star>$z lower limit</c_star>
  </lower>
  <upper>
    <a_star>$x upper limit</a_star>
    <b_star>$y upper limit</b_star>
    <c_star>$z upper limit</c_star>
  </upper>
</resolution>
```

or...

```
<resolution>
  <lower>
    <isotropic>$lower limit<isotropic>
  </lower>
  <upper>
    <isotropic>$upper limit</isotropic>
  </upper>
</resolution>
```

- The pixel size: (ratio is equivelent to yscale)

```
<pixel>
  <size>$size</size>
  <ratio>$ratio x/y</ratio>
</pixel>
```

- The oscillation ranges, allowing for multiple segments:

```
<oscillation>
  <phi>
    <start>$start of the sequence</start>
    <range>$range of an individual image</range>
    <number>$number of images in the sequence</number>
  </phi>
</oscillation>
```

4.1 Dataset

- Wavelength (Å) of the radiation used in the diffraction experiment.
- Crystal to detector distance (mm) is measured from the centroid of the irradiated volume of the crystal to the detector's X-ray sensitive surface measured along a line normal to the detector.

4.1.1 Markup

The dataset information may be marked up as:

```
<dataset>
  <wavelength>$wavelength</wavelength>
  <distance>$distance</distance>
</dataset>
```

5 Spot finding

The process of spot finding is wholly concerned with locating diffraction maxima from images with the intent of using them in an immediately subsequent autoindexing procedure.

In the normal run of events, none of the DNA clients other than the GUI will perform this step outside of an autoindexing command. Further, under normal circumstances for automated processing, it should be unnecessary for the expert systems to provide any parameters for this process as they are all determined directly from the diffraction images by Mosflm itself.

5.1 Input

The following parameters may be supplied to the program as criteria for finding spots located in the spot finding routine:

- ◇ Minimum intensity (I_{min}) for a pixel to be considered part of a spot, expressed as a multiple (sdfac) of the rms variation (σ_{bg}) in the background I_{bg}, so

$$I_{min} = \text{sdfac} \cdot \sigma_{\text{bg}} + \text{I}_{\text{bg}}$$

  I_{bg} and σ_{bg} are calculated by Mosflm, and sdfac can be supplied by the user.

  (dimensionless) for spot search

- ◇ Minimum/maximum radii (mm) (centred on the direct beam co-ordinates) for the spot search.
- ◇ Offset of the radial background measurement box (mm) in slow and fast directions, respectively. The box extends from the minimum to the maximum search radii, and its major axis is usually orthogonal to the oscillation axis.

The following parameters may be supplied to the program as criteria for rejecting spots located in the spot finding routine:

- ◇ minimum size in X (defined as a multiple of the median spot size)
- ◇ maximum size in X
- ◇ minimum size in Y
- ◇ maximum size in Y
- ◇ minimum pixels (dimensionless)
- ◇ minimum spot separation (mm)

5.1.1 Markup

The spot finding request could be marked up as:

```
<spot_search_request>
  <fileinfo>
    <template>$template</template>
    <directory>$directory</directory>
  </fileinfo>
  <image>$image1</image>
         .
         .
  <image>$imagen</image>
  <background>
    <rmin>$rmin</rmin>
    <rmax>$rmax</rmax>
    <offset>$offset</offset>
    <direction>(parallel|perpendicular) (w.r.t. ‘x’)</direction>
  </background>
  <threshold>$minimum threshold</threshold>
  <spot>
```

```
    <xmin>$xmin</xmin>
    <xmax>$xmax</xmax>
    <ymin>$ymin</ymin>
    <ymax>$ymax</ymax>
    <separation>$separation</separation>
    <pixels>$minimum number of pixels</pixels>
  </spot>
</spot_search_request>
```

The response will be described shortly...

5.2 Output

The output of spot finding includes;

- Total number of spots found (dimensionless)
- Number of spots accepted for indexing (dimensionless)

The following ouputs give the number of spots rejected from the spot found list for various reasons;

- too small in X (dimensionless)
- too small in Y (dimensionless)
- too large in X (dimensionless)
- too large in Y (dimensionless)
- too few pixels (dimensionless)
- too close (dimensionless)

5.2.1 Markup

The response from the `spot_search_request` will be a document of the form:

```
<spot_search_response>
  <found>$number of spots found</found>
  <used>$number of acceptable spots</used>
  <rejected>
    $number of rejected spots
    <xmin>$number too small in x</xmin>
    <xmax>$number too large in x</xmax>
    <ymin>$number too small in y</ymin>
    <ymax>$number too large in y</ymax>
    <separation>$number too close</separation>
    <pixels>$number with too few pixels</pixels>
  </rejected>
</spot_search_response>
```

6 Indexing

In the current context, this will always mean "autoindexing". It is the process of determining the orientation matrix A and from this the unit cell (and its orientation in the coordinate frame) from a list of spot positions found on one or more images.

In the context of Mosflm, autoindexing by any of the clients will be performed by the DPS code.

6.1 Input

- Which images are to be used for indexing.
  - ◇ The maximum cell edge (Å) to be considered for autoindexing.
  - ◇ Minimum $I/\sigma(I)$ (dimensionless) for acceptance of spots for autoindexing.
  - ◇ Target space group.
  - ◇ The name for the orientation matrix file (text)
  - ◇ Positional error cutoff for including reflections in refinement (defined as a mutiple of the rms positional error).
  - ◇ Whether or not each solution should be refined (*i.e.* not just the final solution)

6.1.1 Markup

```
<index_request>
  <target>
    <max_cell>$maximum cell edge</max_cell>
    <symmetry>$target symmetry</symmetry>
    <cell/>
  </target>
  <fileinfo>
    <template>$template</template>
    <directory>$directory</directory>
  </fileinfo>
  <image>$image1</image>
            .
            .
            .
  <image>$imagen</image>
  <detector/>
  <beam/>
  <matrix_file>$matrix file name</matrix_file>
</index_request>
```

If `target symmetry` is `all`, refine all solutions, rather than just the optimum one.

6.2 Output

- The number of reflections used from the reflection list for indexing (dimensionless).
- The number of reflections rejected from the reflection list for indexing (dimensionless).

  All the following are concerned only with the chosen solution;

- The rank order (dimensionless).
- The penalty, *i.e.* the goodness of fit (normalized to a maximum value of 999) of a given orientation matrix to a lattice type, according to the formulae in International Tables A, 9.3, pp. 741 for the 44 characteristic lattices
- The characteristic lattice or space group.

- The unit cell dimensions before refinement.
- The unit cell dimensions after refinement.
- The number of reflections used from the reflection list for refining the cell (dimensionless).
- The RMS deviation of predicted to observed spot position (mm).
- The RMS deviation of predicted to observed spot position (°).
- Shift in direct beam co-ordinates from starting values after refinement (mm) and expressed as a decimal fraction (dimensionless) compared to the minimum spot separation within lunes (mm).

6.2.1 Markup

This is the old XML...

```
<index_response>
  <status/>
  <solution>
    <symmetry>$symmetry</symmetry>
    <number>$number of the solution</number>
    <penalty>$penalty</penalty>
    <initial>
      <cell/>
    <initial>
    <orientation>
      <a_matrix/>
      <u_matrix/>
      <cell/>
    </orientation>
    <reflections>
      <used>$number of acceptable spots</used>
      <rejected>
      $number of rejected spots
        <xmin>$number too small in x</xmin>
        <xmax>$number too large in x</xmax>
        <ymin>$number too small in y</ymin>
        <ymax>$number too large in y</ymax>
        <separation>$number too close</separation>
        <pixel>$number with too few pixels</pixel>
      </rejected>
    </reflections>
    <spot_deviation>$spot_deviation</spot_deviation>
    <beam/>
    <refinement>
      <beam_shift>
        <beam>
          <x>$x</x>
          <y>$y</y>
        <beam>
        <relative>
```

```
          <x>$x</x>
          <y>$y</y>
        </relative>
      </beam_shift>
      <reflections>
        <used>$number of reflections used in refinement</used>
      </reflections>
    <refinement>
  </solution>
</index_response>
```

And this is the new and shiny XML for the first half of the `index_request` - obtaining the list of solutions. This may be bypassed if either the desired solution has been specified, or it was left to Mosflm to determine the optimum solution.

```
<index_response>
  <status/>
  <spot_search_response/>  <!-- optional - if a spot search was
                               performed as a part of the autoindexing -->
  <solution/>  <!-- number 1 -->
  <solution/>  <!-- number 2 -->
      .
      .
      .
  <solution/>  <!-- number 44 -->
</index_response>
```

where the `solution` is defined, in this context, as

```
<solution>
  <lattice>$lattice</lattice>
  <number>$number</number>
  <penalty>$penalty</penalty>
  <initial>  <!-- the unrefined cell parameters -->
    <cell/>
  </initial>
  <orientation>  <!-- the refined cell parameters - optional! -->
    <cell/>
  </orientation>
</solution>
```

This will include more information in the next context.

7 Choosing and refining a solution from autoindexing

The autoindexing step produces a set of 44 solutions corresponding to each of the characteristic lattices. Mosflm has the ability to choose one of these solutions based on analysis of the penalty distribution. The client has the choice of either allowing Mosflm to make this choice, or to perform some local analysis and choose a solution of its own.

Several of the items included here are identical to those in the preceding section for the case when solutions are pre-refined prior to choosing.

7.1 Input

- ▷ number of the solution (i.e. rank order)
- ▷ space group
- ◇ whether to refine the cell
- ◇ whether to accept the refined beam position
- ◇ what σ cutoff to use for these refinements
- ◇ The number of reflections used from the reflection list for refining the cell (dimensionless).

7.1.1 Markup

The markup for this request is based very closely on the basic indexing request, and is designed to be compatible to a large degree. For this specification, it is necessary to have already received the previous document, so that it is possible to determine the rank of the desired solution.

```
<index_request>
  <target>
    <symmetry>$summetry</symmetry>
    <number>$number</number>  <!-- the rank of the solution -->
  </target>
  <refine>beam</refine>
  <refine>cell</refine>
  <signal>$signal</signal>  <!-- the sigma cutoff to use -->
</index_request>
```

If an automatically determined solution is desired, some half way house between these `index_request` documents is necessary - illustrate this.

7.2 output

- The unit cell dimensions after refinement (Å, °).
- The number of reflections used from the reflection list for refining the cell (dimensionless).
- The RMS deviation of predicted to observed spot position (mm).
- The RMS deviation of predicted to observed spot position (°).
- Shift in direct beam co-ordinates from starting values after refinement (mm) and expressed as a decimal fraction (dimensionless) compared to the minimum spot separation within lunes (mm).

7.2.1 Markup

```
<index_response>
  <status/>
  <spot_search_response/>  <!-- this is optional -->
  <solution/>
</index_response>
```

where the `solution` element is now of the form

```
<solution>
  <symmetry>$symmetry</symmetry>
  <number>$number</number>
  <initial>
    <cell/>
  </initial>
  <orientation>
    <a_matrix/>
    <u_matrix/>
    <cell/>  <!-- the refined cell - if this is to be refined -->
  </orientation>
  <spot_deviation>$the rms deviation</spot_deviation>
  <beam/>  <!-- the refined beam position - if this is to be refined -->
  <refinement>
    <beam_shift>
      <beam/>  <!-- the shift in the beam in mm -
                    shares the beam structure -->
      <relative/>  <!-- the relative shift in the beam -
                         this shares the beam structure -->
    </beam_shift>
    <reflections>
      <used>$decimal fraction of the number of reflections used</used>
    </reflections>
  </refinement>
</solution>
```

Clearly, this is a much more verbose document. Further, if this is in response to a Mosflm-choose-your-own solution, there could also be a `spot_search_response` document fragment too.

8 Mosaicity and beam divergence

The effective mosaic spread (°) is caused by a combination of the misalignment of domains (mosaicity) within the crystal and the divergence of the incident beam; effectively, the size of each reciprocal lattice point is increased to a finite volume so that it cuts the Ewald sphere under a wider range of conditions.

The mosaicity is a property of the crystal and may be anisotropic. The divergence of the beam is usually only considered in the case of synchrotron light sources, and is usually given two values, corresponding to the divergence in the plane of the storage ring and perpendicular to this plane. While the divergence is not, strictly speaking, a crystal property, it is convenient to treat it at the same time because it affects the diffraction from the crystal in a mathematically similar way to the mosaicity.

In Mosflm, an initial estimate of the mosaicity may be obtained without optical examination of the image.

8.1 Input

Either

- ⋆ a request for an estimate of mosaicity

or

- ◇ a known value of the mosaicity is supplied

8.1.1 Markup

This is a fairly minimal document, which will either set the value for the mosaicity or request that this value be determined.

```
<mosaicity_request>
  <value>0.44</value>  <!-- this is to set the value -->
</mosaicity_request>
```

If the value of the tag is `estimate`, then the program should estimate the mosaicity.

8.2 Output

- If a request for mosaicity estimation has been made, then the estimate is returned.
- If a known value of the mosaicity is supplied, this value is returned.

8.2.1 Markup

```
<mosaicity_response>
  <status/>
  <value>$value</value>
</mosaicity_response>
```

9 Integrate one image

Having obtained an A matrix and an estimate for the mosaicity, it is possible to integrate a single image and provide an estimate of $< I/\sigma(I) >$ as a function of resolution to the Mosflm server which can be passed on to the expert system to determine the effective resolution.

9.1 Input

The image to be processed must be specified to the Mosflm server. In this case, refinement directives will not be supplied, as the fit of the measurement boxes to the observed spots following refinement in the autoindexing step should be adequate.

- Integrate image n
  - Integrate image n, starting angle (°), oscillation range (°)

9.1.1 Markup

From a communications perspective, it is assumed that this will follow an indexing request, which will have included all of the fileinfo information. If this is not the case, allowance will have to be made for inclusion of a `fileinfo` element in this document. The `phi` element is included to allow for the fact that the oscillation information may not be present in the image header.

```
<single_integrate_request>
  <image>$index of the image</image>
  <phi/>  <!-- this is the phi element - explained elsewhere -->
</single_integrate_request>
```

9.2 Output

A summary of the integration as a function of resolution is returned. A list of the following nine items is written for each of the n resolution bins (for Mosflm, default $n = 8$, and also for the image as a whole.

- High resolution limit of the i-th bin (Å)
- For profile fitted reflections:
  - Number of fully recorded reflections
  - Mean $I/\sigma(I)$ of fully recorded reflections
  - Number of partially recorded reflections
  - Mean $I/\sigma(I)$ of partially recorded reflections
- For reflections measured by summation integration:
  - Number of fully recorded reflections
  - Mean $I/\sigma(I)$ of fully recorded reflections
  - Number of partially recorded reflections
  - Mean $I/\sigma(I)$ of partially recorded reflections

9.2.1 Markup

```
<single_integrate_response>
  <status/>
  <bin>  <!-- you have eight of these -->
    <number/>
    <resolution/>  <!-- this defines the upper and lower limits,
                        isotropic or anisotropic -->
  </bin>
  <records>  <!-- the recorded reflection information -->
    <method>profile</method>  <!-- this could be summation -->
    <full>
      <number/>
      <signal/>
    </full>
    <partial>
      <number/>
      <signal/>
    </partial>
  </records>
</single_integrate_response>
```

This is a first draft, and will quite likely change in short order.

10 Strategy

Once an A matrix, crystal system, mosaicity and effective resolution limit have been determined, a strategy for the data collection may be performed by Mosflm. The default option for this is to calculate the maximum completeness available given the current A matrix, space group, mosaicity and effective resolution. The starting and finishing values for phi and the phi ranges for this segment are

returned, as well as the available completeness for both unique and anomalous datasets. Under normal circumstances, different rotation ranges for images will be returned depending on the individual images.

The options exist to calculate the optimum total oscillation range for collecting a complete anomalous dataset, or for the data to be collected in more than one segment.

This and the preceding step (integrate one image) may be superceded by Sasha Popov's routine at a later date.

10.1 Input

- run strategy
  - ◇ calculate the appropriate oscillation ranges for maximum anomalous completeness
  - ◇ number of segments in data collection (default 1)
  - ◇ size of segments (°)
  - ◇ maximum percentage of overlaps allowed
  - ◇ minimum oscillation angle (°)
  - ◇ maximum oscillation angle (°)
  - ◇ maximum rotation range (°)
  - ◇ step size (for calculation purposes) (°)
  - ◇ start rotation angle (°), end rotation angle (°)
  - ◇ speedup factor (can be used to speed up the calculations, but may result in loss of accuracy).
  - ◇ alternate mode (occasionally gives a better starting phi than the default setting, *only* for use with orthorhombic space groups)

10.1.1 Markup

This is rather a nasty structure, as there are a lot of rather unrelated and method orientated parameters in here which can't go anywhere else and have to be included.

```
<strategy_request>
  <segments>
    <number/>
    <range>$size of segment 1</range>
    <range>$size of segment 2</range>
  </segments>
  <strategy_method>
    <step_size>$step_size</step_size>
    <mode>(alternate|standard)</mode>
    <speedup>$speedup</speedup>
    <optimise>(standard|anomalous)</optimise>  <!-- i don't like this -->
  </strategy_method>
  <rotation>
    <start>$start</start>
    <end>$end</end>
    <range>$range</range>  <!-- the maximum mange -->
```

```
    <step>
      <minimum>0.1</minimum>
      <maximum>4</maximum>
    </step>
  </rotation>
  <overlap_limit>0</overlap_limit>  <!-- sore thumb? -->
</strategy_request>
```

10.2 Output

- optimum rotation range for completeness (°) of unique data
- percentage completeness (for unique data) attainable for this range (ignoring overlaps)
- percentage completeness (for anomalous data) attainable for this range (ignoring overlaps)
- suggested detailed strategy for n subsegments including;
  - phi start (°)
  - phi end (°)
  - number of images
  - oscillation angle for each image (°)
  - percentage overlaps
  - percentage of reflections collected as fully recorded

10.2.1 Markup

```
<strategy_response>
  <status/>
  <completeness>
    <standard>$value</standard>
    <anomolous>$value</anomolous>
  </completeness>
  <segment>  <!-- you can have a number of these -->
    <phi/>
    <records>
      <full>
        <number>$percentage fully recorded spots</number>
      </full>
      <overlap>
        <number>$percentage overlapped spots</number>
      </overlap>
    </records>
  </segment>
</strategy_response>
```

11 Refine

text & XML to be added

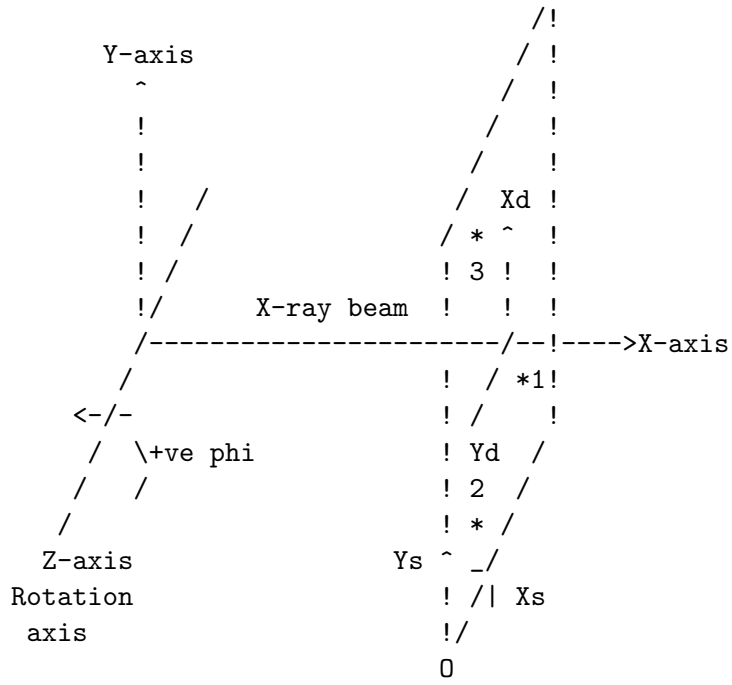

Figure 1
The coordinate systems used by Mosflm

12 Integrate dataset

text & XML to be added

Appendix I: Co-ordinate systems used by MOSFLM

The following is extracted from the *Mosflm User Guide*.

The coordinates systems used by Mosflm are illustrated in Fig. 1. The crystal orientation is defined relative to the laboratory coordinate frame (X,Y,Z), with X along the X-ray beam (away from the source), Z along the rotation axis, and Y forming a right-handed set. A positive phi rotation is anticlockwise viewed from +Z looking towards the origin. The orientation matrices (A and U) are defined relative to this axial system.

Appendix II - Connecting to and Disconnecting from the Server

12.1 Introduction

The connection to the Mosflm server is a two stage process. The initial connection consists of authentication and creation of a dedicated serving port, as each client will have a port of it's own. The identity of the newly created port is then returned to the client, after which the initial connection is broken. The client should then reconnect to the newly created port to process data.

When all of the data processing is complere, the client should issue a dicsonnection request, which will cause the serving thread to exit any child processes and rejoin itself to the main thread. This will also happen automatocally in the event of the channel being broken.

12.2 Connection

In order that the Mosflm processes are run with the appropriate PID, the server must obtain information about the user. This requires the username (minimally) and possibly an authentication method and 'token'. The authentication methods have so far not been considered in much detail. The authentication method and token are necessary for the processing, so in the event of a site having *no authentication*, 'default' and 'none' should be passed respectively.

In keeping with the other request/response pairs, the response will include status information, a code and a message. In the event that the authentication is accepted, the code will be 'ok'. The response document in this instance will also include a new port number for the client to connect to.

12.2.1 Markup

```
<connection_request>
  <username>my username</username>
  <method>default</method>
  <token>none</token>
</connection_request>

<connection_response>
  <status>
    <code>ok</code>
  </status>
  <port>10241</port>
</connection_response>
```

12.3 Disconnection

After the client has connected through the new port, it will have exclusive access to that port, so the need for authentication of individual requests is deminished. The request for disconnection will therefore be a trivial document of the form `<disconnection_request/>`, which will require an equally trivial response, `<disconnection_response/>`. This response is only necessary for the HTTP protocol, which insists that each request is responded to.

Appendix III - The DTD for the XML

This defines the structure of the XML documents described previously.

```
<!-- DTD for the individual elements
     Maintained by: G.Winter
     Started 6th February 2002

     $Id$

     Notes:
     This represents exactly half of the information necessary for
     interfacing with the Mosflm server. There is an associated document
     in which the full definitions of all of these elements are considered.
     It is important to note that these elements define the document
     structure rather than the meaning, which can vary.

     General principle:
```

```
     Reuse - all of the documents share the same elements wherever
     possible, allowing for a rather short DTD, although there are a lot
     of ?marks in it, as there are instances when one document serves
     two purposes (ie index_request/response). This isn't a problem however,
     as there is enough common information to make it useful.

     This document defines only the data structures, not the meaning of
     those structures. In particular, although the structures will generally
     correspond to individual pieces of information, this will not be a
     one-to-one relationship. For instance, the 'phi' tag will include
     elements which can be used to describe either a single image or a
     sequence of images which share the same oscillation angle.

     In terms of the chosen vocabulary, there have been instances where
     a single word shares two crystallographic meanings. In these instances
     a second term has been used to mean the second (on a first come, first
     served basis) instance, for instance the 'orientation' matrices and the
     'orientation' of the background strip.

     There is no attempt at data typing. In some instances 'number' may
     refer to a decimal expansion, in others an integer. The meanings of this
     information are determined from the structure of the document and the
     data description which should accompany this DTD.

-->

<!-- the cell parameter 'structure' -->

<!ELEMENT cell (a, b, c, alpha, beta, gamma)>
  <!ELEMENT a (#PCDATA)>
  <!ELEMENT b (#PCDATA)>
  <!ELEMENT c (#PCDATA)>
  <!ELEMENT alpha (#PCDATA)>
  <!ELEMENT beta (#PCDATA)>
  <!ELEMENT gamma (#PCDATA)>

<!-- the a_matrix (and u_matrix) structures -->

<!ELEMENT a_matrix (e11, e12, e13, e21, e22, e23, e31, e32, e33)>
<!ELEMENT u_matrix (e11, e12, e13, e21, e22, e23, e31, e32, e33)>
  <!ELEMENT e11 (#PCDATA)>
  <!ELEMENT e12 (#PCDATA)>
  <!ELEMENT e13 (#PCDATA)>
  <!ELEMENT e21 (#PCDATA)>
  <!ELEMENT e22 (#PCDATA)>
  <!ELEMENT e23 (#PCDATA)>
  <!ELEMENT e31 (#PCDATA)>
  <!ELEMENT e32 (#PCDATA)>
  <!ELEMENT e33 (#PCDATA)>

<!-- the direct beam position - z is included as the z is used to
```

```
     complete the x, y, z triple for use in other elements -->

<!ELEMENT beam (x, y)>
  <!ELEMENT x (#PCDATA)>
  <!ELEMENT y (#PCDATA)>
  <!ELEMENT z (#PCDATA)> <!-- included here for completeness -->

<!-- the resolution structure - defining the limits on the structure in
     both isotropic and anisotropic situations - hence 'z' above -->

<!-- this is used in two instances, first to define the limits on resolution
     on an entire image, and secondly to define the limits within a single
     resolution bin. in the latter instance the lower limits will most
     likely be irrelevant and so may be omitted -->

<!ELEMENT resolution (lower?, upper)>
  <!ELEMENT lower (isotropic?, a_star?, b_star?, c_star?)>
    <!ELEMENT a_star (#PCDATA)>  <!-- distances in terms of the reciprocal -->
    <!ELEMENT b_star (#PCDATA)>  <!-- lattice -->
    <!ELEMENT c_star (#PCDATA)>
  <!ELEMENT upper (isotropic?, a_star?, b_star?, c_star?)>
    <!ELEMENT isotropic (#PCDATA)>

<!-- the pixel description -->

<!ELEMENT pixel (size, ratio)>
  <!ELEMENT size (#PCDATA)>
  <!ELEMENT ratio (#PCDATA)>

<!-- oscillation ranges, allowing for the possibility of having several
     oscillation angles in a dataset -->

<!-- the 'phi' element is used in a number of situations and will take related
     but differing meanings in each instance, so watch for this. the actual
     definition in each instance should be well defined in the accompanying
     documentation -->

<!ELEMENT oscillation (phi+)>
  <!ELEMENT phi (start, range, number)>
    <!ELEMENT start (#PCDATA)>
    <!ELEMENT range (#PCDATA)>
    <!ELEMENT number (#PCDATA)>

<!-- parameters which define a data set -->

<!ELEMENT dataset (wavelength, distance)>
  <!ELEMENT wavelength (#PCDATA)>
  <!ELEMENT distance (#PCDATA)>

<!-- the fileinfo structure - this will be used in a number of places
     and may include up to 10 (currently) directories in which the images
```

```
     may be found -->

<!ELEMENT fileinfo (template, directory+)>
  <!ELEMENT template (#PCDATA)>
  <!ELEMENT directory (#PCDATA)>

<!-- the 'image' tag - this just gives the index of an image to be processed
     there will often be lists of these -->

<!ELEMENT image (#PCDATA)>

<!-- minima and maxima of things - spatial -->

<!ELEMENT rmin (#PCDATA)>
<!ELEMENT rmax (#PCDATA)>
<!ELEMENT xmin (#PCDATA)>
<!ELEMENT xmax (#PCDATA)>
<!ELEMENT ymin (#PCDATA)>
<!ELEMENT ymax (#PCDATA)>

<!-- status information
     code should be (ok|warning|error)
     message is a human readable message to explain what went wrong -->

<!ELEMENT status (code, message?)>
  <!ELEMENT code (#PCDATA)>
  <!ELEMENT message (#PCDATA)>

<!-- detector information
     possible values for type are given in the other documentation -->

<!ELEMENT detector (type)>
  <!ELEMENT type (#PCDATA)>

<!ELEMENT solution (symmetry?, lattice?, number?, penalty?, initial?,
                    orientation?, spot_deviation?, beam?, refinement?)>
  <!ELEMENT symmetry (#PCDATA)>
  <!ELEMENT lattice (#PCDATA)>
  <!ELEMENT penalty (#PCDATA)>
  <!ELEMENT initial (cell)>
  <!ELEMENT orientation (a_matrix?, u_matrix?, cell?)>
  <!ELEMENT spot_deviation (#PCDATA)>
  <!ELEMENT refinement (beam_shift, reflections?)>
    <!ELEMENT beam_shift (beam, relative)>
      <!ELEMENT relative (x, y)>

<!-- document definitions - for input (request) and output (response) -->

<!-- general considerations
     all input requests have a name which ends in 'request', and do not
     include a 'status' element. output documents all end with 'response'
```

```
     and share the same prefix as the request they are responding to.
     further, all responses contain at the very least a 'status' element -
     this tells the client whether the requested task was successful, or
     whether it failed. in the latter instance this should explain why
     in message -->

<!-- spot_search_request
     request that a spot search is performed, passing in all of the
     necessary information. the 'spot_search_response' document should be
     returned -->

<!ELEMENT spot_search_request (fileinfo, image+, background, threshold, spot)>
  <!ELEMENT background (rmin, rmax, offset, direction)>
    <!ELEMENT offset (#PCDATA)>
    <!ELEMENT direction (#PCDATA)>
  <!ELEMENT threshold (#PCDATA)>
  <!ELEMENT spot (xmin, xmax, ymin, ymax, separation, pixels)>
    <!ELEMENT separation (#PCDATA)>
    <!ELEMENT pixels (#PCDATA)>

<!ELEMENT refine (#PCDATA)>  <!-- what to refine in the event of
                                  an index_request -->

<!-- this illustrates nicely the different uses of for instance the
     xmin element -->

<!-- spot_search_response
     this describes the numbers of spots found, and those which
     were deemed useful and those which were rejected.
     the found, used and rejected may not be returned in the
     event of an error. -->

<!ELEMENT spot_search_response (status?, found?, used?, rejected?)>
  <!ELEMENT found (#PCDATA)>
  <!ELEMENT used (#PCDATA)>
  <!ELEMENT rejected (#PCDATA | xmin | xmax | ymin | ymax |
                      separation | pixels)*>

<!-- index_request
     request that a number of images, specified by fileinfo and image
     elements, are indexed and the resulting possible solutions returned,
     with the refined cell for the optimum or selected solution. -->

<!ELEMENT index_request (target?, fileinfo, image+, detector,
                         beam, matrix_file, refine*, threshold?)>
  <!ELEMENT target (max_cell, number, symmetry, cell)>
    <!ELEMENT max_cell (#PCDATA)>
  <!ELEMENT matrix_file (#PCDATA)>

<!-- index_response
     this is the response to the index_request, and should include 44
```

```
     solutions, with one of them refined and the rest not, hence lots
     of ?marks below. for the optimum solution the refined parameters
     will be included.

     if another solution is desired, then the same initial beam position
     should be used as was used for the initial index_request, so that the
     list of solutions is the same, and the solution specified by the
     line number and spacegroup is the correct one - this may cause
     problems -->

<!-- this is worth thinking about, as it will mean rather large documents
     being passed back and forth - is it really necessary to include all
     of the possible solutions when describing the one that was actually
     wanted? no - so there has been a change made to this, and there are
     two possible 'versions' of the index_response, and two versions
     of the request to accomodate this.

     will have a lot of things like:

     <solution>
       <number>1</number>
       <lattice>iC</lattice>
       <penalty>121</penalty>
       <initial>
         <cell>
           <a>120</a>
           <b>90</b>
           <c>90</c>
           <alpha>90</alpha>
           <beta>90</beta>
           <gamma>90</gamma>
         </cell>
       </initial>
     </solution>

     which will define possible solutions...

     -->

<!ELEMENT index_response (status, spot_search_response?, solution*)>

<!-- mosaicity things - these are simple -->

<!ELEMENT mosaicity_request (value)>
  <!ELEMENT value (#PCDATA)>  <!-- make value estimate to perform the
                                   calculation -->

<!ELEMENT mosaicity_response (status, value)>

<!-- single_integration requests etc -->
```

```
<!-- this is used to determine properties about the image, for example the
     resolution as a function of radius etc -->

<!ELEMENT single_integration_request (image, phi?)>

<!-- a bin is a resolution bin - Mosflm has eight by default -->

<!ELEMENT single_integration_response (status, bin+, records)>
  <!ELEMENT bin (number, resolution)>  <!-- this will be a subset of the
                                            resolution info -->
  <!ELEMENT records (method?, full?, partial?, overlap?)>
    <!ELEMENT method (#PCDATA)>
    <!ELEMENT full (number?, signal?)>
      <!ELEMENT signal (#PCDATA)>
    <!ELEMENT partial (number?, signal?)>
    <!ELEMENT overlap (number?, signal?)>

<!-- strategy things -->

<!-- almost all of this is truly optional - just need to say whether to
     optimise the strategy for the standard or amorphous scattering -->

<!ELEMENT strategy_request (segments?, strategy_method, rotation?,
                            overlap_limit?)>
  <!ELEMENT segments (number, range+)>
  <!ELEMENT strategy_method (step_size?, mode?, speedup?, optimise?)>
    <!ELEMENT step_size (#PCDATA)>
    <!ELEMENT mode (#PCDATA)>
    <!ELEMENT speedup (#PCDATA)>
    <!ELEMENT optimise (#PCDATA)>  <!-- this is used to say what we're
                                       getting the strategy for - required -->
  <!ELEMENT rotation (start, end, range, step
    <!ELEMENT step (maximum, minimum)>
      <!ELEMENT maximum (#PCDATA)>
      <!ELEMENT minimum (#PCDATA)>
  <!ELEMENT overlap_limit (#PCDATA)>

<!ELEMENT strategy_response (status, completeness, segment+)>
  <!ELEMENT completeness (standard, amorphous)>
    <!ELEMENT standard (#PCDATA)>
    <!ELEMENT amorphous (#PCDATA)>
  <!ELEMENT segment (phi, records)>

<!-- connection and disconnection -->

<!ELEMENT connection_request (username, method, token)>
  <!ELEMENT username (#PCDATA)>
  <!ELEMENT method (#PCDATA)>
  <!ELEMENT token (#PCDATA)>

<!ELEMENT connection_response (code, port?)>
```

```
  <!ELEMENT port (#PCDATA)>

<!ELEMENT disconnection_request EMPTY>
<!ELEMENT disconnection_response EMPTY>
```