

3DXRD and TotalCryst Geometry

Version 1.0.8

H.F. Poulsen, S. Schmidt, J. Wright, H.O. Sørensen, J. Oddershede, A. Alpers

This note defines the geometry related to the 3DXRD methodology as implemented at ID11 at ESRF. It is mainly an extension of the work presented in the book by Poulsen [1] and the more recent summary in the book by Banhardt [2]. The reader is expected to be familiar with single crystal diffraction.

The note serve as a standard for the geometry in a suite of programs available under the TotalCryst umbrella¹ such as FABLE, near- and farfield simulators (qnfs and PolyXSim, respectively), Fabian, ImageD11, GrainSpotter, Grainsweeper, and the Monte Carlo routines for grain mapping. For historic reasons, the representation within the program may not always follow the standard, but should apply to the user interfaces.

A few comments on typography: vectors are written with underline, matrices and quaternions in bold² and tensors with no special fonts (but distinguishable by their subscripts).

1 The basic 3DXRD set-up

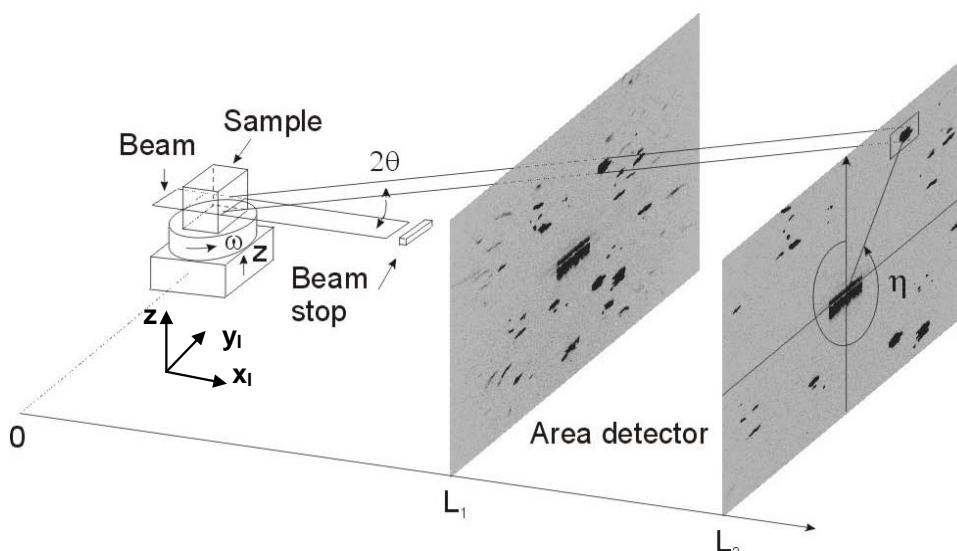


Figure 1.1

Sketch of the 3DXRD principle for the case of the incoming monochromatic beam being focused in one dimension. The Bragg angle 2θ , the rotation angle ω and the azimuthal angle η are indicated for the diffracted beam arising from one grain of a coarse-grained specimen, and for two settings of the near field area detector. In addition a farfield detector may be present. The axes for the laboratory co-ordinate system are also shown.

¹ See www.totalcryst.dk and <http://fable.wiki.sourceforge.net>.

² Unfortunately WORD doesn't allow for bold face in the equations.

The 3DXRD method is an extension of the “rotation method” known from single crystal crystallography. The basic set-up is sketched in Fig 1.1. A monochromatic x-ray beam is constrained to a suitable cross-section by means of focusing and/or the use of absorbing slits. This beam may illuminate the whole sample, or only parts of it.

The sample is mounted on a goniometer. The prototypical set-up considered in this version of the document comprises only one rotation stage, an ω -stage rotating around an axis perpendicular to the incoming beam, see Fig 1.1. Furthermore, for convenience, we shall in the text assume the ω -axis is vertical, although this restriction is not required by the algebra. However, we will provide equations for more general rotations for use e.g. in connection with a four circle or a double tilt below the ω -stage. There are no translations.

Any part of the illuminated structure, which fulfils the Bragg condition, will generate a diffracted beam. This beam is transmitted through the sample and probed by one or more 2D detectors. Essential to 3DXRD is the idea to use such detectors to mimic a 3D detector, similar to the ones used in particle physics. This can be done in two ways. Firstly, by positioning several 2D detectors at different distances to the centre-of-rotation, L , and exposing these either simultaneously (many detectors are semi-transparent to hard x-rays) or subsequently. Secondly, by acquiring images with one 2D detector positioned at several distances to the rotation axis, as illustrated in Fig 1.1.

To probe the complete structure, and not just the part that happens to fulfil the Bragg condition, the sample is rotated around one axis – the ω axis. Hence, exposures are made for equi-angular settings of ω with a step of $\Delta\omega$. To provide a uniform sampling the sample is rotated by $\Delta\omega$ during each exposure. To avoid confusion of terms in the following $\Delta\omega$ is termed the “oscillation range”. We shall adapt the convention that the omega range defined is the total range: e.g. a range of $[0^\circ 10^\circ]$ in 10 steps imply that acquisitions are made around $0.5^\circ, 1.5^\circ, \dots, 9.5^\circ$ – in each case while rotating by ± 0.5 degree around the nominal value. The ω range is defined as a subinterval of $[-180^\circ, 180^\circ]$ (and not $[0^\circ, 360^\circ]$ - which makes a difference in forward projection routines). *This should be enforced in the specs macro: sweepscan etc.*

The azimuthal angle η (see Fig 1.1) is defined to be positive in the cw direction when looking downstream – from sample towards detector – and to be 0 pointing upwards.

With the detectors available, experience has proven two complementary detector configurations to be of particular use. Depending on the issue at hand, they may be used on a stand-alone basis or they can be combined.

Near-field configuration: A detector with a high spatial resolution positioned close to the specimen. Data acquisition may or may not be repeated at several distances (say 3 settings of L in the range of 1-10 mm). The angular resolution is relatively low, implying that the diffraction patterns are not influenced by any elastic strain, as the associated angular perturbations are too small to be observed. Hence, only spatial and orientation degrees of freedom are probed.

Far-field configuration: A detector with a low spatial resolution positioned at a fixed distance to the specimen (not shown in Fig 1.1). The distance is optimised such that the full diffraction pattern appears in the images (say $L = 400$ mm at 50 keV). The diffraction spots now appear on a set of rings – the Debye-Scherrer rings well known from powder diffraction. In this case

the spatial degrees of freedom are to a large extent integrated out, whilst the angular resolution is medium.

The formalism developed below works for all distances and allows the plane of the detectors to be tilted with respect to the incoming beam.

The formalism developed below is subject to the following basic assumptions:

- The beam divergence and the energy band width is neglected
- Kinematical scattering

For more information, see Refs 1 and 2.

2 Diffraction geometry

The algebra for associating diffraction observations with reciprocal space is well described for single crystals. The polycrystal case differs by the need for one extra coordinate system since the sample and the grains are separate objects. In the following equations describing a single scattering event are derived. Mainly the single crystal formalism of Busing and Levy [3] is followed as close as possible³, but for a number of equations alternative – but equivalent - expressions are given, as used by the various programs in the full package. For reasons of simplicity, the relevant part of the sample is assumed fully illuminated at all ω -settings.

The laboratory and rotated coordinate systems

We first consider **the basic set-up** shown in Fig 1 with only one rotation axis, which is directed perpendicular to the incoming beam. In this case we define the laboratory system $(\hat{x}_l, \hat{y}_l, \hat{z}_l)$ as having \hat{x}_l pointing along the incoming beam, \hat{y}_l transverse to it in the horizontal plane and \hat{z}_l positive upwards, parallel to the ω rotation axis. Furthermore, $(x_l, y_l) = (0, 0)$ along the ω -rotation axis. The definition of $z_l = 0$ is made by means of a reference beam of infinitesimal size.

In this system the direction of the diffracted ray can be parameterised by the Bragg angle θ and the azimuthal angle η , both defined in Fig 1.1. Notably η is defined positively in the cw direction, when viewed along the beam direction from the sample towards the detector and $\eta=0$ when pointing upwards (in the positive direction of \hat{z}_l).

Next, we define ω positive ccw when viewed along the z-axis from the top of the instrument. Then

$$\begin{pmatrix} x_l \\ y_l \\ z_l \end{pmatrix} = \Omega \begin{pmatrix} x_\omega \\ y_\omega \\ z_l \end{pmatrix} = \begin{pmatrix} \cos(\omega) & -\sin(\omega) & 0 \\ \sin(\omega) & \cos(\omega) & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x_\omega \\ y_\omega \\ z_l \end{pmatrix}. \quad (2.1)$$

³ However, the sign convention for ω used here is opposite to the one used in [3].

The coordinates $(x_\omega, y_\omega, z_l)$ refer to the rotated system, which is rigidly attached to the ω turntable. This coordinate system is identical to the sample system, except if the user wants to redefine the latter using a matrix \mathbf{S} , see below.

We next consider a **more generalised goniometer**. Following Busing-Levy we shall operate with 3 rotations, namely an outer ω -rotation stage (rotating around the z -axis), a χ -rotation, and an innermost ϕ -stage. The combined rotation is given by

$$\begin{pmatrix} x_l \\ y_l \\ z_l \end{pmatrix} = \Gamma \begin{pmatrix} x_\omega \\ y_\omega \\ z_l \end{pmatrix} = \Omega X \Theta \begin{pmatrix} x_\omega \\ y_\omega \\ z_l \end{pmatrix}. \quad (2.2)$$

This definition can be used to handle e.g. a Eulerian cradle, a Kappa-goniometer or a double-tilt above the ω -axis. We shall leave the exact definition of \mathbf{X} and Θ to the user. We will assume that the ω rotation is used for scanning/oscillating during actual data acquisition, and this axis is perpendicular to the incident beam.

Detector coordinate systems

The detector read-out is in a pixel coordinate system. Different detector systems use different conventions for how to flip the image and where to put the origin. We here define our own standard convention $(y_{\text{raw}}, z_{\text{raw}})$, which is used in equations below. This is a regular grid defined in pixels, as shown in Figure 2.1

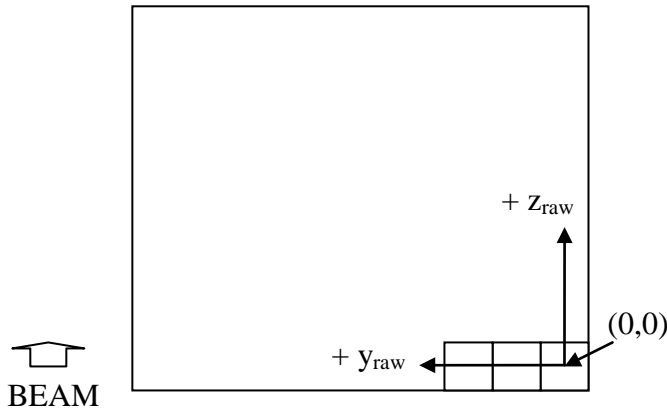


Figure 2.1. Definition of detector pixel coordinate system, see text.

Hence, for the detector plane normal parallel to the beam, y_{raw} is parallel to \hat{y}_l and z_{raw} is parallel to \hat{z}_l . We define $(0,0)$ as corresponding to the centre of the pixel in the lower right corner of the image. This implies that the border of the detection area will have half-integer values.

The detector may be associated with spatial distortion, as given by the operator SC . We define $(y_{\text{det}}, z_{\text{det}})$ to represent the corrected system:

$$(y_{\text{det}}, z_{\text{det}}) = SC((y_{\text{raw}}, z_{\text{raw}})) \quad (2.3)$$

The detector plane normal may be tilted with respect to the incoming beam. We define

ϕ_x : Tilt of detector around x (pos ccw around x: right-hand system)

ϕ_y : Tilt of detector around y (pos ccw around y: right-hand system)

ϕ_z : Tilt of detector around z (pos ccw around z: right-hand system)

Correspondingly we have rotation matrices \mathbf{R}^X , \mathbf{R}^Y and \mathbf{R}^Z :

$$\mathbf{R}^X = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos(\phi_x) & -\sin(\phi_x) \\ 0 & \sin(\phi_x) & \cos(\phi_x) \end{pmatrix}; \mathbf{R}^Y = \begin{pmatrix} \cos(\phi_y) & 0 & \sin(\phi_y) \\ 0 & 1 & 0 \\ -\sin(\phi_y) & 0 & \cos(\phi_y) \end{pmatrix}; \mathbf{R}^Z = \begin{pmatrix} \cos(\phi_z) & -\sin(\phi_z) & 0 \\ \sin(\phi_z) & \cos(\phi_z) & 0 \\ 0 & 0 & 1 \end{pmatrix}; \quad (2.4)$$

Furthermore we use the convention that the detector system is rotated with respect to the laboratory system by ⁴

$$\mathbf{R} = \mathbf{R}^X \mathbf{R}^Y \mathbf{R}^Z. \quad (2.5)$$

We can now express the connection between a point on the detector as defined by pixel coordinates (y_{det} , z_{det}) and the corresponding point (x' , y' , z') in the laboratory system

$$\begin{pmatrix} x' \\ y' \\ z' \end{pmatrix} = \begin{pmatrix} L \\ 0 \\ 0 \end{pmatrix} + \mathbf{R} \begin{pmatrix} 0 \\ P_y(y_{\text{det}} - y_{\text{det}}(0)) \\ P_z(z_{\text{det}} - z_{\text{det}}(0)) \end{pmatrix} \quad (2.6)$$

Here by definition ($y_{\text{det}}(0)$, $z_{\text{det}}(0)$) are the pixel coordinates for the reference beam (the incident ray passing through $(x_1, y_1, z_1) = (0,0,0)$), P_y and P_z is the pixel size in the two directions, and L is the distance from center-of-rotation to the point where the incident ray hits the detector.

Diffraction

The scattering vector associated with the diffraction event is denoted $\underline{\mathbf{G}}$. To describe its relationship with reciprocal space we need five Cartesian coordinate systems: the laboratory system, the omega-system, the rotated system, the sample system, and the Cartesian grain system. These are identified by subscripts l, ω , γ , s and c, respectively. The former three have already been defined. Hence, the scattering vector transforms as $\underline{\mathbf{G}}_l = \underline{\mathbf{\Omega}} \underline{\mathbf{G}}_\omega = \underline{\mathbf{\Gamma}} \underline{\mathbf{G}}_\gamma$. (in case of only one rotation axis: $\underline{\mathbf{\Gamma}} = \underline{\mathbf{\Omega}}$).

The sample system is fixed with respect to the sample, as defined *a priori* by the experimentalist. As an example, in metallurgy the sample coordinates are typically defined by the rolling, transverse and normal directions of a rolled sheet (RD, TD, ND). The orientation of the sample on the ω turntable is given by the \mathbf{S} matrix: $\underline{\mathbf{G}}_\gamma = \mathbf{S} \underline{\mathbf{G}}_s$. By default $\mathbf{S} = \mathbf{I}$, the identity matrix.

⁴ Notably experience shows, that the order in which \mathbf{R}^X , \mathbf{R}^Y and \mathbf{R}^Z appears is important even for relatively small tilts.

The crystallographic orientation of a grain with respect to the sample is represented by \mathbf{U} ⁵, $\underline{\mathbf{G}}_s = \mathbf{U}\underline{\mathbf{G}}_c$. where index c refers to a Cartesian grain system $(\hat{x}_c, \hat{y}_c, \hat{z}_c)$. This is fixed with respect to the reciprocal lattice $(\underline{a}^*, \underline{b}^*, \underline{c}^*)$ in the grain. We use the convention that \hat{x}_c is parallel to \underline{a}^* , \hat{y}_c is in the plane of \underline{a}^* and \underline{b}^* , and \hat{z}_c is perpendicular to that plane. Let $\underline{\mathbf{G}}$ be represented in the reciprocal lattice system by the integer Miller indices $\underline{\mathbf{G}}_{hkl} = (h, k, l)$ [†]. The correspondence between the Cartesian grain system and reciprocal space is then given by the \mathbf{B} matrix: $\underline{\mathbf{G}}_c = \mathbf{B} \underline{\mathbf{G}}_{hkl}$, with

$$\mathbf{B} = \begin{pmatrix} a^* & b^* \cos(\gamma^*) & c^* \cos(\beta^*) \\ 0 & b^* \sin(\gamma^*) & -c^* \sin(\beta^*) \cos(\alpha) \\ 0 & 0 & c^* \sin(\beta^*) \sin(\alpha) \end{pmatrix} \quad (2.7)$$

and⁶

$$\cos(\alpha) = \frac{\cos(\beta^*) \cos(\gamma^*) - \cos(\alpha^*)}{\sin(\beta^*) \sin(\gamma^*)}. \quad (2.8)$$

Here $(a, b, c, \alpha, \beta, \gamma)$ and $(a^*, b^*, c^*, \alpha^*, \beta^*, \gamma^*)$ symbolize the lattice parameters in direct and reciprocal space, respectively.

With these definitions we have

$$\underline{\mathbf{G}}_l = \Gamma \text{SUB} \underline{\mathbf{G}}_{hkl}. \quad (2.9)$$

At times it is relevant to operate with normalised scattering vectors instead. Let $\underline{\mathbf{u}} = \underline{\mathbf{G}}_l / \|\underline{\mathbf{G}}_l\|$, $\underline{\mathbf{y}} = \underline{\mathbf{G}}_s / \|\underline{\mathbf{G}}_s\|$ and $\underline{\mathbf{h}} = \frac{\mathbf{B}\underline{\mathbf{G}}_{hkl}}{\|\mathbf{B}\underline{\mathbf{G}}_{hkl}\|}$. Then

$$\underline{\mathbf{u}} = \Gamma S \underline{\mathbf{y}} = \Gamma S U \underline{\mathbf{h}} = \begin{pmatrix} -\sin(\theta) \\ -\cos(\theta) \sin(\eta) \\ \cos(\theta) \cos(\eta) \end{pmatrix}. \quad (2.10)$$

For completeness we mention that Bragg's law provides an alternative way to determine the norm of the G-vectors. Using a formalism that includes 2π in the reciprocal space definition, the corresponding equation for $\underline{\mathbf{h}}$ is

$$\begin{pmatrix} h_1 \\ h_2 \\ h_3 \end{pmatrix} = \frac{\lambda}{4\pi \sin(\theta)} \mathbf{B} \begin{pmatrix} h \\ k \\ l \end{pmatrix}. \quad (2.11)$$

⁵ In the texture community it is customary to operate with $\mathbf{g} = \mathbf{U}^{-1}$. Notably this \mathbf{g} is not related to the metric \mathbf{g} in Eq. 2.22 nor the metric in Eq. 3.4.

⁶ There is an error in this formula in the book by Poulsen [1].

Forward projection

Let the diffraction event take place at position (x_l, y_l, z_l) in the sample system and be associated with a certain orientation \mathbf{U} . Assume further we know \mathbf{X} and Θ , the spacegroup, the lattice parameters (and therefore \mathbf{B}) and the x-ray wavelength. The task at hand is then to find the detector coordinates for the diffraction spot associated with a set of Miller indices \underline{G}_{hkl} .

We may proceed as follows. For each \underline{G}_{hkl} first we find \underline{G}_ω . The ω -position at which the diffraction condition is fulfilled is then given by

$$\sin(\theta) = -u_1 = -[\Omega \mathbf{G}_\omega]_1. \quad (2.12)$$

with $\sin(\theta)$ defined by Braggs law. This is a quadratic equation for determination of ω given \underline{G}_ω and \underline{h} . The two possible solutions are⁷

$$\cos \omega = \frac{ac \pm b\sqrt{D}}{a^2 + b^2}, \quad \sin \omega = \frac{bc \mp a\sqrt{D}}{a^2 + b^2}. \quad (2.13)$$

where

$$a = \frac{G_{\omega,1}}{|G_\omega|}, b = \frac{-G_{\omega,2}}{|G_\omega|}, c = \frac{\cos(2\theta) - 1}{\sqrt{2 - 2\cos(2\theta)}}, \quad \text{and} \quad \sqrt{D} = a^2 + b^2 - c^2. \quad (2.14)$$

Evidently the expression for c can be transformed in various ways. A different but equivalent formula can be found in the code in appendix.

Depending on ω -range 0, 1 or 2 of these solutions may be real. Also note that (2.13) has to be interpreted carefully to obtain ω , because taking only the arcos or arcsin will not do. The proper ω has to be chosen fulfilling both equations.

Knowing Ω we can compute \underline{G}_1 and \underline{u} . In the **case of an untilted detector** we have

$$\begin{aligned} P_Y(y_{\text{det}} - y_{\text{det}}(0)) &= y_l - (L - x_l) \tan(2\theta) \sin(\eta) = y_l + \frac{(L - x_l) \tan(2\theta)}{\cos(\theta)} u_2 = y_l + \frac{(L - x_l) \lambda}{2\pi \cos(2\theta)} G_{l,2}; \\ P_Z(z_{\text{det}} - z_{\text{det}}(0)) &= z_l + (L - x_l) \tan(2\theta) \cos(\eta) = z_l + \frac{(L - x_l) \tan(2\theta)}{\cos(\theta)} u_3 = z_l + \frac{(L - x_l) \lambda}{2\pi \cos(2\theta)} G_{l,3} \end{aligned} \quad (2.15)$$

Detector tilts:

Unfortunately, there are two different specifications of detector tilts around. They are equivalent (except for pixel rounding errors), but care has to be taken while converting from one representation to the other.

⁷ The derivation of Eqs. 2.13 and 2.14 is due to S. Schmidt.

The tilt specifications differ in two aspects. The origins of the detector coordinate systems are defined in two different reference systems, and in one case the origin has to be interpreted as the origin after tilting – in the other case before tilting. Details follow below.

First, let us introduce (and recap) some notations. The detector width and height in pixels are given by d_{width} and d_{height} ; the corresponding pixel sizes are denoted by P_y and P_z . The detector tilt angles are specified by the matrix \mathbf{R} in (2.5). We want to compute the actual detector pixels y_{det} and z_{det} recording the diffraction, which takes place at $(x_l, y_l, 0)$ in the laboratory system. The unit directional vector \mathbf{v} , along which diffraction occurs, is given by

$$\mathbf{v} = \begin{pmatrix} \cos(2\theta) \\ -\sin(2\theta)\sin(\eta) \\ \sin(2\theta)\cos(\eta) \end{pmatrix}. \quad (2.16)$$

Detector tilts I (used in PolyXSim, qnfs, ImageD11):

In this specification, the detector origin $(y_{\text{det}}(0), z_{\text{det}}(0))$ is defined as that pixel in the tilted detector coordinate system that corresponds to the $(L, 0, 0)^T$ beam center.

In the laboratory system, the diffracted beam thus hits the detector as expressed by the following equation:

$$\begin{pmatrix} L \\ 0 \\ 0 \end{pmatrix} + \mathbf{R} \begin{pmatrix} 0 \\ P_Y (y_{\text{det}} - y_{\text{det}}(0)) \\ P_Z (z_{\text{det}} - z_{\text{det}}(0)) \end{pmatrix} = \begin{pmatrix} x_l \\ y_l \\ 0 \end{pmatrix} + t\mathbf{v}. \quad (2.17)$$

We want to solve these three row equations in (2.16) for $t, y_{\text{det}}, z_{\text{det}}$. The solution is

$$t = \frac{R_{11}(L - x_l) - R_{21}y_l - R_{31}z_l}{R_{11}v_1 + R_{21}v_2 + R_{31}v_3}. \quad (2.18)$$

From which we find:

$$\begin{aligned} P_Y (y_{\text{det}} - y_{\text{det}}(0)) &= R_{12}((x_l - L) + tv_1) + R_{22}(y_l + tv_2) + R_{32}(z_l + tv_3) \\ P_Z (z_{\text{det}} - z_{\text{det}}(0)) &= R_{13}((x_l - L) + tv_1) + R_{23}(y_l + tv_2) + R_{33}(z_l + tv_3). \end{aligned} \quad (2.19)$$

Detector tilts II (used in the Grainsweeper and Monte Carlo programs):

However, this time the origin of the detector coordinate system $(y_{\text{det}}(0), z_{\text{det}}(0))$ is specified within the laboratory system (and not in the detector coordinate system as in the case above).

In the laboratory system, the diffracted beam thus hits the detector as expressed by the following equation:

$$\mathbf{R} \begin{pmatrix} 0 \\ P_Y(y_{\det} - d_{width}/2 + 1) \\ P_Z(z_{\det} - d_{height}/2 + 1) \end{pmatrix} + \begin{pmatrix} L \\ y_{\det}(0) \\ z_{\det}(0) \end{pmatrix} = \begin{pmatrix} x_l \\ y_l \\ 0 \end{pmatrix} + t\mathbf{v}. \quad (2.20)$$

From this equation we easily compute t (by left-multiplication with the transposed of the first column vector of \mathbf{R})

$$t = \frac{R_{11}(L - x_l) - R_{21}(y_{\det}(0) - y_l) - R_{31}z_{\det}(0)}{R_{11}v_1 + R_{21}v_2 + R_{31}v_3}. \quad (2.21)$$

Left-multiplication with the transposed of the second and third column vector of \mathbf{R} , respectively, yields

$$P_Y(y_{\det} - d_{width}/2 + 1) = (R_{12}, R_{22}, R_{32}) \cdot \left(t\mathbf{v} + \begin{pmatrix} x_l - L \\ y_l - y_{\det}(0) \\ -z_{\det}(0) \end{pmatrix} \right) \quad (2.22)$$

and

$$P_Z(z_{\det} - d_{height}/2 + 1) = (R_{13}, R_{23}, R_{33}) \cdot \left(t\mathbf{v} + \begin{pmatrix} x_l - L \\ y_l - y_{\det}(0) \\ -z_{\det}(0) \end{pmatrix} \right), \quad (2.23)$$

from which we immediately obtain (y_{\det}, z_{\det}) .

Conversion between both detector tilt specifications:

We actually just need to convert between (2.17) and (2.20). For notational convenience we write $y_{\det}(0)'$, $z_{\det}(0)'$ and L' for $y_{\det}(0)$, $z_{\det}(0)$ and L in (2.17). Then in (2.17) we get

$$\mathbf{R} \begin{pmatrix} 0 \\ P_Y(y'_{\det} - y'_{\det}(0)) \\ P_Z(z'_{\det} - z'_{\det}(0)) \end{pmatrix} = \mathbf{R} \begin{pmatrix} 0 \\ P_Y(y'_{\det} - d_{width}/2 + 1) \\ P_Z(z'_{\det} - d_{height}/2 + 1) \end{pmatrix} - \mathbf{R} \begin{pmatrix} 0 \\ P_Y(y_{\det}(0)' - d_{width}/2 + 1) \\ P_Z(z_{\det}(0)' - d_{height}/2 + 1) \end{pmatrix} \quad (2.24)$$

Thus we clearly get $(y_{\det}, z_{\det}) = (y'_{\det}, z'_{\det})$ from (2.17) and (*) if

$$\begin{pmatrix} L' \\ 0 \\ 0 \end{pmatrix} - \mathbf{R} \begin{pmatrix} 0 \\ P_Y(y_{\det}(0)' - d_{width}/2 + 1) \\ P_Z(z_{\det}(0)' - d_{height}/2 + 1) \end{pmatrix} = \begin{pmatrix} L \\ y_{\det}(0) \\ z_{\det}(0) \end{pmatrix}, \quad (2.25)$$

or equivalently

$$\mathbf{R} \begin{pmatrix} L' \\ 0 \\ 0 \end{pmatrix} - \begin{pmatrix} 0 \\ P_Y(y_{\det}(0)' - d_{width}/2 + 1) \\ P_Z(z_{\det}(0)' - d_{height}/2 + 1) \end{pmatrix} = \begin{pmatrix} L \\ y_{\det}(0) \\ z_{\det}(0) \end{pmatrix}. \quad (2.27)$$

Since \mathbf{R} is invertible (and $R_{11} \neq 0$, since y- and z-tilts are less than 90 degrees) one can uniquely convert between $(L', y_{\text{det}}(0)', z_{\text{det}}(0)')$ and $(L, y_{\text{det}}(0), z_{\text{det}}(0))$. Plugging $(L', y_{\text{det}}(0)', z_{\text{det}}(0)')$ into (2.27) immediately gives $(L, y_{\text{det}}(0), z_{\text{det}}(0))$. On the other hand, given $(L, y_{\text{det}}(0), z_{\text{det}}(0))$ we set

$$\mathbf{d} := \begin{pmatrix} d_1 \\ d_2 \\ d_3 \end{pmatrix} = \mathbf{R}^T \begin{pmatrix} L \\ y_{\text{det}}(0) \\ z_{\text{det}}(0) \end{pmatrix},$$

And deduce from (2.27):

$$\begin{aligned} L' &= d_1 / R_{11}, \\ y_{\text{det}}(0)' &= d_{\text{width}} / 2 - 1 + (R_{12}L' - d_2) / P_Y, \\ z_{\text{det}}(0)' &= d_{\text{height}} / 2 - 1 + (R_{13}L' - d_3) / P_Z. \end{aligned}$$

In summary: If the detector origin is specified in the detector pixel reference frame, then tilt specification I used, otherwise specification II. Conversion between these two systems means that three parameters need to be converted, namely $(L, y_{\text{det}}(0), z_{\text{det}}(0))$.

Flipping:

Different detector choices in the actual experimental might entail additional y and/or z flipping of the detector coordinate. If flipping is needed, it should be performed as final step (also after detector tilting, as described above).

A Test Case for Verifying Forward Projection Calculations

Debugging one's own forward projection code can turn out to be quite demanding. To help in this process we include here one example, which demonstrates what results should be obtained.

Input:

The calculations assume the following setting.

```
//The energy of the monochromatic X-ray (in keV).
DIFFR_ENERGY      69.533

// **** Y and Z coordinates of the detector origin (0,0) on the detector (in mm).
DIFFR_D_0_Y       0.18296
DIFFR_D_0_Z       -0.08347

// **** Sample to detector distance (in mm).
DIFFR_L_S2D       9.27058

// **** Size of a pixel on the detector in Y and Z directions (in mm).
DIFFR_PX_SIZE_Y   0.0043
DIFFR_PX_SIZE_Z   0.0043
```

```

// **** Detector tilts (in degrees)
DET_TILT_X   1.0988
DET_TILT_Y   2.085
DET_TILT_Z   3.473

// **** Space group ***
DIFFR_SPACE_GROUP 225

// **** Lattice parameters for space group "DIFFR_SPACE_GROUP" in direct space.
DIFFR_LATTICE_A      4.05
DIFFR_LATTICE_B      4.05
DIFFR_LATTICE_C      4.05

// **** Y size of the detector (in pixels), that is the width of the detector.
DIFFR_D_WIDTH        1536

// **** Z size of the detector (in pixels), that is the height of the detector.
DIFFR_D_HEIGHT       1024

```

Note that the detector tilt corresponds to the tilt specification II. For convenience, we provide the proper detector origin according to tilt specification I (using the conversion (2.27)). The detector origin in this case is $(L', y_{\text{det}}(0), z_{\text{det}}(0)) = (9.284758, 724.953252, 531.210607)$.

Consider the point $(x_l, y_l, 0) = (-0.0602, 0.215, 0)$ in the sample.

We assume that the grain orientation at this point is given by the Euler angles $(\varphi_1, \phi, \varphi_2) = (209.423715, 26.208917, 126.576384)$, or equivalently by the U matrix

$$U = \begin{pmatrix} 0.872986 & 0.436832 & -0.216965 \\ -0.334822 & 0.860185 & 0.384678 \\ 0.354669 & -0.263174 & 0.897190 \end{pmatrix}.$$

Output:

For the $\{-2, -2, 2\}$ hkl-lattice plane we obtain diffraction at a position specified by the following angles (in degrees)

$$\begin{aligned} \omega &= 79.793676, \\ \theta &= 4.373314, \\ \eta &= 62.190963, \end{aligned}$$

which hits the following pixel on the detector $(y_{\text{det}}, z_{\text{det}}) = (418, 698)$.

Note that you might have to possibly flip the detector (around y and/or z) to obtain the same pixel coordinates.

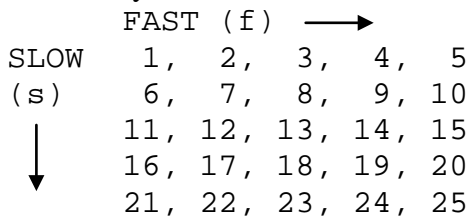
Detector orientation parameter

All the peak positions are calculated above in the detector coordinates (y_{det}, z_{det}). Reading the intensities differs from detector to detector (differ from which corner it is read out and whether it reads columns or rows first) and from setup to setup. Therefore a transformation need to be performed to get the intensities get the coordinate set of pixel intensities to meet the definitions above. Reading the pixels from a detector we get one long row of pixel values 1,2,3,4,...., $d_{width} * d_{height}$. We call the raw pixel coordinates read from the detector (slow, fast) or (s,f). fast for the fastest changing index and and slow for the

E.g. A 5 by 5 image

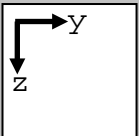
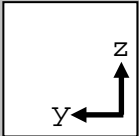

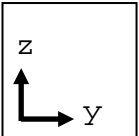
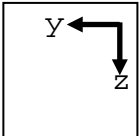

1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11,, 25

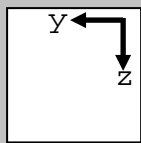
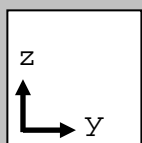

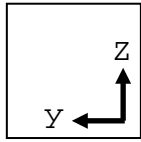
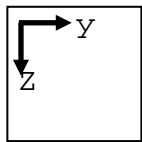

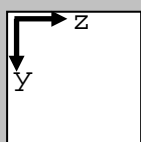
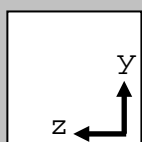

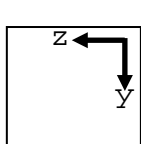
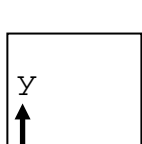

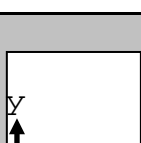
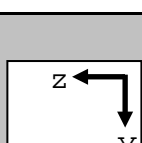
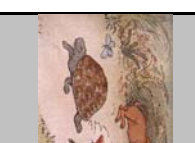
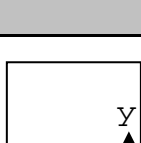
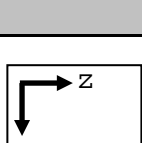
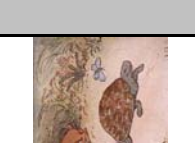
this 1d array will be made into a 5x5 image as



$$\begin{pmatrix} o11 & o12 \\ o21 & o22 \end{pmatrix} \begin{pmatrix} s \\ f \end{pmatrix} = \begin{pmatrix} z_{det} \\ y_{det} \end{pmatrix}$$

In the table below (y, z) is just short for (z_{det}, y_{det})

Orientation matrix	Standard Image setting (0,0) top left	3DXRD setting (0,0) lower right (180° rotation)	Example picture after operating by O in the 3DXRD setting shown in the former column	Transformation to make image in standard orientation – to read the image array as (z,y).	Transformation to make image in 3DXRD orientation. Image viewed as looking onto the detector along the beam
$\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$				none	fliplr+flipud
$\begin{pmatrix} -1 & 0 \\ 0 & 1 \end{pmatrix}$				flipud	fliplr

$\begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$				fliplr	flipud
$\begin{pmatrix} -1 & 0 \\ 0 & -1 \end{pmatrix}$				fliplr+flipud	none
$\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$				transpose	transpose + fliplr + flipud
$\begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix}$				transpose+ flipud	transpose+ fliplr
$\begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix}$				transpose+ fliplr	transpose+ flipud
$\begin{pmatrix} 0 & -1 \\ -1 & 0 \end{pmatrix}$				transpose + fliplr + flipud	transpose

The orientation matrix elements ($o11$, $o12$, $o21$, $o22$) can be determined using the ImageViewer program of the FABLE suite. If the image has specific known features (shadows of beamstop or other equipment) one can choose the 8 different possibilities, when the detector image has the orientation "up is up and left side is left" as if looking along the beam on the detector the correct o-matrix has been determined.

Indexing⁸

A main task for 3DXRD programs is to index polycrystalline materials. Here we introduce some notation – as used primarily in the program ImageD11 – for the case of indexing a single crystal. We shall assume that the \mathbf{B} matrix is not a priori known.

For each spot on the detector we know $\mathbf{\Omega}$, \mathbf{X} and $\mathbf{\Theta}$ and as such we can determine $\underline{\mathbf{G}}_\gamma$. Associating these with a set of Miller indices $\underline{\mathbf{G}}_{hkl}$, we have

$$\underline{\mathbf{G}}_\gamma = \mathbf{\Theta}^{-1} \mathbf{X}^{-1} \mathbf{\Omega}^{-1} \underline{\mathbf{G}}_{hkl} = (\mathbf{UB}) \underline{\mathbf{G}}_{hkl}. \quad (2.28)$$

The process of indexing is then the process of determining (\mathbf{UB}) or $(\mathbf{UB})^{-1}$ so that $\underline{\mathbf{G}}_{hkl}$ are simple integers.

To continue we introduce the metric tensor $\mathbf{g}^{-1} = (\mathbf{UB})^t(\mathbf{UB})$

$$\mathbf{g} = \begin{pmatrix} a^2 & ab \cos(\gamma) & ac \cos(\beta) \\ ab \cos(\gamma) & b^2 & bc \cos(\alpha) \\ ac \cos(\beta) & bc \cos(\alpha) & c^2 \end{pmatrix}. \quad (2.29)$$

The unit cell volume is given by the determinant of \mathbf{g} . Adding a “*” to all symbols we get the same relations in reciprocal space with $\mathbf{g}^* = \mathbf{g}^{-1}$.

It turns out that (since $\mathbf{U}^t = \mathbf{U}^{-1}$)

$$\mathbf{g}^* = (\mathbf{UB})^t(\mathbf{UB}) = \mathbf{B}^t \mathbf{U}^t \mathbf{UB} = \mathbf{B}^t \mathbf{B} \quad (2.30)$$

Hence, knowing \mathbf{g} one can determine \mathbf{U} and \mathbf{B} by Cholesky decomposition [4]. Alternatively QR decomposition can be used to determine \mathbf{U} and \mathbf{B} from \mathbf{UB} . For a real square matrix \mathbf{A} , QR decomposition is the process of determining \mathbf{Q} orthogonal and \mathbf{R} upper triangular such that $\mathbf{A} = \mathbf{QR}$. If \mathbf{UB} is nonsingular and we require that the diagonal elements of \mathbf{B} are positive (cf. the definition of \mathbf{B} (2.7)), then the factorization is unique.

Units

We adapt the convention that all angles in user input/output are given in degrees. Linear dimensions will be specified in each module, but will preferably be in mm and μm .

Possible extensions

In later versions we will aim at introducing

- Addition of x-, y-, and z-translations as well as partly illuminated samples (stripes, grains moving in and out of the beam)
- Additional rotations, such as a tilt below the omega stage.
- Inclusion of effects related to beam divergence and energy spread.

⁸ This section by J. Wright

3. Representation of crystallographic orientation

Crystallographic orientations can be expressed in numerous ways, as described in detail in the literature on texture [5,6]. We will use four representations. For algebra the natural choice is the 3x3 orthogonal matrix \mathbf{U} or its inverse (transpose) \mathbf{g} , as defined above. For visualisation and sampling a representation by three parameters is preferable. We have chosen three alternatives: Euler angles, Rodrigues vectors and quaternions. In the following we provide definitions and summarise the most important transforms and geometric properties for each of the representations. A discussion of crystal symmetry will be given in chapter 4.

Euler angles (Bunge definition)

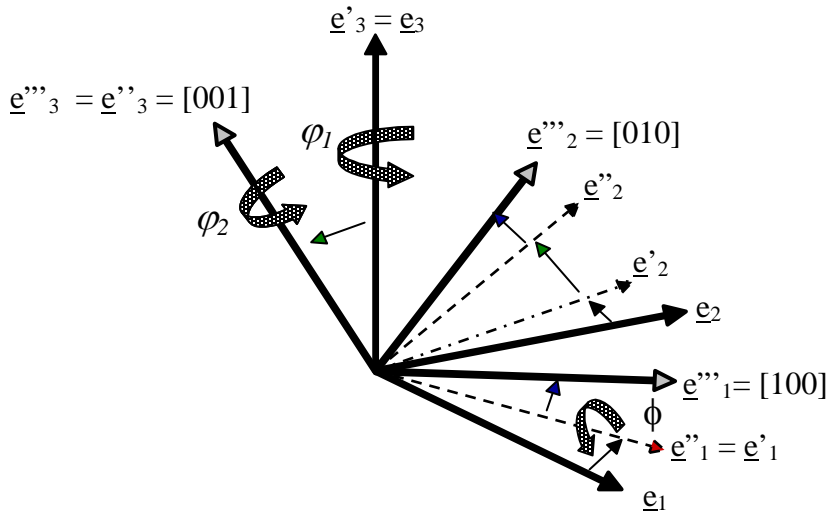


Figure 3.1

Definition of the Euler angles ($\varphi_1, \phi, \varphi_2$) according to Bunge [7]. The sample system ($\underline{e}_1, \underline{e}_2, \underline{e}_3$) is rotated first around the third axis by φ_1 , then around the new first axis \underline{e}'_1 by ϕ and finally around the new third axis \underline{e}''_3 by φ_2 to match the Cartesian grain system ($\underline{e}'''_1, \underline{e}'''_2, \underline{e}'''_3$). For cubic systems the latter set is identical to the reciprocal axes ([100], [010], [001]).

Traditionally, orientations are parameterized by a set of Euler angles ($\varphi_1, \phi, \varphi_2$), expressing subsequent rotations around three axes, cf. Fig 3.1. With the definitions of the angles as provided by Bunge [7]:

$$\mathbf{U} = \begin{bmatrix} c(\varphi_1)c(\varphi_2) - s(\varphi_1)s(\varphi_2)c(\phi) & -c(\varphi_1)s(\varphi_2) - s(\varphi_1)c(\varphi_2)c(\phi) & s(\varphi_1)s(\phi) \\ s(\varphi_1)c(\varphi_2) + c(\varphi_1)s(\varphi_2)c(\phi) & -s(\varphi_1)s(\varphi_2) + c(\varphi_1)c(\varphi_2)c(\phi) & -c(\varphi_1)s(\phi) \\ s(\varphi_2)s(\phi) & c(\varphi_2)s(\phi) & c(\phi) \end{bmatrix} \quad (3.1)$$

Here we have used the shorthands c and s for \cos and \sin , respectively. The reverse relationship is given by (code is provided in an appendix):

$$\begin{aligned}
\varphi_1 &= -\arctan(U_{13}, U_{23}), \\
\phi &= \arccos(U_{33}), \\
\varphi_2 &= \arctan(U_{31}, U_{32}).
\end{aligned}
\tag{3.2}$$

Note that the two-argument inverse tangent $\arctan(x,y)$ computes $\arctan(x/y)$, but returns the correct angle by taking into account in which quadrant (x,y) is in.

Calculating the combined rotation of two individual rotations is very cumbersome in the Euler angle representation. This should be done using one of the other representations.

Sampling Euler space is non-linear with singularities at $\phi = 0$ and π . In order to sample it uniformly use of the metric $d\mathbf{g}$ is required [7]

$$d\mathbf{g}(\phi, \varphi_1, \varphi_2) = \frac{1}{8\pi^2} \sin(\phi) d\phi d\varphi_1 d\varphi_2. \tag{3.3}$$

Projection lines: From a single diffraction event, only the direction of the scattering vector is probed. Measurements will be invariant to a rotation of the sample around the vector. The set of orientations, which for given \underline{y} and \underline{h} fulfils $\underline{y} = \mathbf{U} \underline{h}$ (cf. Eq 2.10) constitutes a curve in orientation space, the so-called projection line for pole-figure inversion. It is evident from Eq. 3.1 that such lines are curved in Euler space.

Rodrigues vectors

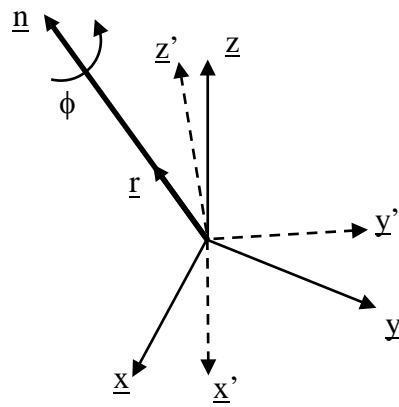


Figure 3.2

Definition of the Rodrigues vector \underline{r} . The coordinate system (x,y,z) is rotated around \underline{n} by angle ϕ into (x',y',z') . \underline{r} is parallel to \underline{n} : $\underline{r} = \underline{n} \tan(\phi/2)$.

The Rodrigues representation is in several ways more elegant and better suited for numerical work in connection with diffraction data [8-10]. It is based on the fact that any rotation can be represented in a unique way by a rotation axis \underline{n} and a rotation angle ϕ , defined on $[0 \pi]$. In the Rodrigues parameterisation these are coupled in the definition of the Rodrigues vector [11]

$$\underline{r} = \tan(\phi/2)\underline{n}. \tag{3.4}$$

The definition is illustrated in Fig 3.2.

The vector \underline{r} can be treated as a vector in \mathbf{R}^3 , with the exception of points with a rotation angle of π , which are represented by two opposite points in infinity. The axes of this space are co-linear with those of the sample system in the sense, that a vector $\underline{r} = (r_1, 0, 0)$ describes a rotation around the sample x-axis.

The relationship to $\mathbf{g} = \mathbf{U}^{-1}$ is given by⁹

$$g_{ij} = \frac{1}{r^2} [(1 - r^2)\delta_{ij} + 2r_i r_j - 2\varepsilon_{ijk} r_k], \quad (3.5)$$

where $r^2 := r_k r_k$, and ε_{ijk} is the permutation tensor:

$$\varepsilon_{ijk} = \begin{cases} +1 & \text{if } (i, j, k) \text{ is } (1, 2, 3), (2, 3, 1) \text{ or } (3, 1, 2), \\ -1 & \text{if } (i, j, k) \text{ is } (3, 2, 1), (1, 3, 2) \text{ or } (2, 1, 3), \\ 0 & \text{otherwise: } i = j \text{ or } j = k \text{ or } k = i, \end{cases}$$

The reverse relationship (determining \underline{r} given \mathbf{U}) is defined by:

$$r_i = \frac{\varepsilon_{ijk} g_{jk}}{1 + g_{mm}}, \quad (3.6)$$

where g_{mm} is the trace of g . The result \underline{r}_3 of two rotations, first \underline{r}_1 and then \underline{r}_2 is

$$\underline{r}_3 = \frac{\underline{r}_1 + \underline{r}_2 - \underline{r}_1 \times \underline{r}_2}{1 - \underline{r}_1 \cdot \underline{r}_2}$$

Sampling The metric d_{ij} and the volume element dV are

$$d_{ij} = \frac{1}{(1 + r^2)^2} [(1 + r^2)\delta_{ij} - 2r_i r_j]; \quad (3.7)$$

$$dV = \sqrt{\det(d_{ij})} = \frac{dr_1 dr_2 dr_3}{1 + r^2}. \quad (3.8)$$

Evidently this leads to complications for $r \rightarrow \infty$ (that is for low crystal symmetry, cf. Chapter 4). On the other hand, for $\phi \rightarrow 0$ rotations become commutative. This is reflected in the fact that the metric becomes linear. As an example, for orientation distributions characterized by $\phi < 10$ deg, the space is Euclidean within an accuracy of better than 1%.

⁹ In Eqs 3.5 and 3.6 we use Einstein summation rule. That is the sign for summation over vector and tensor suffices is omitted. Summation is understood with respect to all suffices that appear twice in a given term.

Projection lines A key fact is that the projection lines for pole-figure inversion are straight lines. Specifically, for a given set of vectors \underline{h} and \underline{y} , the projection line is given by

$$\underline{r} = \underline{r}^0 + t \frac{\underline{h} + \underline{y}}{1 + \underline{h} \cdot \underline{y}}; \quad -\infty < t < \infty \quad (3.9)$$

$$\underline{r}^0 = \frac{\underline{h} \times \underline{y}}{1 + \underline{h} \cdot \underline{y}} = \tan\left(\frac{\phi_0}{2}\right) \underline{n}. \quad (3.10)$$

where \underline{r}^0 is the rotation from \underline{h} into \underline{y} with the minimum rotation angle ϕ_0 . The second term in Eq. 3.7 corresponds to an arbitrary rotation around the symmetric position $\underline{h} + \underline{y}$.

For $\phi \rightarrow 0$ the expression for the projection line simplifies to

$$\underline{r} = \frac{1}{2}(\underline{h} \times \underline{y}) + \frac{t}{2}(\underline{h} + \underline{y}). \quad (3.11)$$

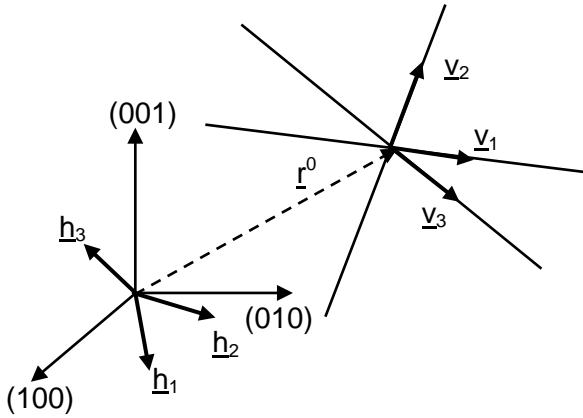


Figure 3.3.

Symmetry of projection lines through point \underline{r}^0 in Rodrigues space. The directions of three normalised (h,k,l)-vectors (\underline{h}_1 , \underline{h}_2 , \underline{h}_3) is marked. The associated projection lines pass through \underline{r}^0 with directions (\underline{v}_1 , \underline{v}_2 , \underline{v}_3).

As a first application of the Rodrigues vector formalism let us address the question: for a given orientation \underline{r}^0 , what is the geometry of the projection lines observed?

The answer is illustrated in Fig 3.3, where the full Rodrigues space is used. Assume that detection geometry and intensity concerns enable the observation of the set of reflections (\underline{h}_1 , \underline{h}_2 , ..., \underline{h}_N). Then from Eq. 3.8 it follows that the associated projection lines are straight line passing through \underline{r}^0 with the i'th line pointing in the direction

$$\underline{v}_i = \frac{(I + U)\underline{h}_i}{1 + \underline{h}_i U \underline{h}_i}. \quad (3.12)$$

As the expression conserves angles between vectors, the set $(\underline{v}_1, \underline{v}_2, \dots, \underline{v}_N)$ is a rotation of the set $(\underline{h}_1, \underline{h}_2, \dots, \underline{h}_N)$. In other words the projection lines associated with a given orientation exhibit the underlying crystal symmetry.

Quaternions

Quaternions and their applications to orientations have a well-developed theory [12-16]. In this section we give only a skeleton development, restricting our attentions to those definitions and facts that we absolutely need for our purpose.

Quaternion basics: A quaternion \mathbf{q} is a 4-tuple (a, b, c, d) of real numbers. The product of two quaternions is defined by

$$(a_1, b_1, c_1, d_1) (a_2, b_2, c_2, d_2) = (a_3, b_3, c_3, d_3), \quad (3.13)$$

where

$$\begin{aligned} a_3 &= a_1a_2 - b_1b_2 - c_1c_2 - d_1d_2, \\ b_3 &= a_1b_2 + b_1a_2 + c_1d_2 - d_1c_2, \\ c_3 &= a_1c_2 - b_1d_2 + c_1a_2 + d_1b_2, \\ d_3 &= a_1d_2 + b_1c_2 - c_1b_2 + d_1a_2. \end{aligned} \quad (3.14)$$

Since the non-commutative multiplication is associative, we can use unambiguously the notation \mathbf{pqr} for the product of the three quaternions \mathbf{p} , \mathbf{q} , and \mathbf{r} . Note also that $(1, 0, 0, 0)$ is an identity element; i.e., for any quaternion \mathbf{q} we have $(1, 0, 0, 0)\mathbf{q} = \mathbf{q} = \mathbf{q}(1, 0, 0, 0)$.

The norm of the quaternion (a, b, c, d) is defined by

$$\| (a, b, c, d) \| = \sqrt{a^2 + b^2 + c^2 + d^2}. \quad (3.15)$$

We now specify two useful subsets of the set of quaternions. For both of these it is helpful to introduce the notation (a, \mathbf{b}) , where \mathbf{b} is the 3D vector (b, c, d) , to abbreviate the quaternion (a, b, c, d) . The conjugate of the quaternion $\mathbf{q} = (a, \mathbf{b})$ is defined to be $\bar{\mathbf{q}} = (a, -\mathbf{b})$. The conjugate of the product of two quaternions is the reversed product of their conjugates; i.e., $\overline{\mathbf{q}_1\mathbf{q}_2} = \bar{\mathbf{q}}_2\bar{\mathbf{q}}_1$.

Relation between unit quaternions and orientations:

The set of unit quaternions consists of quaternions whose norm is 1. They form $SO(3)$, the 3-sphere in 4-space, similar to e.g. the Earth surface being a 2-sphere in 3-space. Apparently, unit quaternion space is bounded. The product of two unit quaternions is a unit quaternion, and that the conjugate of a unit quaternion is also a unit quaternion.

Every unit quaternion \mathbf{q} defines an orientation as it can be expressed as:

$$\mathbf{q} = (a, \mathbf{b}) = (\cos(\varphi/2), \mathbf{n} \sin(\varphi/2)), \quad (3.16)$$

where the unit vector $\mathbf{n} = \mathbf{b}/\|\mathbf{b}\|$ defines the axis and $\varphi = 2\arccos(a/\|\mathbf{b}\|)$ the angle of rotations. Note that \mathbf{q} and $-\mathbf{q}$ define the same orientation. In fact, this mapping from $SO(3)$ to orientation space (i.e., Rodrigues space, or Euler angle space, etc.) is a 2-1 mapping.

A particularly useful and elegant consequence of this approach is that the rotation represented by the unit quaternion \mathbf{q}_1 followed by the rotation represented by the unit quaternion \mathbf{q}_2 is the composite rotation represented by $\mathbf{q}_2\mathbf{q}_1$.

It follows from Eqs. 3.4 and 3.16 that there is a gnostic relationship between Rodrigues vectors and unit quaternions. This implies that one can go forth and back, e.g. for visualisation in Rodrigues space and for calculations in quaternion space.

Via (3.16) we can compute for every unit quaternion \mathbf{q} the corresponding \mathbf{n} and ϕ . Setting $\underline{r} = \tan(\phi/2)\underline{n}$, with $\underline{n}=\mathbf{n}$ and $\phi=\phi$, we obtain the corresponding Rodrigues vector \underline{r} . Vice versa, given \underline{r} we compute \mathbf{q} via (3.16) with $\phi=2\arctan(\|\underline{r}\|)$ and $\mathbf{n}=\underline{r}/\|\underline{r}\|$.

The relationship from $\mathbf{q}=(a,b,c,d)$ to \mathbf{U} is given by

$$\mathbf{U} = \begin{pmatrix} 2(a^2 + b^2) - 1 & 2(bc - ad) & 2(bd + ac) \\ 2(bc + ad) & 2(a^2 + c^2) - 1 & 2(cd - ab) \\ 2(bd - ac) & 2(cd + ab) & 2(a^2 + d^2) - 1 \end{pmatrix}. \quad (3.17)$$

The inverse relationship (determining \mathbf{q} given \mathbf{U}) is, of course due to the \mathbf{q} and $-\mathbf{q}$ ambiguities, not uniquely defined. We give only one formula, which works if and only if $U_{11}+U_{22}+U_{33} \geq -1$, for the other cases see appendix.

$$\begin{aligned} a &= \frac{1}{2}\sqrt{U_{11} + U_{22} + U_{33} + 1}, \\ b &= \frac{1}{4a}(U_{32} - U_{23}), \\ c &= \frac{1}{4a}(U_{13} - U_{31}), \\ d &= \frac{1}{4a}(U_{21} - U_{12}). \end{aligned} \quad (3.18)$$

Distance

Following [17] we define the distance d between two orientations \mathbf{q}_1 and \mathbf{q}_2 by

$$d(q_1, q_2) = \min_{s_1, s_2 \in S} r(q_1 s_1, q_2 s_2) \quad (3.19)$$

where S is the set of symmetry rotations given by crystal symmetry – see chapter 4 – and r is

$$r(q_1, q_2) = 1 - |a_3| = 1 - |\cos(\phi_{12})|. \quad (3.20)$$

In other words, the distance d is the minimal r when comparing all permutations of symmetry operations. For computing it is relevant to note that it is sufficient to loop over the symmetry operations once, as we can rewrite (3.17) to be

$$d(q_1, q_2) = \min_{s \in S} r\left((1,0,0,0), sq_2 \bar{q}_1\right) \quad (3.19)$$

Usually in the materials science literature, we specify the distance $d(\mathbf{q}_1, \mathbf{q}_2)$ by the associated angle of disorientation, which is defined as that ϕ in the range $[0^\circ, 180^\circ]$ for which $\cos(\phi/2) = 1 - d$.

Sampling

Knowing that $a^2 + b^2 + c^2 + d^2 = 1$, a discretisation by three parameters is evidently possible. The superior way to do so is not known to the authors. In [17,18] one has chosen to sample in a regular grid over the components b , c , and d in the interval $[-1, 1]$ and allowing only values that satisfy $b^2 + c^2 + d^2 \leq 1$. This sampling leads to neighbouring voxel elements having misorientations that increases with the distance to $\mathbf{q} = (1,0,0,0)$.

Projection lines

In quaternion space the projection lines become circles. Following [16] the circle for a given set of \underline{h} and \underline{r} is defined by (under condition $\underline{h} \times \underline{r} \neq 0$) the two orthogonal unit quaternions \mathbf{q}_1 and \mathbf{q}_2

$$q_1 = \left(\cos\left(\frac{\phi}{2}\right), \frac{\underline{h} \times \underline{r}}{\|\underline{h} \times \underline{r}\|} \sin\left(\frac{\phi}{2}\right) \right); \quad q_2 = \left(0, \frac{\underline{h} + \underline{r}}{\|\underline{h} + \underline{r}\|} \right) \quad (3.20)$$

where ϕ is defined as the angle between \underline{h} and \underline{r} , that is:

$$\|\underline{h} + \underline{r}\| = 2 \cos\left(\frac{\phi}{2}\right). \quad (3.21)$$

With these definitions the circle is

$$C(q_1, q_2) = \{q(t) = q_1 \cos(t) + q_2 \sin(t) : t \in [0, 2\pi]\} \quad (3.22)$$

4. Representation of elastic strain¹⁰

Definition of strain

The strain is a perturbation of the local lattice. It is represented by a symmetric 3x3 matrix, the strain tensor, $\boldsymbol{\varepsilon}$. Notably, $\boldsymbol{\varepsilon}$ is defined in direct space co-ordinates, not in reciprocal space. Hence, for each position we define a Cartesian system with axes $(\hat{x}_d, \hat{y}_d, \hat{z}_d)$, and with \hat{x}_d parallel to \underline{a} , \hat{y}_d in the plane of \underline{a} and \underline{b} , and \hat{z}_d perpendicular to that plane. By analogy with (2.7) the transformation between the two systems is given by a matrix:

$$\mathbf{A} = \begin{pmatrix} a & b \cos \gamma & c \cos \beta \\ 0 & b \sin \gamma & -c \sin \beta \cos \alpha^* \\ 0 & 0 & c \sin \beta \sin \alpha^* \end{pmatrix}. \quad (4.1)$$

¹⁰ This section was added by J.Oddershede based on Chapter 3.5 of reference 1 and Appendix 2 of reference 20.

Now chose a direct space reference coordinate system and let A_0 relate the undeformed Cartesian grain system to this reference while A relates the deformed Cartesian grain system to the same reference:

$$\begin{array}{|c|} \hline \text{undeformed} \\ \hline \text{sample} \\ \hline \text{system} \\ \hline \end{array} \begin{array}{c} \mathbf{U}_0^T \\ \rightleftharpoons \\ \mathbf{U}_0 \end{array} \begin{array}{|c|} \hline \text{undeformed} \\ \hline \text{grain} \\ \hline \text{system} \\ \hline \end{array} \begin{array}{c} \mathbf{A}_0^{-1} \\ \rightleftharpoons \\ \mathbf{A}_0 \end{array} \begin{array}{|c|} \hline \text{direct space} \\ \hline \text{reference} \\ \hline \text{system} \\ \hline \end{array} \begin{array}{c} \mathbf{A} \\ \rightleftharpoons \\ \mathbf{A}^{-1} \end{array} \begin{array}{|c|} \hline \text{deformed} \\ \hline \text{grain} \\ \hline \text{system} \\ \hline \end{array} \begin{array}{c} \mathbf{U} \\ \rightleftharpoons \\ \mathbf{U}^T \end{array} \begin{array}{|c|} \hline \text{deformed} \\ \hline \text{sample} \\ \hline \text{system} \\ \hline \end{array} \quad (4.2)$$

Let T be given by:

$$T = \mathbf{A}\mathbf{A}_0^{-1}. \quad (4.3)$$

Then, by definition the strain tensor is:

$$\boldsymbol{\varepsilon}_{ij} = \frac{1}{2}(\mathbf{T}_{ij} + \mathbf{T}_{ji}) - \mathbf{I}_{ij}, \quad (4.4)$$

where \mathbf{I} is the identity matrix. This strain tensor defined as the deformation of the lattice relative to the undeformed Cartesian grain system is sometimes known as the linear Lagrangian strain tensor [19].

With this general formalism, the relation between the diffraction geometry and the strain is via the metric \mathbf{B} , cf. (2.11). If \mathbf{B} is known, \mathbf{A} can be derived, from which follows $\boldsymbol{\varepsilon}$ by (4.3) and (4.4). And the other way around; if $\boldsymbol{\varepsilon}$ is known T can be obtained by rearranging (4.4), and then \mathbf{B} can be derived from $\mathbf{A}=\mathbf{T}\mathbf{A}_0$.

Note that the elastic strain tensor calculated as outlined above refers to the local Cartesian grain coordinate system. Under the assumption that the orientation of the grain does not change with the deformation, that is $\mathbf{U}=\mathbf{U}_0$, the strain tensor in the sample coordinate system is given by (cf. (4.2)):

$$\boldsymbol{\varepsilon}_{\text{sample}} = \mathbf{U}\boldsymbol{\varepsilon}\mathbf{U}^T \quad (4.5)$$

Strain to stress conversion¹¹

The strain tensor $\boldsymbol{\varepsilon}$ is related to the stress tensor $\boldsymbol{\sigma}$ by means of the stiffness tensor \mathbf{C} , which is a 6x6 symmetric matrix where the number of independent terms varies from 21 for a triclinic to 3 for a cubic crystal. The easiest way to convert from strain to stress is by writing $\boldsymbol{\varepsilon}$ and $\boldsymbol{\sigma}$ in the Mandel-Voigt notation:

$$\boldsymbol{\varepsilon}^{MV} = \begin{pmatrix} \varepsilon_{11} \\ \varepsilon_{22} \\ \varepsilon_{33} \\ \sqrt{2}\varepsilon_{23} \\ \sqrt{2}\varepsilon_{13} \\ \sqrt{2}\varepsilon_{12} \end{pmatrix}, \quad \boldsymbol{\sigma}^{MV} = \begin{pmatrix} \sigma_{11} \\ \sigma_{22} \\ \sigma_{33} \\ \sqrt{2}\sigma_{23} \\ \sqrt{2}\sigma_{13} \\ \sqrt{2}\sigma_{12} \end{pmatrix} \quad (4.6)$$

¹¹ The authors owe Joel Bernier, Lawrence Livermore National Lab. USA many thanks for the derivations leading this section.

We now define the orthogonal rotation matrix in the Mandel-Voigt frame as:

$$\mathbf{U}^{MV} = \begin{pmatrix} [\mathbf{U}_1^{MV}] & [\mathbf{U}_2^{MV}] \\ [\mathbf{U}_3^{MV}] & [\mathbf{U}_4^{MV}] \end{pmatrix} \quad (4.7)$$

where

$$\mathbf{U}_1^{MV} = \begin{bmatrix} U_{11}^2 & U_{12}^2 & U_{13}^2 \\ U_{21}^2 & U_{22}^2 & U_{23}^2 \\ U_{31}^2 & U_{32}^2 & U_{33}^2 \end{bmatrix} \quad (4.8)$$

$$\mathbf{U}_2^{MV} = \sqrt{2} \begin{bmatrix} U_{12}U_{13} & U_{11}U_{13} & U_{11}U_{12} \\ U_{22}U_{23} & U_{21}U_{23} & U_{21}U_{22} \\ U_{32}U_{33} & U_{31}U_{33} & U_{31}U_{32} \end{bmatrix} \quad (4.9)$$

$$\mathbf{U}_3^{MV} = \sqrt{2} \begin{bmatrix} U_{21}U_{31} & U_{22}U_{32} & U_{23}U_{33} \\ U_{11}U_{31} & U_{12}U_{32} & U_{13}U_{33} \\ U_{11}U_{21} & U_{12}U_{22} & U_{13}U_{23} \end{bmatrix} \quad (4.10)$$

$$\mathbf{U}_4^{MV} = \begin{bmatrix} U_{23}U_{32} + U_{22}U_{33} & U_{23}U_{31} + U_{21}U_{33} & U_{32}U_{21} + U_{31}U_{22} \\ U_{13}U_{32} + U_{12}U_{33} & U_{13}U_{31} + U_{11}U_{33} & U_{31}U_{12} + U_{32}U_{11} \\ U_{12}U_{23} + U_{13}U_{22} & U_{21}U_{13} + U_{23}U_{11} & U_{12}U_{21} + U_{11}U_{22} \end{bmatrix} \quad (4.11)$$

Then (4.5) can be rewritten as:

$$\boldsymbol{\varepsilon}_{sample}^{MV} = \mathbf{U}^{MV} \boldsymbol{\varepsilon}^{MV} \quad (4.12)$$

If the stiffness tensor, which is normally given in the Voigt notation, is now rescaled according to:

$$\mathbf{C}_{ij}^{MV} = \begin{cases} \mathbf{C}_{ij} & \forall i, j \in \{1, 2, 3\} \\ \sqrt{2}\mathbf{C}_{ij} & \forall i \in \{1, 2, 3\}, j \in \{4, 5, 6\} \\ \sqrt{2}\mathbf{C}_{ij} & \forall i \in \{1, 2, 3\}, j \in \{4, 5, 6\} \\ 2\mathbf{C}_{ij} & \forall i, j \in \{4, 5, 6\} \end{cases} \quad (4.13)$$

the conversion goes as:

$$\boldsymbol{\sigma}^{MV} = \mathbf{C}^{MV} \boldsymbol{\varepsilon}^{MV} \quad (4.14)$$

and:

$$\boldsymbol{\sigma}_{sample}^{MV} = \mathbf{U}^{MV} \boldsymbol{\sigma}^{MV} = \mathbf{U}^{MV} \mathbf{C}^{MV} \boldsymbol{\varepsilon}^{MV} = \mathbf{U}^{MV} \mathbf{C}^{MV} (\mathbf{U}^{MV})^T \boldsymbol{\varepsilon}_{sample}^{MV} \quad (4.15)$$

The reason for using the Mandel-Voigt notation rather than the traditional Voigt notation is that the former preserves the inner product of the tensors while the latter does not. This is of particular importance because in the elastic regime the work is defined as the inner product of $\boldsymbol{\sigma}$ and $\boldsymbol{\varepsilon}$. Therefore a representation which preserves the inner product is also termed work-conjugate.

Symmetry equivalents

If the system under study is of monoclinic or higher symmetry in the undeformed state, then several different symmetry equivalent versions of U can be used to describe the orientation of the grain relative to the sample frame. However, when the system is deformed we observe weak symmetry breaking in the lattice permutations. Consequently, in order to extract the evolution of the strain tensor for an individual grain one must choose a well-defined reference frame for the orientation such that the lattice parameters are not permuted from one strain level to the next. This is also important when comparing strains in neighbouring grains. The way to do this is to choose the symmetry equivalent version of U which lies in the fundamental zone (5.1) for each grain in the undeformed state and then use this same U at later strain states.

5. Crystal symmetry

In texture literature one introduces both crystal and sample symmetry. For multigrain work – as for single grain work - there is no sample symmetry (or stated differently: the sample symmetry is triclinic).

The crystal symmetry operations associated with a given space group can be found in International Tables of Crystallography, Volume A. The relevant generators have been extracted by calls to the library xfab.splib.

The crystal symmetry implies that orientation space is divided into a set of N equivalent sub-spaces, with N being the number of rotational symmetries (e.g. 24 for cubic systems). We identify one of these as the *fundamental zone*: the one with superior geometric properties such as orientations space being nearly flat. Typically output from programs should refer to the symmetry equivalent in the fundamental zone.

Fundamental zone:

The relevant numbers for the Euler representation is given in Table 4.1. Notably, the 3-fold axis in cubic materials does not lead to sub-volumes that have a rectangular shape in Euler space. The Phi-interval listed in the table is the one that generate the smallest rectangular box comprising an irreducible volume.

Bravais class	phi1	Phi	Phi2
Cubic	[0 360]	[54 90]	[0 90]
Tetragonal	[0 360]	[0 90]	[0 90]
Orthorhombic	[0 360]	[0 90]	[0 180]
Hexagonal	[0 360]	[0 90]	[0 60]
Trigonal	[0 360]	[0 90]	[0 120]
Monoclinic	[0 360]	[0 90]	[0 360]
Triclinic	[0 360]	[0 180]	[0 360]

Table 3.1

Irreducible part of Euler-space for the seven Bravais classes. From [6].

Rodrigues space: The fundamental zone is centred around (0,0,0).

Quaternions: The fundamental zone is centred around (1,0,0,0).

Determining the orientation in the fundamental zone:

Given an arbitrary orientation we can determine all the symmetry equivalents. For each of these we derive the matrix \mathbf{U}_k . We then calculate the misorientation with respect to the Identity matrix, that is the misorientation with respect to the center of the fundamental zone:

$$\beta_k = \cos^{-1}\left(\frac{\text{Tr}(\mathbf{U}_k) - 1}{2}\right). \quad (5.1)$$

Here $\text{Tr}(\mathbf{U}_k)$ is the trace of \mathbf{U}_k , that is the sum of the three diagonal elements:

$$\text{Tr}(\mathbf{U}) = U_{11} + U_{22} + U_{33}.$$

The symmetry equivalent with the minimum β is the one representing the fundamental zone.

References

1. H.F. Poulsen. *Three dimensional x-ray diffraction microscopy*. (Springer, Berlin, 2004)
2. Tomo-book
3. W.R. Busing, H.A. Levy. *Acta Cryst.* (1967). 22, 457
4. W.A. Paciorek, M. Meyer, G. Chapuis. *Acta Cryst.* (1999) **A55**, 543-557.
5. U.F. Kocks, C.N. Tome, and H.R. Wenk. *Texture and Anisotropy* (Cambridge University Press, Cambridge, 1998).
6. V. Randl, O. Engler. *Introduction to texture analysis, Macrotecture, Microtexture and Orientation Mapping*. (Gordon and Breach, London, 2000)
7. H.J. Bunge. *Matematische Methoden der Texturanalyse* (Akademie Verlag, Berlin, 1969).
8. P. Neumann. *Text. Microstruct.* (1991) **14-18**, 53-58.
9. A. Morawiec and D.P. Field. *Phil. Mag. A*, (1996) **73**, 1113-1130.
10. A. Kumar and P.R. Dawson. *Acta Mater.* (2000) **48**, 2719-2736.
11. F.C. Frank. *Metall. Trans. A* (1988) **19**, 403.
12. Altmann, S.L.: *Rotations, Quaternions, and Double Groups*. (Clarendon Press, Oxford, 1986).
13. Conway, J.H., Smith, D.A.: *On Quaternions and Octonions: Their Geometry, Arithmetic, and Symmetry*. (A. K. Peters, Natick, MA, 2003).
14. Kuipers, J.B. *Quaternions and Rotation Sequences* (Princeton University Press, Princeton, 1999).
15. Morawiec, A. *Orientations and Rotations. Computations in Crystallographic Textures*. (Springer, Berlin, 2004).
16. L. Meister, H. Schaeben. *Math. Meth. Appl. Sci.* (2005) **28**, 101-126
17. A. Alpers, L. Rodek, H.F. Poulsen, E. Knudsen, G.T. Herman. In: *Advances in Discrete Tomography, Ed. G.T. Herman, A. Kuba* (Birkhäuser, Boston, 2007), pp. 271-302
18. A. Kulshreshtha, G.T. Herman, E. Knudsen, L. Rodek, A. Alpers, H.F. Poulsen. Submitted (2007)
19. J.L. Schlenker, G.V. Gibbs, M.B. Boisen Jr. *Acta Cryst.* (1978) **A34**, 52-54.
20. W.F. Hosford *The Mechanics of Crystals and Textured Polycrystals*. (Oxford University Press, NY, 1993)

Appendices

1. C Code for deriving Euler angles from U
2. C Code for deriving U from Euler angles
3. Python Code for deriving a Rodrigues vector from U
4. Python Code for deriving U from a Rodrigues vector
5. C Code for deriving a unit quaternion from U
6. C Code for deriving U from a unit quaternion
7. C Code for entire forward projections (using detector tilt specification II)

1. Code for deriving Euler angles from U

```
double sin_cos_to_angle (double s, double c)
// Calculates the angle given its sine (or the sign of its sine) and its cosine.
{
    return (s >= 0.0) ? acos(c) : 2.0 * M_PI - acos(c);
}

void mat_to_Euler_zxz (const double **m, double *psi, double *phi, double *theta)
{
    double sph;

    *phi = acos(m[2][2]);
    sph = sin(*phi);
    if (fabs(sph) < EPS)
    {
        *psi = 0.0;
        *theta = (fabs(m[2][2] - 1.0) < EPS) ? sin_cos_to_angle(m[1][0], m[0][0]) : sin_cos_to_angle(-m[1][0],
m[0][0]);
    } else
    {
        *psi = sin_cos_to_angle(m[0][2] / sph, -m[1][2] / sph);
        *theta = sin_cos_to_angle(m[2][0] / sph, m[2][1] / sph);
    }
}
```

2. Code for deriving U from Euler angles

```
void Euler_zxz_to_mat (double psi, double phi, double theta, double **m_out)
{
    double    cps, cph, cth, sps, sph, sth;

    cps = cos(psi) ; cph = cos(phi); cth = cos(theta);
    sps = sin(psi); sph = sin(phi); sth = sin(theta);

    m_out[0][0] = cth * cps - sth * cph * sps;
    m_out[0][1] = -cth * cph * sps - sth * cps;
    m_out[0][2] = sph * sps;
    m_out[1][0] = cth * sps + sth * cph * cps;
    m_out[1][1] = cth * cph * cps - sth * sps;
    m_out[1][2] = -sph * cps;
    m_out[2][0] = sth * sph;
    m_out[2][1] = cth * sph;
```

```

    m_out[2][2] = cph;
}

```

3. Code for deriving a Rodrigues vector from U (Python)

```

def u_to_rod(U):
    """
    Get Rodrigues vector from U matrix (Busing Levy)
    INPUT: U 3x3 matrix
    OUTPUT: Rodrigues vector

    Function taken from GrainsSpotter by Soeren Schmidt
    """

    ttt = 1+U[0, 0]+U[1, 1]+U[2, 2]
    if abs(ttt) < 1e-16:
        raise ValueError, 'Wrong trace of U'
    a = 1/ttt
    r1 = (U[1, 2]-U[2, 1])*a
    r2 = (U[2, 0]-U[0, 2])*a
    r3 = (U[0, 1]-U[1, 0])*a
    return n.array([r1, r2, r3])

```

4. Code for deriving U from a Rodrigues vector (Python)

```

def rod_to_u(r):
    """
    rod_to_u calculates the U orientation matrix given an orientation
    represented in Rodrigues space. r = [r1, r2, r3]
    """
    g = n.zeros((3, 3))
    r2 = n.dot(r , r)

    for i in range(3):
        for j in range(3):
            if i == j:
                fac = 1
            else:
                fac = 0
            term = 0
            for k in range(3):

```

```

        if [i, j, k] == [0, 1, 2] or \
            [i, j, k] == [1, 2, 0] or \
            [i, j, k] == [2, 0, 1]:
            sign = 1
        elif [i, j, k] == [2, 1, 0] or \
            [i, j, k] == [0, 2, 1] or \
            [i, j, k] == [1, 0, 2]:
            sign = -1
        else:
            sign = 0
        term = term + 2*sign*r[k]
        g[i, j] = 1/(1+r2) * ((1-r2)*fac + 2*r[i]*r[j] - term)
    return n.transpose(g)

```

5. Code for deriving a unit quaternion from U

```

void mat_to_quat(const double **m, quat_s *q_out)
// Calculates the unit quaternion "q_out" corresponding to U matrix "m".
{
    double a, e;

    a = q_out->a = sqrt(max_d(m[0][0] + m[1][1] + m[2][2] + 1.0, 0.0)) / 2.0;
    if (a < EPS) {
        if (fabs(m[0][0] - (-1.0)) < EPS) {
            q_out->b = 0.0;
            q_out->c = sqrt(max_d((m[1][1] + 1.0) / 2.0, 0.0));
            q_out->d = sqrt(max_d((m[2][2] + 1.0) / 2.0, 0.0)) * sgn_b_d(m[1][2]);
        } else if (fabs(m[1][1] - (-1.0)) < EPS) {
            q_out->c = 0.0;
            q_out->b = sqrt(max_d((m[0][0] + 1.0) / 2.0, 0.0));
            q_out->d = sqrt(max_d((m[2][2] + 1.0) / 2.0, 0.0)) * sgn_b_d(m[0][2]);
        } else if (fabs(m[2][2] - (-1.0)) < EPS) {
            q_out->d = 0.0;
            q_out->b = sqrt(max_d((m[0][0] + 1.0) / 2.0, 0.0));
            q_out->c = sqrt(max_d((m[1][1] + 1.0) / 2.0, 0.0)) * sgn_b_d(m[0][1]);
        } else {
            e = M_SQRT2 * m[0][1] * sqrt(max_d(m[0][2] * m[1][2] / m[0][1], 0.0));
            q_out->b = e / (2.0 * m[1][2]);
            q_out->c = e / (2.0 * m[0][2]);
            q_out->d = e / (2.0 * m[0][1]);
        }
    } else {
        a *= 4.0;
        q_out->b = (m[2][1] - m[1][2]) / a;
        q_out->c = (m[0][2] - m[2][0]) / a;
        q_out->d = (m[1][0] - m[0][1]) / a;
    }
}

```

6. Code for deriving U from a unit quaternion

```

void quat_to_mat( const quat_s *q, double **m_out)

```

```

{
  m_out[0][0] = 2.0 * (sqr_d(q->a) + sqr_d(q->b)) - 1.0;
  m_out[1][0] = 2.0 * (q->b * q->c - q->a * q->d);
  m_out[2][0] = 2.0 * (q->b * q->d + q->a * q->c);

  m_out[0][1] = 2.0 * (q->b * q->c + q->a * q->d);
  m_out[1][1] = 2.0 * (sqr_d(q->a) + sqr_d(q->c)) - 1.0;
  m_out[2][1] = 2.0 * (q->c * q->d - q->a * q->b);

  m_out[0][2] = 2.0 * (q->b * q->d - q->a * q->c);
  m_out[1][2] = 2.0 * (q->c * q->d + q->a * q->b);
  m_out[2][2] = 2.0 * (sqr_d(q->a) + sqr_d(q->d)) - 1.0;
}

```

7. Code for entire forward projections (using detector tilt specification II)

```

void calc_diff_r_spot(const double *U, double x, double y, int *num_of_points, det_point_s *points)
{
  double omega,theta,eta,omegas[2];
  double B[9],gd[9],gtmp[9];
  double tmp[3], h[3],gt[3],v[3],dg[3];
  double xl,yl,hlen;
  int *hkl;
  int i, j;
  int pixel_y, pixel_z, pixel_omega;
  double point_omega;
  double a,b,tau,c,t;
  double ydet,zdet;

  lattice_params_to_B(B,diffr_space_group,diffr_lattice_a,diffr_lattice_b,diffr_lattice_c);

  *num_of_points = 0;
  hkl = (int *)diffr_hkl_list;

  for (i = 0; i < DIFFR_HKL_NUM; hkl += 3, i++)
  {
    // matrix multiplication U*B*hkl
    // matrices are stored like this:
    // m(0) m(1) m(2)
    // M = m(3) m(4) m(5)
    // m(6) m(7) m(8)
    tmp[0]=(B[0]* (*hkl) + B[1]* *(hkl+1) + B[2]* *(hkl+2))/(2*M_PI);
    tmp[1]=(B[3]* (*hkl) + B[4]* *(hkl+1) + B[5]* *(hkl+2))/(2*M_PI);
    tmp[2]=(B[6]* (*hkl) + B[7]* *(hkl+1) + B[8]* *(hkl+2))/(2*M_PI);
    h[0]=U[0]*tmp[0] + U[1]* tmp[1] + U[2]* tmp[2];
    h[1]=U[3]*tmp[0] + U[4]* tmp[1] + U[5]* tmp[2];
    h[2]=U[6]*tmp[0] + U[7]* tmp[1] + U[8]* tmp[2];

    hlen = sqrt(tmp[0]*tmp[0]+tmp[1]*tmp[1]+tmp[2]*tmp[2]);
    theta=asin(hlen*DIFFR_HC_1/(2.0*diffr_energy)); // theta fulfilling Bragg's law

    c=-hlen*hlen*DIFFR_HC_1/(2.0*diffr_energy); // compute omegas, note: returned omegas are in (-pi,pi)
    b=h[0]*h[0]+h[1]*h[1];
    if (b>0)
    {
      a=sqrt(1/b);
    }
  }
}

```

```

tau=(h[1] >= 0.0) ? acos(h[0]*a) : -acos(h[0]*a);
omegas[0]=acos(c*a)-tau;
if (omegas[0]<-M_PI) omegas[0]+=2.0*M_PI;
if (omegas[0]>M_PI) omegas[0]-=2.0*M_PI; // Now it's in the interval [-pi,pi]
omegas[1]=-acos(c*a)-tau;
if (omegas[1]<-M_PI) omegas[1]+=2.0*M_PI;
if (omegas[1]>M_PI) omegas[1]-=2.0*M_PI; // Now it's in the interval [-pi,pi]

for (j=0;j<2;j++) //use both omega angles
{
omega=omegas[j];

//gt is the scattering vector, gt=OMEGAMATRIX*U*h
gt[0] = cos(omega)*h[0] - sin(omega)*h[1];
gt[1] = sin(omega)*h[0] + cos(omega)*h[1];
gt[2] = h[2];

// -- compute eta -- (in (-pi,pi) range)
if (gt[2]>0) eta = -atan(gt[1]/gt[2]);
else if ( (gt[2]<0) && (gt[1]<=0)) eta = M_PI-atan(gt[1]/gt[2]);
else if ( (gt[2]<0) && (gt[1]>0)) eta = -atan(gt[1]/gt[2]) - M_PI;
else if ( (gt[2]==0) && (gt[1]<0)) eta = M_PI/2.0;
else if ( (gt[2]==0) && (gt[1]>0)) eta = -M_PI/2.0;

if (eta<0) eta+=2.0*M_PI;

xl=x*cos(omega) - y*sin(omega);
yl=x*sin(omega) + y*cos(omega);

//construct unit vector v - along which diffracted photons travel
v[0]=cos(2*theta);
v[1]=-sin(eta)*sin(2*theta);
v[2]=cos(eta)*sin(2*theta);

// Rotation, first around x, then y, then z to get from diffraction vector the point on the detector
double gx[9]={ 1.0,0.0,0.0,0.0,cos(det_tilt_x),-sin(det_tilt_x), 0.0,sin(det_tilt_x),cos(det_tilt_x)};
double gy[9]={ cos(det_tilt_y),0.0,sin(det_tilt_y),0.0,1.0,0.0,sin(det_tilt_y),0.0,cos(det_tilt_y)};
double gz[9]={ cos(det_tilt_z),-sin(det_tilt_z),0.0, sin(det_tilt_z),cos(det_tilt_z),0.0, 0.0,0.0,1.0};

gtmp[0]=gy[0]*gz[0]+gy[1]*gz[3]+gy[2]*gz[6];
gtmp[3]=gy[3]*gz[0]+gy[4]*gz[3]+gy[5]*gz[6];
gtmp[6]=gy[6]*gz[0]+gy[7]*gz[3]+gy[8]*gz[6];
gtmp[1]=gy[0]*gz[1]+gy[1]*gz[4]+gy[2]*gz[7];
gtmp[4]=gy[3]*gz[1]+gy[4]*gz[4]+gy[5]*gz[7];
gtmp[7]=gy[6]*gz[1]+gy[7]*gz[4]+gy[8]*gz[7];
gtmp[2]=gy[0]*gz[2]+gy[1]*gz[5]+gy[2]*gz[8];
gtmp[5]=gy[3]*gz[2]+gy[4]*gz[5]+gy[5]*gz[8];
gtmp[8]=gy[6]*gz[2]+gy[7]*gz[5]+gy[8]*gz[8];
gd[0]=gx[0]*gtmp[0]+gx[1]*gtmp[3]+gx[2]*gtmp[6];
gd[3]=gx[3]*gtmp[0]+gx[4]*gtmp[3]+gx[5]*gtmp[6];
gd[6]=gx[6]*gtmp[0]+gx[7]*gtmp[3]+gx[8]*gtmp[6];
gd[1]=gx[0]*gtmp[1]+gx[1]*gtmp[4]+gx[2]*gtmp[7];
gd[4]=gx[3]*gtmp[1]+gx[4]*gtmp[4]+gx[5]*gtmp[7];
gd[7]=gx[6]*gtmp[1]+gx[7]*gtmp[4]+gx[8]*gtmp[7];
gd[2]=gx[0]*gtmp[2]+gx[1]*gtmp[5]+gx[2]*gtmp[8];
gd[5]=gx[3]*gtmp[2]+gx[4]*gtmp[5]+gx[5]*gtmp[8];
gd[8]=gx[6]*gtmp[2]+gx[7]*gtmp[5]+gx[8]*gtmp[8]; // gd=gx*gy*gz = R_x * R_y * R_z = R

```

```

        t=(gd[0]*(diff_1_s2d-xl)+gd[3]*(diff_d_0_y-yl)+gd[6]*(diff_d_0_z-0.0))/
(gd[0]*v[0]+gd[3]*v[1]+gd[6]*v[2]);
        dg[0]=xl + t*v[0] - diff_1_s2d;
        dg[1]=yl + t*v[1] - diff_d_0_y;
        dg[2]=0.0 + t*v[2] - diff_d_0_z;

        ydet= gd[1]*dg[0] + gd[4]*dg[1] + gd[7]*dg[2]+diff_d_0_y;
        zdet= gd[2]*dg[0] + gd[5]*dg[1] + gd[8]*dg[2]+diff_d_0_z;

        pixel_y = r_int(diff_d_width/2.0-1-(diff_d_0_y-ydet)/diff_px_size_y);
        pixel_z = r_int(diff_d_height/2.0-1-(diff_d_0_z-zdet)/diff_px_size_z);

        point_omega = 180.0 * omega / M_PI;
        if (
            (((point_omega >= diff_start_omega1) && (point_omega <= diff_start_omega1+(diff_proj_num1-
2)*diff_degree_steps1))
            || ((point_omega >= diff_start_omega2) && (point_omega <= diff_start_omega2+(diff_proj_num2-
2)*diff_degree_steps2)))
            &&
            (pixel_y >= 0) && (pixel_y <= diff_d_width - 1) &&
            (pixel_z >= 0) && (pixel_z <= diff_d_height - 1) &&
            ((pixel_z>deadzone_max_z) || (pixel_z<deadzone_min_z)) ){

            //pixel_y=(diff_d_width-1)-pixel_y; //Detector flipping (if required)
            //pixel_z=(diff_d_height-1)-pixel_z;

            points[*num_of_points].x = pixel_y;
            points[*num_of_points].y = pixel_z;
            if ((point_omega >= diff_start_omega1) && (point_omega <= diff_start_omega1+(diff_proj_num1-
2)*diff_degree_steps1))
                pixel_omega=roundrange(point_omega, diff_start_omega1, diff_degree_steps1);
            else pixel_omega=diff_proj_num1+roundrange(point_omega, diff_start_omega2, diff_degree_steps2);
            points[*num_of_points].omega = pixel_omega;

            points[*num_of_points].intensity = calc_intensity(theta,eta,i);
            (*num_of_points)++;
        }
    }
}
}
}
}

```