Experimental and Computational Investigation into Light Scattering by Atmospheric Ice Crystals

by

Christopher Thomas Collier

Submitted to the University of Hertfordshire in partial fulfilment of the requirements of the degree of PhD

October 2014

Supervisors: Dr. E. Hesse and Prof. Z. J. Ulanowski

Contents

0. Abstract

An investigation was carried out into light scattering by Gaussian rough ice crystals. Gaussian rough crystal geometries were generated using roughness parameters derived from mineral dust grains, which have been reported to be suitable proxies for rough ice crystals. Light scattering data for these geometries was computed using the discrete dipole approximation (DDA) method.

Phase functions, 2D scattering patterns, degree of linear polarisation patterns and asymmetry parameters were computed for smooth, moderately rough and highly rough crystals with a variety of orientations and size parameters.

A sodium fluorosilicate ice analogue crystal with three partially roughened prism facets was created using focused ion beam (FIB) milling and 2D scattering patterns were collected from it using the small ice detector (SID) 3 cloud probe.

It was found that roughness reduces features in the phase function compared to scattering by smooth hexagonal prisms, particularly when the roughness features were horizontally much larger than the wavelength. However, the most effective roughness model also takes account of horizontal features whose size is closer to that of the wavelength. Horizontal features smaller than the wavelength have very little effect.

1. Introduction

A hospitable climate is vital to our long-term survival, and so the study of it is very important. The conclusion that the climate is being changed rapidly by human activity [1] makes this study even more urgent, because rapid climate change reduces biodiversity [2], increases the number of people at risk of hunger [3] and makes parts of the world less habitable, by causing more extremes in temperature and rainfall in different parts of the world [1], and by causing sea levels to rise [1]. The effects of this change are already being felt [1], and will continue to increase in severity.

Several methods have been used to study climate change. Data from the past, including recorded temperature measurements [4] and proxies [5,6], have been analysed to produce plots of climate behaviour over the past which can be compared to ice core samples [7] of the atmospheric composition over the same time period. Future climate behaviour can be predicted by the use of climate models [1] which work well for predicting the global climate decades into the future; however, there are necessarily assumptions and simplifications involved that limit the accuracy of these methods, making longer term predictions unreliable, making it more difficult to plan how to mitigate and adapt to climate change, and preventing the feasibility of predictions of future local weather [8].

One of the largest sources of error within these models is the interaction of clouds with radiation [9]. It is known that they have a large effect [10,11], but their interaction with the climate is complex; their overall effect depends on the balance between reflected, absorbed and transmitted shortwave (from the sun) and longwave (from the surface and lower clouds) radiation [18].

Ice clouds are particularly difficult to model accurately, as the particles within them are non-spherical and their scattering behaviour is largely determined by both the interactions of these individual particles with incoming radiation and the average size, shape and the number density of these crystals within the cloud [10,13]; this makes their light scattering behaviour much more difficult to quantify [1,14].

Although cirrus clouds allow most incident sunlight to pass through them, their extent

makes them a major factor; cirrus coverage is typically 30% [21]; over the tropics it is typically 70% [20]. An additional source of complexity is the fact that ice crystals found in nature have recently been found to have rough surfaces; these crystals make up a sizable proportion of all crystals within cirrus clouds [12,15]. Ice crystals that are pristine (i.e. have not undergone melting or aggregation) have a regular hexagonal prism shape with smooth facets; due to the 60° prism angle this should produce a halo at 22° (assuming an adequately large size parameter) - however, this is quite rarely seen from real cirrus clouds [19]. Roughness in ice crystal geometry has a large effect on the radiative properties – more roughness leads to a lower asymmetry parameter, which means more light scattered back towards the light source [22]. Understanding the extent to which roughness causes this effect is important for better characterising ice crystals in clouds.

Previous work on characterising ice crystal roughness has focused on light scattering simulations, since direct imaging in clouds is not accurate enough to characterise it [27]. These have been performed using geometric optics ray tracing simulations, with facets being randomly tilted as a ray hits [16]. This method can be improved upon, as tilted facets are not usable within exact light scattering models, the simulation is not exactly repeatable and it does not account for more complex ray paths, as the tilting only occurs when the ray hits. This issue has been looked at previously [17], however the parameters used in such studies are not derived from physical measurements and do not take account of multiple roughness scales.

Work on ice crystal roughness has also been performed by growing them inside scanning electron microscopes [28] and cloud chambers [29]. Ice crystal analogue-based laboratory experiments have also been performed [22]; these analogues are hexagonal crystals made of sodium fluorosilicate. Their refractive index is very similar to that of ice, but they don’t melt at room temperature and therefore make it possible to study how light scattering by ice crystals behaves in detailed physical experiments.

Therefore, the aim of the work described here is to generate roughened crystal geometries for use with light scattering computations; roughened ice analogue crystals will also be manufactured using the same geometry. The parameters of this roughness will be derived from analysis of suitable proxy materials, since in-situ measurements of cloud ice crystals are unfeasible to the required degree of accuracy. The rough ice analogue will be created by using Focused Ion Beam (FIB) milling to etch the roughness geometry onto the prism facets of ice crystal analogues. Light scattering computations will be performed for crystals with a range of orientations, size parameters and roughness types and magnitudes, and light scattering experiments will be performed on both the smooth and rough ice analogues. These results will be analysed and compared and conclusions drawn about the effect of this roughness on the phase function, degree of linear polarisation and asymmetry parameter.

2. Theory

2.1 Cirrus clouds and ice crystals

Cirrus clouds (Fig. 2-1) are wispy clouds that are made up of ice crystals. They form at heights between 4km and 20km above sea level [31], above most other cloud types. Their thickness varies between 100m and 8km [31], and they cover approximately 30% of the Earth [21], with 70% coverage over the tropics [20].

Fig. 2-1: Cirrus clouds over Hatfield, Hertfordshire.

These clouds form when water vapour undergoes deposition, and begins below -20°C in the presence of aerosols for nucleation to start on, or below -40°C otherwise. This process is initiated when warm air is forced upwards – the lower temperature at higher altitudes causes the moisture to cool down rapidly. The crystals grow as more water and water vapour nucleates on them. A type of cirrus known as a contrail can also form from aircraft exhaust; this results in much smaller crystals than normal cirrus.

The ice crystals they are composed of are formed from water molecules that arrange into hexagonal crystals when they turn into ice. These crystals can become columns, plates, rosettes or conglomerations of all these. The lengths of these can vary from hundredths of a millimetre to millimetres.

Cirrus clouds sometimes cause optical effects that can be seen in the sky – halos, sundogs and arcs are sometimes seen (Fig. 2-2).

Fig. 2: A halo caused by ice crystals in a cirrus cloud over Stockholm, Sweden. The 22° halo can be seen, and the inset shows the path light takes through ice crystals for it to form. Photograph credit: Peter Rosén via Spaceweather.com

2.2 Light scattering theory and conventions

2.2.1 Stokes Parameters

The Stokes Parameters describe the intensity and polarisation state of the electric field of a beam of light. They are as follows:

$$I = E_p E_p^* + E_s E_s^* \qquad (2.1)(a)$$

$$Q = E_p E_p^* - E_s E_s^* \qquad (2.1)(b)$$

$$U = -E_p E_s^* - E_s E_p^* \qquad (2.1)(c)$$

$$V = i(E_s E_p^* - E_p E_s^*) \qquad (2.1)(d)$$

Where E_p is the parallel part of the electric field and E_p^* is its complex conjugate; E_s is the perpendicular part of the electric field and E_s^* is its complex conjugate . I describes the energy flux of the beam, *Q* and *U* describe the linear polarisation of the beam and *V* describes the circular polarisation of the beam. For convenience, they are usually described in a column matrix:

$$S = \begin{bmatrix} I \\ Q \\ U \\ V \end{bmatrix} \qquad (2.2)$$

2.2.2 Scattering Matrix

The scattering matrix describes how the intensity and polarisation of a light beam is changed by scattering caused by a single particle. The notation used for the matrix elements can be seen below:

$$\begin{bmatrix} p_{11} & p_{12} & p_{13} & p_{14} \\ p_{21} & p_{22} & p_{23} & p_{24} \\ p_{31} & p_{32} & p_{33} & p_{34} \\ p_{41} & p_{42} & p_{43} & p_{44} \end{bmatrix} \qquad (2.3)$$

It can be multiplied by the Stokes matrix of incident light to give the Stokes matrix of scattered light.

$$\begin{bmatrix} I_{sca} \\ Q_{sca} \\ U_{sca} \\ V_{sca} \end{bmatrix} = \frac{1}{k^2 r^2} \begin{bmatrix} p_{11} & p_{12} & p_{13} & p_{14} \\ p_{21} & p_{22} & p_{23} & p_{24} \\ p_{31} & p_{32} & p_{33} & p_{34} \\ p_{41} & p_{42} & p_{43} & p_{44} \end{bmatrix} \begin{bmatrix} I_{inc} \\ Q_{inc} \\ U_{inc} \\ V_{inc} \end{bmatrix} \qquad (2.4)$$

Where *r* is the distance of the scattered light from the particle and *k* is the wavenumber, $k = 2\pi/\lambda$.

For enough randomly oriented symmetric scattering objects, symmetries will occur between different particles. This means that the scattering matrix can be simplified so only 6 independent terms remain [35]:

$$\begin{bmatrix} p_{11} & p_{12} & 0 & 0 \\ p_{12} & p_{22} & 0 & 0 \\ 0 & 0 & p_{33} & p_{34} \\ 0 & 0 & -p_{34} & p_{44} \end{bmatrix} \qquad (2.5)$$

2.2.3 Phase function and asymmetry parameter

The phase function is an azimuthally averaged description of the intensity of scattered light (p_{11} from the scattering matrix) as a function of scattering angle θ. The way the phase function's angular orientation for scattering events is defined can be seen in Fig. 2-3. It is normalised using the following:

$$\frac{1}{2}\int_0^\pi d\theta \sin\theta \, p_{11}(\theta) = 1 \qquad (2.6)$$

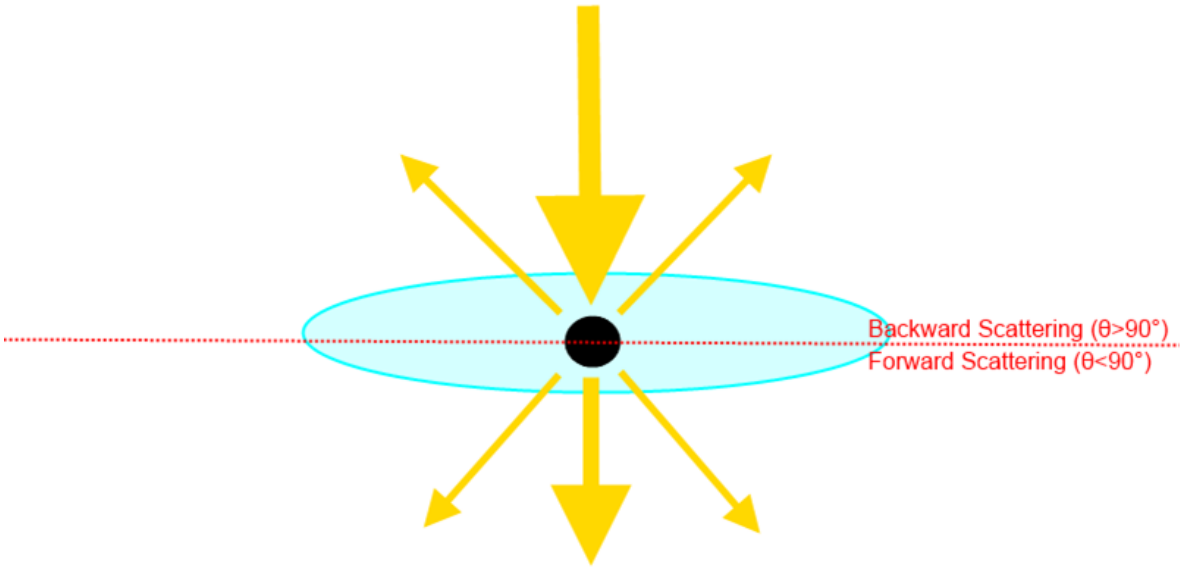

Fig. 2-3: The angular orientation of the phase function for a scattering event. The light source is at the top; backward scattered light is light directed back towards the light source, whereas forward scattered light is light directed away from the light source.

The asymmetry parameter, g, is given by the following:

$$g = \frac{1}{2}\int_0^{\pi} d\theta \sin\theta \cos\theta \, p_{11}(\theta) \qquad (2.7)$$

It describes the extent to which light is scattered forward or backward. If it is positive, more light is scattered forward than backward; if it is negative, more light is scattered backward than forward.

2.2.4 Degree of linear polarization

The degree of linear polarisation is derived from the Stokes parameters and is given by:

$$DLP = \frac{\sqrt{Q^2 + U^2}}{I} \qquad (2.8)$$

When the simplified scattering matrix for multiple randomly oriented scattering objects is multiplied by the Stokes matrix for unpolarised light, the following is found:

$$\begin{bmatrix} p_{11} & p_{12} & 0 & 0 \\ p_{12} & p_{22} & 0 & 0 \\ 0 & 0 & p_{33} & p_{34} \\ 0 & 0 & -p_{34} & p_{44} \end{bmatrix} \begin{bmatrix} I_0 \\ 0 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} p_{11} I_0 \\ p_{12} I_0 \\ 0 \\ 0 \end{bmatrix} \qquad (2.9)$$

Therefore, the degree of linear polarisation for unpolarised light can be defined as:

$$DLP = -\frac{p_{12}}{p_{11}} \qquad (2.10)$$

2.2.5 Size parameter

This describes the size of a scattering object relative to the wavelength of light that passes through it. It is given by the equation below, where *X* is the size parameter, *r* is the longest dimension of the scattering object and λ is the wavelength of the incident light.

$$X = \frac{2\pi r}{\lambda} \qquad (2.11)$$

2.2.6 Rayleigh scattering

Rayleigh scattering is the process by which light beams are deflected by particles which are smaller than the wavelength of the propagating light. As a light wave passes a small particle, it induces the charges in the particle to oscillate at the same frequency as the wave.

Violet and blue light is more efficiently scattered by the particles in the Earth's atmosphere than red and green light. This is what gives rise to the blue colour of the sky.

2.2.7 Mie theory

This method gives the exact result for light being scattered by a perfect sphere. It uses a spherical wave equation with boundary conditions set at the sphere's surface; this enables the separation of the variables and gives a solvable partial differential equation. This method is not suitable for computing light scattering by facetted particles.

2.2.8 T-matrix

T-matrix [32] is a light scattering method that solves the Maxwell equations for light traversing objects. It uses a transformation matrix to relate the spherical wave functions that describe the incident and scattered fields. It is useful for exactly computing light scattering by small particles with rotational symmetry; however, computational time requirements increase dramatically as scatterer complexity and size increase. This means that it is not a suitable method for calculating light scattering by roughened particles.

2.2.9 Discrete Dipole Approximation (DDA)

The discrete dipole approximation [33] is a light scattering theory that exactly solves the Maxwell equations for objects that are divided into discrete dipoles. Usually, the object is split into 10 dipoles for each wavelength distance in each dimension, which has a slight impact on the accuracy of the results as compared to the T-matrix method. A scattering computation is performed by simulating how these dipoles respond to both the incident field and the fields produced by all the other dipoles. The scattered field is then found by considering the contributions of all the dipoles. DDA is able to handle objects of any complexity, but its computational demands increase dramatically with particle size. This makes it suitable for computing light scattering by small rough particles.

2.2.10 Geometric Optics (GO)

Geometric optics is an approximate light scattering theory that uses the concept of ray tracing – that is, treating light as rays, and tracing how these rays pass through a scattering medium using Snell's law and the Fresnel equations, which are described below. The results of this simulation are combined with the diffraction caused by the scattering object's projected cross-section. This method trades accuracy for computational speed; it is a much less accurate method than those which solve the Maxwell equations, but light scattering simulations can be performed much more quickly. It is useful for particles with large size parameters, as these are less sensitive to its

inaccuracy and are impractical for computation with methods that solve the Maxwell equations.

Snell's law describes how the direction of the propagation of a ray in geometric optics is altered by passing from a medium with one refractive index into a medium with another refractive index. Here, the refractive index of the first medium is given by n_1, the refractive index of the second medium is given by n_2, the angle of incidence is given by θ_1 and the angle of refraction is given by θ_2.

$$n_1 \sin\theta_1 = n_2 \sin\theta_2 \qquad (2.12)$$

Two different outcomes are possible; either the ray will partially be transmitted into the second medium at the angle θ_2 from the normal and partially be reflected back into the first medium by the angle θ_1 from the normal (Fig. 2-4); or, if the incident angle is at or exceeds the critical angle, the ray will be totally reflected at the angle θ_1 from the normal. The critical angle can be found by setting θ_2 equal to 90°.

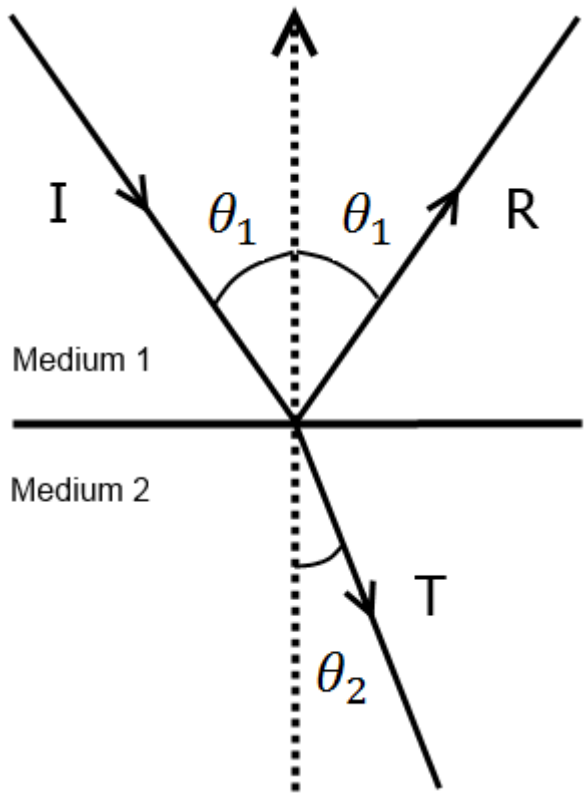

Fig. 2-4: Diagram showing the incident (I), transmitted (T) and reflected (R) ray paths for light passing through a boundary between media with different refractive indices.

The Fresnel equations [36] calculate the ratios of the transmitted and reflected electric fields to that of the incident field and their polarisation states. Intensity is the square of these values.

Where s-polarisation is perpendicular to the plane of incidence (the plane containing the incident ray and the normal vector of the surface), and p-polarisation is parallel to the plane of incidence:

$$r_p = \frac{\tan(\theta_1 - \theta_2)}{\tan(\theta_1 + \theta_2)} \qquad (2.13)(a)$$

$$r_s = -\frac{\sin(\theta_1 - \theta_2)}{\sin(\theta_1 + \theta_2)} \qquad (2.13)(b)$$

$$t_p = \frac{2\sin\theta_2 \cos\theta_1}{\sin(\theta_1 + \theta_2)\cos(\theta_1 + \theta_2)} \qquad (2.13)(c)$$

$$t_s = \frac{2\sin\theta_2 \cos\theta_1}{\sin(\theta_1 + \theta_2)} \qquad (2.13)(d)$$

Where $r_p$is the reflected p-polarisation, r_s is the reflected s-polarisation, t_p is the transmitted p-polarisation and t_s is the transmitted s-polarisation.

2.2.11 Ray Tracing with Diffraction on Facets (RTDF)

This is an adaptation [34] of the geometric optics method that increases its accuracy without massively increasing the computational overhead. Each facet is treated as a set of slits which cause the ray path to deviate from what would be expected using ray tracing alone. The concept of energy flow lines is used to model this deviation; slit size, distance between where the ray has hit and where the edges of the slit are and the wavelength of the incident light are all taken into account.

It is useful for simulating scattering by particles that are too big to be computed in a reasonable amount of time using methods that solve the Maxwell equations and too small to be accurately modelled using geometric optics.

3. Measurement Techniques

3.1 Small Ice Detector (SID) 3

SID3 (Fig. 3-1) is a device which is used to determine the light scattering caused by particles which pass through its 532nm laser beam (Fig. 3-2). The laser beam is emitted from the probe arm at the top right of Fig. 3-1, down towards the head at the bottom right of Fig. 3-1, where a CCD camera is positioned to image the resultant scattering. It is primarily intended as a cloud-based probe, but can also be set up for use in the laboratory.

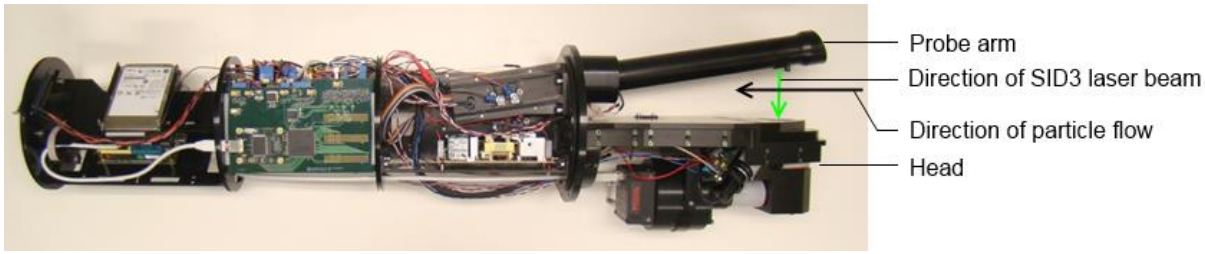

Fig. 3-1: The Small Ice Detector (SID) 3 cloud probe, without the protective cover it would have in cloud-based operation. The laser beam extends down (in the orientation shown) from the arm on the right towards the head, in which the CCD collector sits. In cloud-based operation, particles pass through from the right and travel through the beam, triggering the CCD to take an image.

Cloud-based operation requires a particle to trip two trigger detectors, mounted in the head. If this happens, an image is captured by the CCD camera in the head, which has a resolution of 780x582 pixels and is positioned directly below the laser beam on Fig. 3-1. Lab-based operation is performed by placing a particle in the scattering volume and manually triggering the detector.

Two dimensional forward scattering patterns are produced when the CCD is triggered. The highest scattering angle it can record is 25°, which means the 22° ice crystal halo can be seen. There is a beamstop to block out the main forward scattering diffraction peak, which means the lowest scattering angle it can record is 6 deg.

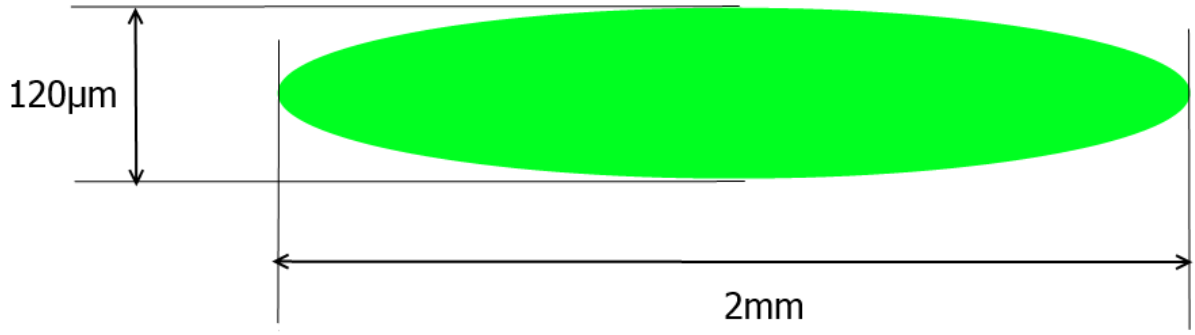

Fig. 3-2: Dimensions of the SID3 laser beam (not to scale); edge is defined as being the Gaussian beam profile. Relative to the diagram, particles flow vertically.

The CCD produces radially distorted images (Fig. 3-3) because it is flat – which means that scattering angle does not vary linearly with distance from the centre of the image. To create images that are linear in scattering angle, a correction routine was created. This involved creating a square of projector transparency with a grid of dots printed on it and placing it within the optics of the detector. A cluster of pollen grains mounted on a fibre was placed in the scattering volume and scattering images were collected of this grain at many orientations. This was averaged to get an image with the whole angular range of the detector illuminated, save for the dots. Scattering images were also captured of the fibre without the pollen cluster, and this was subtracted from the averaged result. The brightness within the scattering range was then inverted and a star detection algorithm was used to find the centroids of the dots. A quartic correction routine was created to and its parameters were found by finding the least square error between the dot locations and where they would be in an undistorted image. Use of this method reduced the mean error of the dot positions by 90%.

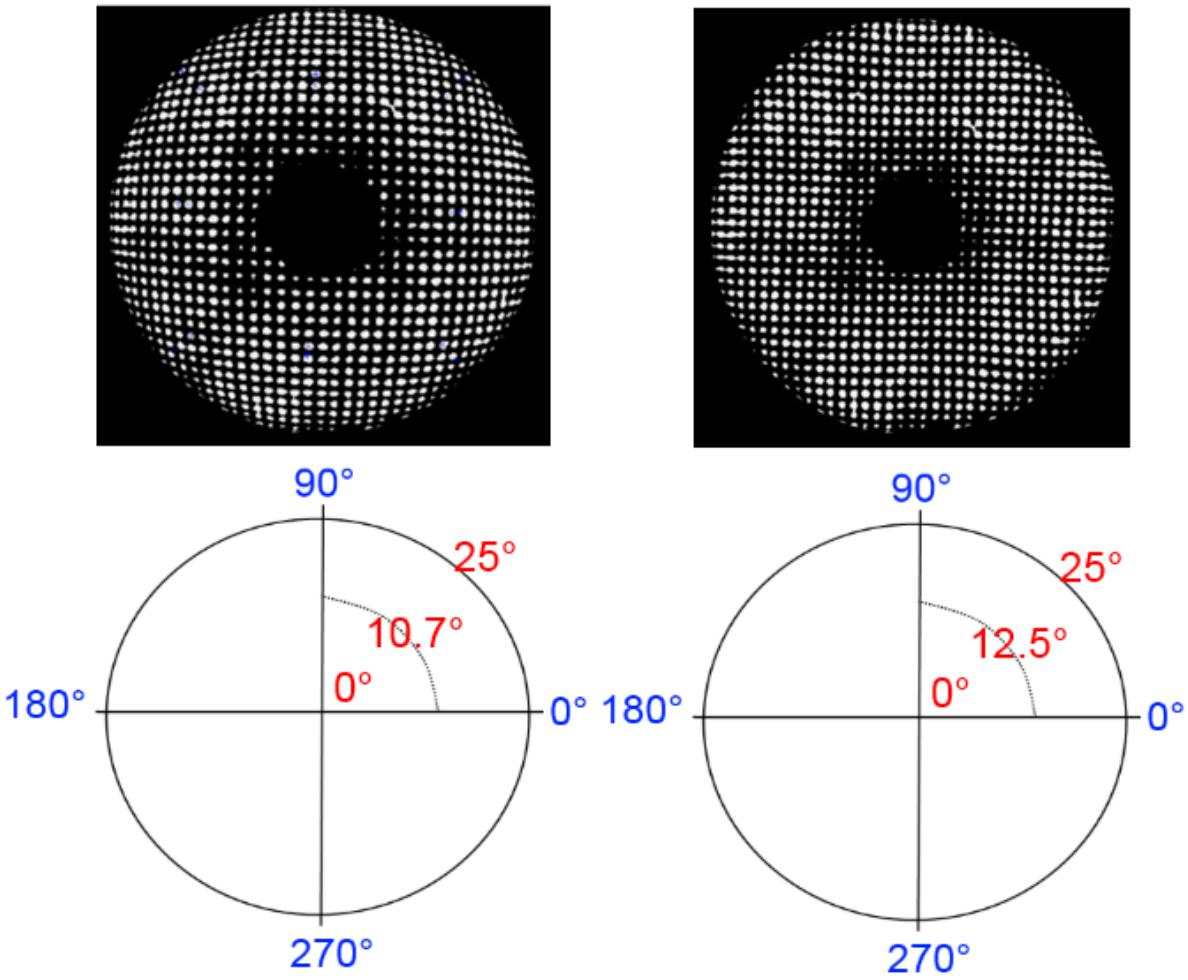

Fig. 3-3: The dots array before (left) and after (right) the quartic correction. Below are the scales – it can be seen that the scattering angle does not vary linearly.

3.2 Ice Analogues

Ice analogue crystals are made of sodium fluorosilicate, and are grown from solutions of sodium carbonate, fluorosilicic acid and triethanol amine [30]. They have quasi-hexagonal structure and a similar refractive index to that of ice, but do not melt at room temperature; this means they are a suitable model for hexagonal ice crystals and can be used for light scattering experiments in the laboratory. Some ice analogues can be seen in Fig. 3-4.

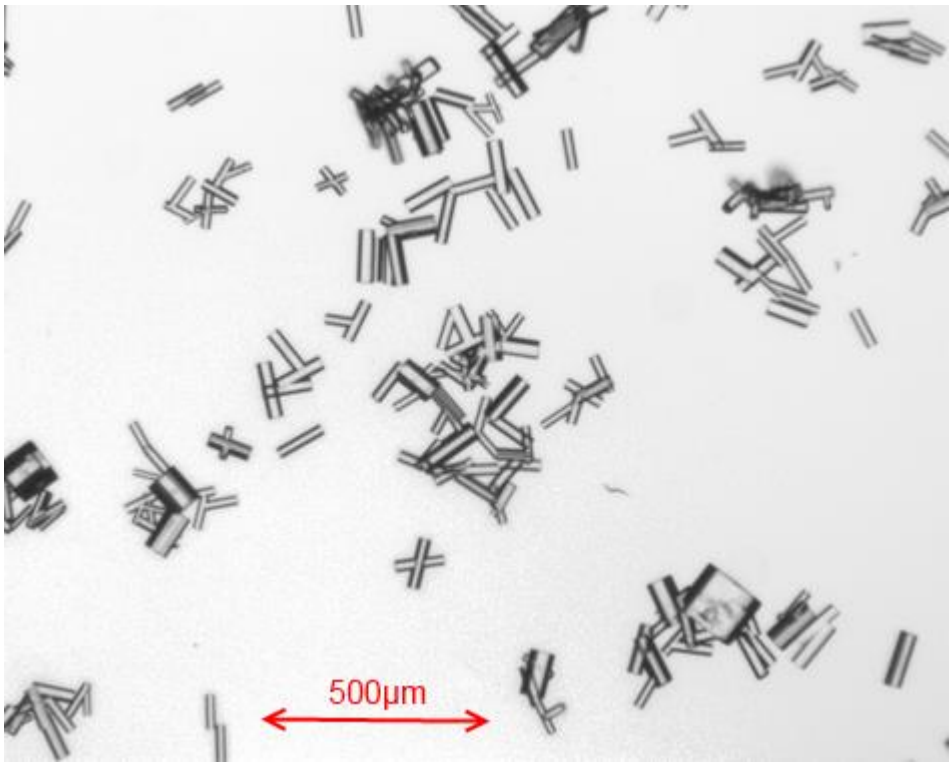

Fig. 3-4: Ice analogue crystals viewed through an optical microscope at 4X magnification.

4. Gaussian Random Crystals

4.0 Why? What are they?

Ice crystal roughness, and its effect on light scattering by cirrus clouds, has not been adequately investigated. Light scattering computations exploring this area have mostly used crystal geometries with facets that randomly tilt as light hits them [16], an artificial, non-physical construct. Within this model the same facet will tilt in a different direction each time it interacts with light. This means each light scattering computation is not exactly repeatable and doesn't allow for more complex light interactions at the crystal surface, such as diffraction.

In the laboratory, ice roughness has been investigated using ice crystal analogues [22]; in general, rougher analogues show a closer agreement to scattering images taken using the SID3 cloud probe than smoother ones. However, the roughness present is introduced as the analogue crystal is growing and cannot be made to obey parameters which describe the dominant frequencies and standard deviation of the required roughness.

A Gaussian random crystal (Fig. 4-1) is constructed from a Gaussian random surface (Fig. 4-2). This is a surface for which the height varies as a function of the lateral x and y dimension; each point's height is calculated as a Fourier series in which each frequency term is multiplied by a Gaussian random amplitude. Gaussian roughness has previously been used to describe roughness for cylinders [23] and spheres [24]. The fundamental parameters in Gaussian roughness are a correlation length, which describes the dominant frequencies, and a standard deviation, which describes the variation in height.

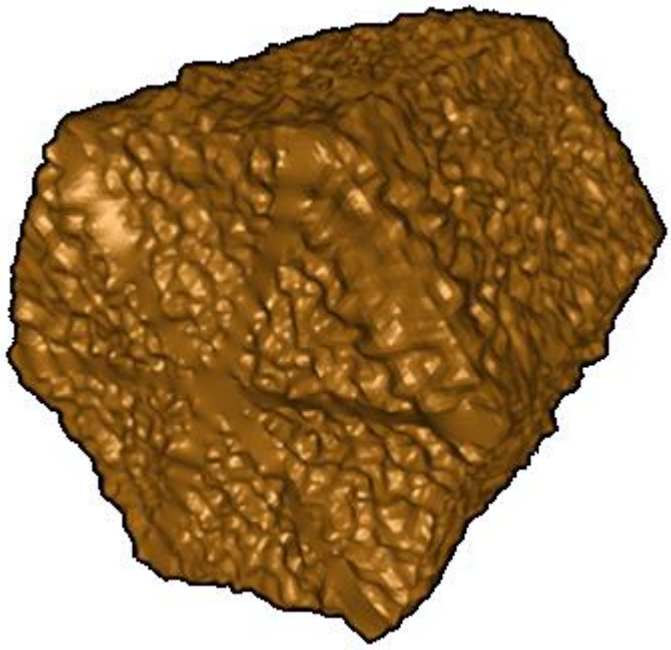

Fig. 4-1: A Gaussian random crystal.

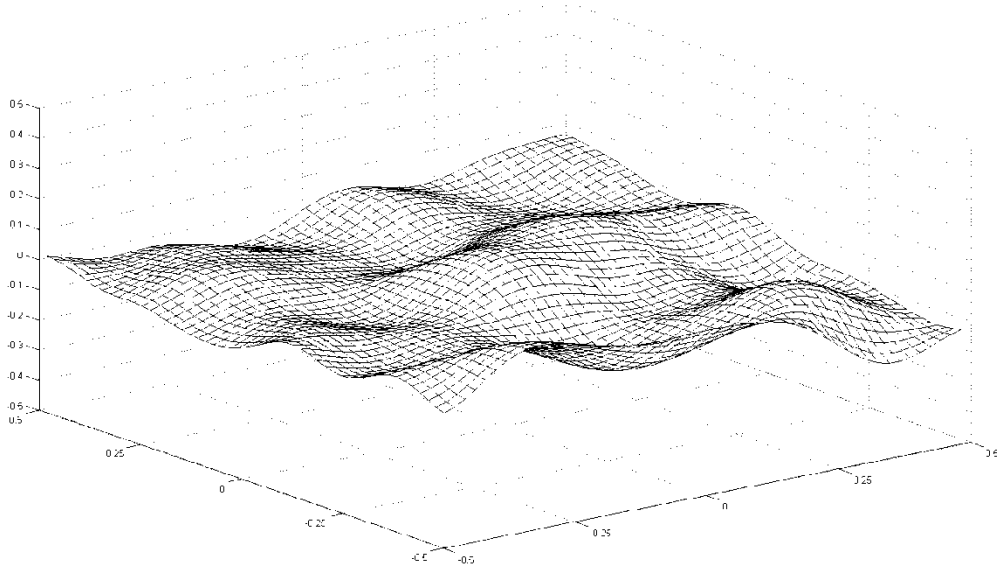

Fig. 4-2: A Gaussian random surface.

4.1 As input for light scattering computations

4.1.1 Gaussian random surface

A method was devised for creating input files describing the geometry of roughened hexagonal prisms in the Macke [16] crystal format for use with computational light scattering models.

The implementation of this involved the use of a Gaussian random surface creation method, adapted from previous work by Muinonen & Saarinen [23], which makes use of a 2D Fourier series method, takes as its parameters the correlation length and standard deviation to create roughness across a surface. This is used to create a rough facet which can then be folded into a hexagonal prism shape and joined together.

Gaussian random surfaces can be created by applying a two dimensional Fourier series across a flat surface to generate roughened height values. Theoretically, the height of one coordinate is given by the following:

$$z(x,y) = \sum_{p=-\infty}^{\infty} \sum_{q=-\infty}^{\infty} z_{pq}\, e^{i(pKx+qKy)} \qquad (4.1)$$

Where $K = \pi/L$, and L is half a period in x & y, which must be chosen to be large compared to the correlation length. z_{pq} are independent Gaussian random complex numbers for each p,q integer pair, generated between -1 and 1. To get z_{pq}, the Marsaglia polar method is used; this involves calculating two uniformly distributed random numbers, G and H, between -1 and 1. As long as $D = G^2 + H^2$ is less than 1, $F = \sqrt{\frac{-2\ln D}{D}}$ is calculated. Two Gaussian random numbers can be found by multiplying each of these uniform random numbers by F.

The computed z_{pq} values are modified by the variance, σ^2, and Kronecker deltas of the p and q values:

$$Var\left(Re\left(z_{pq}\right)\right) = \frac{1}{8}\left(1 + \delta_{p0} + \delta_{q0} + 5\delta_{p0}\delta_{q0}\right)c_{pq}\sigma^2 \qquad (4.2)(a)$$

$$Var\left(Re\left(z_{pq}\right)\right) = \frac{1}{8}\left(1 + \delta_{p0} + \delta_{q0} - 3\delta_{p0}\delta_{q0}\right)c_{pq}\sigma^2 \qquad (4.2)(b)$$

and the same z_{pq} values are used for all *x,y*. c_{pq} are cosine series coefficients, and are defined later.

However, practicality requires that *p* & *q* must both only take a finite number of values, and so the correlation statistics of the surface must be taken into account to calculate suitable limits for these values. This makes use of the correlation function chosen to represent Gaussian random surfaces:

$$C(\zeta, \eta) = e^{-\left[\frac{\zeta^2+\eta^2}{2l^2}\right]} \qquad (4.3)$$

Where ζ is the difference between the *x* positions of two points, η is the difference between the *y* position of the same two points and *l* is the correlation length.

It also requires the surface's two dimensional Fourier expansion:

$$C(\zeta, \eta) = \sum_{p=-\infty}^{\infty} \sum_{q=-\infty}^{\infty} c_{pq} \cos pK\zeta \cos qK\eta \qquad (4.4)$$

Instead of iterating the Fourier expansion through infinite *p* and *q*, they will only be iterated until both equations are equal within a suitable tolerance. This tolerance is set as 1E-6, and both equations are evaluated for ζ=0, η=0. This results in *p* & *q* being iterated through until the following is satisfied:

$$1 - \sum_{\mathrm{p}=-p_{max}}^{p_{max}} \sum_{\mathrm{q}=-q_{max}}^{q_{max}} \mathrm{c_{pq}} < 1 \times 10^{-6} \qquad (4.5)$$

Where q_{max} must be less than or equal to p_{max} and c_{pq} is given by this equation:

$$c_{pq} = \left[\left(2 - \delta_{p0}\right)\sqrt{\frac{\pi}{2}}\frac{l}{L}e^{-(\frac{1}{2}p^2\pi^2\frac{l^2}{L^2})}\right]\left[\left(2 - \delta_{q0}\right)\sqrt{\frac{\pi}{2}}\frac{l}{L}e^{-(\frac{1}{2}q^2\pi^2\frac{l^2}{L^2})}\right] \qquad (4.6)$$

4.1.2 Crystal Creation

The Gaussian random hexagonal prism is created by generating a roughened surface that can be folded to form a crystal. This is made up of six adjacent rectangular parent facets which form the prism facets and twelve equilateral triangular parent facets – two per prism facet – which eventually form the basal facets (Fig. 4-3). Firstly the coordinates required to describe the unroughened prism facets are created. The length and width values are set separately, as are the values describing the number of points along the length and width of each prism facet. After the roughening has been completed quadrilateral Macke format subfacets are generated, which can be split into two triangular subfacets each if required.

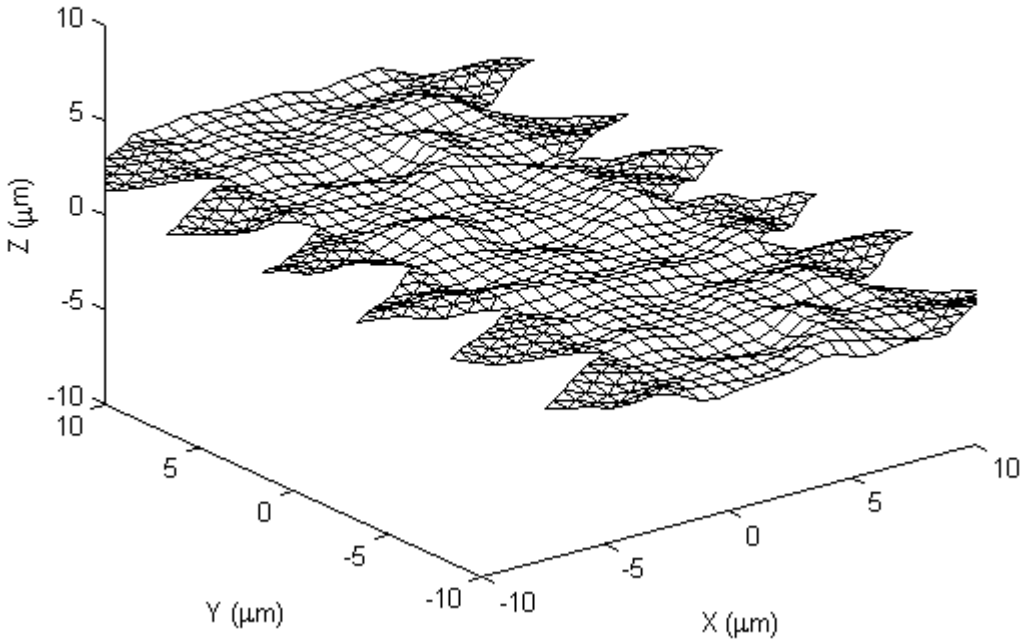

Fig. 4-3: The crystal in its unfolded state. The triangles will be folded down, and then the rectangle will be folded to create the prism facets. The correlation length is 1 μm, and the standard deviation is 0.3 μm.

Secondly, the coordinates describing the triangular parent facets are added, with each triangle being added in turn. Coordinates are added in rows, beginning with the row connecting the triangle to its adjacent prism facet and concluding with the final coordinate forming the parent triangular facet's point. After the roughening has been completed equilateral triangular Macke format subfacets are formed out of these coordinates.

At the stage of the roughening and subfacet creation being completed, if a composite surface (i.e., one with multiple roughness scales) is required, this is the stage at which it is most easily created; as long as both component surfaces have an equal number of coordinates and the same size before being roughened, all that is required is to add together the *z*-values for each coordinate.

Thirdly, the triangular parent facets are rotated downwards by 90° around the prism facet-triangle facet edge y-value (Fig. 4-4). A discontinuity can be seen, as the edge has not been rotated; to correct for this, the edge is rotated around its own unroughened height by 45°.

Coordinate rows either side of the boundary can also be rotated. For example, if it was required to rotate the two prism facet rows and the two basal facet rows nearest to the boundary; the furthest of the two prism facet rows from the boundary would be rotated towards the basal facet (around its own unroughened position) by 15°, and the nearest would be rotated towards the basal facet (again, around its own unroughened position) by 30°. Similarly, the furthest of the two basal facet rows would be rotated towards the prism facet (around its own unroughened position) by 15° and the nearest of the prism facet rows would be rotated towards the prism facet (around its own unroughened position) by 30°.

Fourthly, the individual prism facets are rotated into position. The first step of this involves rotating five adjacent prism facets and their adjacent triangular parent facets around the unroughened edge between the first and second prism facets by 60° (Fig. 4-5). The next iteration involves rotating four adjacent prism facets and their adjacent triangular parent facets by 60° around the unroughened edge between the second and third prism facet, leaving the first and second prism facets and their adjacent triangular parent facets in place. This is continued until the iteration in which just the sixth prism facet and its adjacent triangles are rotated, at which point all the parent facets will be in position.

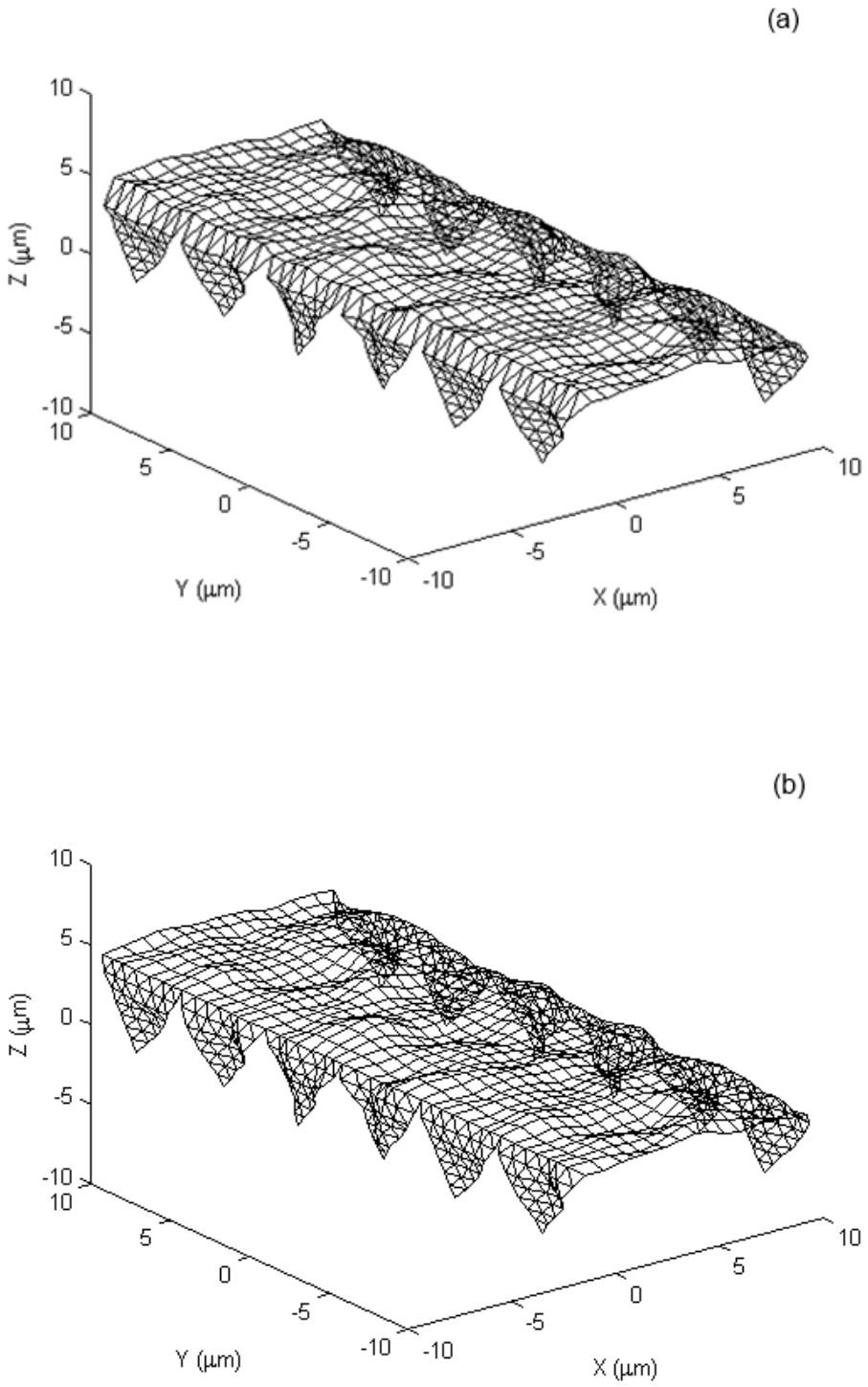

Fig. 4-4: The crystal with its triangles folded down. (a) shows the crystal without the boundary between the prism facets and the triangular parent facets rotated; (b) shows it with the boundary rotated. The correlation length is 1μm, and the standard deviation is 0.1 μm.

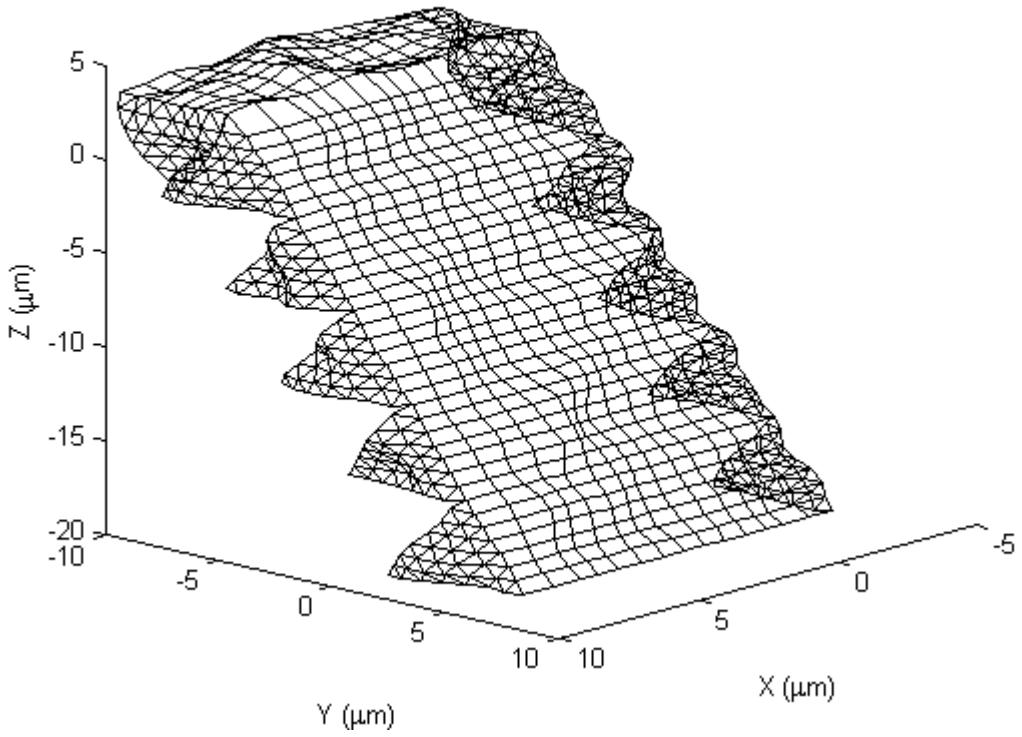

Fig. 4-5: The crystal with one prism facet rotation competed. The triangles on each side will fit together to form the hexagonal surface. Correlation length is 1μm, standard deviation is 0.1μm.

As with the 90° rotation of the triangular parent facets, rotation of the prism facets will have resulted in discontinuities at the edges; to correct for this, the edge is rotated by 30° around its own unroughened position.

Once the rotations are complete, there are gaps left due to the roughness that need to be joined together. The first and last prism facets are connected using a 2D interpolation method that takes the non-edge values of both facets and uses these to generate values which can be applied to the edges of both to join them together.

The triangles at either end of the prism facets also need to be joined together to create the basal facets. This is done by interpolating for each point over all the triangles for that basal facet, excluding the points along the edges of each triangle (Fig. 4-6).

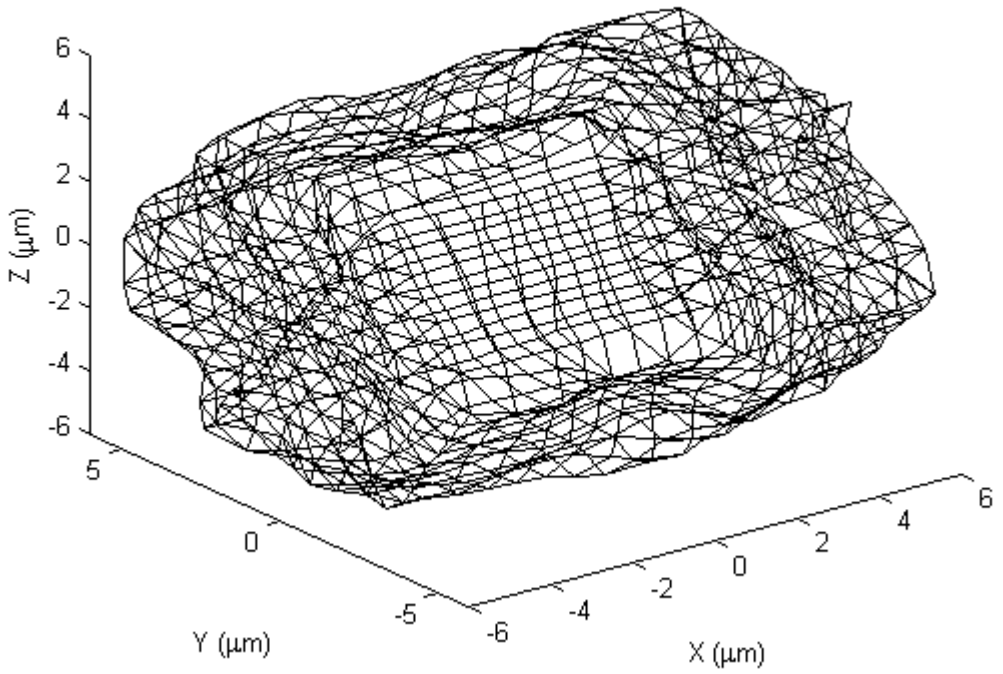

Fig. 4-6: The crystal with all the sections folded into position and the unconnected edges joined together. It has a correlation length of 1μm and a standard deviation of 0.1μm.

4.1.3 Sand Grain Microscopy

To obtain suitable parameters for roughness generation, analysis of the surface of an ice crystal in a cirrus cloud would be ideal. However, in situ cloud imaging methods aren't able to provide the required optical resolution, and so a suitable proxy is needed. Measures of roughness show that mineral dust grain roughness has a similar effect on light scattering as ice crystal roughness [25]; therefore, work was done to derive these parameters from mineral dust grain samples.

Samples (e.g. (Fig. 4-7)) were prepared for detailed microscopy work using the Park Systems XE-100 Atomic Force Microscope (AFM) at Cardiff University. For this, suitably sized dust grains were selected – they needed to be less than approximately 50μm in

size but also as large as possible within that constraint to get as large a scan area as possible (grains measured varied from 40μm to 54μm). 50μm is approximately the size of grain that shows similar scattering pattern-derived roughness as that of naturally occurring cirrus ice crystals; since roughness producing processes can change with particle size (for example, roughening can be introduced by collisions between grains in air; at smaller sizes, van der Waal's forces have a larger effect and the mass involved in each collision is decreased as compared to larger particles), much larger grains may not give suitable properties.

For preparation, firstly a suitably sized rough desert dust grain from a sample collected in Mitribah, Kuwait was selected using optical microscopy. Secondly, a glass coverslip had epoxy glue applied to its surface using a thin strand of carbon fibre; this was necessary to create suitably small spots of glue on the glass so a dust grain could be stuck to the spot without sinking into it.

The grain was then picked up and deposited onto a glue spot smaller than itself on the previously prepared coverslip using a fine tungsten needle and a micromanipulator. Optical microscope images were taken at x4, x10, x20 and x40 magnifications to enable the grain to be found in subsequent work; the underside of the coverslip was then glued to an electron microscope stub.

Images were taken of the topography of the sample grains using a Scanning Electron Microscope (SEM). This was done to simplify the AFM work, which is otherwise forced to be done semi-blind as it is difficult to tell where the imaging probe is on the grain surface using the instrument's camera.

Measurements were then done using the AFM. For all the samples, the maximum area that could be scanned was 20μm x 20μm.

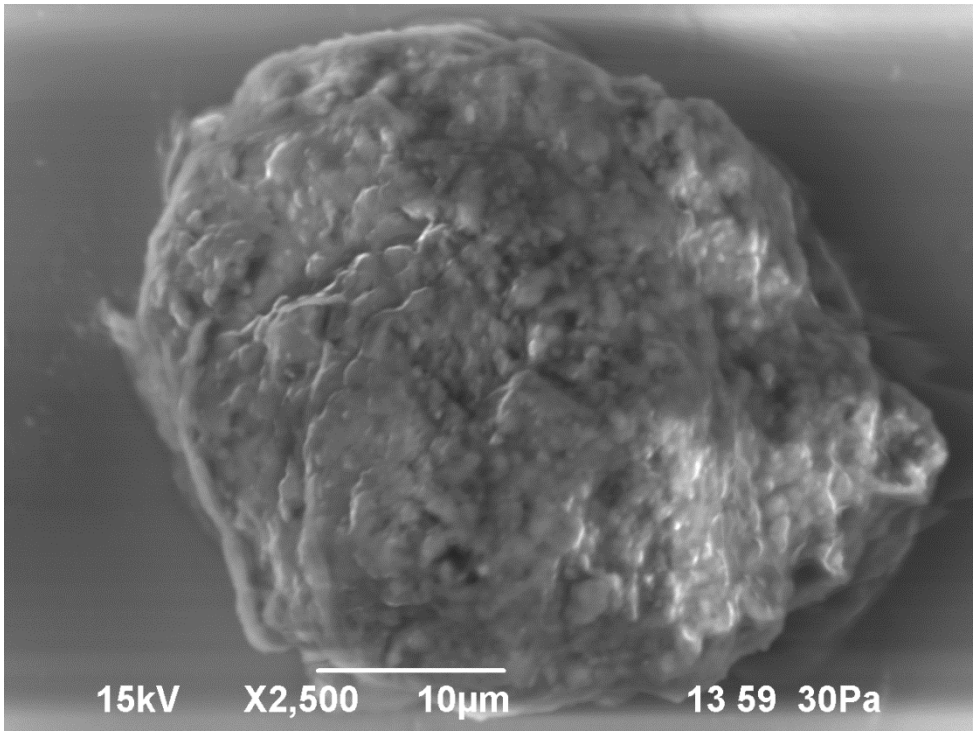

Fig. 4-7: An SEM scan of a sand grain from Mitribah, Kuwait, of length 40µm.

4.1.4 Retrieval of roughness parameters

Data gained from the AFM work on dust grains was analysed to derive the correlation length and standard deviation values. Firstly, scanning artifacts were averaged out and the surface had a fitted 2D polynomial subtracted from it to remove the grain's overall long-range profile and leave just the roughness (Fig. 4-8).

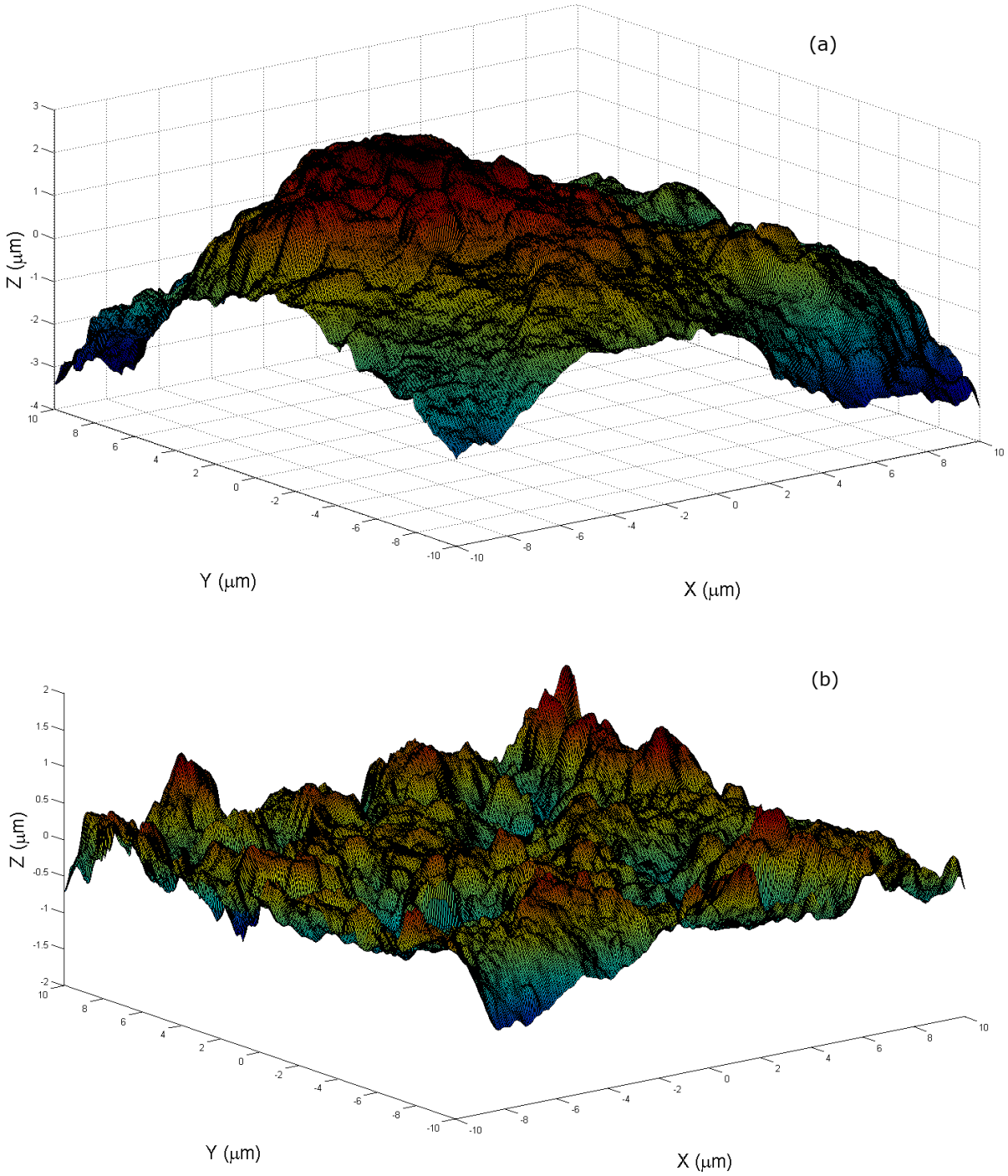

Fig. 4-8: (a) shows the surface before it has had the overall profile removed. (b) shows the surface after the overall long-range profile has been removed.

Secondly, the Fourier spectrum of the surface was generated and then all frequencies above a cutoff were nulled (Fig. 4-9). The remaining spectrum was inversely transformed and a new surface was created that contained only the low frequencies (Fig. 4-10). This was visually compared with the fitted surface, and the process was repeated with the cutoff being varied until the created surface appeared to match the large-scale

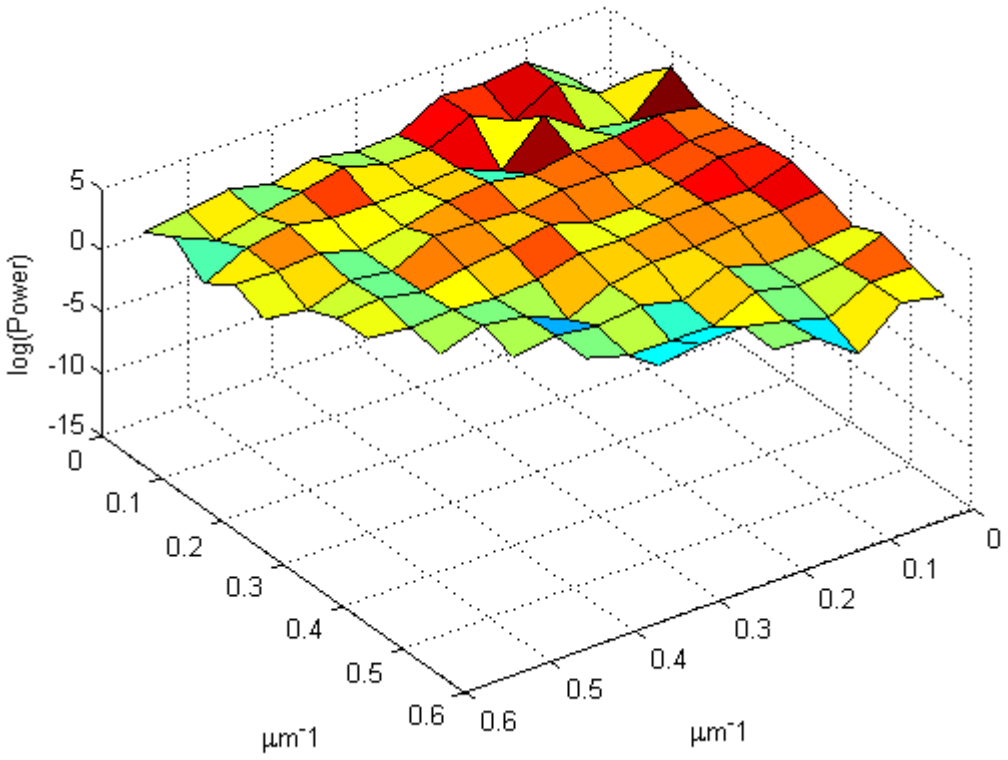

Fig. 4-9: Power spectrum of low frequencies of the measured surface. The frequency cutoff is at 0.6µm^{-1}.

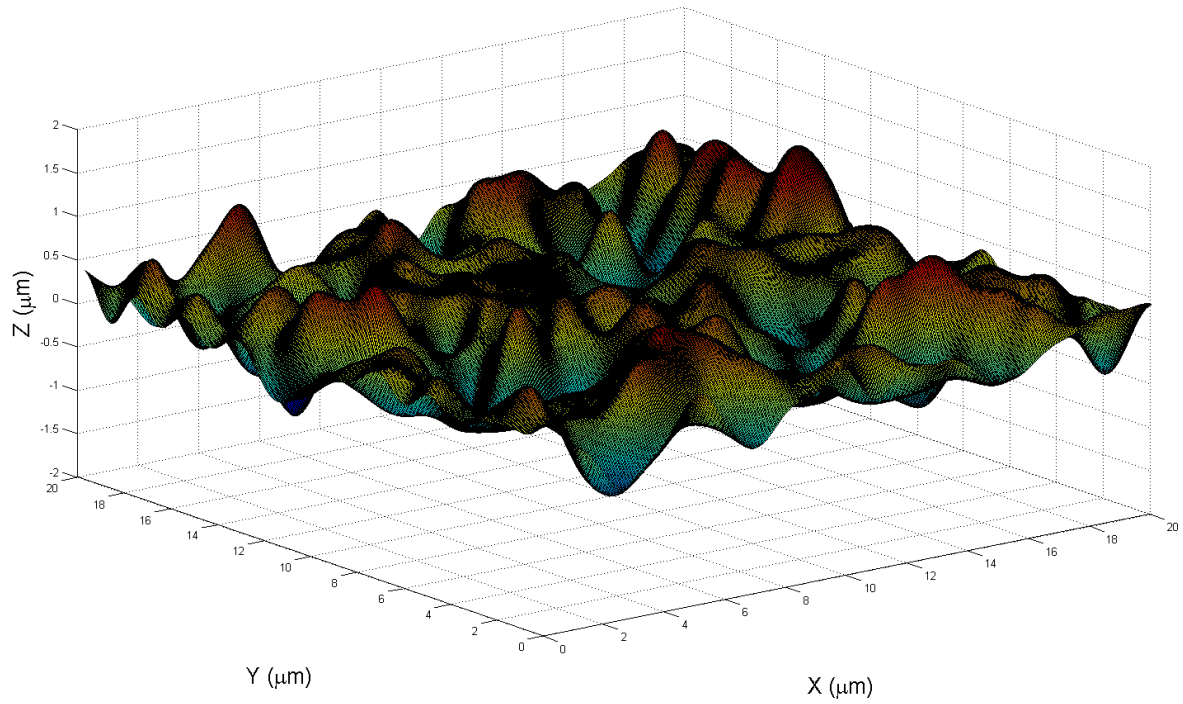

Fig. 4-10: Simulated surface containing only low frequencies from the measured surface.

features in the fitted surface. A high frequency surface (Fig. 4-12) was then created using the previously discarded part of the spectrum (Fig. 4-11).

Thirdly, these surfaces were analysed to obtain values for the standard deviation and correlation length. Correlation lengths for both surfaces were retrieved by calculating autocorrelation and using a rearranged form of equation 3. The surface created using only high frequencies shows edge effects, and so the edges were not considered in deriving these parameters.

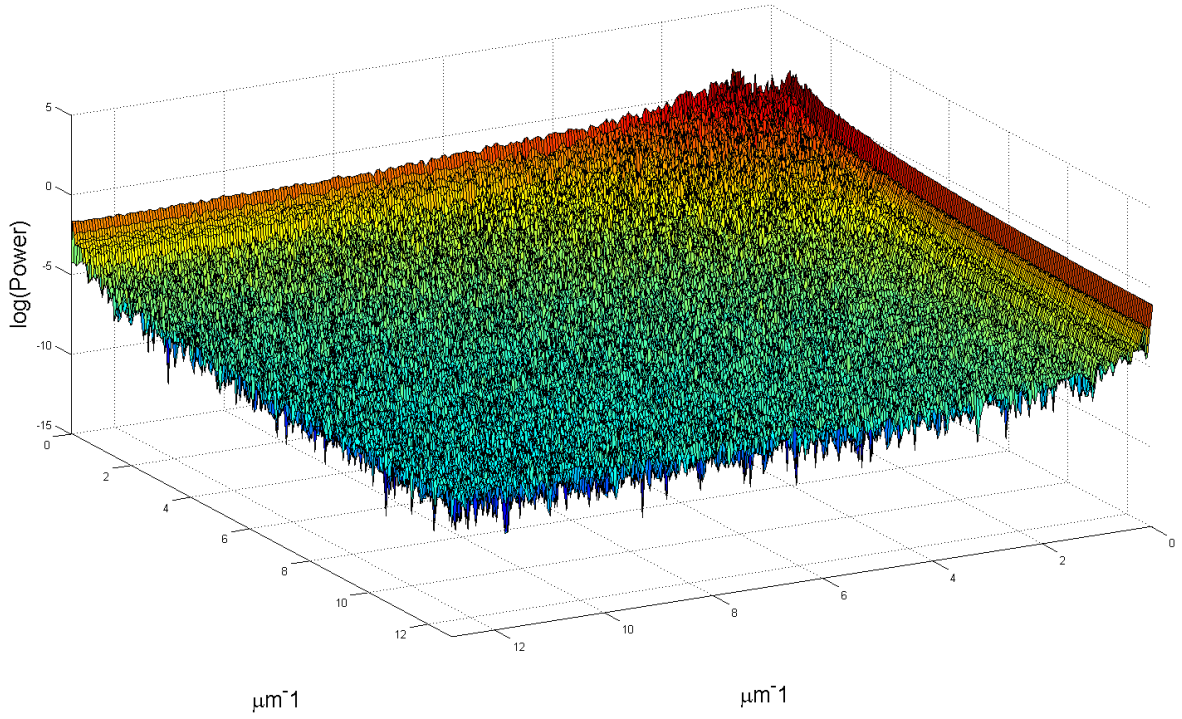

Fig. 4-11: Power spectrum of high frequencies of the measured surface. The frequency cutoff is at 0.6μm^{-1}.

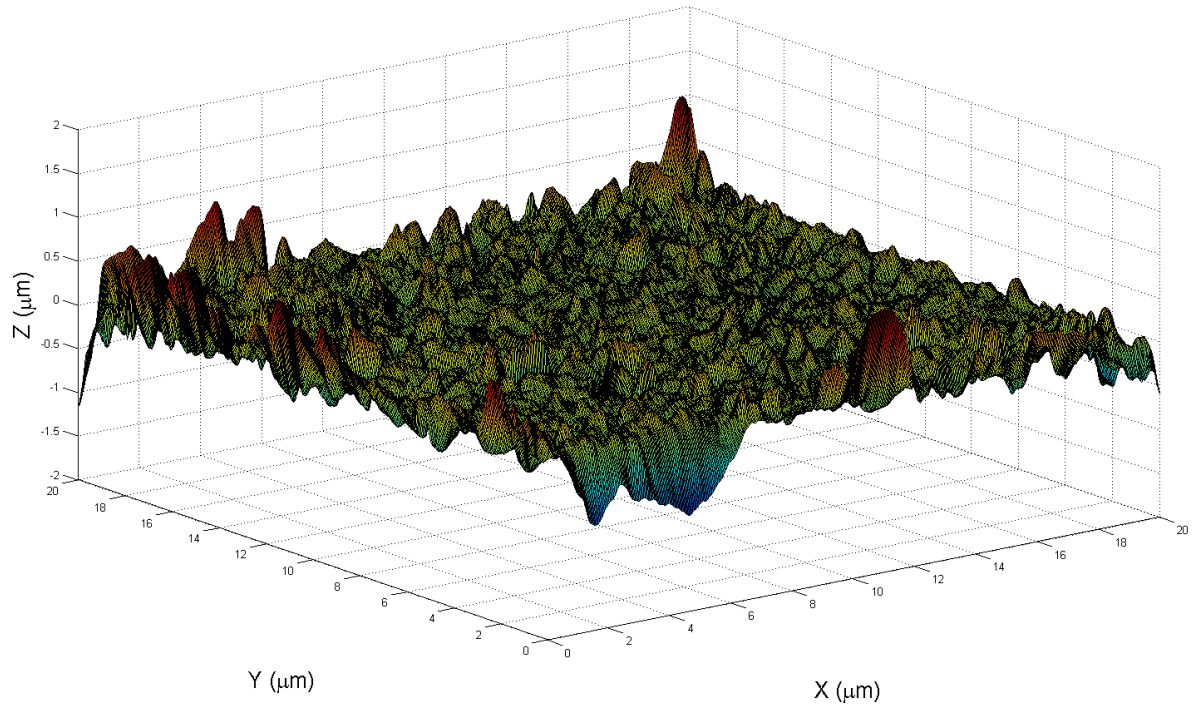

Fig. 4-12: Surface containing only high frequencies from the measured surface.

4.2 *Manufacturing of Gaussian rough ice crystal analogues*

4.2.1 Ice analogue selection

Ice analogues selected must have a large aspect ratio to reduce the effect on light scattering of the basal facets, which are not formed smoothly; and to reduce the proportion of the crystal length that will later be attached to a glass fibre. They must also be large enough to be easily handled using the micromanipulator and be small enough that light scattering computations can realistically be done using the discrete dipole approximation. The prismatic facets should be smooth and the crystal should not show any obvious defects from the required hexagonal prism shape (Fig. 4-13).

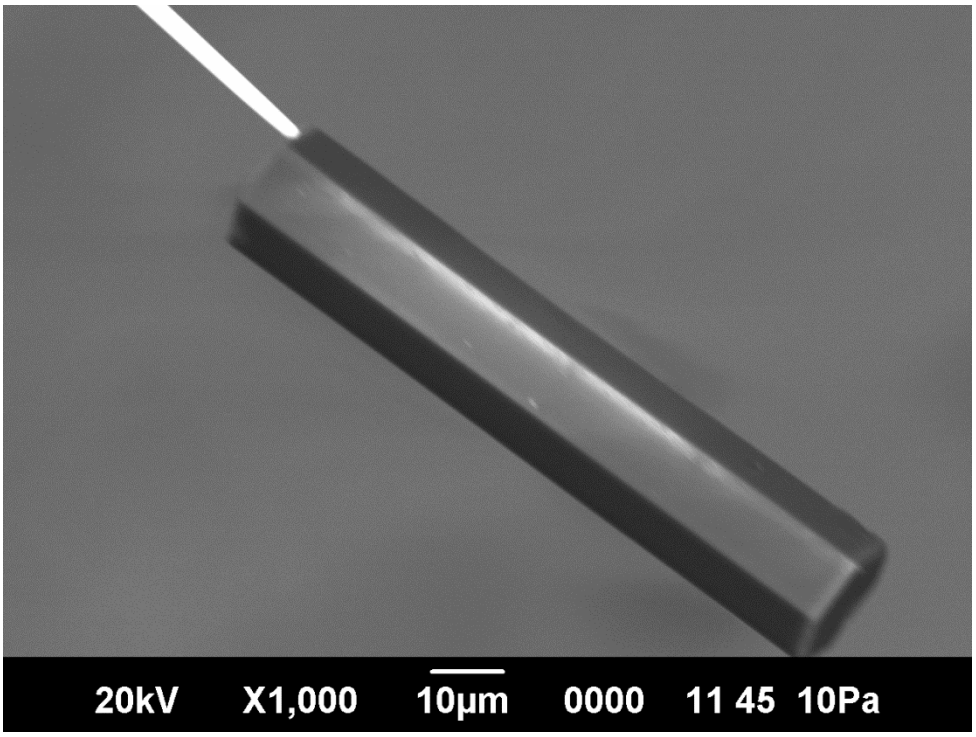

Fig. 4-13: An SEM image of a fibre-mounted ice analogue that satisfies the requirements. The length is approximately 90µm and the prism diameter is approximately 20µm.

4.2.2 Holder preparation

A holder is made up of three parts. The first part is a mild steel rod of length 7mm and diameter 1.55mm. At both ends, the cross-section was filed down to a semi-circle – this means the crystal will better stay in position when rotated and is necessary for rotation with the SEM mount.

The second part is copper wire – approximately 3mm in length and of 50μm diameter. This was glued onto one of the filed-down flats on the steel rod using conducting epoxy glue. The unattached end was made to project out as an extension of the cylinder axis.

The third part is a glass fibre. This was pulled using a Kopf Instruments Model 720 needle/pipette puller with the solenoid set to full and a heater value of 10 to produce an end that tapers down to a radius of curvature of just a few μm. Once the fibre was pulled, a 3mm length of it including the point was placed between two clean glass slides. This setup was held in place and the part sticking out was flexed up and down until it snapped off – the remainder of the fibre was then discarded. The glass slides ensured that the snap occurred at the required position along the fibre.

To attach the fibre to the rest of the holder, the remainder of the fibre was gently moved so that the thicker end slightly overhung the glass slide it was on. The free end of the copper wire had epoxy glue applied to it and this was pressed against the thicker end of the glass fibre so that the two attached. When the glue set, the copper wire was bent so that the fibre point was in line with the steel rod axis.

Finally, the holder was sputter coated with gold using an Emitech SC7620 sputter coater, making all surfaces conductive. This allows easier examination of the fibre with a scanning electron microscope (SEM) (Fig. 4-14). The holder is only handled with inverted forceps; this means that accidents are less likely which is important because it is very fragile.

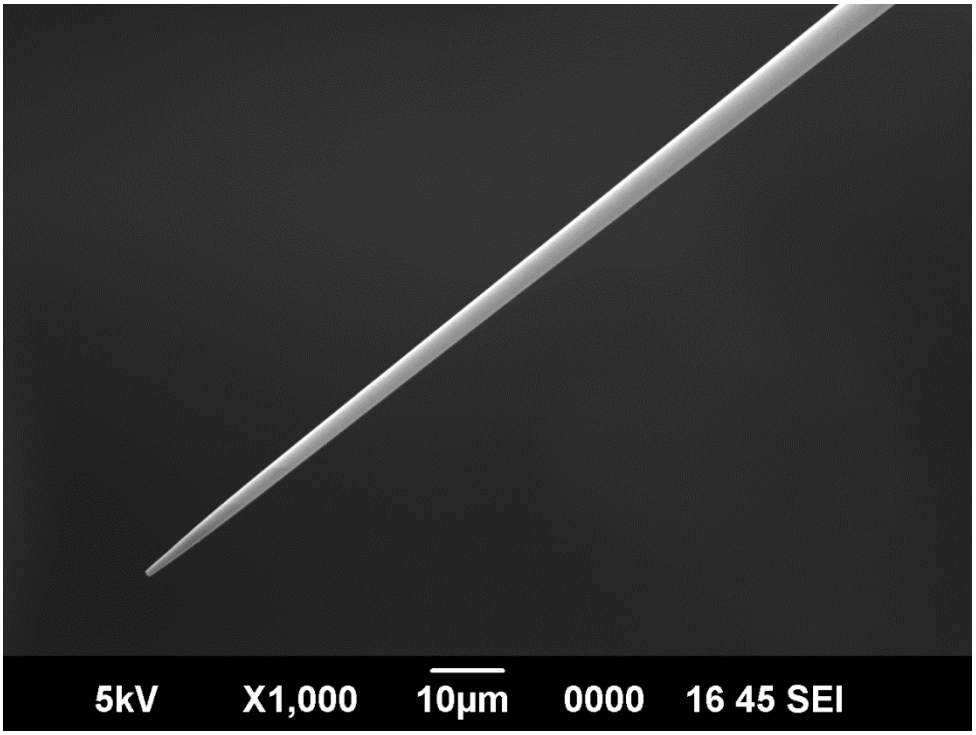

Fig. 4-14: An SEM image of a gold sputter coated glass fibre.

4.2.3 Analogue mounting

Crystal mounting was done using micromanipulators and an optical microscope. The analogue was first picked up using a tungsten probe, and usually attached to the probe by van der Waal's force; where this alone was not sufficient for the crystal to stick, the probe was dipped in a 3.5% solution of glycerol to make the crystal easier to pick up. This probe was then placed into a remote hydraulic micromanipulator (a Narishige MHW-103), but not yet placed in the optical microscope.

Once this was done, the mount was set up in a different micromanipulator (same make and model) and placed in an inverted optical microscope (Olympus CK2). Epoxy glue was prepared; glue with a setting time of 30 minutes was used as enough time was needed to allow the attaching procedure to be carried out. A separate probe was dipped in the glue, and then mounted in the optical microscope.

The probe with the glue and the probe with the holder were both positioned in the optical microscope's view (Fig. 4-15); the micromanipulator was then operated to dip the

point of the fibre into the glue (Fig. 4-16). The glue-bearing probe and its mount were then removed and the glue was wiped off the probe. The glue spot can be seen at higher magnification in figure 4-17.

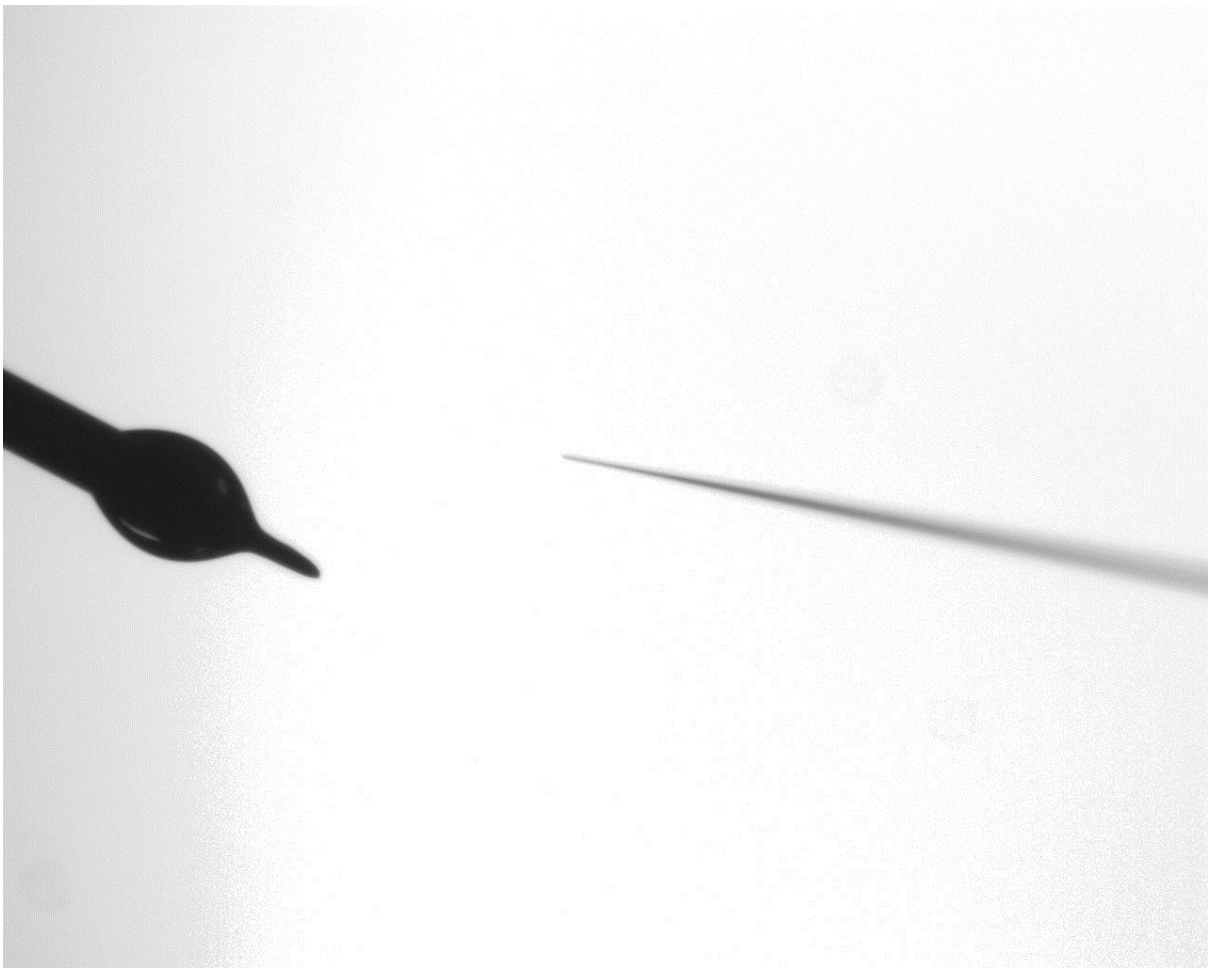

Fig. 4-15: Optical microscope image of a glue spot and a fibre at 4X magnification.

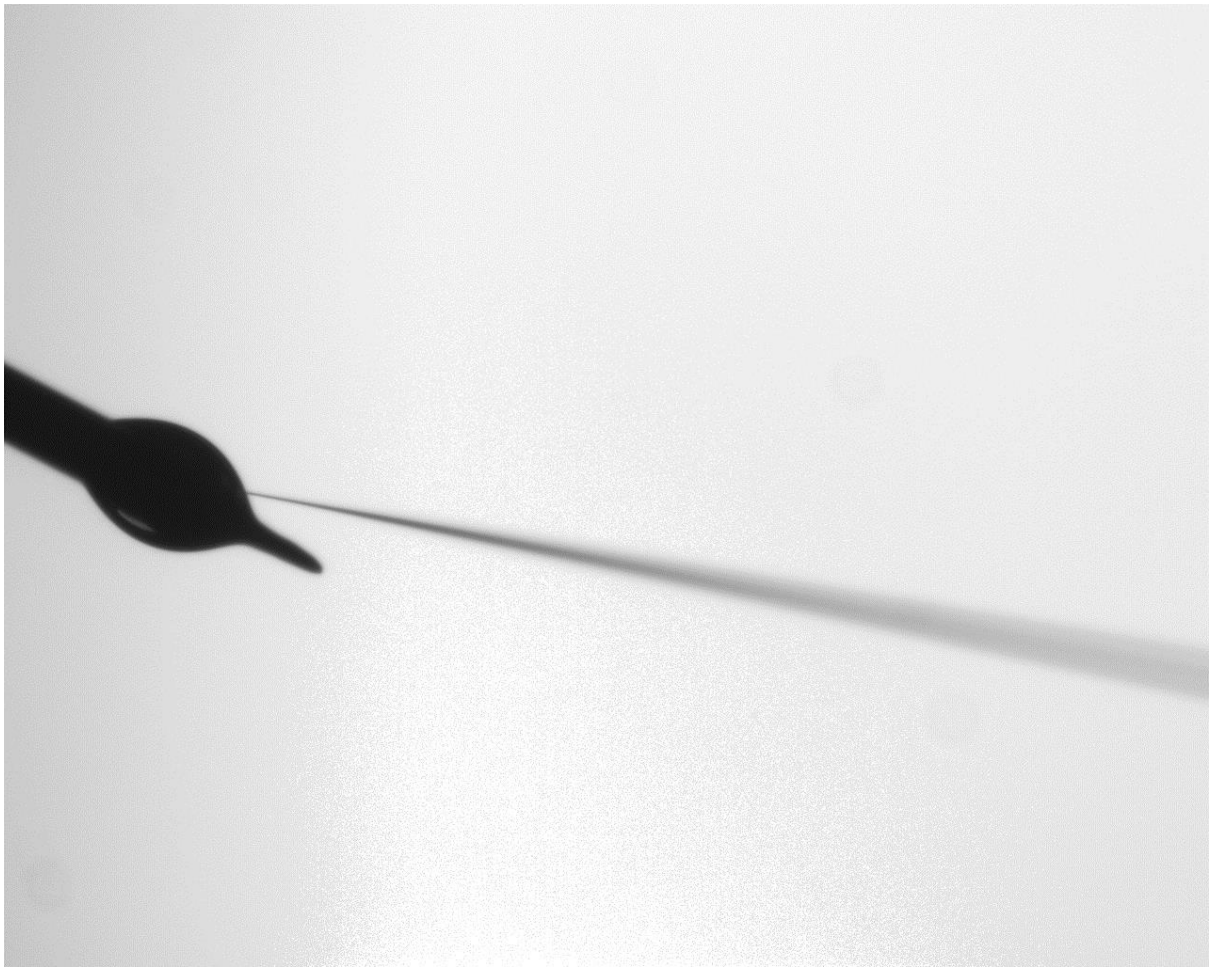

Fig. 4-16: Optical microscope image of a fibre with its point dipped into a glue spot at 4X magnification.

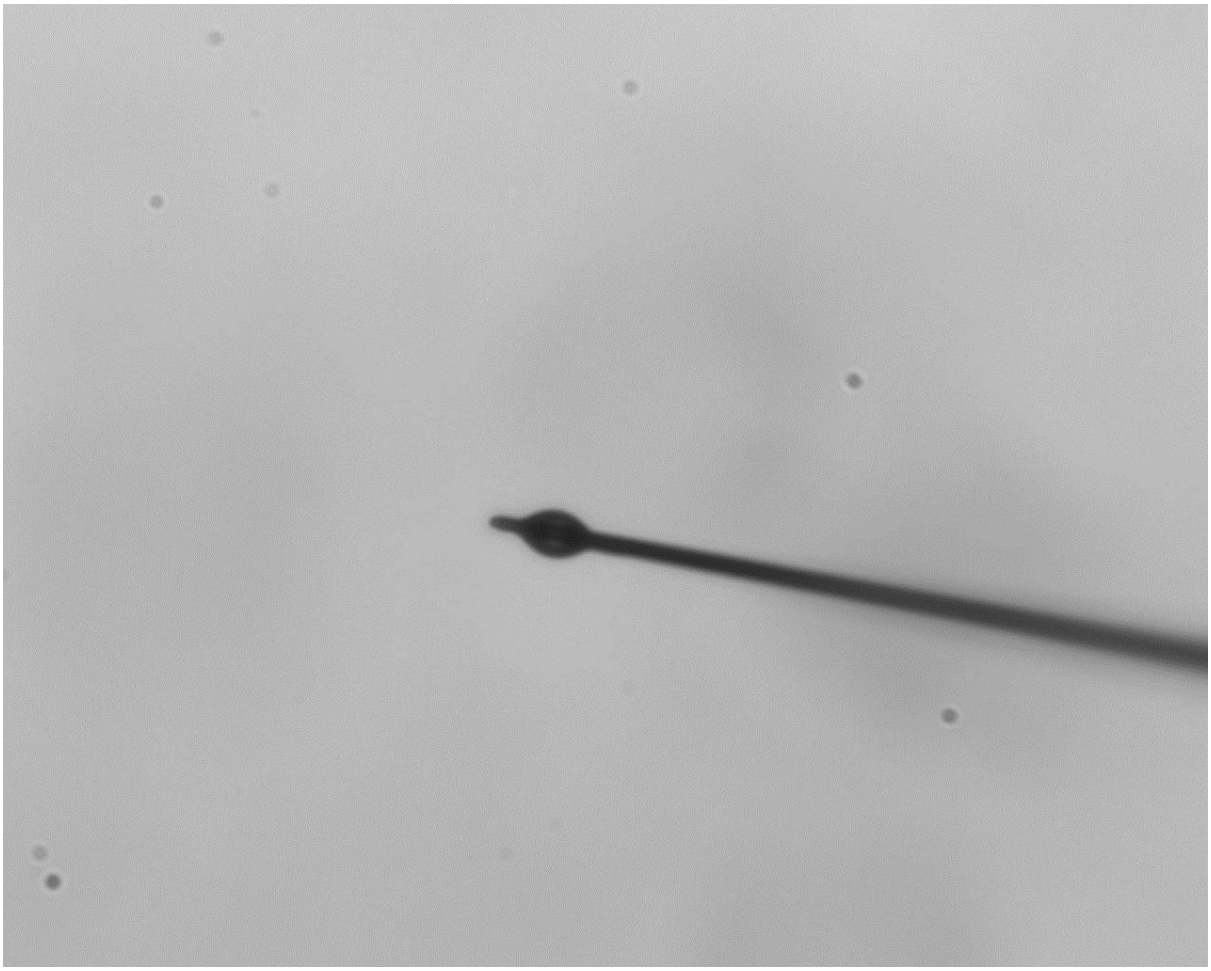

Fig. 4-17: A glue spot on a fibre at 40x magnification.

Fig. 4-18: Optical microscope image of a glue-dipped sputtered fibre and a probe-mounted ice analogue crystal at 4X magnification.

Next, the micromanipulator with the probe bearing the crystal was placed in the optical microscope. The fibre and the crystal were brought into view (Fig. 4-18), and the micromanipulators were used to manoeuvre the fibre so that it touched a small proportion of the length of one of the prism facets (Fig. 4-19). The micromanipulators were then used to separate the fibre and crystal from the probe the crystal was attached to before.

Fig. 4-19: Optical microscope image of an ice analogue crystal being attached to a glue-dipped sputter coated fibre at 4X magnification.

The crystal should point along the rod axis. This was often not the case at this stage, and the crystal quite often bent at a 90° angle to what was desired. This can be rectified using the probe to which the crystal was originally attached to push it back into the desired position (Fig. 4-20). In the case that the whole crystal couldn't be in focus at the same time, the probe with the holder was rotated by 90° within its micromanipulator and the crystal was again pushed into the desired position.

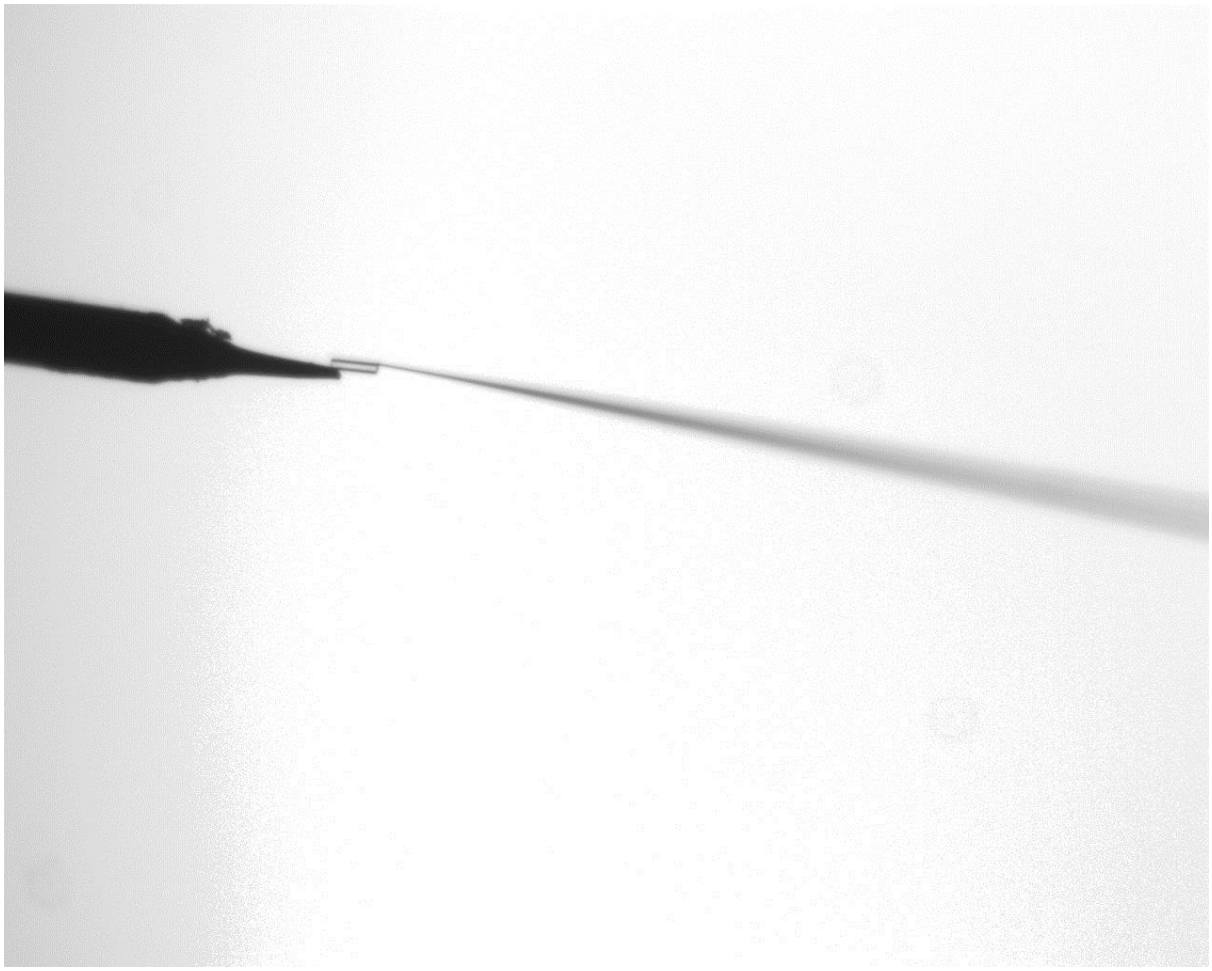

Fig. 4-20: Optical microscope image of an ice analogue crystal attached to a fibre with a probe being used to correct the crystal orientation at 4X magnification.

4.2.4 Ion beam milling

An ice analogue (Fig. 4-25) had three facets roughened using a Carl Zeiss Gemini 1540XB FIB/SEM system. This system uses a focused beam of gallium ions to sputter a defined pattern into a surface, and so it can be used to create Gaussian roughness on the surface of a mounted ice crystal analogue.

Six facets would have been better; however this wasn't possible because the available milling time was limited. The work was also difficult to do; the crystal had to be carefully aligned by hand for each individual facet and charge build-up meant that the milling had to be done slowly.

This crystal was used for scattering experiments using SID3 – the results can be seen in Fig. 5-13 and Fig. 5-15.

To make this possible, an adaptor was created to allow mounted ice crystal analogues to be held in place inside SEM systems (Figs. 4-21 & 4-22). It is an aluminium cylinder with a collar and a cut-out at the top, a hole extending horizontally from the cut out part to the edge of the cylinder and a threaded hole with a 2mm diameter grub screw which enters the remaining part of the cylinder top and reaches down to the first hole.

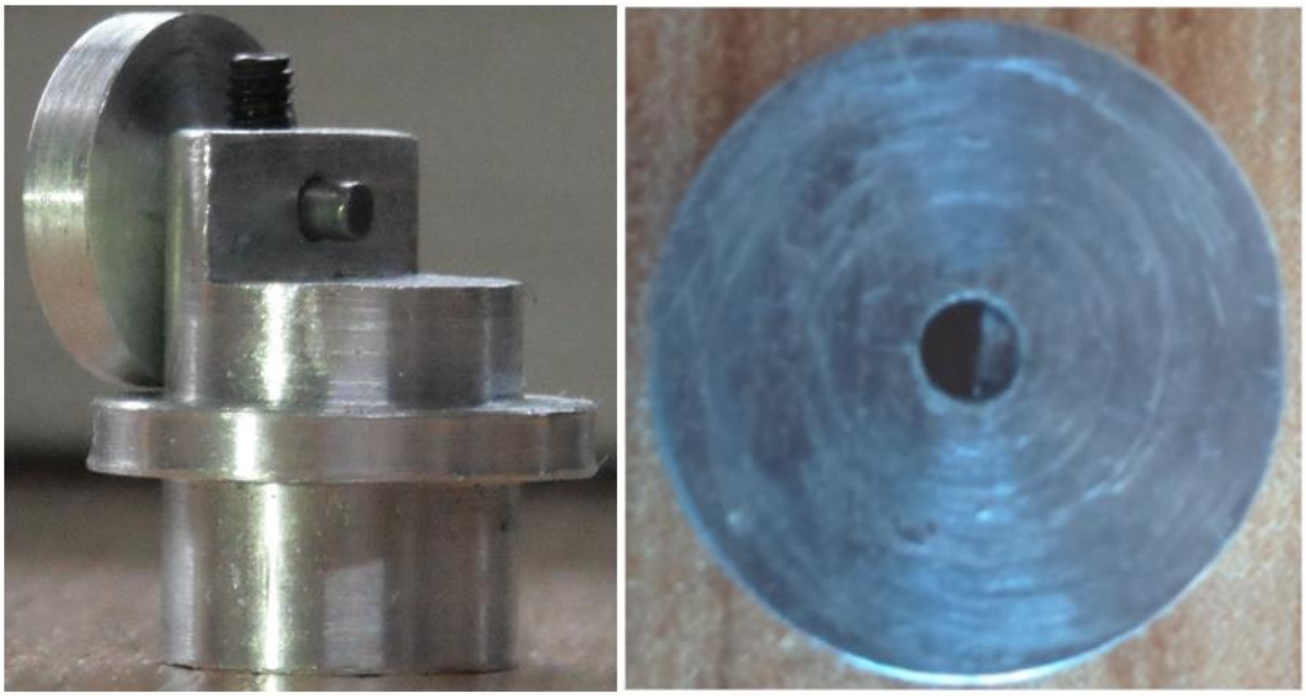

Fig. 4-21: Left - an SEM adaptor for mounted ice crystal analogues. The rotation disc can also be seen attached to a rod. Right – the rotation disc.

There is also a disc which is used for rotation while the crystal is mounted. This has a hole through its centre which has a flat piece of metal fixed inside. This addition means that the hole can lock on to the flat on the steel rod part of the ice analogue holders and this enables it to be used to rotate the crystal on the SEM mount.

To use the SEM mount, the free end of the rod part of the ice analogue holder rod was pushed into the SEM mount hole, taking care not to touch the fibre. The rod was pushed far enough through so that the filed-down part protruded. The grub screw was then tightened to lock the holder in place and the rotation disc was attached to the flat part of the rod. Using a reflecting optical microscope and the rotation disc, the crystal was rotated to have a prism facet facing up. This process can be repeated so that each prism facet in turn can be upward facing, loosening and tightening the grub screw each time.

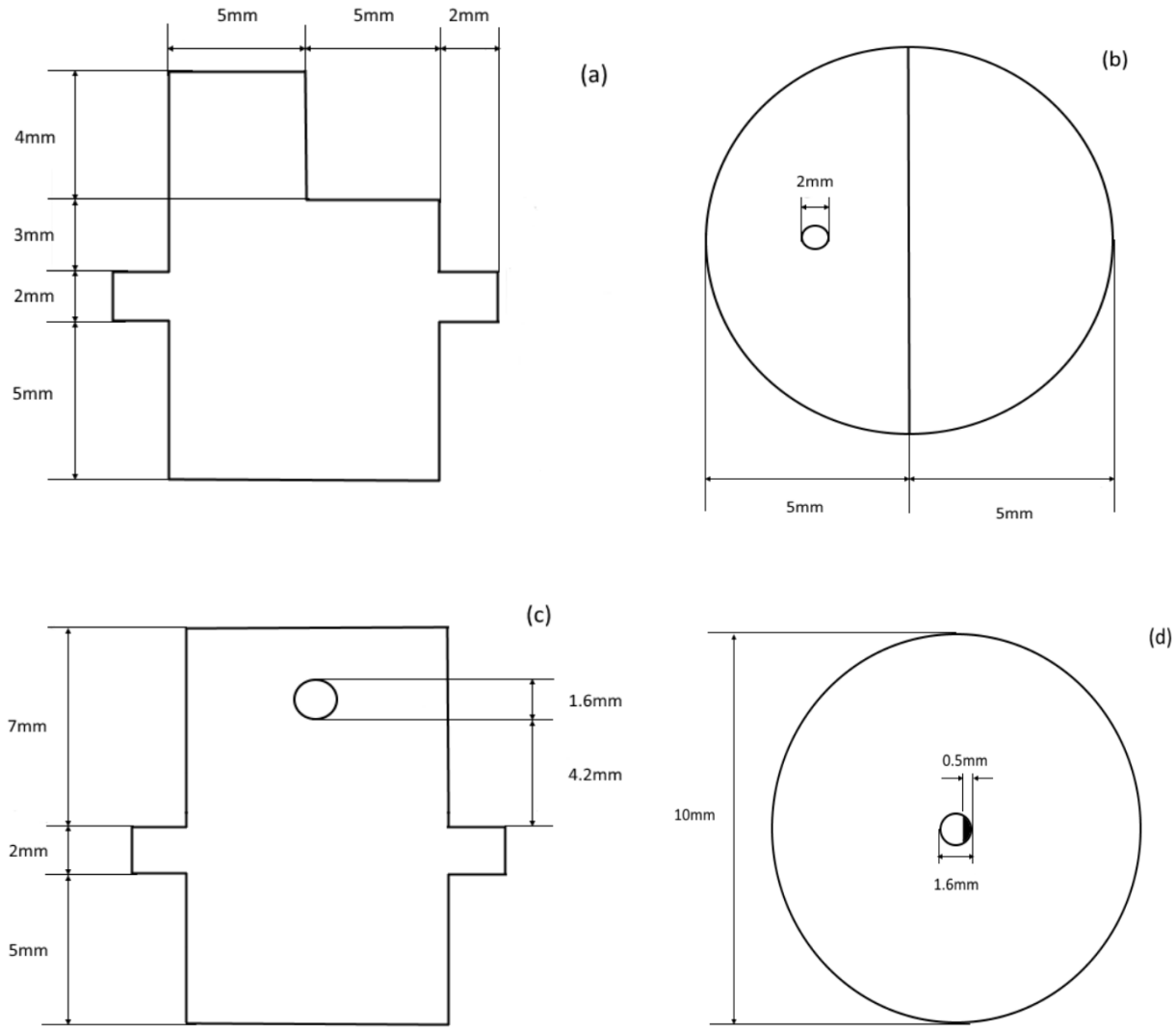

Fig. 4-22: Schematics of the holder. (a) is the front view, (b) is the top view, (c) is the side view and gives a view of the hole, and (d) is the disc.

Examples of milled ice analogues can be seen in Figs. 4-23, 4-24 and 4-25. The ice analogue in Fig. 4-23 is attached to a substrate and was milled prior to the initiation of this project as a test of the feasibility of ion beam milling on sodium fluorosilicate, using a variable pitch eggbox pattern.

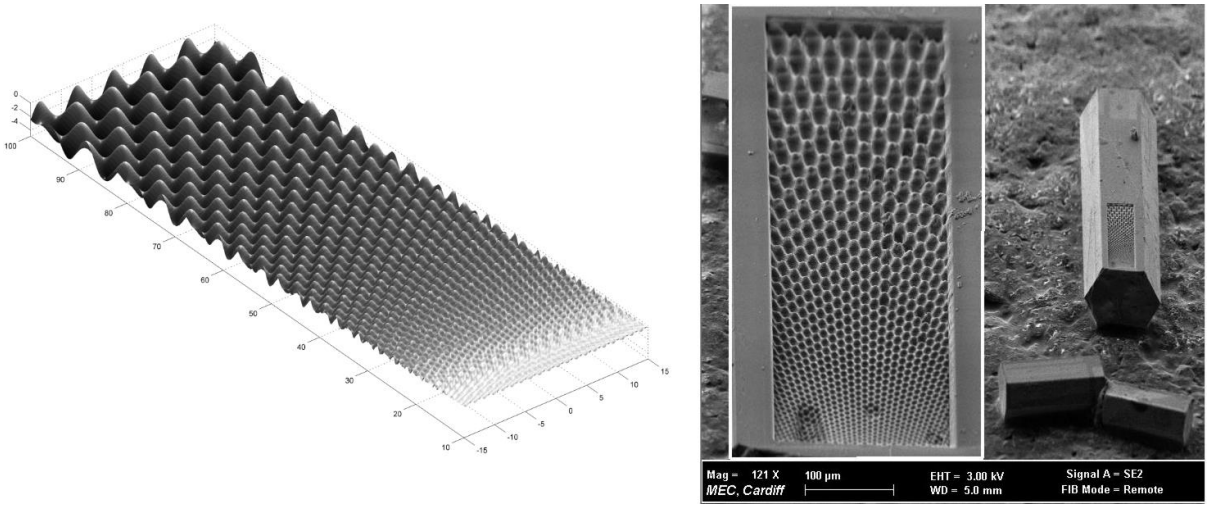

Fig. 4-23: Left: An eggbox geometry test pattern. Right: An ice analogue crystal, affixed to a substrate, with the same eggbox pattern sputtered into it. Images courtesy of Joseph Ulanowski.

The ice analogue in Fig. 4-24 has a Gaussian rough pattern milled into it. It is a test crystal, glued to a substrate and so cannot be used in scattering experiments. Fig. 4-25 shows a fibre mounted ice analogue with a pattern sputtered into it on three facets.

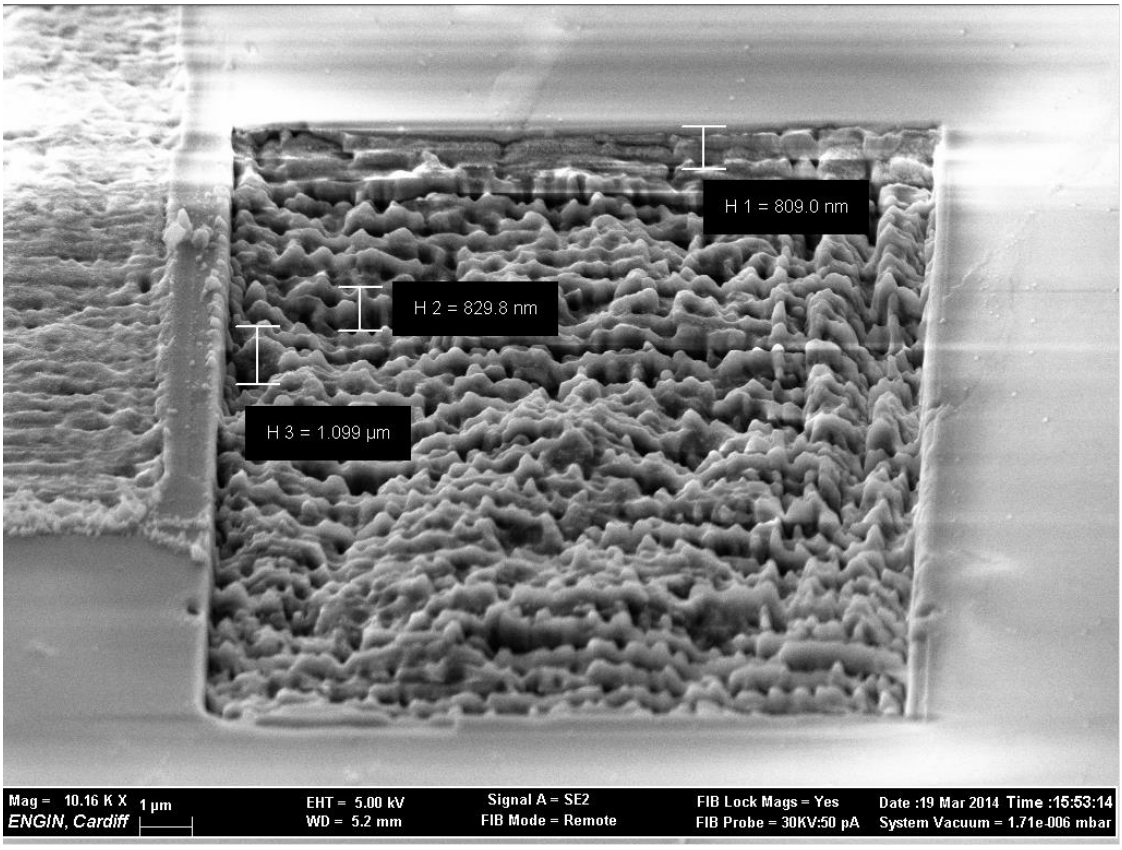

Fig. 4-24: An ice analogue crystal, affixed to a substrate, with a Gaussian rough surface sputtered into it.

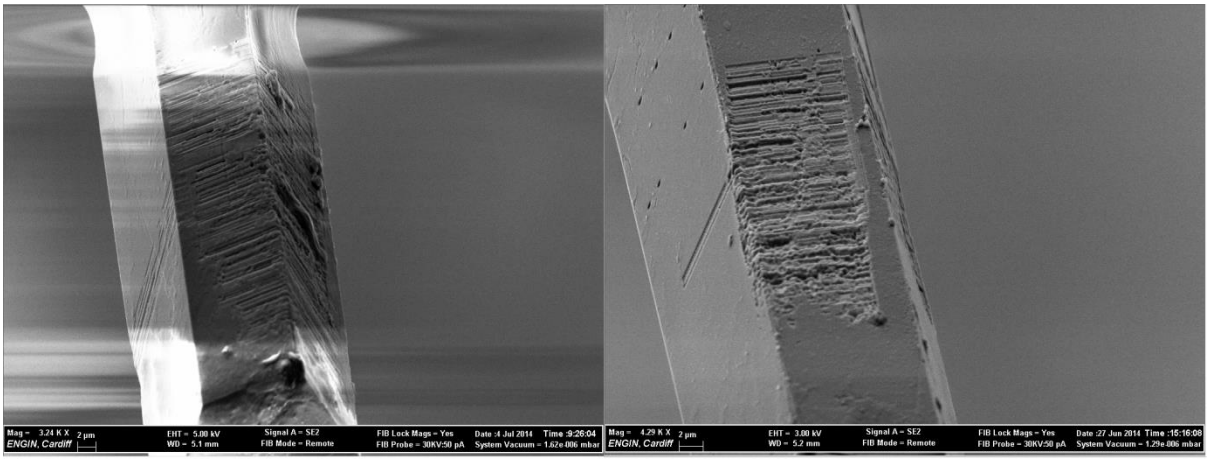

Fig. 4-25: A fibre mounted ice analogue crystal with a pattern sputtered into three adjacent prism facets.

5. Light scattering by Gaussian random crystals

5.1 Computations performed using the discrete dipole approximation

Computations were carried out using the ADDA [26] implementation of the discrete dipole approximation light scattering model to find the light intensity as a function of azimuthal angle and scattering angle for smooth, moderately rough and highly rough hexagonal columns at a wavelength of 532nm. Four different beam orientations were considered (Fig. 5-1) for a fixed crystal orientation (Fig. 5-2).

Using this model, the incident beam originally propagates along the *x*-axis; Fig. 5-2 shows the rotations performed to achieve these beam orientations; all make an angle of 30° with the *x* axis in the *x*-*y* plane, and make an initial rotation of 0°, 10°, 20° or 30° with the *y* axis in the *y*-*z* plane projection. Four crystal size parameters were considered; 20, 40, 60 and 100, with roughness being scaled proportionately with size.

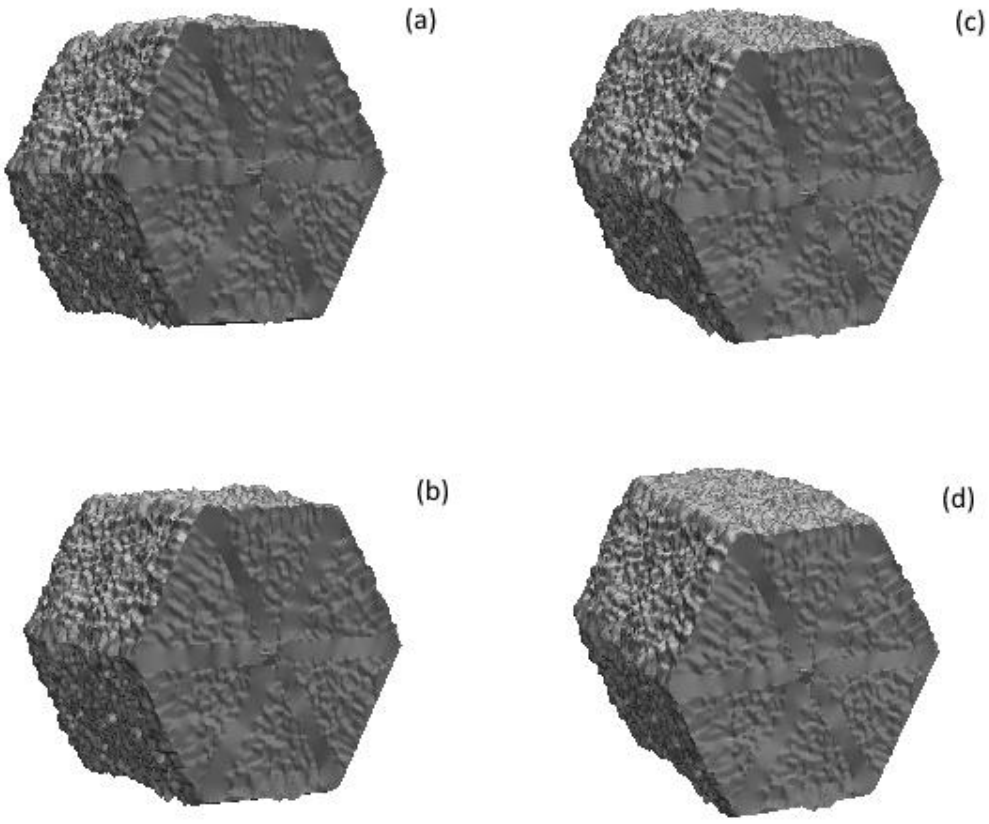

Fig. 5-1: The crystal orientations used in this work as seen by the incident beam, shown for a two-scale rough crystal. All beam rotations are at 30° in the *x*-*y* plane; (a) has no other rotation, (b) is rotated 10° in the *y*-*z* plane, (c) is rotated 20° in the *y*-*z* plane and (d) is rotated 30° in the *y*-*z* plane. The coordinate system used can be seen in Fig. 5-2.

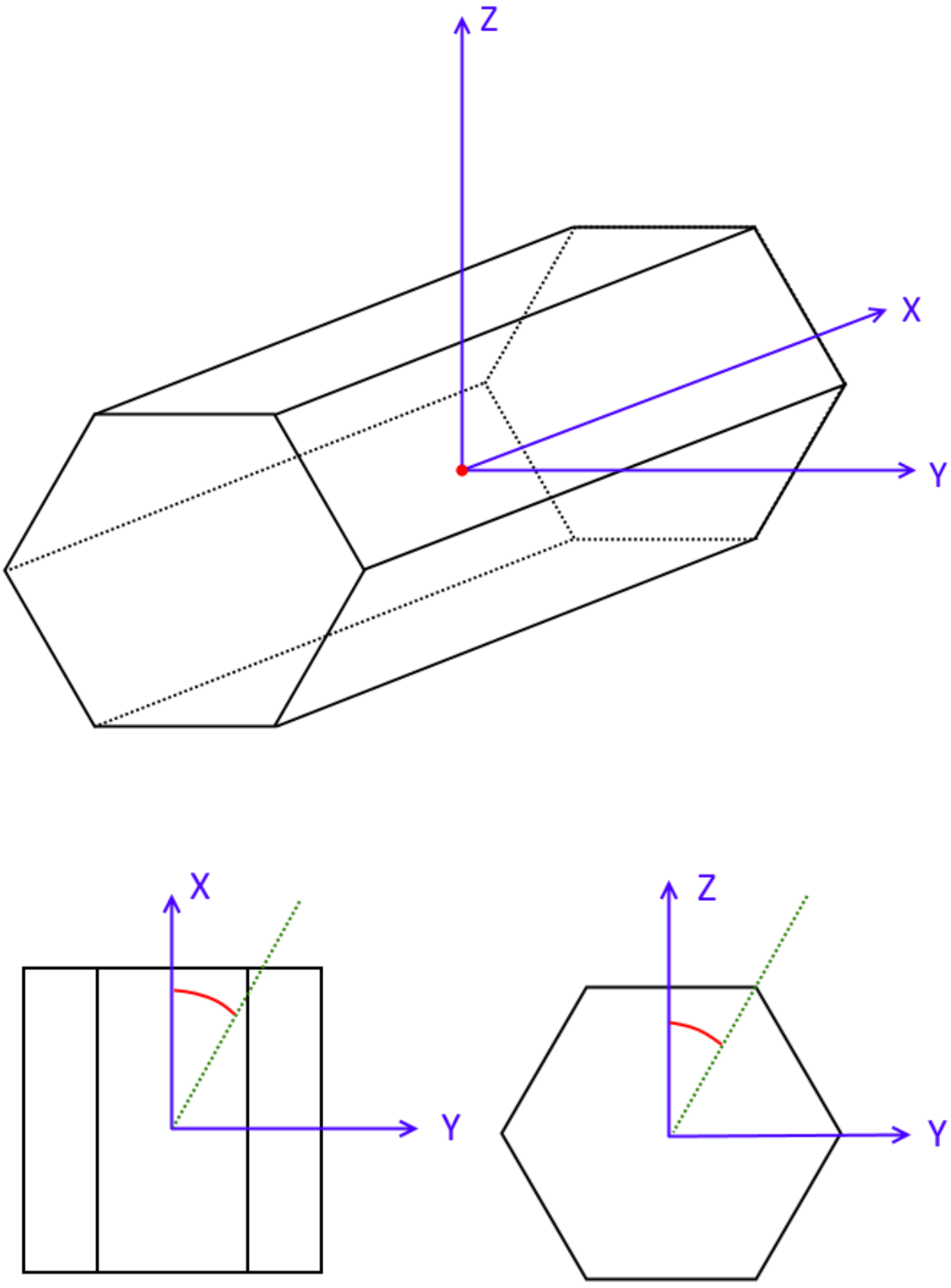

Fig. 5-2: The top image shows the coordinate system in which the beam sits. Below left is the rotation around the *z*-axis, and below right is the rotation around the *x*-axis.

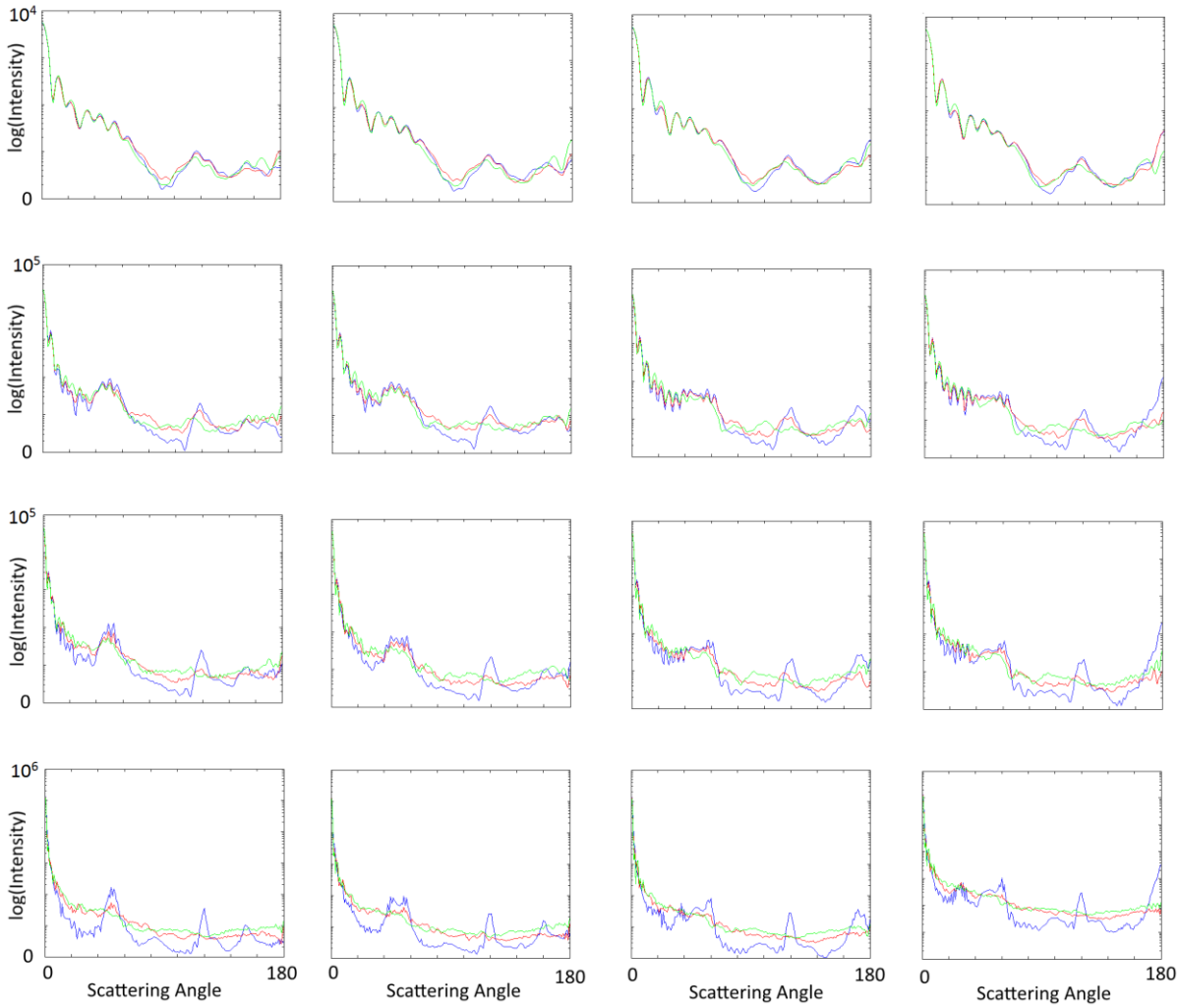

Fig. 5-3: Phase functions of smooth (blue lines), one-scale rough (red lines) and two-scale rough (green lines) crystals. Different columns in the diagram represent different beam orientations - from left to right, the angle in the *y*-*z* plane increases from 0° to 30° in steps of 10°. Different rows represent different size parameters - from top to bottom, the size parameter is 20, 40, 60 and 100.

Fig. 5-3 shows the phase functions from DDA calculations.

At a size parameter of 20, roughness makes little difference to the result with respect to the pristine crystal's scattering pattern – this can also be seen from the 2D scattering patterns in Fig. 5-5 and Fig. 5-6, which show little difference to the smooth case other than features being slightly smudged out. This is due to the fact that its surface roughness is small compared to the wavelength, and so light scattering is not sensitive to it. Most of the difference that can be seen is in the backward direction (i.e., a scattering angle >90°), which is due to the fact that forward scattering is dominated by external diffraction, which has a low sensitivity to surface roughness because it only

depends on the crystal's 3-dimensional contour. Roughness also increases side scattering (i.e., at a scattering angle of ~90°).

At a size parameter of 40, some roughness effects can be seen in the forward direction – features present in the smooth case are still present in the rough cases, but less pronounced. This is most noticeable in the 20°-40° range, where the depth of the troughs is noticeably reduced. Below 20°, no difference can be seen – external diffraction is still dominant here. This agrees with the 2D scattering pattern in Fig. 5-5, which shows little difference for one-scale roughness and slightly more for two-scale roughness compared to the smooth pattern. However, a large effect can be seen in the backward direction – features present for the smooth case are much reduced in the rough cases, particularly for the two-scale rough crystal; those peaks that remain seem to be shifted towards lower scattering angles. This is evident in the 2D scattering patterns in Fig. 5-6, which show that one-scale roughness distorts the features that appear for the smooth crystal and that two-scale roughness largely turns these features into speckle.

At a size parameter of 60, roughness affects forward scattering with respect to its smooth counterpart more than for smaller size parameters. The depth of the troughs (with respect to the smooth crystal's scattering pattern) in the range of 20°-40° is much reduced, as is the height of the peaks in the range of 40°-60°. The 2D scattering patterns in Fig. 5-5 also show this; it is particularly evident for two-scale roughness. Little remains of the features in the backward scatter direction; the peak at 120° has mostly disappeared, particularly for the two-scale rough crystal – which has no features left apart from a slight peak beyond 160°. Fig. 5-6 confirms that the features have disappeared, with only speckle remaining.

At a size parameter of 100, most of the features from the smooth pattern in the forward direction beyond 10° are greatly reduced - particularly for the two-scale rough crystal, for which the features in the 20°-60° range have disappeared. This agrees with what can

be seen in Fig. 5-5. In the backward direction, no features remain for either one or two-scale roughness. Fig. 5-6 shows that only speckle remains.

For the pristine crystal at this size parameter the size parameter is closest to geometric optics ray tracing, and several peaks can be identified for each orientation. The primary contribution for each is identified below; it is important to note that more complex ray interactions can produce the same exit angle (e.g. external reflection and refraction-internal reflection-refraction interactions).

For the orientation at 0° in the *x*-*y* plane, peaks can be seen for the smooth crystal at 51° (caused by reflection off the prism facets), 80° (due to light passing through the two prism facets and passing through the two opposite them), 120° (due to light reflecting off the basal facet) and 152° (due to light passing through the basal facet facing the beam, internally reflecting off the other basal facet and exiting back through the first basal facet).

For the orientation at 10° in the *x*-*y* plane, peaks can be seen for the smooth crystal at 10° (due to reflection off the prism facet that is slightly revealed by the rotation), 48° (due to reflection off the prism facet that is slightly rotated away), 55° (due to reflection off the remaining upward facing prism facet), 120° (due to reflection off the basal facet) and 160° (due to light passing through the basal facet facing the beam, internally reflecting off the other basal facet and exiting back through the first basal facet).

For the orientation at 20° in the *x*-*y* plane, peaks can be seen for the smooth crystal at 20° (due to reflection off the prism facet that is being revealed by the rotation), 38° (due to reflection off the prism facet that is being rotated away from the beam), 58° (due to reflection off the remaining upward facing prism facet), 120° (due to reflection off the basal facet) and 170° (due to light passing through the basal facet facing the beam, internally reflecting off the other basal facet and exiting back through the first basal facet).

For the orientation at 30° in the *x*-*y* plane, peaks can be seen for the smooth crystal at 28° (due to reflection off the two prism facets that now face the beam to the same extent), 60° (due to reflection off the remaining upward facing prism facet), 120° (due to reflection off the basal facet) and 180° (due to light passing through the basal facet facing the beam, internally reflecting off the other basal facet and exiting back through the first basal facet).

Size parameter 60					
	Roughness				
	Orientation		Smooth	One-scale	Two-scale
		off30x0	0.7113	0.7215	0.7024
		off30x10	0.7272	0.7386	0.7002
		off30x20	0.7253	0.7382	0.7098
		off30x30	0.7242	0.7459	0.7215
		mean	0.7222	0.7363	0.7086

Size parameter 100					
	Roughness				
	Orientation		Smooth	One-scale	Two-scale
		off30x0	0.7641	0.7581	0.7258
		off30x10	0.7612	0.7584	0.7134
		off30x20	0.7708	0.7678	0.7237
		off30x30	0.766	0.7673	0.7288
		mean	0.7656	0.763	0.7229

Size parameter 20					
	Roughness				
	Orientation		Smooth	One-scale	Two-scale
		off30x0	0.7251	0.7236	0.744
		off30x10	0.7219	0.7185	0.7402
		off30x20	0.7201	0.7202	0.7357
		off30x30	0.7196	0.7175	0.7343
		mean	0.7217	0.72	0.7386

Size parameter 40					
	Roughness				
	Orientation		Smooth	One-scale	Two-scale
		off30x0	0.7102	0.6939	0.7162
		off30x10	0.7004	0.689	0.7141
		off30x20	0.6942	0.6975	0.716
		off30x30	0.6978	0.7051	0.7228
		mean	0.7008	0.6964	0.7173

Fig. 5-4: Tables of asymmetry parameters calculated for the crystals for which light scattering was modelled.

In general, it can be seen that roughness causes peaks and troughs that appear for smooth particles in the phase function of large crystals to be smoothed out, particularly two-scale roughness. As crystal size reduces, this roughness effect is lessened.

Fig. 5-4 shows the asymmetry parameters derived from DDA calculations. At size parameters of 20 and 40, the asymmetry parameter reduces slightly for one-scale roughness and increases for two-scale roughness with respect to that for the smooth crystal case. The decrease for one-scale roughness would suggest a slight increase in reflectance. The increase in asymmetry parameter for the two-scale case suggests that the cloud becomes less reflective, allowing more energy to reach the ground.

At a size parameter of 60, one-scale roughness increases the asymmetry parameter – implying a decrease in cloud reflectance; whereas two-scale roughness reduces it as compared with the smooth crystal – implying an increase in cloud reflectance.

At a size parameter of 100, the asymmetry parameter slightly reduces for one-scale roughness and drops markedly for two-scale roughness. This means that in both cases, but particularly in the two-scale rough case, the reflectance of the cloud increases - causing less radiation to be scattered towards the ground.

Experimental results on rough ice analogues show that the asymmetry parameter would be expected to reduce for observed rough crystals compared to smooth ones [22], so this suggests that two-scale roughness is a better model for real ice crystal roughness.

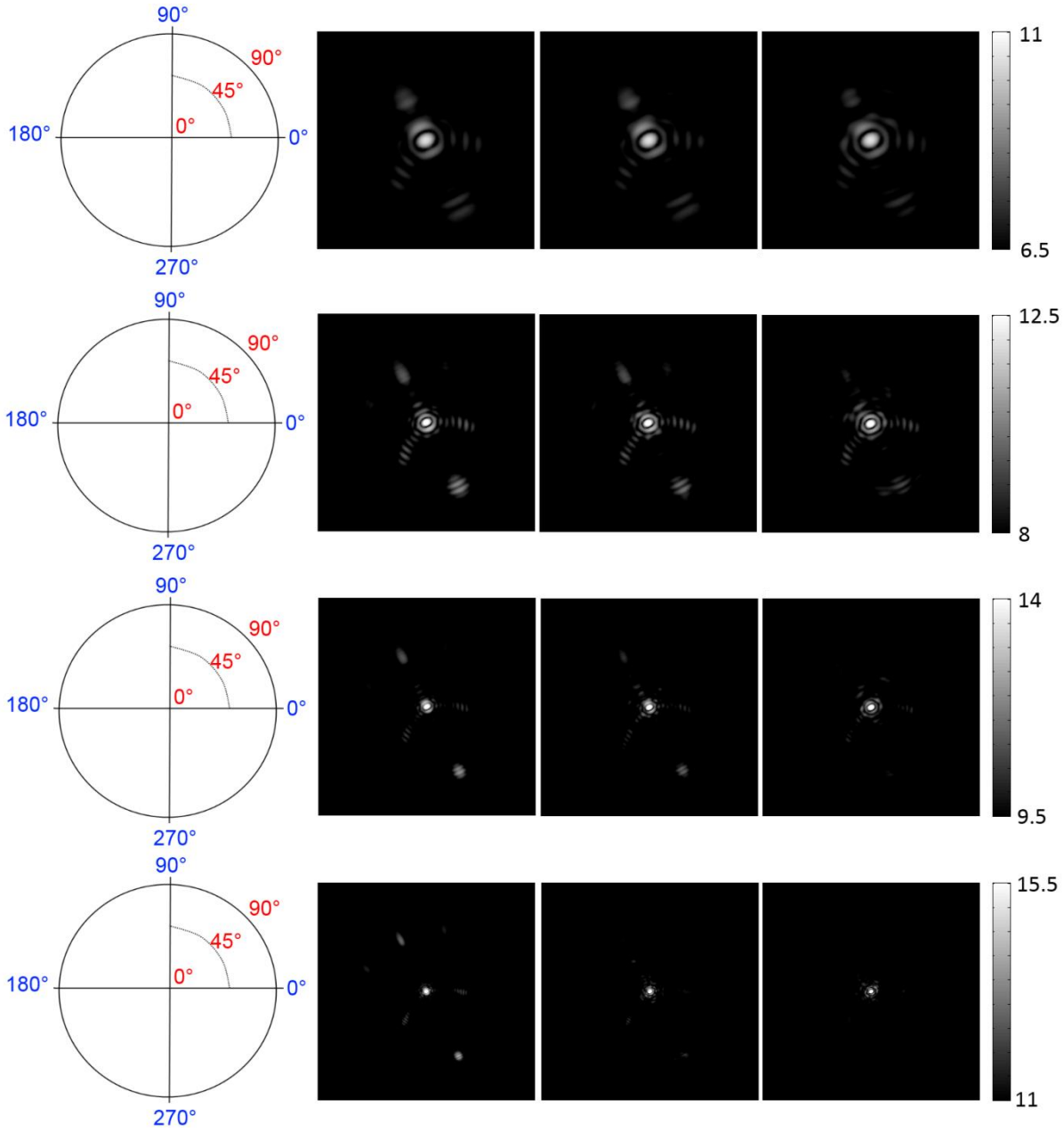

Fig. 5-5: Forward light scattering images for the crystals at one orientation. The orientation considered is where the beam orientation is rotated 30° in the *x*-*y* plane and 30° in the *y*-*z* plane. Different rows represent different crystal size parameters; from top to bottom, the rows represent the 20, 40, 60 and 100 size parameters. Different columns represent different roughnesses; from left to right the rows represent the smooth, one-scale rough and two-scale rough cases. The scale for the scattering pattern can be seen on the left; blue represents azimuthal angles and red represents scattering angles. Since brightness increases with crystal size parameter, grey-scale ranges were varied to best show the features at each size parameter.

Fig. 5-5 shows the forward scattering patterns produced by DDA calculations. As the size parameter increases, the features in the scattering pattern all become more concentrated for the smooth crystal. This is due to the fact that the prism facets act as diffracting apertures; as aperture size increases, diffraction features get smaller.

At a size parameter of 20, very little difference can be seen between either of the roughness scale crystals and the smooth crystal. This is due to the roughness mostly being smaller than the wavelength, and can be seen in the phase function in Fig. 5-3.

At a size parameter of 40, a little difference can be seen for the one-scale rough case for the feature with an azimuth of 120° and a scattering angle of 45°, with respect to the smooth crystal. For the two-scale rough case, changes are apparent at large scattering angles - at an azimuth of 300° the features spread out, whereas at an azimuth of 120° the features have almost disappeared. Also, the two-scale rough pattern shows more deviation from the smooth pattern at low scattering angles than for the one-scale rough crystal at the same azimuth.

At a size parameter of 60, the one-scale rough pattern shows slight differences to the smooth pattern at low scattering angles, particularly the feature at an azimuth of 120° and a scattering angle of 45°. Intensity is reduced for larger scattering angles. The two-scale rough pattern shows more deviation, particularly at larger scattering angles where the features almost disappear.

At a size parameter of 100, the one-scale rough pattern clearly shows a speckle pattern – this is a result of the pristine crystal's diffraction features being broken up by interference modified by the rough surface. At larger scattering angles, brightness is much reduced for all features. For the two-scale rough pattern, all features beyond very small scattering angles have disappeared; only the feature at the centre, caused by external diffraction, remains.

Overall, it can be seen that roughness spreads out the intensity of features in the forward scattering pattern compared to smoothness, apart from the direct forward

scattering peak. Two-scale roughness has a larger effect than one-scale roughness, and the effect is increased as crystal size parameter increases.

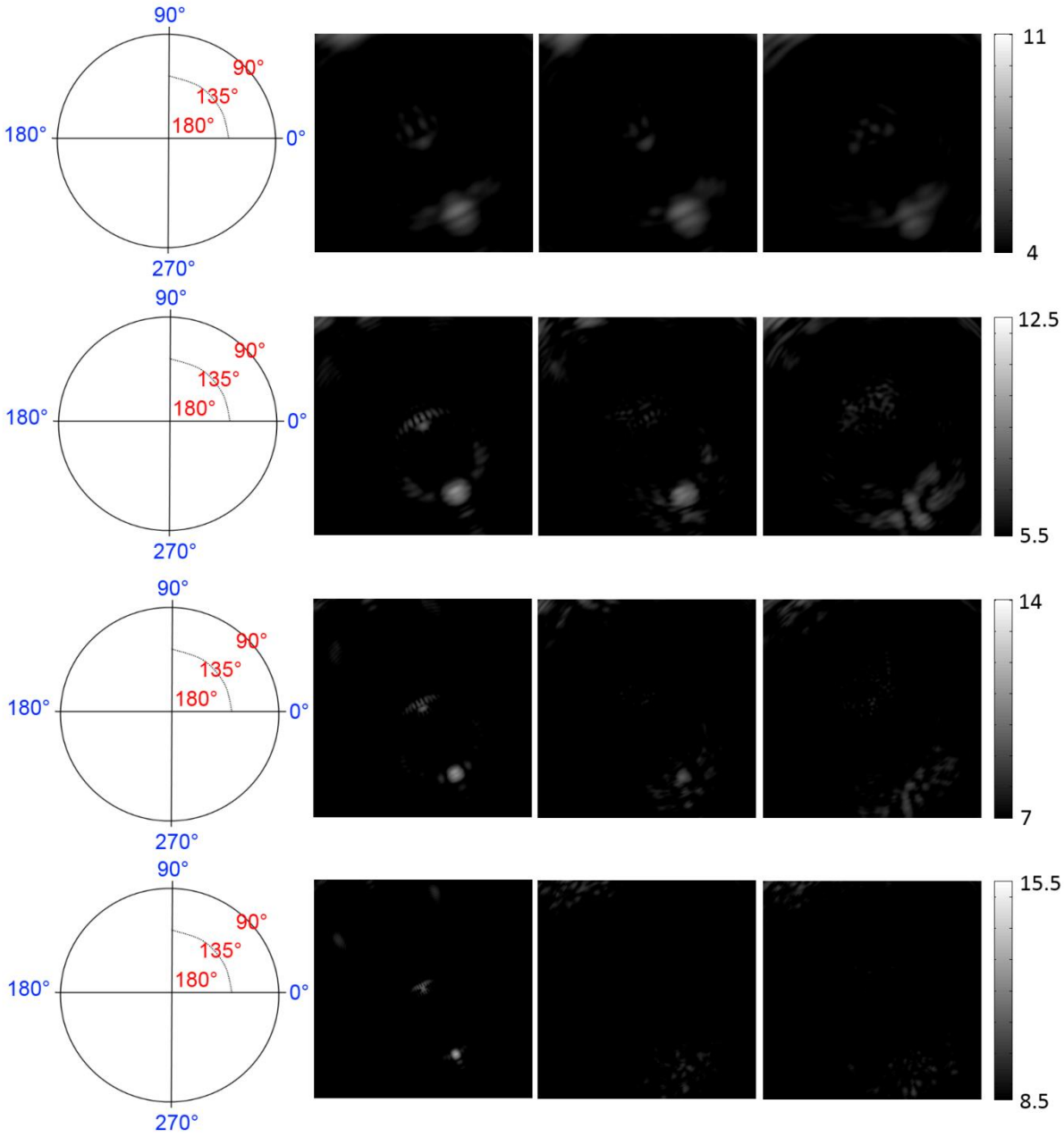

Fig. 5-6: Backward light scattering images for the crystals at one orientation. The orientation considered is where the beam orientation is rotated 30° in the *x*-*y* plane and 30° in the *y*-*z* plane. Different rows represent different crystal size parameters; from top to bottom, the rows represent the 20, 40, 60 and 100 size parameters. Different columns represent different roughnesses; from left to right the rows represent the smooth, one-scale rough and two-scale rough cases. The scale for the scattering pattern can be seen on the left; blue represents azimuthal angles and red represents scattering angles. Since brightness increases with crystal size parameter, grey-scale ranges were varied to best show the features at each size parameter.

Fig. 5-6 shows the backward scattering patterns produced by DDA calculations. As size parameter increases, features become more concentrated for the smooth crystal, as discussed for the forward scattering hemisphere.

At a size parameter of 20, only slight smudging out of the features from the smooth case can be seen in the one-scale rough scattering pattern. This is slightly more pronounced for the two-scale rough crystal.

At a size parameter of 40, one-scale roughness causes the sideward scattering features to be smeared out and features near the centre to almost disappear. For two-scale roughness, there is no single backscattering peak but features near the centre are broken up into speckle. Near the edges, features are smeared out.

At a size parameter of 60, one-scale roughness causes features near the centre to disappear; features near the edges are smeared out a lot. Two-scale roughness results in some speckle near the centre and causes all the features near the edges to be turned into speckle.

At a size parameter of 100, in both roughness cases none of the features from the smooth crystal pattern remain; only speckle can be seen.

In general, roughness smudges out, blurs and removes features that are prominent in the backward scattering pattern of the pristine crystal. Two-scale roughness shows more of an effect than one-scale roughness, and the effect increases as the crystal size parameter is increased.

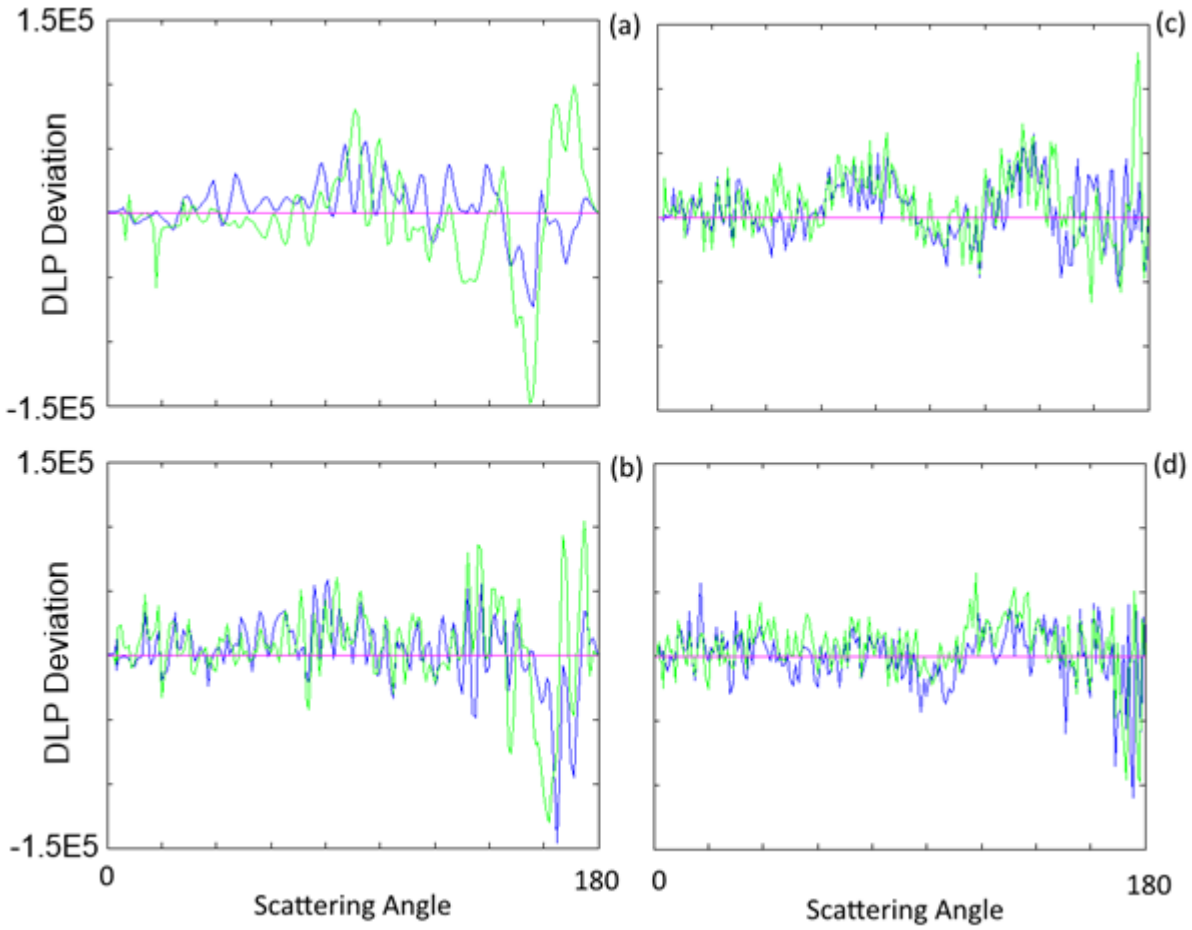

Fig. 5-7: Graphs showing the difference between average degree of linear polarization for a smooth crystal and its rough counterparts of the same size and at the same orientation. One-scale rough (blue) and two-scale rough (green) crystal results, both with the smooth results subtracted, are shown. The orientation considered is where the crystal is rotated 30° in the *x*-*y* plane and 30° in the *x*-*z* plane. (a) has a size parameter of 20, (b) has a size parameter of 40, (c) has a size parameter of 60 and (d) has a size parameter of 100.

Fig. 5-7 shows azimuthally-integrated graphs of the value of degree of linear polarization (DLP). This is defined as $-p_{12}/p_{11}$. For the largest size parameter, comparison between DLP and the corresponding phase function graph from Fig. 5-3 shows that the largest changes in DLP introduced by roughness are in similar angular positions to the peaks in the phase function for the smooth crystal that get removed by roughness- for example, this can be consistently seen for the scattering angle range 160°-180°.

For the size parameter of 20, it can be seen that DLP increases for both roughness scales around 90°, which in both cases corresponds with an increase in intensity (see Fig. 5-3).

One-scale roughness reduces the DLP value in the range 140-180°. Two-scale roughness causes a reduction between 120° and 160°, and an increase between 160° and 180°.

For the size parameter of 40, it can be seen that one-scale roughness causes a large decrease between 155° and 175° and a slight increase around 60°. Two-scale roughness again causes a slight increase around 60. It also causes a noticeable increase in the DLP value at 135°, a large drop between 155° and 165° and a large increase between 165° and 175°.

For the size parameter of 60, it can be seen that one-scale roughness causes a drop in DLP at 40°, an increase in the range 60°-90°, drops at 100° and 170°, and an increase around 140°. Two-scale roughness causes an increase at 40°, an increase in the range 60°-90°, an increase around 140°, a decrease around 160° and a large increase at 175°.

For the size parameter of 100, it can be seen that one-scale roughness causes a slight increase at 40°, a decrease around 100° a slight increase in the range 120°-150° and a large drop in the range 165°-180°. Two-scale roughness causes a higher increase at 40°, a slight decrease around 100°, an increase in the range 120°-160° and a large decrease between 170° and 180°.

Overall, it appears that changes in the degree of linear polarization can be observed for all size parameters and for both roughness types as compared to pristine crystals. The effect is stronger for two-scale roughness than for one-scale roughness, particularly for backward scattering and smaller size parameters.

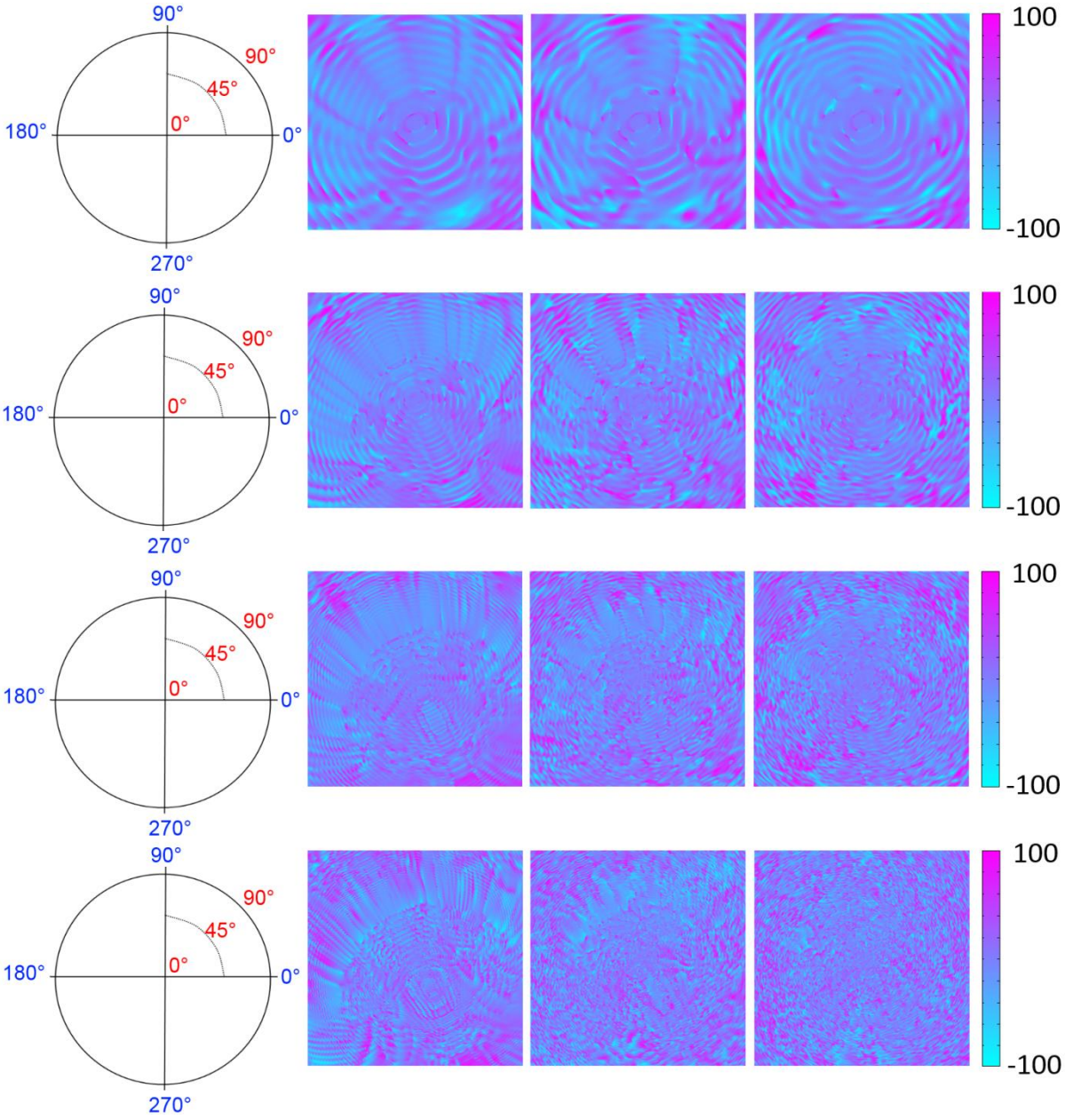

Fig. 5-8: Forward degree of linear polarization images for the crystals at one orientation. The orientation considered is where the beam is rotated 30° in the *x*-*y* plane and 30° in the *y*-*z* plane. Different rows represent different crystal size parameters; from top to bottom, the rows represent the 20, 40, 60 and 100 size parameters. Different columns represent different roughnesses; from left to right the rows represent the smooth, one-scale rough and two-scale rough cases. The scale for the scattering pattern can be seen on the left; blue represents azimuthal angles and red represents scattering angles. Since brightness increases with crystal size parameter, grey-scale ranges were varied to best show the features at each size parameter.

Fig. 5-8 shows the degree of linear polarization patterns for the forward scattering hemisphere, derived from DDA calculations. It can be seen that light with a scattering angle of below approximately 20° has little polarization; this is due to it being dominated by external diffraction. The effects of the external diffraction can also be seen at and near the centre of the images in Fig. 5-5 and at scattering angles below 20° in Fig. 5-3. Features in the 2D forward scattering intensity patterns cannot be readily identified in these patterns, however it can be seen that the patterns are all oriented towards the same azimuth of 125°.

Patterns of high and low polarisation can be seen for scattering by smooth crystals that vary based on both azimuthal angle (especially for smaller size parameters) and scattering angle. The azimuthal variation is noticeably distorted by the introduction of roughness (particularly two-scale roughness) for all size parameters considered; this effect increases as size parameter increases. The scattering angle variation only becomes noticeably distorted at size parameter 100 – this is more noticeable for two-scale roughness.

At a size parameter of 20, roughness slightly smears out the azimuthal pattern in the smooth case, particularly at scattering angles between 20° and 90°. It also slightly increases polarization in regions that are unpolarized for the smooth crystal. These effects are more prominent for two-scale roughness than for one-scale roughness.

At a size parameter of 40, one-scale roughness causes much distortion in the azimuthal pattern as compared to smoothness; this is apparent for the whole scattering angle range considered (0°-90°), and is even more pronounced for two-scale roughness. Both increase polarization in regions that are unpolarized for the smooth crystal.

At a size parameter of 60, one-scale roughness causes great distortion to the azimuthal pattern. This is even more pronounced for the two-scale rough crystal pattern.

At a size parameter of 100, one-scale roughness causes a huge distortion in the azimuthal pattern and some loss of the scattering angle pattern at scattering angles up

to 45°. For two-scale roughness, the azimuthal pattern can no longer be seen and the scattering angle pattern completely disappears.

In general, roughness removes azimuthal dependence and, for larger crystals, scattering angle dependence from the degree of linear polarization pattern for forward scattering and redistributes polarisation. Two-scale roughness has more of an effect than one-scale roughness, as compared to smoothness. The effect of roughness increases as size parameter increases.

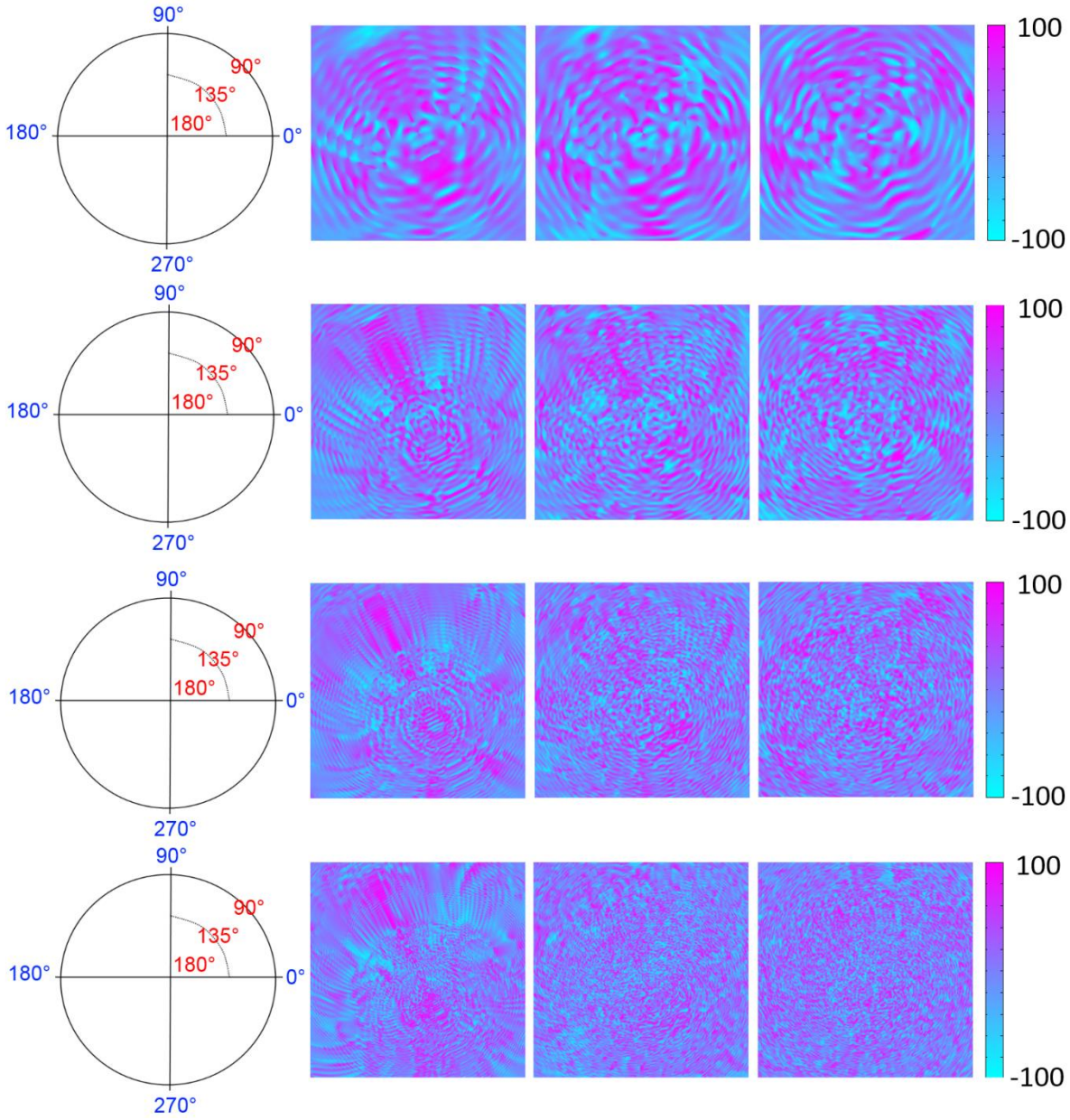

Fig. 5-9: Backward degree of linear polarization images for the crystals at one orientation. The orientation considered is where the beam is rotated 30° in the *x*-*y* plane and 30° in the *y*-*z* plane. Different rows represent different crystal size parameters; from top to bottom, the rows represent the 20, 40, 60 and 100 size parameters. Different columns represent different roughnesses; from left to right the rows represent the smooth, one-scale rough and two-scale rough cases. The scale for the scattering pattern can be seen on the left; blue represents azimuthal angles and red represents scattering angles. Since brightness increases with crystal size parameter, grey-scale ranges were varied to best show the features at each size parameter.

Fig. 5-9 shows the degree of linear polarization patterns for the backscattering hemisphere, derived from DDA calculations. As with DLP for forward scattering, high and low polarisation can be seen to alternate with both scattering angle and azimuthal angle for smooth crystals.

At a size parameter of 20, one-scale roughness distorts the azimuthal variation, as compared to the smooth crystal. This effect is increased for two-scale roughness, which also shows some distortion in the scattering angle polarization variation at scattering angles greater than 160°.

At a size parameter of 40, one-scale roughness greatly distorts the azimuthal pattern with respect to the pattern for the smooth crystal and causes some distortion in the scattering angle polarization pattern at angles above 150°. Two-scale roughness causes the azimuthal pattern to become even more distorted and distorts the scattering angle polarization pattern at angles greater than 135°.

At a size parameter of 60, one-scale roughness distorts most of the azimuthal pattern to the extent that it is unrecognizable and disrupts the scattering angle pattern at angles above 135°. Two-scale roughness completely distorts the azimuthal pattern and warps the scattering angle pattern for angles greater than 120°.

At a size parameter of 100, neither roughness scale shows any pattern in the backward azimuthal range any longer. One scale roughness shows no pattern in scattering angle DLP above 100°; two-scale roughness shows no pattern in scattering angle DLP at all for backward scattering.

In general, roughness removes structure in the degree of linear polarization pattern for backward scattered light and redistributes polarisation. This is more pronounced for larger crystals and for two-scale roughness. Roughness has a greater distorting effect on DLP for backward scattering than forward scattering; this is also true for the phase function (Fig. 5-3).

5.2 Light scattering experiments on smooth and rough ice analogues

Light scattering experiments were carried out on smooth and rough hexagonal ice analogue crystals using the SID3 probe. Data from these experiments was compared with DDA (for small columns of length 20 μm and hexagon radius 10 μm) and RTDF (for large columns of length 100μm and hexagon radius 10 μm as well as small columns) modelling results, all at a wavelength of 532nm, to look for similarities and differences between these two approaches. This is necessary because DDA is the more accurate of the two methods, but DDA-produced data is not available for the large column due to computational restrictions. The DDA data will be used to check the accuracy of RTDF data for a pristine crystal of the same size parameter. This will be used to inform a comparison of ice analogue crystal used in the SID3 experiments with RTDF data for a pristine crystal of the same size parameter.

Two different crystal orientations were considered; one with a crystal prism facet perpendicular to the beam, and one with the edge between two crystal prism facets perpendicular to the beam (Fig. 5-10).

Phase functions were not calculated; rather, the maximum brightness within an azimuthal angular range (Fig. 5-11) was considered. This is because SID3 has a limited scattering angle range, meaning the phase function cannot be found as it would need to be normalized for all possible scattering angles. Modelling results are also examined using this method to enable comparison with SID3.

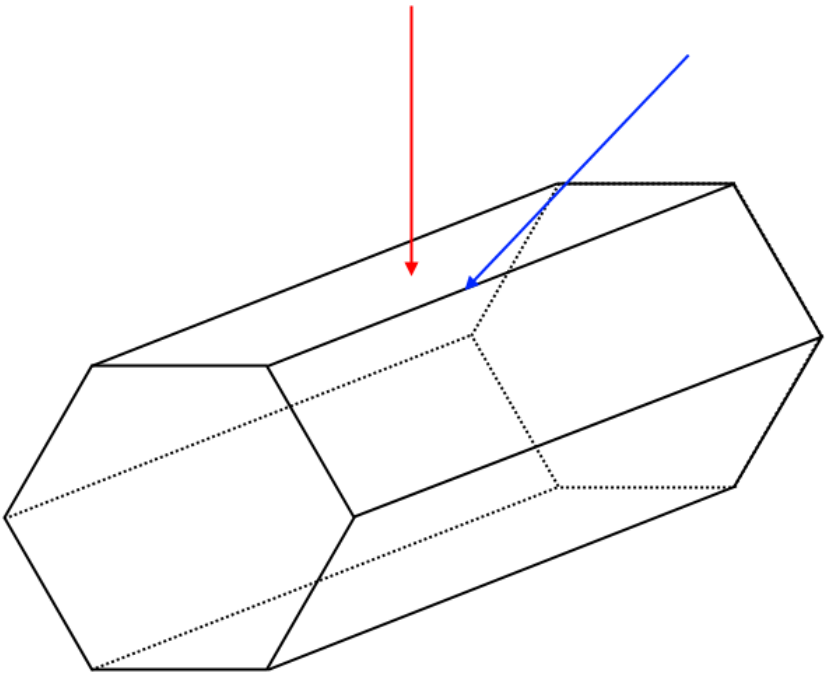

Fig. 5-10: The orientations considered. Blue shows the beam direction relative to the crystal for edge-on results; red shows the beam direction relative to the crystal for facet-on results.

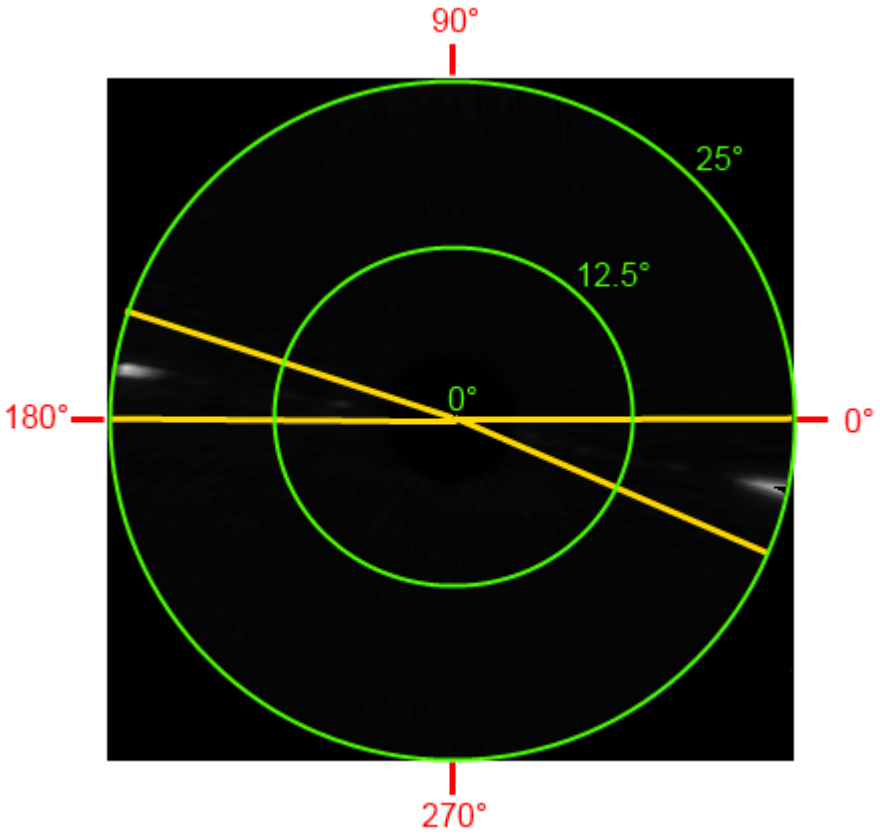

Fig. 5-11: Image showing the azimuthal angular ranges for an example SID3 scattering pattern. Azimuthal angles are shown in red. Two azimuthal ranges can be seen marked in yellow, both centred on the opposite "arms" of the scattering pattern. The scattering angle is shown in green.

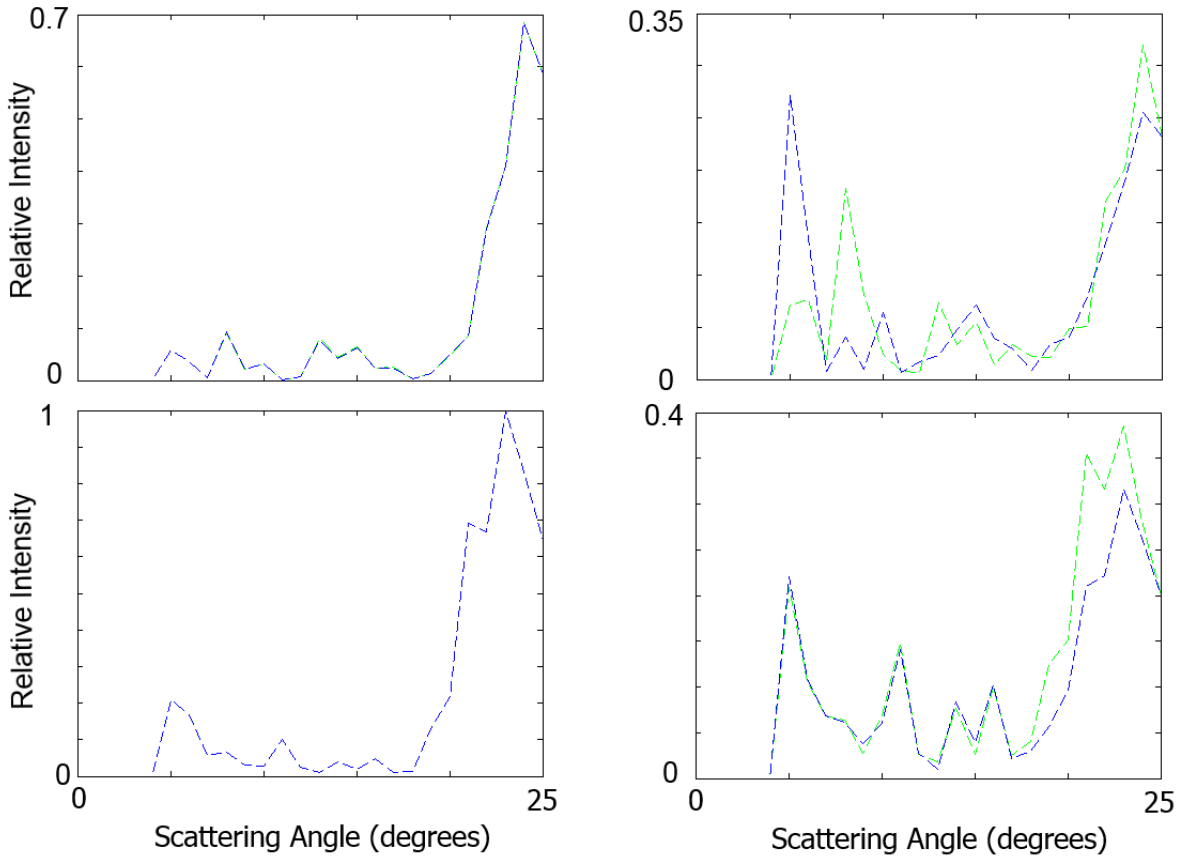

Fig. 5-12: Graphs of relative (normalised) intensity as a function of scattering angle computed using DDA for two-scale rough (right column) and smooth (left column) crystals at a facet-on orientation (top row) and an edge-on orientation (bottom row). In all cases, the crystal is of length 20μm and hexagon radius 10μm. These graphs are not phase functions – they instead show the maximum intensity within a defined azimuthal range for each scattering angle; the two azimuthal angles considered are represented separately on the graph by different colours.

It can be seen from Fig. 5-12 that roughness reduces the intensity of the 22° halo for both edge-up and facet-up orientations and that the intensity is higher for the edge-up orientation than the facet-up orientation.. For the facet-up orientation, the diffraction peaks move and no longer appear at the same scattering angles for the two opposite azimuths (represented by different colours - see (Fig. 5-11) for an explanation of opposite azimuths), and the intensity of them increases. For the edge-up orientation, the diffraction pattern only slightly changes; the intensity is very similar and the scattering angles at which the peaks appear are unchanged.

Comparing the graphs for the smooth crystals with the top row of Fig. 5-13 shows how RTDF and DDA differ; RTDF appears to overestimate the intensity of diffraction peaks, particularly below 10°. For upward facing facets, it estimates the halo peak to be broader than it really is, but it does a good job of replicating the positions of diffraction peaks and the halo peak.

Bearing these comments in mind, comparisons can be drawn between RTDF results for a smooth crystal of length 100μm and hexagon radius of 10μm, and SID3 scattering results for a smooth ice analogue of the same size. Also, the way in which roughness changes the experimental results for ice analogues will be compared with the way roughness changes the DDA results.

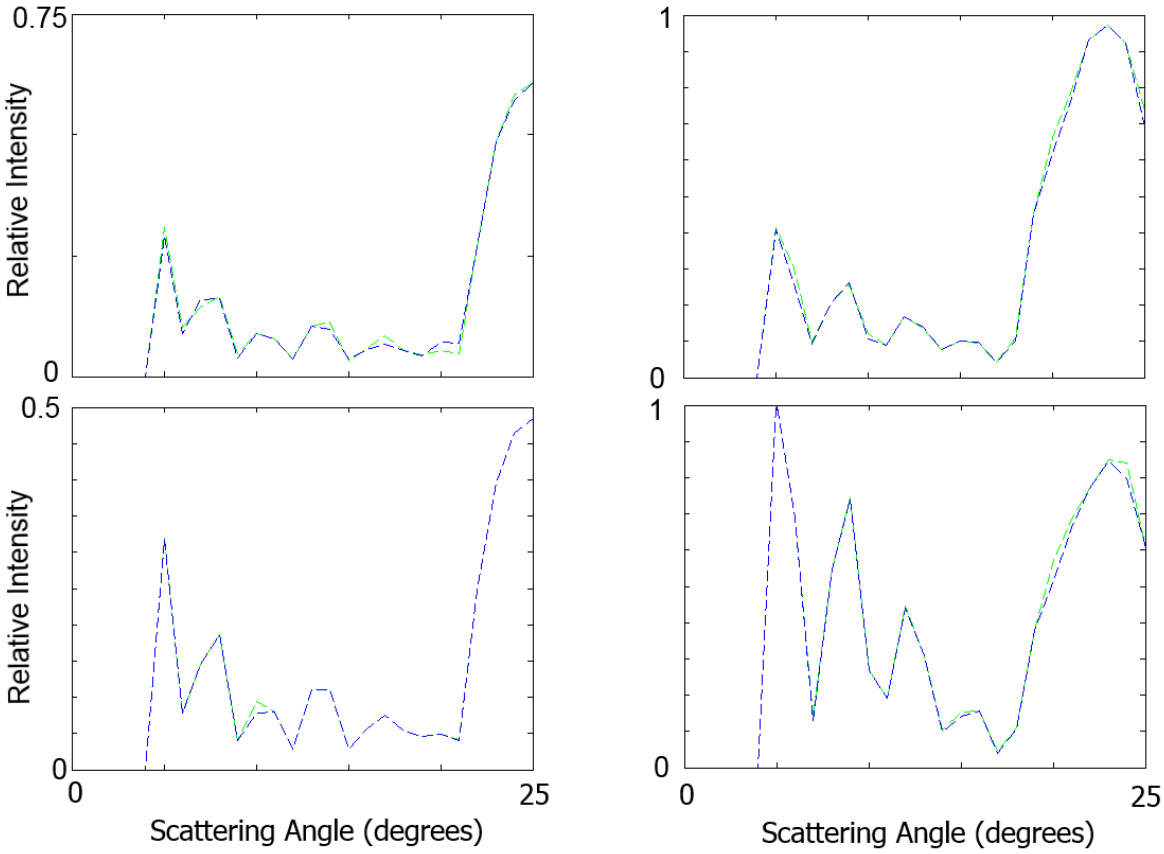

Fig. 5-13: Graphs of relative (normalised) intensity as a function of scattering angle computed using RTDF for crystals at a facet-on orientation (left column) and an edge-on orientation (right column). The top row show results for a crystal of length 20μm and hexagon radius 10μm; the bottom row shows results for a crystal of length 100μm and hexagon radius 10μm. In all cases, simulations were performed using 200000 rays. The green and blue lines represent opposite azimuthal "arms" (see Fig. 11). The smaller crystal is normalised relative to the DDA results, whereas the larger crystal is normalised relative to the SID3 results (see Fig. 5-14 for these) – relative to the smaller crystal, the larger crystal is much brighter.

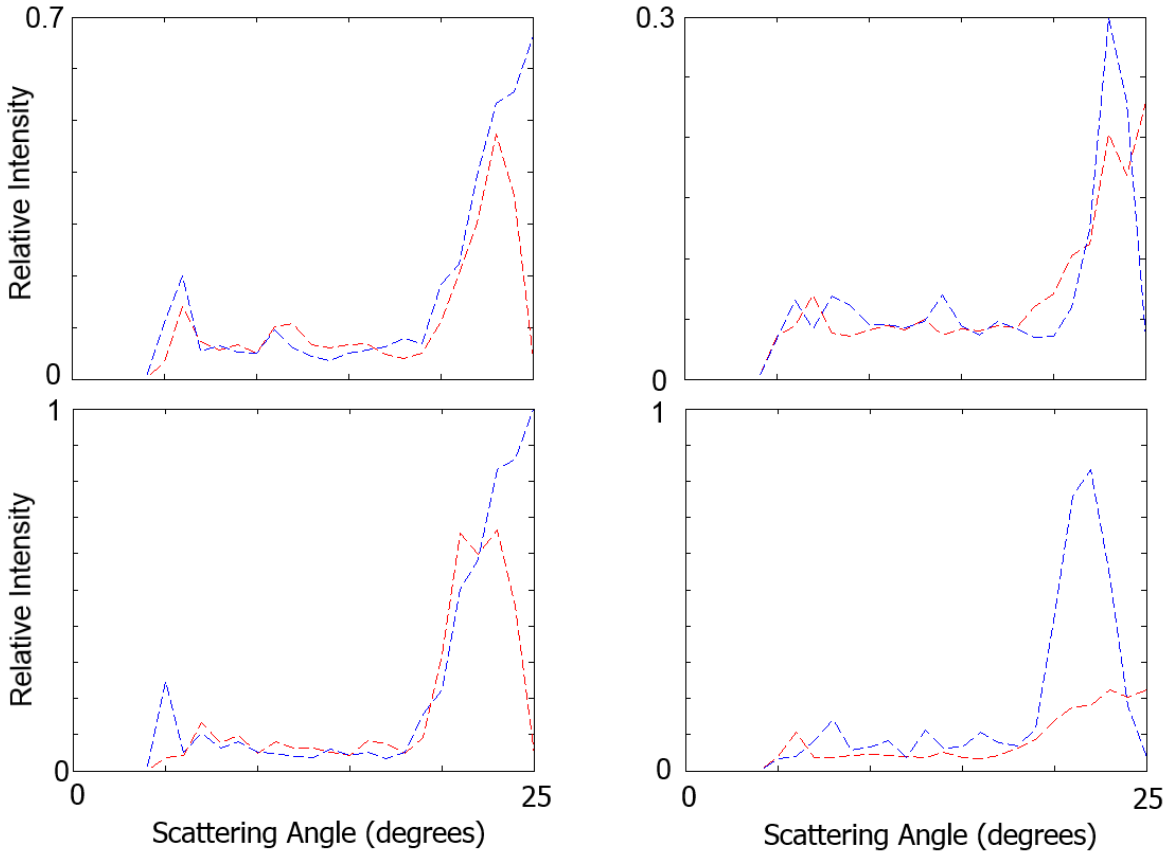

Fig. 5-14: Graphs showing relative (normalised) intensity as a function of scattering angle for ice crystal analogues of length 100μm and hexagon radius 10μm, captured using SID3. Smooth crystals are in the left column and rough ones in the right column; the top row shows crystals with a prism facet facing up and the bottom row shows crystals with the edge facing up. Blue and red represent the two opposite azimuths.

Comparing the bottom row of Fig. 5-13 with the left column of Fig. 5-14, it can be seen that a good agreement is achieved between them regarding the position of the halo peak, however this peak extends into a higher scattering angle for one of the azimuths of the experimental result. This is likely due to a crystal alignment problem. The other azimuth shows, as expected from the earlier comparison, that the halo peak is less broad than RTDF would suggest. Diffraction peaks are less apparent for the experimental results than the modelling results, but the peaks at 5° and 10° for facet-up scattering, as well as the halo peak can be discerned. The reduction in the brightness of the diffraction peaks shown by the previous comparison between RTDF and DDA enables the diffraction peaks to be better matched with the experimental results, particularly in the case of the edge facing the beam.

It can be seen from Fig. 5-14 that, for the facet-up case, roughness reduces the overall intensity as compared to the smooth crystal graph; it can also be seen that the positions of the diffraction peaks are different for the two different azimuths. This agrees with what would be expected, based on Fig. 5-12.

For the edge-up case, overall intensity is again reduced and the diffraction peaks shift; as would be expected when comparing with Fig. 5-12. It can be seen that there is a big difference between the results for the two different azimuths; this is likely due to the difference in the quality of the roughening between the two facets that collect the incident beam (Fig. 4-25).

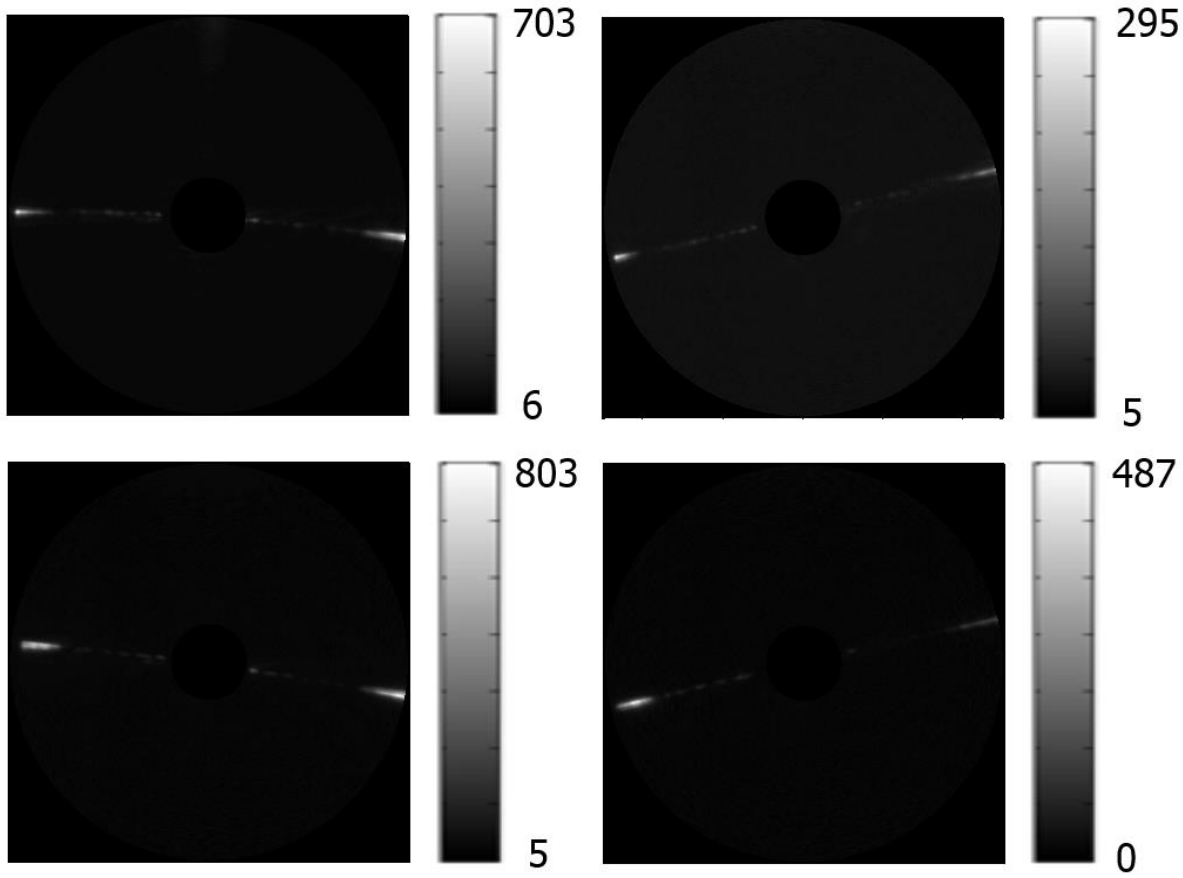

Fig. 5-15: 2D scattering patterns from an ice analogue crystal of length 100μm and hexagon radius 10μm, captured using SID3. Smooth crystals are in the left column and rough ones in the right column; the top row shows crystals with the facet facing up and the bottom row shows crystals with the edge facing up. Scattering angle varies linearly between 0° and 25°, and is represented by the radial distance from the centre of the scattering pattern. Azimuthal angle varies as shown in Fig. 5-11. The scale for these patterns can be seen in Fig. 5-11.

Fig. 5-15 shows 2D scattering patterns for rough and smooth ice analogues, derived from experimental results. This confirms the overall drop in intensity in the investigated angular region, in both the facet-up and edge-up cases. The difference between the two azimuths in the rough edge-up result is clear to see, especially when compared with the smooth result.

6. Conclusions

Phase function, asymmetry parameter and degree of linear polarisation results were computed for smooth, one-scale Gaussian rough and two-scale Gaussian rough crystals of various orientations and size parameters of 20, 40, 60 and 100. This was done using the DDA method; the computations were done at the University of Helsinki by Antti Penttilä.

Phase functions show that the effects of roughness increase with particle size in the investigated range; little change is seen (compared to the smooth crystal case) for rough crystals at a size parameter of 20; however at a size parameter of 100, most of the peaks and troughs (apart from the peak at 0°) have disappeared. These effects are more potent, and appear at lower size parameters, for crystals with two-scale roughness.

Asymmetry parameters are changed inconsistently by roughness compared to those for smooth crystals at low size parameters, but the asymmetry parameter is consistently lower at higher size parameters for two-scale roughness than for the smooth crystal. The latter agrees with previous results [22] and suggests that two-sale roughness is the better model for real roughness.

2D scattering patterns show that roughness causes characteristic features related to smooth crystals to blur and disappear at large enough size parameters for both forward and backward scattering. The effect becomes stronger as the crystal size parameter increases, and is more apparent for two-scale roughness than for one-scale roughness. With two-scale roughness at a size parameter of 100 considered, only the external diffraction peak with some speckle remains for forward scattering; for backward scattering, only speckle remains.

Degree of linear polarisation 2D patterns show that, compared to results for smooth crystals, roughness disrupts patterns in scattering angle and azimuth and makes light more polarised (although the opposite also occurs in some angular regions) for forward and backward scattered light. Backward scattering is affected more than forward

scattering; the same effect is seen for intensity. These effects become more pronounced for larger crystals and for two-scale roughness.

The surface of an ice analogue was roughened using focused ion beam milling (FIB). This was not an ideal reproduction of a rough ice analogue; only three of the facets were roughened, the roughness had deviations from the Gaussian random pattern, two of the facets had only a small proportion of their surfaces roughened and the depth of the roughness had to be compromised on. This is largely a result of the ice analogue crystal needing to be fibre-mounted in order to enable light scattering measurements; ion beam irradiation-induced charging on the crystal was a particular problem, and overall this was pushing right at the limit of what FIB could achieve. The amount of milling that could be done was also limited by the availability of the FIB/SEM machine and operator. Nevertheless, an interesting result was found from SID measurements for the crystal orientation which had the edge between two prism facets pointing up – one of these prism facets was relatively well-roughened, and so greatly reduced the intensity of the halo peak at the azimuth it contributed to, as well as removing most of the higher order external diffraction peaks, whereas the other facet doesn't show this.

Overall, it can be concluded that Gaussian roughness with the investigated parameters removes features from the phase function as compared with smooth hexagonal prisms and reduces the asymmetry parameter, as long as the roughness features are horizontally at least as large as the wavelength – this appears to begin happening noticeably at a size parameter of 40. Larger roughness features cause more deviation in light scattering from scattering observed for smooth crystals, but the most effective roughness model investigated here takes account of both these large features and features whose size is closer to that of the wavelength.

Several possibilities exist for future research based on the work presented in this thesis. The modelling work could be extended by performing simulations at more orientations and by looking at how DLP and phase function change when only the standard deviation

and only the correlation length of a crystal is varied. Creation of a bounding box method for RTDF would enable simulations for larger size parameters. The crystal creation routines could be modified to enable the creation of extra crystal geometries such as rosettes and aggregates. The roughness parameters could be made more realistic by looking at the roughness of ice grown inside an SEM using a method such as photogrammetry to get the three dimensional data. Rough ice analogue manufacturing would be improved by making the attachment between the mount and the crystal. One method for achieving this could be to replace the glass fibre with a very thin metal wire coated in another metal which has a low melting point (lower than that of the crystal) – the crystal would be fixed in place by heating up the area and then allowing it to cool.

7. Bibliography

[1] Contribution of Working Group I to the Fifth Assessment Report of the Intergovernmental Panel on Climate Change (2013)

[2] Thomas C.D., et al (2004) Extinction Risk from Climate Change, Nature 427, 145-148

[3] Rosenzweig C, Parry M.L. (1994) Potential Impact of Climate Change on World Food Supply, Nature 367, 133-138

[4] Zhang J, Crowley T.J. (1989) Historical Climate Records in China and Reconstruction of Past Climates, J. Climate 2, 833-849

[5] Rixen M., et al (2005) The Western Mediterranean Deep Water: A Proxy for Climate Change, Geophys. Res. Lett. 32, L12608, doi:10.1029/2005GL022702

[6] Zhang Q-B, et al (2003) A 2,326-Year Tree-Ring Record of Climate Variability on the Northeastern Qinghai-Tibetan Plateau, Geophys. Res. Lett. 30(14), 1739, doi:10.1029/2003GL017425

[7] Petit J.R., et al (1999) Climate and Atmospheric History of the Past 420,000 Years From the Vostok Ice Core, Antarctica, Nature 399, 429-436

[8] Wilby R.L., Wigley T.M.L. (1997) Downscaling General Circulation Model Output: A Review of Methods and Limitations, Progress Phys. Geog. 21, 530-548

[9] Colman, R. (2003) A Comparison of Climate Feedbacks in General Circulation Models, Climate Dynamics, 20, 865-873

[10] Yang P., Takano Y., Liou K.N. (1999) In: Mishchenko M.I., Hovenier J.W., Travis L.D., Light Scattering by Nonspherical Particles (New York: Academic Press), 417-449

[11] Baran A.J. (2004) On the Scattering and Absorption Properties of Cirrus Cloud, J. Quant. Spect. Rad. Trans. 89, 17-36

[12] Ulanowski Z., et al (2010) Light Scattering by Ice Particles in the Earth's Atmosphere and Related Laboratory Measurements, Electromagnetic and Light Scattering Conference XII. Web page, accessed 03/06/2011: http://www.helsinki.fi/els/articles/74/

[13] Mitchell D.L., et al. (2008) Impact of Small Ice Crystal Assumptions on Ice

Sedimentation Rates in Cirrus Clouds and GCM Simulations, Geophys. Res. Lett. 35, L09806, doi:10.1029/2008GL033552

[14] Waliser D.E., et al. (2009) Cloud Ice: A Climate Model Challenge with Signs and Expectations of Progress, J. Geophys. Res. 114, D00A21, doi:10.1029/2008JD010015

[15] Cole, B.H., et al. (2014) Ice particle habit and surface roughness derived from PARASOL polarization measurements, Atmos. Chem. Phys., 14, 3739–3750,

[16] Macke, A., et al. (1996) Single Scattering Properties of Atmospheric Ice Crystals. J. Atmos. Sci., 53, 2813–2825

[17] Lui, C., Panetta R.L., Yang, P. (2013) The effects of surface roughness on the scattering properties of hexagonal columns with sizes from the Rayleigh to the geometric optics regimes. J. Quant. Spect. Rad. Trans., 129, 169-185

[18] Wendisch, M., et al. (2007) Effects of ice crystal habit on thermal infrared radiative properties and forcing of cirrus, J. Geophys. Res., 112, *D08201*

[19] Mischenko, M.I. et al (1999) How big should hexagonal ice crystals be to produce halos? Applied Optics 38, 9

[20] Nazaryan, H., et al (2009) Global characterization of cirrus clouds using CALIPSO data, J. Geophys. Res., 113, *D16211*

[21] Wylie, D.P., et al (1994) Four Years of global Cirrus Cloud Statistics Using HIRS. Journal of Climate, 7, 12, 1972-1986

[22] Ulanowski, Z., et al. (2006) Light scattering by complex ice-analogue crystals, J. Quant. Spect. Rad. Trans. 100(1-3), 382-392

[23] Muinonen K., Saarinen K. (2000) Ray Optics Approximation for Gaussian Random Cylinders, J. Quant. Spect. Rad. Trans. 64, 201-218

[24] McCall D.S. (2010) Measurement and Modelling of Light Scattering by Small to Medium Size Parameter Airborne Particles, PhD thesis, University of Hertfordshire

[25] Ulanowski, Z. et al. (2012) Rough and irregular ice crystals in mid-latitude clouds, 16th Int. Conf. Clouds Precipitation, Leipzig

[26] Yurkin, M.A. and Hoekstra, A.G. (2011) The discrete-dipole-approximation code ADDA: capabilities and known limitations, J. Quant. Spect. Rad. Trans. 112, 2234-2247

[27] Conolly, P.J., et al. (2007) Calibration of the Cloud Particle Imager Probes Using Calibration Beads and Ice Crystal Analogs: The Depth of Field. J. Atmos. Ocean. Tech. 24, 1860-1879

[28] Pfalzgraff, W.C., Hulscher, R.M., Neshyba, S.P. (2010) Scanning Electron Microscopy and Molecular Dynamics of Surfaces of Growing and Ablating Hexagonal Ice Crystals, Atmos. Chem. Phys. 10, 2927-2935

[29] Wagner, R., et al. (2009) A Review of Optical Measurements at the Aerosol and Cloud Chamber AIDA, J. Quant. Spect. Rad. Trans. 110, 930-949

[30] Ulanowski, Z. et al. (2003) Scattering of Light from Atmospheric Ice Analogues. J Quant. Spect. Rad. Trans. 79-80C, 1091-1102

[31] Dowling, D.R., Radke, L.F. (1990) A summary of the Physical Properties of Cirrus Clouds. J. Appl. Meteor. 29, 970-978

[32] Waterman, P.C. (1971) Symmetry, Unitarity, and Geometry in Electromagnetic Scattering, Physical Review D 3, 825

[33] Purcell, E.M. and Pennypacker, C.R. (1973) Scattering and Absorption of Light by Nonspherical Dielectric Grains, Astrophysical Journal 186, 705-714

[34] Hesse, E., et al. (2009) Application of RTDF to Particles with Curved Surfaces. J. Quant. Spect. Rad. Trans. 110, 1599-1603

[35] Van de Hulst, H.C. (1981) Light Scattering by Small Particles. New York, United States: Dover Publications

[36] Hecht, E. (2002) Optics: Fourth Edition. San Francisco, United States: Addison Wesley

8. Acknowledgements

Many thanks are due to the following people:

Dr E. Hesse (Principal Supervisor)

Dr Z.J. Ulanowski (Second supervisor)

Mr K. Henman, for help in the laboratory, particularly with gold sputtering of fibres

My other colleagues in the CAIR group: Dr E. Hirst, Dr R. Greenaway, Prof. P. Kaye, Dr C. Stopford, Dr W. Stanley, Georg Ritter, Laurence Taylor, Paolo Dandini, Jenna Thornton and Paul Burns.

Dr. Antti Penttilä, for performing light scattering simulations using the DDA code.

Dr. Emmanuel Brousseau, for performing AFM scans and helping with the rough ice analogue.

Dr. Hassan Hirshy, for his work on the rough ice analogue.

My family; both those who passed away during the course of this project and those who remain.

Funding for this work was provided by the Natural Environment Research Council in the form of a PhD studentship.

9. Appendix

9.0 Introduction to the appendix

Below is the code used to generate a Gaussian random crystal. The process first requires that an input file be created, describing the hexagon radius, number of facets along the hexagon radius, prism facet length, number of facets along the prism facet length, correlation length, standard deviation and an offset value, which tells the code how many facets to offset by and is intended to avoid problems caused by calculating *z(x,y)* at and near *x*=0 & *y*=0. The name used for this file is vals.in, and it looks like this:

```
5.      !length of each hexagon side
7       !number of polygons along each hexagon edge - should be large enough to correctly plot the correllation length
10.     !length of prism surfaces
14      !number of polygons along prism facet length edge - should be large enough to correctly plot the correllation length
1.0     !correllation length
0.3     !standard deviation
100     !number of facets to offset by - to deal with corner effects for each surface
!
!all length values in µm
```

Once this has been created, the required output filename should be set inside the code grfstrip.f90; it should then be compiled and run. This generates the roughened strip described in section 4.1.2.

If more roughness scales are required the file *vals.in* should be edited to change the correlation length and standard deviation. The other values should be left alone – the number of facets along the edges should be large enough to accommodate the smaller of the correlation lengths. The output filename in *grfstrip.f90* must be changed; the code should then be re-compiled and run again. Once this is complete, the source code of *comp.f90* should be edited to point to the correct strip filenames and to produce the required output filename. It should then be compiled and run, producing a strip file in which each z(x,y) is the sum of the same z(x,y) from the input files.

Whether or not a multiple roughness scale is required, the subsequent procedure is the same. The code *crystal.f90* should be edited to point to the output of *grfstrip.f90* (in the case of a single roughness scale) or of *comp.f90* (in the case of multiple roughness scales). The output filename must also be set. This is then compiled and run. The output file will describe a crystal which does not yet have its edges joined together. It will also output a file, *mlvals.in*, which is the same as vals.in except that the Fortran 90 comments are removed, to make the values readable for Matlab.

Finally, the Matlab file *join.m* should be edited so that the input filename is the same as the output filename of *crystal.f90;* the output filename should also be set – this will be the final crystal.
If the crystal produced has a small number of facets (<5000), the Matlab file *srfh.m* can be used to display it – each facet is represented by a wire frame, as in figures 4-2 to 4-6. To do this, the filename of the crystal should be set in the code and it should then be run.

All code used to create and display the crystal is reproduced below. Section 9.1 contains the code for *grfstrip.f90*, section 9.2 contains the code for *comp.f90*, section 9.3 contains the code for *crystal.f90*, section 9.4 contains the code for *join.m* and its associated functions, and section 9.5 contains the code for *srfh.m*.

9.1 grfstrip.f90

```
!Generate a Gaussian random rectangle
!Copyright Chris Collier
!Adapted from Appendix A of JQSRT 64 (2000) 201-218, Muinonen & Saarinen

PROGRAM GRFStrip
IMPLICIT NONE

COMPLEX(8) :: i, zxy, ipKxqKy
REAL(8) :: pi, targetdiff, clen, hlength, pflength, stdev
INTEGER :: nfhl, nfsl, npfw, noffset, nfsw, nl, nw, nfacets, eval, nptsrow, ntp, ntf, szfc
INTEGER, DIMENSION(:,:), ALLOCATABLE :: e1, e2, e3, e4
REAL(8) :: L, K, swidth, lstep, wstep, loffset, woffset, a, xval, ystart
INTEGER :: j1, j2, j3, j4, itlim, dp0, dq0, p, q, pqval, n, nzxy, n2, szedges
REAL(8) :: p1, p1sq, p2, cc, ccp1, ccp2, diff, diffold, zpqr, zpqi, sdzr, sdzi, zx, zy, Kx, Ky, rn, pKx, psq
COMPLEX, ALLOCATABLE, DIMENSION(:,:) :: zpq
INTEGER :: m, m2, m3, nr
REAL(8), DIMENSION(:,:), ALLOCATABLE :: facetcoords
INTEGER, DIMENSION(:), ALLOCATABLE :: lwedge, rwedge, ledges, tpts
REAL :: rnd1, rnd2

!set constants
pi=4.*ATAN(1.)                  !pi
i=CMPLX(0.,1.)                  !set i
targetdiff=10.**(-6.0)  !required difference between correlation function and its expansion
CALL set_rseed()                !seed the random number generator

!read in input values
OPEN(UNIT=3, FILE='vals.in', STATUS='OLD')
READ(UNIT=3, FMT=*) hlength             !hexagon radius
READ(UNIT=3, FMT=*) nfhl                !number of subfacets along hexagon radius
READ(UNIT=3, FMT=*) pflength  !prism facet length
READ(UNIT=3, FMT=*) nfsl                !number of subfacets along prism facet length
READ(UNIT=3, FMT=*) clen                !correlation length
READ(UNIT=3, FMT=*) stdev               !standard deviation
READ(UNIT=3, FMT=*) noffset             !number of subfacets to offset by - to deal with edge effects
CLOSE(3)

!set various things
nfsw=6*nfhl                             !number of subfacets along 6 hexagon radii
nl=nfsl+1                               !number of vertices along prism facet length
nw=nfsw+1                               !number of vertices along 6 hexagon radii
npfw=nfhl+1                             !number of vertices along hexagon radius
nfacets=nfsl*nfsw               !number of subfacets in 6 pr
L=2.*hlength                    !L from Muinonen & Saarinen
K=pi/L                                  !K from Muinonen & Saarinen
swidth=6.*hlength               !"width" of the strip - 6*hexagon radius
ntp=6*(npfw**2-npfw)  !number of vertices in the triangles
szfc=nl*nw+ntp                  !size of the coordinates array
ALLOCATE(facetcoords(szfc,3))
```

```
lstep=pflength/REAL(nfsl)                    !distance between adjacent vertices in x
wstep=swidth/REAL(nfsw)                      !distance between adjacent vertices in y
loffset=REAL(noffset+1)*lstep    !distance from 0 to use as minimum x
woffset=REAL(noffset+1)*wstep!distance from 0 to use as minimum y

OPEN(UNIT=1, FILE='strip.crystal', STATUS='REPLACE')
!OPEN(UNIT=2, FILE='points.dat', STATUS='REPLACE')

!generate the initial rectangle
ALLOCATE(lwedge(nw), rwedge(nw), ledges(7*nl))
n=0
n2=0
DO j1=1+noffset,nw+noffset      !offsets included to avoid boundary effects
        DO j2=1+noffset,nl+noffset
                n=n+1
                facetcoords(n,1)=REAL(j2)*lstep
                facetcoords(n,2)=REAL(j1)*wstep
                facetcoords(n,3)=0.             !set z values as 0 to begin with
                !remember which values describe the left & right of the strip
                IF (j2 .EQ. 1+noffset) THEN
                        lwedge(j1-noffset)=n
                ELSE IF (j2 .EQ. nl+noffset) THEN
                        rwedge(j1-noffset)=n
                END IF
                !remember top & bottom of prism facets
                IF (MOD(j1-noffset-1,nfhl) .EQ. 0) THEN
                        n2=n2+1
                        ledges(n2)=n
                END IF
        END DO
END DO

!determine edges of each prism facet
eval=1+nfsw/6
ntf=12*(eval-1)**2
ALLOCATE(e1(6,eval), e2(6,eval))!, e3(6,nl), e4(6,nl))
DO j1=1,6
        e1(j1,:)=lwedge(2-j1+eval*(j1-1):j1*eval)
        e2(j1,:)=rwedge(2-j1+eval*(j1-1):j1*eval)
        !e3(j1,:)=ledges(1+(j1-1)*nl:j1*nl)
        !e4(j1,:)=ledges(j1*nl+1:(j1+1)*nl)
END DO

DEALLOCATE(lwedge, rwedge, ledges)

a=wstep*SQRT(3.)/2. !length of triangle subfacets
!generate triangles
DO j1=1,6
        CALL trigen(-a, e1(j1,1), eval, wstep, szfc, facetcoords, n, 1, 2*(j1-1)+1)
        CALL trigen(a, e2(j1,1), eval, wstep, szfc, facetcoords, n, 2, 2*j1)
END DO
```

```
nzxy=nl*nw+ntp          !total number of vertices in the strip

PRINT*, 'Initial facet created'

!calculate the limits in p & q
itlim=0                                                 !iteration limit
p1=SQRT(pi/2.)*clen/L                   !precalculate parts of the function
p1sq=p1**2.                                             !       "                       "       "
        "       "
p2=-0.5*(pi**2.)*((clen/L)**2.)  !      "                       "       "       "       "
j1=0
DO      !loop through setting a new max p,q (j1) each time
        cc=0.
        DO p=0,j1!*FLOOR(AR)
                dp0=0
                IF (p==0) THEN
                        dp0=1
                END IF
                ccp1=REAL(2-dp0)*p1sq*EXP((REAL(p)**2.)*p2)
                DO q=0,j1
                        dq0=0
                        IF (q==0) THEN
                                dq0=1
                        END IF
                        !calculate cc
                        ccp2=REAL(2-dq0)*EXP((REAL(q)**2.)*p2)
                        cc=cc+ccp1*ccp2
                        !IF (cc>1.02) THEN
                        !       STOP
                        !END IF
                END DO
        END DO
        !calculate the difference between the correlation function and the correlation expansion
        diffold=diff
        diff=ABS(1.-cc)
        IF (diff .LT. targetdiff) THEN
                itlim=j1
                EXIT
        END IF
        IF (cc > 1.02) THEN
                STOP '(p,q) will never be found. Exiting...'
        END IF
        IF (diff .EQ. diffold) THEN
  PRINT*, p
  STOP '(p,q) will never be found. Exiting...'
 END IF
        j1=j1+1
END DO
PRINT*, 'Limit in p & q is:', itlim
```

```
pqval=itlim+1

!calculate all z_pq
ALLOCATE(zpq(2*itlim+1,2*itlim+1))
!calculate z_pq for p=0, q=0
CALL RNDG(rnd1)
rn=rnd1
zpq(pqval,pqval)=p1*stdev*rn
!calculate z_pq for p=0 q=/=0
DO q=1,itlim
        cc=p1sq*2.*EXP((REAL(q)**2.)*p2)
        sdzr=SQRT((1./4.)*cc)*stdev
        sdzi=sdzr
        CALL RNDG(rnd1)
        rn=rnd1
        zpqr=rn*sdzr
        CALL RNDG(rnd2)
        rn=rnd2
        zpqi=rn*sdzi
        zpq(pqval,pqval+q)=zpqr+i*zpqi
        zpq(pqval,pqval-q)=zpqr-i*zpqi
END DO
!calculate z_pq for all other p,q
DO p=1,itlim
        psq=REAL(p)**2.
        DO q=-itlim,itlim
                dq0=0
                IF (q==0) THEN
                        dq0=1
                END IF
                !correlation coefficients
                cc=2.*REAL(2-dq0)*p1sq*EXP(p2*(psq + REAL(q)**2.))
                !std dev of the real & imag parts of z_pq
                sdzr=SQRT((1./8.)*REAL(1+dq0)*cc)*stdev
                sdzi=sdzr
                !calculate z_pq & z_-p-q & save into array zpq
                CALL RNDG(rnd1)
                rn=rnd1
                zpqr=rn*sdzr
                CALL RNDG(rnd2)
                rn=rnd2
                zpqi=rn*sdzi
                zpq(pqval+p,pqval+q)=zpqr+i*zpqi
                zpq(pqval-p,pqval-q)=zpqr-i*zpqi
        END DO
END DO
PRINT*, 'Coefficient values set'

!calculate z_xy for all points
n=0
DO j1=1,nzxy
```

```
        zxy=0.
        n=n+1
        Kx=K*facetcoords(n,1)
        Ky=K*facetcoords(n,2)
        !p=0, q=0
        zxy=zpq(pqval,pqval)
        !p=0, q=/=0
        DO q=1,itlim
                !add contributions for z_0q & z_0-q
                ipKxqKy=i*REAL(q)*Ky
                zxy=zxy+zpq(pqval,pqval+q)*EXP(ipKxqKy)+zpq(pqval,pqval-q)*EXP(-ipKxqKy)
        END DO
        !remaining p,q
        DO p=1,itlim
                pKx=REAL(p)*Kx
                DO q=-itlim,itlim
                        !add contributions for z_pq & z_-p-q
                        ipKxqKy=i*(pKx + (REAL(q)*Ky))
                        zxy=zxy+zpq(pqval+p,pqval+q)*EXP(ipKxqKy)+zpq(pqval-p,pqval-q)*EXP(-
ipKxqKy)
                END DO
        END DO
        facetcoords(n,3)=REAL(zxy)
END DO
DEALLOCATE(zpq)
PRINT*, "Real and imaginary parts of final z_xy:", zxy

!shift the facet to zero position
zx=loffset + pflength/2.
zy=woffset + swidth/2.
facetcoords(:,1) = facetcoords(:,1)-zx
facetcoords(:,2) = facetcoords(:,2)-zy

!save the generated facet
PRINT*, 'Saving the surface'

!!separate file, just a list of x,y,z values - for visualisation
!m=0
!DO j1=1,nw
!       DO j2=1,nl
!               m=m+1
!               WRITE(UNIT=2, FMT=*) facetcoords(m,1), '', facetcoords(m,2), '', facetcoords(m,3)
!       END DO
!END DO
!!add the triangles
!DO j1=1,ntp
!       m=m+1
!       WRITE(UNIT=2, FMT=*) facetcoords(m,1), '', facetcoords(m,2), '', facetcoords(m,3)
!END DO
!CLOSE(UNIT=2)
```

```
!write the number of facets
WRITE(UNIT=1, FMT=*) nfacets+ntf

!write the number of vertices for each facet
DO j1=1,nfacets
	WRITE(UNIT=1, FMT=*) 4
END DO
DO j1=1,ntf
	WRITE(UNIT=1, FMT=*) 3
END DO

!write the x,y,z values for each vertice of each subfacet - rectangles
m=0
DO j1=1,nfsw
	IF (j1 .NE. 1) m=m+1
	DO j2=1,nfsl
		m=m+1
		WRITE(UNIT=1, FMT=*) facetcoords(m,1), '', facetcoords(m,2), '', facetcoords(m,3)
		WRITE(UNIT=1, FMT=*) facetcoords(m+1,1), '', facetcoords(m+1,2), '',
facetcoords(m+1,3)
		WRITE(UNIT=1, FMT=*) facetcoords(m+nl+1,1), '', facetcoords(m+nl+1,2), '',
facetcoords(m+nl+1,3)
		WRITE(UNIT=1, FMT=*) facetcoords(m+nl,1), '', facetcoords(m+nl,2), '',
facetcoords(m+nl,3)
	END DO
END DO

!add the triangles
m=nl*nw
ALLOCATE(tpts(eval*(eval+1)/2))
!left triangles
DO j1=1,6
	!create an array containing row numbers for all points in the triangle
	tpts(1:eval)=e1(j1,:)
	DO j2=m+1,m+ntp/12
		tpts(eval+j2-m)=j2+(2*j1-2)*ntp/12
	END DO
	m2=0
	DO j2=1,eval-1
		!# points on this row:
		nr=eval+1-j2
		IF (j2 .NE. 1) m2=m2+1
		m3=m2+1
		IF (nr .NE. 2) THEN	!handle the last row
			DO j3=2,nr-1	!generate triangles that point away from the main triangle
point
				m3=m3+1
				WRITE(UNIT=1, FMT=*) facetcoords(tpts(m3),:)
				WRITE(UNIT=1, FMT=*) facetcoords(tpts(m3+nr),:)
				WRITE(UNIT=1, FMT=*) facetcoords(tpts(m3+nr-1),:)
			END DO
```

```
                END IF
                m3=m2
                !loop through points in the row
                DO j3=1,nr-1     !generate triangles that point towards the main triangle point
                        m3=m3+1
                        WRITE(UNIT=1, FMT=*) facetcoords(tpts(m3),:)
                        WRITE(UNIT=1, FMT=*) facetcoords(tpts(m3+1),:)
                        WRITE(UNIT=1, FMT=*) facetcoords(tpts(m3+nr),:)
                END DO
                m2=m3
        END DO
END DO
DEALLOCATE(e1)
!right triangles
DO j1=1,6
        !create an array containing row numbers for all points in the triangle
        tpts(1:eval)=e2(j1,:)
        DO j2=m+1,m+ntp/12
                tpts(eval+j2-m)=j2+(2*j1-1)*ntp/12
        END DO
        m2=0
        DO j2=1,eval-1
                !# points on this row:
                nr=eval+1-j2
                IF (j2 .NE. 1) m2=m2+1
                m3=m2+1
                IF (nr .NE. 2) THEN        !handle the last row
                        DO j3=2,nr-1     !generate triangles that point away from the main triangle
point
                                m3=m3+1
                                WRITE(UNIT=1, FMT=*) facetcoords(tpts(m3),:)
                                WRITE(UNIT=1, FMT=*) facetcoords(tpts(m3+nr),:)
                                WRITE(UNIT=1, FMT=*) facetcoords(tpts(m3+nr-1),:)
                        END DO
                END IF
                m3=m2
                !loop through points in the row
                DO j3=1,nr-1     !generate triangles that point towards the main triangle point
                        m3=m3+1
                        WRITE(UNIT=1, FMT=*) facetcoords(tpts(m3),:)
                        WRITE(UNIT=1, FMT=*) facetcoords(tpts(m3+1),:)
                        WRITE(UNIT=1, FMT=*) facetcoords(tpts(m3+nr),:)
                END DO
                m2=m3
        END DO
END DO
DEALLOCATE(tpts, e2, facetcoords)

CLOSE(UNIT=1)

END PROGRAM GRFStrip
```

```
! Random number generation:
! RNDG: Gaussian distribution with zero mean and unit standard deviation
! set_rseed: Seed the random number generator

subroutine RNDG(r1)

! Returns a normally distributed random deviate with zero mean and
! unit variance. Version 2002-12-16.
! Copyright (C) 2002 Karri Muinonen

implicit none
integer :: flg,irnd,xrandom
real :: RNDU,q1,q2,r1,r2
save flg, r2
data flg/0/
common irnd

if (flg.eq.1) then
        r1=r2
        flg=0
        return
endif

flg=0
q1 = 0.

do while ((q1.ge.1. .or. q1.le.0.))
        CALL RANDOM_NUMBER(r1)
        CALL RANDOM_NUMBER(r2)
        r1=2.*r1-1.
        r2=2.*r2-1.
        q1=r1**2.+r2**2.
end do

q2=sqrt(-2.*log(q1)/q1)
r1=r1*q2
r2=r2*q2
flg=1

end

!Seed the inbuilt random number generator - only needs to be called once per program
!Copyright Daniel Brown, Aberystwyth University
SUBROUTINE set_rseed()

  IMPLICIT NONE
  REAL :: r
  INTEGER :: a, b, c, size, i
  INTEGER, ALLOCATABLE, DIMENSION(:) :: seed
```

```
 CALL RANDOM_SEED(size)
 ALLOCATE(seed(size))

 CALL SYSTEM_CLOCK(a,b,c)
 DO i=1,size
   CALL RANDOM_NUMBER(r)
   seed(i) = NINT(r*REAL(a))
 END DO

 CALL RANDOM_SEED(PUT=seed)

 DEALLOCATE(seed)

END SUBROUTINE set_rseed

!generate a triangle
!IN:-
!a: length of polygons within the triangle
!sp: list of prism facet edges
!eval: number of unique points along the edge facing the prism facet
!wstep: gap between points along the width
!szfc: number of rows in the vertice array
!lr: indicate which side of the prism facets the triangles should be
!tas: indicate which triangle this is
!INOUT:-
!n: vertice row number
!te: row numbers of points on the triangle edge
!fc: vertice array
SUBROUTINE trigen(a, sp, eval, wstep, szfc, fc, n, lr, tas)
IMPLICIT NONE

INTEGER, INTENT(IN) :: eval, szfc, lr, sp, tas
REAL(8), INTENT(IN) :: a, wstep
INTEGER, INTENT(INOUT) :: n
REAL(8), DIMENSION(szfc,3), INTENT(INOUT) :: fc
INTEGER :: j1, j2, nptsrow, aa, m
REAL(8) :: x, y, ys


x=fc(sp,1)        !starting position in x
ys=fc(sp,2)       !starting position in y
y=ys

m=(tas-1)*(eval-1)+1     !track the row having its vertices calculated
DO j1=1,eval-1                    !loop through the vertice rows
        nptsrow=eval-j1                   !number of vertices in the row
        x=x+a                             !x position of the row
        y=y+wstep/2.              !y position of the start of the row
        DO j2=1,nptsrow                   !loop through all vertice in the row
                n=n+1                     !identify the vertice
                fc(n,1)=x                 !x position of the vertice
```

```
                fc(n,2)=y+REAL(j2-1)*wstep        !y position of the vertice
                fc(n,3)=0.                        !z position
        END DO
        m=m+1
END DO

END SUBROUTINE
```

9.2 comp.f90

```
!Create composite surface
!vertices must all be in the same place in x&y

PROGRAM comp

IMPLICIT NONE
INTEGER :: j1, j2, m, s1nf, s2nf, s1np, s2np
INTEGER, DIMENSION(:), ALLOCATABLE :: s1f, s2f
REAL(8), DIMENSION(:,:), ALLOCATABLE :: s1p, s2p, sp
REAL(8) :: sd1, sd2

!read in 1st surface
OPEN(UNIT=1, FILE='strip1.crystal', STATUS='OLD')
READ(UNIT=1, FMT=*)  s1nf
ALLOCATE(s1f(s1nf))
DO j1=1,s1nf
        READ(UNIT=1, FMT=*)  s1f(j1)
END DO
s1np=SUM(s1f)
ALLOCATE(s1p(s1np,3))
m=0
DO j1=1,s1nf
        DO j2=1,s1f(j1)
                m=m+1
                READ(UNIT=1, FMT=*)  s1p(m,1), s1p(m,2), s1p(m,3)
        END DO
END DO
CLOSE(1)

!read in 2nd surface
OPEN(UNIT=2, FILE='strip2.crystal', STATUS='OLD')
READ(UNIT=2, FMT=*)  s2nf
ALLOCATE(s2f(s2nf))
DO j1=1,s2nf
        READ(UNIT=2, FMT=*)  s2f(j1)
END DO
s2np=SUM(s2f)
ALLOCATE(s2p(s2np,3))
m=0
DO j1=1,s2nf
        DO j2=1,s2f(j1)
                m=m+1
                READ(UNIT=2, FMT=*)  s2p(m,1), s2p(m,2), s2p(m,3)
        END DO
END DO
CLOSE(2)

!superimpose the two surfaces
ALLOCATE(sp(s1np,3))
sp(:,1)=s1p(:,1)
```

```
sp(:,2)=s1p(:,2)
sp(:,3)=s1p(:,3)+s2p(:,3)

!output the new surface
OPEN(UNIT=3, FILE='strip.crystal', STATUS='REPLACE')
WRITE(UNIT=3, FMT=*) s1nf
DO j1=1,s1nf
        WRITE(UNIT=3, FMT=*) s1f(j1)
END DO
DO j1=1,s1np
        WRITE(UNIT=3, FMT=*) sp(j1,1), sp(j1,2), sp(j1,3)
END DO
CLOSE(3)

DEALLOCATE(s1f, s2f, s1p, s2p, sp)

END PROGRAM comp
```

9.3 *crystal.f90*

```
!Fold a grfstrip-generated strip into a crystal shape
PROGRAM crystal
IMPLICIT NONE

REAL(8) :: pi, hlength, pflength, clen, stdev, L, K, swidth, lstep, wstep, ndevmax
INTEGER :: nfhl, nfsl, noffset, nfsw, nl, nw, npfw, npf, ntp, szfc, loffset, woffset, szle
INTEGER :: nf, np, m, j1, j2, eval, v, u, np1pf, sztn
INTEGER, ALLOCATABLE, DIMENSION(:) :: f, le, tn, pfwe
REAL(8), ALLOCATABLE, DIMENSION(:,:) :: p
REAL(8), DIMENSION(6,2) :: bv

pi=4.*ATAN(1.)

!open the output file
OPEN(UNIT=1, FILE='fc.crystal', STATUS='REPLACE')

!read in input values
OPEN(UNIT=3, FILE='vals.in', STATUS='OLD')
READ(UNIT=3, FMT=*) hlength
READ(UNIT=3, FMT=*) nfhl
READ(UNIT=3, FMT=*) pflength
READ(UNIT=3, FMT=*) nfsl
READ(UNIT=3, FMT=*) clen
READ(UNIT=3, FMT=*) stdev
READ(UNIT=3, FMT=*) noffset
CLOSE(UNIT=3)

!set various things
nfsw=6*nfhl
nl=nfsl+1
nw=nfsw+1
npfw=nfhl+1
npf=nfsl*nfsw
L=2.*hlength
K=pi/L
swidth=6.*hlength
ntp=6*(npfw**2-npfw)
np1pf=npf*2/3
szfc=nl*nw+ntp
lstep=pflength/REAL(nfsl)
wstep=swidth/REAL(6*nfhl)
loffset=REAL(noffset+1)*lstep
woffset=REAL(noffset+1)*wstep
eval=1+nfhl                    !number of vertices along hexagon radius

!matlab version of input values file
OPEN(UNIT=10, FILE='mlvals.in', STATUS='REPLACE')
WRITE(UNIT=10, FMT=*) hlength
WRITE(UNIT=10, FMT=*) nfhl
```

```
WRITE(UNIT=10, FMT=*) pflength
WRITE(UNIT=10, FMT=*) nfsl
WRITE(UNIT=10, FMT=*) clen
WRITE(UNIT=10, FMT=*) stdev
WRITE(UNIT=10, FMT=*) noffset
CLOSE(UNIT=10)

!!!!!!!!!!!!!!!!!!!!!
!!!Read in surfaces!!!
!!!!!!!!!!!!!!!!!!!!!
PRINT*, 'Reading in surface...'
!read in hexagons
OPEN(UNIT=2, FILE='strip.crystal')
READ(UNIT=2, FMT=*)nf
ALLOCATE(f(nf))
DO j1=1,nf
        READ(UNIT=2, FMT=*) f(j1)
END DO
np=SUM(f)
ALLOCATE(p(np,3))
m=0
DO j1=1,nf
        DO j2=1,f(j1)
                m=m+1
                READ(UNIT=2, FMT=*) p(m,1), p(m,2), p(m,3)
        END DO
END DO
CLOSE(UNIT=2)
PRINT*, 'read in'
CALL cog(p, np, 0)

!get points along prism facet width edges
ALLOCATE(pfwe(24*nfhl))
DO j1=1,6*nfhl
        pfwe(4*j1-3)=1+(j1-1)*nfsl*4
        pfwe(4*j1-2)=4+(j1-1)*nfsl*4
        pfwe(4*j1-1)=j1*nfsl*4-2
        pfwe(4*j1)=j1*nfsl*4-1
END DO

!make most negative z the base z value
ndevmax=0.0
DO j1=1,np
        IF (p(j1,3)<ndevmax) THEN
                ndevmax=p(j1,3)
        END IF
END DO
PRINT*, 'roughness offset =', ndevmax
p(:,3)=p(:,3)-ndevmax
DEALLOCATE(pfwe)
```

```
!!!!!!!!!!!!!!!!!!!!!
!!!Crystal creation!!!
!!!!!!!!!!!!!!!!!!!!!

PRINT*, 'Creating the crystal'

!rotate the triangles to point down
v=4*npf
u=3*(eval-1)**2
DO j1=1,12
        CALL trotate(p(v+1:v+u,:), u, eval)
        v=v+u
END DO

!perform rotations at prism facet-hex facet edges
CALL tedgerotate(p, np, nfhl, nfsl)

!get boundary values
bv(:,2)=0.          !set unroughened z values
DO j1=1,5
        bv(j1,1)=p(1+np1pf*j1,2)          !unroughened y values
END DO
bv(6,1)=p(np1pf*6,2)

!perform rotations at prism facet-prism facet edges
DO j1=1,5
        CALL pfedgerotate(p, np, nfhl, nfsl, bv, j1)
END DO

!rotate to form the crystal
DO j1=1,5
        sztn=12-2*j1
        ALLOCATE(tn(sztn))
        DO j2=1,sztn/2
                tn(j2)=j1+j2
                tn(sztn/2+j2)=6+j1+j2
        END DO
        !perform a rotation
        CALL crotate(p(j1*np1pf+1:np,:), np-j1*np1pf, np1pf*(6-j1), tn, sztn, nfsl, bv, j1)
        DEALLOCATE(tn)
END DO

PRINT*, 'Saving the crystal'

!write the number of facets
WRITE(UNIT=1, FMT=*) nf

!write the number of vertices for each facet
DO j1=1,nf
        WRITE(UNIT=1, FMT=*) f(j1)
END DO
```

```
!write the x,y,z values for each corner of each facet
DO j1=1,np
	WRITE(UNIT=1, FMT=*) p(j1,1), '', p(j1,2), '', p(j1,3)
END DO
CLOSE(UNIT=1)

END PROGRAM

!rotate a triangle
!IN:-
!np: number of points
!eval: number of unique points along the edge facing the prism facet
!INOUT:-
!p: coordinate array
SUBROUTINE trotate(p, np, eval)
IMPLICIT NONE

INTEGER, INTENT(IN) :: np, eval
REAL(8), DIMENSION(np, 3), INTENT(INOUT) :: p
REAL(8), DIMENSION(np, 3) :: tempp
INTEGER, DIMENSION(3*eval-4) :: el
INTEGER :: j1, j2, flag, etfa
REAL(8) :: xval, xv

!create list of points not to rotate - i.e. edge points
DO j1=1,eval-2
	el(j1)=3*j1-2
END DO
etfa=3*eval-6    !last coordinate row in a triangle pointing away from the main triangle point
DO j1=1,eval-1
	el(j1+eval-2)=3*j1-2+etfa
	el(j1+2*eval-3)=3*j1-1+etfa
END DO

!get x value of the triangle base
xval=p(el(1),1)

!assign temporary array
tempp=p

IF (xval>0.) THEN
	tempp(:,1)=-tempp(:,1)  !if the triangles are on the right, make sure they rotate the correct
way
	xv=-xval
ELSE
	xv=xval
	tempp(:,3)=-tempp(:,3)
END IF

!compare row numbers to the previous list
```

```
!only rotate rows that aren't in that list
DO j1=1,np
	flag=0
	DO j2=1,3*eval-4
		IF (j1==el(j2)) THEN
			flag=1
			EXIT
		END IF
	END DO
	IF (flag==0) THEN
		p(j1,1)=tempp(j1,3)+xval
		p(j1,2)=tempp(j1,2)
		p(j1,3)=tempp(j1,1)-xv
	END IF
END DO

END SUBROUTINE

!rotate a section of the crystal
!IN:-
!np: number of points
!pfe: last points array row describing a polygon in a prism facet
!t2r: triangles to rotate
!szt2r: numbers of triangles to rotate
!nfsl: number of facets along the strip length
!nr: n'th rotation - describes which bv row to use
!INOUT:-
!bv: boundary values in y & z - describe axis to rotate around
!p: points array
SUBROUTINE crotate(p, np, pfe, t2r, szt2r, nfsl, bv, nr)
IMPLICIT NONE

REAL(8), PARAMETER :: pi=4.*ATAN(1.) ,angle=-pi/3., ca=COS(angle), sa=SIN(angle)
INTEGER, INTENT(IN) :: np, szt2r, pfe, nfsl, nr
REAL(8), DIMENSION(np,3), INTENT(INOUT) :: p
INTEGER, DIMENSION(szt2r), INTENT(IN) :: t2r
REAL(8), DIMENSION(np,3) :: tp
REAL(8), DIMENSION(6,2), INTENT(INOUT) :: bv
REAL(8), DIMENSION(6,2) :: bvtmp
INTEGER :: j1, j2, sz1t, flag, sct, ect, dr, ev, v1, v2
REAL(8) :: yval, zval
INTEGER, DIMENSION(2*nfsl) :: el

!nr: n'th rotation
!shift the points so they are rotating around the axis
yval=bv(nr,1)
zval=bv(nr,2)
p(:,2)=p(:,2)-yval
p(:,3)=p(:,3)-zval

!assign temporary array
```

```
tp=p

!number of points in one triangle
sz1t=(np-pfe)/12

!work out which triangle points to not rotate
ev=1+INT(SQRT(REAL(sz1t/3)))
v1=3*(ev-2)+1

!rotate the triangles
DO j1=1,szt2r
	!start of the current triangle
	sct=pfe+sz1t*(t2r(j1)-1)+1
	!end of the current triangle
	ect=pfe+sz1t*t2r(j1)
	IF ((j1 .EQ. 1) .OR. (j1 .EQ. szt2r/2 +1)) THEN
		DO j2=sct,ect
			IF ((j2-sct+1 .NE. v1)) THEN! .AND. (j2 .NE. v2)) THEN
				p(j2,2)=ca*tp(j2,2) - sa*tp(j2,3)
				p(j2,3)=sa*tp(j2,2) + ca*tp(j2,3)
			END IF
		END DO
	ELSE
		p(sct:ect,2)=ca*tp(sct:ect,2) - sa*tp(sct:ect,3)
		p(sct:ect,3)=sa*tp(sct:ect,2) + ca*tp(sct:ect,3)
	END IF
END DO

!calculate which prism facet points to not rotate
DO j1=1,nfsl
	el(j1)=4*j1-3
	el(j1+nfsl)=4*j1-2
END DO

!compare row numbers to the previous list
!rotate coordinate rows that aren't in that list
DO j1=1,pfe
	flag=0
	DO j2=1,2*nfsl
		IF (j1==el(j2)) THEN
			flag=1
			EXIT
		END IF
	END DO
	IF (flag==0) THEN
		p(j1,2)=ca*tp(j1,2) - sa*tp(j1,3)
		p(j1,3)=sa*tp(j1,2) + ca*tp(j1,3)
	END IF
END DO

!shift the points back from the rotation axis
```

```
p(:,2)=p(:,2)+yval
p(:,3)=p(:,3)+zval

!rotate the boundary values array
bv(:,1)=bv(:,1)-yval
bv(:,2)=bv(:,2)-zval
bvtmp=bv
bv(:,1)=ca*bvtmp(:,1) - sa*bvtmp(:,2)
bv(:,2)=sa*bvtmp(:,1) + ca*bvtmp(:,2)
bv(:,1)=bv(:,1)+yval
bv(:,2)=bv(:,2)+zval

END SUBROUTINE


!centre of gravity calculation
!IN:-
!szarr: number of rows in coordinate array
!zs: z switch - 1 if the CoG is also required in z
SUBROUTINE cog(arr, szarr, zs)
IMPLICIT NONE
REAL(8), DIMENSION(3) :: cogval
INTEGER, INTENT(IN) :: szarr, zs  !zs=1 if z cog also required
REAL(8), DIMENSION(szarr,3), INTENT(INOUT) :: arr

!find centre of gravity of these points
cogval(1)=SUM(arr(1:szarr,1))/REAL(szarr)
cogval(2)=SUM(arr(1:szarr,2))/REAL(szarr)
IF (zs==1) THEN
        cogval(3)=SUM(arr(1:szarr,3))/REAL(szarr)
END IF

!shift the array to the new zero value
arr(:,1)=arr(:,1)-cogval(1)
arr(:,2)=arr(:,2)-cogval(2)
IF (zs==1) THEN
        arr(:,3)=arr(:,3)-cogval(3)
END IF

END SUBROUTINE

!calculate points for triangle-prism facet edge rotations & rotate
SUBROUTINE tedgerotate(p, np, nfpfw, nfsl)
IMPLICIT NONE

REAL(8), PARAMETER :: pi=4.*ATAN(1.)
INTEGER, INTENT(IN) :: np, nfpfw, nfsl
REAL(8), DIMENSION(np,3), INTENT(INOUT) :: p
INTEGER, DIMENSION(:,:), ALLOCATABLE :: te, pfwe, rp
INTEGER :: eval, j1, j2, npsl, srp, trp, ept, etfa, sz1t, v, szrp, im, s, epra, tpae, nr
REAL(8) :: xr1, xr2, angle, ac, zz, p1, p2
```

```
REAL(8), DIMENSION(np,3) :: tp
INTEGER, DIMENSION(:,:), ALLOCATABLE :: er

!set value for 45 degree rotation
angle=pi/4.

!get points along prism facet width edges
ALLOCATE(pfwe(12*nfpfw,2))
npsl=nfsl*4
DO j1=1,6*nfpfw
        pfwe(2*(j1-1)+1,1)=1+(j1-1)*npsl
        pfwe(2*j1,1)=4+(j1-1)*npsl
        pfwe(2*(j1-1)+1,2)=j1*npsl-2
        pfwe(2*j1,2)=j1*npsl-1
END DO

!get rotation axes
xr1=p(pfwe(1,1),1)
xr2=p(pfwe(1,2),1)

!get points connecting triangles with prism facets
eval=nfpfw+1
sz1t=3*(eval-1)**2
ALLOCATE(te(3*eval-4,12))
ept=24*nfpfw*nfsl
etfa=3*eval-6    !last coordinate row in a triangle pointing away from the main triangle point
DO j2=1,12
        DO j1=1,eval-2
                te(j1,j2)=3*j1-2+ept
        END DO
        DO j1=1,eval-1
                te(j1+eval-2,j2)=3*j1-2+etfa+ept
                te(j1+2*eval-3,j2)=3*j1-1+etfa+ept
        END DO
        ept=ept+sz1t
END DO

!group together points to be rotated around the same axis
szrp=18*eval-24+12*nfpfw
ALLOCATE(rp(szrp,2))!3*(6*eval-8+6*nfpfw),2))
rp=0
srp=2*nfpfw*6
rp(1:srp,:)=pfwe
trp=3*eval-4         !trp: # edge points per triangle
                                        !srp: # edge points on each side of the strip
DO j1=1,6
        rp(srp+1+(j1-1)*trp:srp+j1*trp,1)=te(:,j1)
        rp(srp+1+(j1-1)*trp:srp+j1*trp,2)=te(:,j1+6)
END DO

zz=0.
```

```
CALL ter(szrp, np, rp, p, xr1, xr2, zz, angle)

DEALLOCATE(rp, te, pfwe)

!rotations on triangles
tp=p
im=1    !number of iterations each side of the edge
ac=-pi/(4.*REAL(im+1))
DO j1=1,im
        s=24*nfpfw*nfsl
        v=6*(eval-1-j1)
        ALLOCATE(te(v,12))
        te=0
        DO j2=1,12
                CALL tfr(s, eval, j1, v, te(:,j2))
                s=s+sz1t
        END DO
        ALLOCATE(rp(6*v,2))
        DO j2=1,6
                rp(1+(j2-1)*v:j2*v,1)=te(:,j2)
                rp(1+(j2-1)*v:j2*v,2)=te(:,j2+6)
        END DO
        angle=ac*REAL(im+1-j1)
        p1=p(rp(1,1),3)
        !rotate a row
        CALL ter(6*v, np, rp, p, xr1, xr2, p1, angle)
        !correct for y position
        nr=3
        ALLOCATE(er(nr,2))
        epra=24*nfpfw*nfsl
        DO j2=2,j1
                epra=epra+6*(eval-j2)+3
        END DO
        tpae=3*(eval-j1-1)
        er(1,1)=epra+3
        er(2,1)=epra+tpae+3
        er(3,1)=epra+tpae+6*(eval-j1)-5
        er(1,2)=epra+tpae-1
        er(2,2)=epra+6*(eval-j1)-3
        er(3,2)=epra+12*(eval-j1)-13
        CALL yc(nr, er, np, p, tp, sz1t)
        !DEALLOCATE(er)
        !nr=6
        !ALLOCATE(er(nr,2))
        !er(1,1)=epra
        !er(1,2)=epra+tpae
        !er(1,3)=
        DEALLOCATE(rp, te, er)
END DO

!rotations on prism facets
```

```
ALLOCATE(pfwe(24*nfpfw,2))
ac=pi/(4.*REAL(im+1))
DO j1=1,im
        pfwe=0
        CALL pfr(nfpfw*6, nfsl, j1, 24*nfpfw, pfwe(:,1))
        CALL pfr(nfpfw*6, nfsl, nfsl-j1, 24*nfpfw, pfwe(:,2))
        angle=ac*REAL(im+1-j1)
        p1=p(pfwe(1,1),1)
        p2=p(pfwe(1,2),1)
        CALL ter(24*nfpfw, np, pfwe, p, p1, p2, zz, angle)
END DO
DEALLOCATE(pfwe)

END SUBROUTINE tedgerotate

!get a prism facet row for the edge rotation
SUBROUTINE pfr(nfsw, nfsl, dfe, szpfe, pfe)
IMPLICIT NONE

INTEGER, INTENT(IN) :: nfsw, nfsl, dfe, szpfe
INTEGER, DIMENSION(szpfe), INTENT(OUT) :: pfe
INTEGER :: plr, j1, j1plr

plr=4*nfsl

DO j1=1,nfsw
        j1plr=(j1-1)*plr+4*(dfe-1)
        pfe(j1)=j1plr+2
        pfe(j1+nfsw)=j1plr+3
        pfe(j1+2*nfsw)=j1plr+5
        pfe(j1+3*nfsw)=j1plr+8
END DO

END SUBROUTINE

!get a triangular facet row for the edge rotation
SUBROUTINE tfr(s, eval, dfe, szte, te)
IMPLICIT NONE

INTEGER, INTENT(IN) :: eval, dfe, szte, s
INTEGER, DIMENSION(szte), INTENT(OUT) :: te
INTEGER :: j1, tj1, epra, df, tpae

df=dfe-1
tpae=3*eval-6-3*df      !end of triangles pointing away in one row

!determine where to start from
epra=s
DO j1=2,dfe
        epra=epra+6*eval-9-6*(j1-2)
END DO
```

```
!get rows along line
DO j1=1,eval-2-df
	tj1=3*j1
	te(j1)=epra+tj1
	te(j1+eval-2-df)=epra+tj1-1
END DO
DO j1=1,eval-1-df
	te(j1+2*(eval-df)-4)=epra+tpae+3*j1
END DO

!move to the next row
epra=epra+6*eval-9-6*df
tpae=tpae-3

DO j1=1,eval-3-df
	te(j1+3*(eval-df)-5)=epra+3*j1-2
END DO
DO j1=1,eval-2-df
	tj1=3*j1
	te(j1+4*(eval-df)-8)=epra+tpae-2+tj1
	te(j1+5*(eval-df)-10)=epra+tpae-1+tj1
END DO

END SUBROUTINE

!perform edge rotations at triangle-prism facet boundary
SUBROUTINE ter(nr, np, rp, p, xr1, xr2, zr, angle)
IMPLICIT NONE

INTEGER, INTENT(IN) :: nr, np
INTEGER, DIMENSION(nr,2), INTENT(IN) :: rp
REAL(8), DIMENSION(np,3), INTENT(INOUT) :: p
REAL(8), DIMENSION(np,3) :: tp
REAL(8), INTENT(IN) :: angle, xr1, xr2, zr
INTEGER :: j1
REAL(8) :: sa, ca, zrr

!set sin & cos
sa=SIN(angle)
ca=COS(angle)

!rotate points with -ve x
tp=p
zrr=zr !for some reason zr changes in the 1st iteration of the loop
tp(:,1)=tp(:,1)-xr1
tp(:,3)=tp(:,3)-zrr
DO j1=1,nr
	IF (tp(rp(j1,1),3)>0.) THEN
		p(rp(j1,1),3)=ca*tp(rp(j1,1),3) + sa*tp(rp(j1,1),1) + zrr
		p(rp(j1,1),1)=-sa*tp(rp(j1,1),3) + ca*tp(rp(j1,1),1) + xr1
```

```
        ELSE
                p(rp(j1,1),3)=tp(rp(j1,1),3) + ca*tp(rp(j1,1),3) + sa*tp(rp(j1,1),1) + zrr
                p(rp(j1,1),1)=sa*tp(rp(j1,1),3) + ca*tp(rp(j1,1),1) + xr1
        END IF
END DO

!rotate points with +ve x
!white in vish
tp(:,1)=p(:,1)-xr2
tp(:,3)=p(:,3)-zrr
DO j1=1,nr
        IF (tp(rp(j1,2),3)>0.) THEN
                p(rp(j1,2),1)=sa*tp(rp(j1,2),3) + ca*tp(rp(j1,2),1) + xr2
                p(rp(j1,2),3)=ca*tp(rp(j1,2),3) - sa*tp(rp(j1,2),1) + zrr
        ELSE
                p(rp(j1,2),1)=-sa*tp(rp(j1,2),3) + ca*tp(rp(j1,2),1) + xr2
                p(rp(j1,2),3)=tp(rp(j1,2),3) + ca*tp(rp(j1,2),3) - sa*tp(rp(j1,2),1) + zrr
        END IF
END DO

END SUBROUTINE ter

!correct y value for rotated triangle subfacet points
!compare unrotated z with rotated z, and calculate new y
SUBROUTINE yc(ner, er, np, p, tp, sz1t)
IMPLICIT NONE

REAL(8), PARAMETER :: pi=4.*ATAN(1.)
INTEGER, INTENT(IN) :: np, ner, sz1t
REAL(8), DIMENSION(np,3), INTENT(INOUT) :: p
REAL(8), DIMENSION(np,3), INTENT(IN) :: tp
INTEGER, DIMENSION(ner, 2), INTENT(IN) :: er
INTEGER :: j1, j2, s
REAL(8) :: zgapn, zgapp, t60

!calculate tan of 60 deg
t60=TAN(pi/3.)

s=0
!calculate change in z
DO j1=1,12
        DO j2=1,ner
                zgapn=p(er(j2,1)+s,3)-tp(er(j2,1)+s,3)
                zgapp=p(er(j2,2)+s,3)-tp(er(j2,1)+s,3)
                !calculate change in y
    p(er(j2,1)+s,2)=p(er(j2,1)+s,2)-zgapn/t60
    p(er(j2,2)+s,2)=p(er(j2,2)+s,2)+zgapp/t60
        END DO
  s=s+sz1t
END DO
```

```
END SUBROUTINE yc


!calculate points for pf-pf edge rotation & rotate
SUBROUTINE pfedgerotate(p, np, nfpfw, nfsl, bv, itn)
IMPLICIT NONE
REAL(8), PARAMETER :: pi=4.*ATAN(1.)
REAL(8) :: angle
INTEGER, INTENT(IN) :: np, nfpfw, nfsl, itn
REAL(8), DIMENSION(np,3), INTENT(INOUT) :: p
REAL(8), DIMENSION(6,2), INTENT(IN) :: bv
REAL(8), DIMENSION(np,3) :: tp
INTEGER, DIMENSION(:), ALLOCATABLE :: e
INTEGER :: eval, epf, m, j1, j2, pfe, sz1t, fj1, im
REAL(8) :: yval, zval, ac, yv

tp=p

!get points along prism facet length edges
ALLOCATE(e(4*nfsl+4))
epf=4*nfsl*(itn*nfpfw-1)
CALL ev(e, m, nfsl, epf)
yv=p(e(1),2)
!get points from triangles to rotate
eval=nfpfw+1
pfe=24*nfsl*nfpfw
sz1t=3*(eval-1)**2
e(m+1)=pfe+(itn-1)*sz1t+6*eval-10
e(m+2)=pfe+itn*sz1t+3*eval-5
e(m+3)=pfe+(itn+5)*sz1t+6*eval-10
e(m+4)=pfe+(itn+6)*sz1t+3*eval-5
!perform 30 degree rotation
angle=-pi/6.
CALL pfer(np, p, 4*nfsl+4, e, tp(e(1),2), bv(itn,2), angle)

DEALLOCATE(e)
ALLOCATE(e(4*nfsl+6))

!rotations on unrotated prism facet
im=5     !iterations each side of the edge
ac=-pi/(6.*REAL(im+1))  !angle change on each iteration
DO j1=1,im
        e=0
        epf=4*nfsl*(itn*nfpfw-1-j1)
        CALL ev(e, m, nfsl, epf)
        CALL ev2a(e, 4*nfsl+6, m, pfe, itn, sz1t, eval, j1)
        angle=ac*REAL(im+1-j1)
        CALL pfer(np, p, 4*nfsl+6, e, tp(e(1),2), bv(itn,2), angle)
END DO

!rotations on rotated prism facet
```

```
ac=-ac
DO j1=1,im
        e=0
        epf=4*nfsl*(itn*nfpfw+j1-1)
        CALL ev(e, m, nfsl, epf)
        CALL ev2b(e, 4*nfsl+6, m, pfe, itn, sz1t, eval, j1)
        angle=ac*REAL(im+1-j1)
        CALL pfer(np, p, 4*nfsl+6, e, tp(e(1),2), bv(itn,2), angle)
END DO

DEALLOCATE(e)

END SUBROUTINE pfedgerotate

!vertices from the triangle to rotate (left side)
SUBROUTINE ev2a(e, ne, m, pfe, itn, sz1t, eval, dfe)
IMPLICIT NONE

INTEGER, INTENT(IN) :: ne, m, pfe, itn, sz1t, eval, dfe
INTEGER, DIMENSION(ne), INTENT(INOUT) :: e
INTEGER :: l

l=(dfe-1)*3

e(m+1)=pfe+(itn-1)*sz1t+3*eval-8-l
e(m+2)=pfe+(itn-1)*sz1t+6*eval-11-l
e(m+3)=pfe+(itn-1)*sz1t+6*eval-13-l
e(m+4)=pfe+(itn+5)*sz1t+3*eval-8-l
e(m+5)=pfe+(itn+5)*sz1t+6*eval-11-l
e(m+6)=pfe+(itn+5)*sz1t+6*eval-13-l

END SUBROUTINE ev2a

!vertices from the triangle to rotate (right side)
SUBROUTINE ev2b(e, ne, m, pfe, itn, sz1t, eval, dfe)
IMPLICIT NONE

INTEGER, INTENT(IN) :: ne, m, pfe, itn, sz1t, eval, dfe
INTEGER, DIMENSION(ne), INTENT(INOUT) :: e
INTEGER :: l

l=(dfe-1)*3

e(m+1)=pfe+itn*sz1t+1+l
e(m+2)=pfe+itn*sz1t+3*eval-4+l
e(m+3)=pfe+itn*sz1t+3*eval-2+l
e(m+4)=pfe+(itn+6)*sz1t+1+l
e(m+5)=pfe+(itn+6)*sz1t+3*eval-4+l
e(m+6)=pfe+(itn+6)*sz1t+3*eval-2+l

END SUBROUTINE ev2b
```

```
!get row numbers of edge values along the prism facet
SUBROUTINE ev(e, m, nfsl, epf)
IMPLICIT NONE

INTEGER, INTENT(IN) :: nfsl, epf
INTEGER, INTENT(OUT) :: m
INTEGER, DIMENSION(4*nfsl+6), INTENT(OUT) :: e
INTEGER :: fj1, j1

m=0
DO j1=1,nfsl
   fj1=4*j1+epf
         e(m+1)=fj1
         e(m+2)=fj1-1
         m=m+2
END DO
DO j1=1,nfsl
   fj1=4*j1+epf+4*nfsl
         e(m+1)=fj1-3
         e(m+2)=fj1-2
         m=m+2
END DO
END SUBROUTINE

!perform a pf-pf edge rotation
SUBROUTINE pfer(np, p, nr, r, yval, zval, angle)
IMPLICIT NONE

INTEGER, INTENT(IN) :: np, nr
REAL(8), INTENT(IN) :: yval, zval, angle
REAL(8), DIMENSION(np,3), INTENT(INOUT) :: p
INTEGER, DIMENSION(nr) :: r
REAL(8), DIMENSION(np,3) :: tp
REAL(8) :: ca, sa

ca=COS(angle)
sa=SIN(angle)
p(:,2)=p(:,2)-yval
p(:,3)=p(:,3)-zval
tp=p

!perform the rotation
p(r(:),2)=ca*tp(r(:),2) - sa*tp(r(:),3)
p(r(:),3)=sa*tp(r(:),2) + ca*tp(r(:),3)

!shift the points back from the rotation axis
p(:,2)=p(:,2)+yval
p(:,3)=p(:,3)+zval

END SUBROUTINE
```

9.4 join.m

```
%Join all facets together using interpolation

%output file name
out='fcc.crystal';

%read in crystal
'Read in data'
% data=dlmread('fc-comp.crystal');
data=dlmread('fc.crystal');
nf=data(1,1);                  %number of facets
nv=data(2:nf+1,1);             %number of vertices for each facet
xyzp=data(nf+2:size(data(:,1)),:);  %vertices coordinates
np=size(xyzp,1);               %total number of vertices

%read in input values
vals=dlmread('mlvals.in');
hlen=vals(1);                  %hexagon radius x6
nfhl=vals(2);                  %number of facets along prism facet width
pflen=vals(3);                 %prism facet length
nfsl=vals(4);                  %number of facets along prism facet length
clen=vals(5);                  %correlation length
stdev=vals(6);                 %standard deviation
offset=vals(7);                %offset to avoid

%calculate various things
nfsw=6*nfhl;
nl=nfsl+1;
nw=nfsw+1;
npfw=nfhl+1;
npf=nfsl*nfsw;
L=2.*hlen;
K=pi/L;
swidth=6.*hlen;
ntp=6*(npfw^2-npfw);
np1pf=npf*2/3;
szfc=nl*nw+ntp;
lstep=pflen/nfsl;
wstep=swidth/(6*nfhl);
loffset=(offset+1)*lstep;
woffset=(offset+1)*wstep;
eval=1+nfhl;                   %number of vertices along hexagon radius
pfe=4*nfsw*nfsl;               %last vertice describing a prism facet
sz1t=(np-pfe)/12;              %number of vertices in one triangle
npe=3*(eval-2);                %number of vertices along the triangle edge
etpa=3*(eval-2);               %number of vertices on subtriangles pointing away from the end of the
main triangle
ygap=xyzp(4,2)-xyzp(1,2);      %distance in y between adjacent vertices

%!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
%!  Join first & last prism facets  !
```

```
%!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

'Join prism facets'

%get edge unroughened y, z values
yv=xyzp(1,2);
zv=0;

%calculate row values of points on the join between pf1 & pf6
pfedge=zeros(2*nfsl,2);
sp=4*nfsl*(nfsw-1);
for j1=1:nfsl
   pfedge(j1,1)=4*j1-3;
   pfedge(j1,2)=sp+4*j1;
   pfedge(nfsl+j1,1)=4*j1-2;
   pfedge(nfsl+j1,2)=sp+4*j1-1;
end %for

% %calculate row values of other points to be interpolated for
% pfev=zeros(2*nfsl);
% for j1=1:2
%    epf=(4*nfsl)*(j1-1);
%    e=ev(nfsl,epf);
%    m=4*nfsl;
%    e=ev2a(e, m, pfe, sz1t, eval, j1);
% end %for
% for j1=1:2
%    epf=pfe-(4*nfsl)*(j1+1);
%    e=ev(nfsl,epf);
%    m=4*nfsl;
%    e=ev2b(e, m, pfe, sz1t, eval, j1);
% end %for

%end points - x,y,z positions & row values
cp=zeros(2,1);
cp(1)=1;
cp(2)=4*nfsl-2;

%define the two prism facets
ppf1=xyzp(1:pfe/6,:);
ppf6=xyzp(5*pfe/6+1:pfe,:);

%exclude points to be interpolated for
ppf1e=zeros(pfe/6-2*nfsl,3);
ppf6e=zeros(pfe/6-2*nfsl,3);
m1=0;
m2=0;
for j1=1:pfe/6
   flag1=0;
   flag2=0;
  for j2=1:2*nfsl
```

```
    if (pfedge(j2,1)==j1)
      flag1=1;
    end %if
    if (pfedge(j2,2)-5*pfe/6==j1)
      flag2=1;
    end %if
  end %for
  if (flag1==0)
    m1=m1+1;
     ppf1e(m1,:)=ppf1(j1,:);
  end %if
  if (flag2==0)
    m2=m2+1;
    ppf6e(m2,:)=ppf6(j1,:);
  end %if
end %for

%get unique x,y,z points for the two prism facets
ppf1eu=unique(ppf1e, 'rows');
ppf6eu=unique(ppf6e, 'rows');

%rotate prism facet 6
yrv=xyzp(1,2);
ca=cos(-pi/3);
sa=sin(-pi/3);
ppf6eu(:,2)=ppf6eu(:,2)-yrv;
tppf6eu=ppf6eu;
ppf6eu(:,2)=ca*tppf6eu(:,2) - sa*tppf6eu(:,3);
ppf6eu(:,3)=sa*tppf6eu(:,2) + ca*tppf6eu(:,3);
ppf6eu(:,2)=ppf6eu(:,2)+yrv;

%create interpolation object
pf=unique(vertcat(ppf1eu, ppf6eu), 'rows');
pftsi=TriScatteredInterp(pf(:,1),pf(:,2),pf(:,3), 'natural');

%interpolate for pf1's edge points
xyzp(pfedge(:,1),3)=pftsi(xyzp(pfedge(:,1),1), xyzp(pfedge(:,1),2));

%half-rotate the edge points
xyzp=pfer(xyzp, pfedge(:,1), yrv, 0, pi/6);

%perform rotations on pf1
im=5;
ac=pi/(6*im+6);
for j1=1:im
  angle=ac*(im+1-j1);
  epf=(4*nfsl)*(j1-1);
  e=ev(nfsl,epf);
  m=4*nfsl;
  e=ev2a(e, m, pfe, sz1t, eval, j1);
  xyzp=pfer(xyzp, e, xyzp(e(1),2), zv, angle);
```

```
end %for

%perform rotations on pf6
ac=-ac;
dz=ygap*cos(pi/3);
dy=ygap*sin(pi/3);
yval=yv;
zval=zv;
for j1=1:im
   yval=yval-dy;
   zval=zval-dz;
   angle=ac*(im+1-j1);
   epf=pfe-(4*nfsl)*(j1+1);
   e=ev(nfsl,epf);
   m=4*nfsl;
   e=ev2b(e, m, pfe, sz1t, eval, j1);
   xyzp=pfer(xyzp, e, yval, zval, angle);
end %for

%apply edge changes to pf6
xyzp(pfedge(:,2),:)=xyzp(pfedge(:,1),:);

%!!!!!!!!!!!!!!!!!!!!!!!!!
%!  Join triangle edges  !
%!!!!!!!!!!!!!!!!!!!!!!!!!
'Create basal facets'

%save a copy of the data points
xyzpold=xyzp;

%calculate row values of points on triangle edges
hnte=36*eval-60;%24*(eval-2);
te=zeros(hnte,2);
m=0;
for j1=1:12
   eva=eval;
   epr=(j1-1)*sz1t;    %identity of last vertice in previous row
   for j2=1:eval-1
      if (j2 ~= 1)
         epr=epr+6*(eval-j2)+3;
         eva=eva-1;
      end %if
      etpa=3*eva-6;
      %edge points, triangles pointing away
      if (eva~=2)
         m=m+1;
         te(m,1)=epr+3;
         te(m,2)=epr+etpa-1;
      end %if
      if (j2 ~= 1)
         %1st & 3rd edge points, triangles pointing towards
```

```
        m=m+1;
        te(m,1)=epr+etpa+1;
        te(m,2)=epr+etpa+3*eva-4;
      end %if
      %2nd & 4th edge points, triangles pointing towards
      m=m+1;
      te(m,1)=epr+etpa+3;
      te(m,2)=epr+etpa+3*eva-3;
    end %for
end %for

%calculate row values of points within the triangle to be interpolated for
d=2;
q=0;
szintp=12*(6*2*d*(eval-d-2)+3*d^2-6*d);   %number of main triangles*vals per point*points per
row*n rows
intripts=zeros(szintp,1);
for j3=1:12
   er=(j3-1)*sz1t;
   for j1=1:eval-2-d
      epr=er;
      etpu=er+3*(eval-j1)-3;
      er=er+6*(eval-j1)-3;
      for j2=1:d
         tj2=3*j2;
         if (j1~=1)
            intripts(q+12)=epr+tj2-2;
            intripts(q+11)=etpu+tj2-1;
            intripts(q+10)=etpu+tj2+1;
            intripts(q+9)=etpu-tj2+1;
            intripts(q+8)=er-tj2-1;
            intripts(q+7)=er-tj2+1;
         end %if
         intripts(q+6)=epr+tj2+3;
         intripts(q+5)=etpu+tj2+3;
         intripts(q+4)=epr+tj2-1;
         intripts(q+3)=er-tj2;
         intripts(q+2)=etpu-tj2-1;
         intripts(q+1)=etpu-tj2+3;
         if(j1==1)
            q=q+6;
         else
            q=q+12;
         end %if
      end %for
   end %for
   ect=sz1t*j3;
   intripts(q+1)=ect-5;
   intripts(q+2)=ect-7;
   intripts(q+3)=ect-11;
   intripts(q+4)=ect-15;
```

```
   intripts(q+5)=ect-21;
   intripts(q+6)=ect-25;
   intripts(q+7)=ect-14;
   intripts(q+8)=ect-16;
   intripts(q+9)=ect-23;
   intripts(q+10)=ect-17;
   intripts(q+11)=ect-19;
   intripts(q+12)=ect-26;
   q=q+12;
end %for

%define the 2 hexagons
h1=xyzp(pfe+1:pfe+6*sz1t,:);
h2=xyzp(pfe+6*sz1t+1:np,:);

%create arrays for the two hexagons
%exclude triangle edges and other points to be interpolated for
h1e=zeros(6*(sz1t-6*eval+11),3); %6*sz1t-24*(eval-1)
h2e=zeros(6*(sz1t-6*eval+11),3);
m1=0;
%1st hexagon
for j1=1:6*sz1t%loop through the 1st 6 triangles
   flag1=0;
   for j2=1:hnte/2%12*(eval-1)%loop through the edges of the 1st 6 triangles
      if ((te(j2,1)==j1) || (te(j2,2)==j1))%if j1 is an edge point, flag it to not be saved
         flag1=1;
         break
      end %if
   end %for
   for j2=1:szintp/2
      if(intripts(j2)==j1)
         flag1=1;
         break
      end %if
   end %for
   if (flag1==0)
      m1=m1+1;
      h1e(m1,:)=h1(j1,:);
   end %if
end %for
%2nd hexagon
m1=0;
for j1=6*sz1t+1:12*sz1t
   flag1=0;
   for j2=hnte/2+1:hnte%12*(eval-1)+1:24*(eval-1)%loop through the edges of the 1st 6 triangles
      if ((te(j2,1)==j1) || (te(j2,2)==j1))%if j1 is an edge point, flag it to not be saved
         flag1=1;
         break
      end %if
   end %for
   for j2=szintp/2+1:szintp
```

```
      if(intripts(j2)==j1)
         flag1=1;
         break
      end %if
   end %for
   if (flag1==0)
      m1=m1+1;
      h2e(m1,:)=h2(j1-6*sz1t,:);
   end %if
end %for

%get unique points for each hexagon
h1eu=unique(h1e, 'rows');
h2eu=unique(h2e, 'rows');

%create interpolation objects
tsih1=TriScatteredInterp(h1eu(:,2),h1eu(:,3),h1eu(:,1), 'natural');
tsih2=TriScatteredInterp(h2eu(:,2),h2eu(:,3),h2eu(:,1), 'natural');

%make te & intripts applicable to the whole crystal
te=te+pfe;
intripts=intripts+pfe;

%create array of hexagon centre points
npe1=npe+1;
temh1=vertcat(te(npe1,1),te(2*npe1,1),te(3*npe1,1),te(4*npe1,1),te(5*npe1,1),te(6*npe1,1));
temh2=vertcat(te(7*npe1,1),te(8*npe1,1),te(9*npe1,1),te(10*npe1,1),te(11*npe1,1),te(12*npe1,1))
;

%interpolate for the centre point
xyzp(temh1(:),1)=tsih1(xyzp(temh1(1),2),xyzp(temh1(1),3));
xyzp(temh2(:),1)=tsih2(xyzp(temh2(1),2),xyzp(temh2(1),3));

%create array of edge points, not including the hexagon centres
te1=vertcat(te(1:npe,:),te(npe+2:2*npe+1,:),te(2*npe+3:3*npe+2,:),te(3*npe+4:4*npe+3,:),te(4*npe
+5:5*npe+4,:),te(5*npe+6:6*npe+5,:));
te2=vertcat(te(6*npe+7:7*npe+6,:),te(7*npe+8:8*npe+7,:),te(8*npe+9:9*npe+8,:),te(9*npe+10:10*
npe+9,:),te(10*npe+11:11*npe+10,:),te(11*npe+12:12*npe+11,:));

%interpolate for the points in the hexagon on one side of each triangle
xyzp(te1(:,1),1)=tsih1(xyzp(te1(:,1),2),xyzp(te1(:,1),3));
xyzp(te2(:,1),1)=tsih2(xyzp(te2(:,1),2),xyzp(te2(:,1),3));

%join the triangles together
xyzp(te1(1:npe,2),:)=xyzp(te1(npe+1:2*npe,1),:);
xyzp(te1(npe+1:2*npe,2),:)=xyzp(te1(2*npe+1:3*npe,1),:);
xyzp(te1(2*npe+1:3*npe,2),:)=xyzp(te1(3*npe+1:4*npe,1),:);
xyzp(te1(3*npe+1:4*npe,2),:)=xyzp(te1(4*npe+1:5*npe,1),:);
xyzp(te1(4*npe+1:5*npe,2),:)=xyzp(te1(5*npe+1:6*npe,1),:);
xyzp(te1(5*npe+1:6*npe,2),:)=xyzp(te1(1:npe,1),:);
xyzp(te2(1:npe,2),:)=xyzp(te2(npe+1:2*npe,1),:);
```

```
xyzp(te2(npe+1:2*npe,2),:)=xyzp(te2(2*npe+1:3*npe,1),:);
xyzp(te2(2*npe+1:3*npe,2),:)=xyzp(te2(3*npe+1:4*npe,1),:);
xyzp(te2(3*npe+1:4*npe,2),:)=xyzp(te2(4*npe+1:5*npe,1),:);
xyzp(te2(4*npe+1:5*npe,2),:)=xyzp(te2(5*npe+1:6*npe,1),:);
xyzp(te2(5*npe+1:6*npe,2),:)=xyzp(te2(1:npe,1),:);

%interpolate for the points within the triangles
xyzp(intripts(1:szintp/2),1)=tsih1(xyzp(intripts(1:szintp/2),2),xyzp(intripts(1:szintp/2),3));
xyzp(intripts(szintp/2+1:szintp),1)=tsih2(xyzp(intripts(szintp/2+1:szintp),2),xyzp(intripts(szintp/2+1:s
zintp),3));

%deal with NaNs
%coded for the composite crystal - NOT general!
a=find(isnan(xyzp(:,1)));
if (size(a,1)>0)
   xyzp(a(1:3),1)=sum(xyzpold(a(1:3),1))/3;
   xyzp(a(1:3),2)=sum(xyzpold(a(1:3),2))/3;
   xyzp(a(1:3),3)=sum(xyzpold(a(1:3),3))/3;
   xyzp(a(4:6),1)=sum(xyzpold(a(4:6),1))/3;
   xyzp(a(4:6),2)=sum(xyzpold(a(4:6),2))/3;
   xyzp(a(4:6),3)=sum(xyzpold(a(4:6),3))/3;
end %if

%!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
%!  Corners of joined prism facets  !
%!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
'Deal with corners of pf1 & pf6'

%define prism facet corners
cp1(1)=1;
cp1(2)=sp+4;
cp2(1)=4*nfsl-2;
cp2(2)=sp+4*nfsl-1;

%define triangle corners
t=zeros(4,1);
t(1)=pfe+npe+1;
t(2)=pfe+5*sz1t+npe+3*eval-4;
t(3)=pfe+6*sz1t+npe+1;
t(4)=pfe+11*sz1t+npe+3*eval-4;

% xyzp(cp1(:),:)
% xyzp(t(1:2),:)
%
% xyzp(cp2(:),:)
% xyzp(t(3:4),:)
%
% xyzp(cp2(1),1)
% xyzp(t(4),1)

%calculate means
```

```
xyzp(cp1(1),:)=(xyzp(t(1),:)+xyzp(t(2),:))/2;   %xyzp(cp1(1),:)+xyzp(cp1(2),:)+
xyzp(cp2(1),:)=(xyzp(t(3),:)+xyzp(t(4),:))/2;   %xyzp(cp2(1),:)+xyzp(cp2(2),:)+

%apply to the other corner points
xyzp(cp1(2),:)=xyzp(cp1(1),:);
xyzp(cp2(2),:)=xyzp(cp2(1),:);
xyzp(t(1),:)=xyzp(cp1(1),:);
xyzp(t(2),:)=xyzp(cp1(1),:);
xyzp(t(3),:)=xyzp(cp2(1),:);
xyzp(t(4),:)=xyzp(cp2(1),:);

%!!!!!!!!!!!!!!!!!!!!!!!!!!
%!  Save output crystal  !
%!!!!!!!!!!!!!!!!!!!!!!!!!!
'Create output file'

%reverse normals of prism subfacets
xyzpo=xyzp;
m=0;
for j1=1:pfe/4
   xyzp(m+2,:)=xyzpo(m+4,:);
   xyzp(m+4,:)=xyzpo(m+2,:);
   m=m+4;
end %for
%reverse normals of hex polygons at -ve x
for j1=1:6*sz1t/3
   xyzp(m+2,:)=xyzpo(m+3,:);
   xyzp(m+3,:)=xyzpo(m+2,:);
   m=m+3;
end %for

%centre of gravity
xcog=sum(xyzp(:,1))/np;
ycog=sum(xyzp(:,2))/np;
zcog=sum(xyzp(:,3))/np;
xyzp(:,1)=xyzp(:,1)-xcog;
xyzp(:,2)=xyzp(:,2)-ycog;
xyzp(:,3)=xyzp(:,3)-zcog;

% %split squares into triangles
% nfold=nf;
% nvold=nv;
% pfeold=pfe;
% xyzpold=xyzp;
% nf=nf+pfe/4;
% nv=zeros(nf,1);
% nv(:)=3;
% pfe=3*pfeold/2;
% xyzp=zeros(3*nf,3);
% m=0;
% n=0;
```

```
% for j1=1:pfeold/4;   %split the squares
%     xyzp(m+1,:)=xyzpold(n+1,:);
%     xyzp(m+2,:)=xyzpold(n+2,:);
%     xyzp(m+3,:)=xyzpold(n+3,:);
%     xyzp(m+4,:)=xyzpold(n+1,:);
%     xyzp(m+5,:)=xyzpold(n+3,:);
%     xyzp(m+6,:)=xyzpold(n+4,:);
%     m=m+6;
%     n=n+4;
% end %for
% xyzp(pfe+1:3*nf,:)=xyzpold(pfeold+1:pfeold+12*sz1t,:); %re-add hexagon subfacets

%output the finished crystal
dlmwrite(out, nf, 'precision', '%6d')
dlmwrite(out, nv, '-append')
dlmwrite(out, xyzp, '-append', 'precision', '%16.15f', 'delimiter', ' ')

% profile report
% profile off



%ev.m
function [ e ] = ev( nfsl, epf )
%get row numbers of edge values along the prism facet

e=zeros(4*nfsl+6,1);

m=0;
for j1=1:nfsl
   fj1=4*j1+epf;
          e(m+1)=fj1;
          e(m+2)=fj1-1;
          m=m+2;
end %for
for j1=1:nfsl
   fj1=4*j1+epf+4*nfsl;
          e(m+1)=fj1-3;
          e(m+2)=fj1-2;
          m=m+2;
end %for

end
```

```
%ev2a.m
function [ e ] = ev2a( e, m, pfe, sz1t, eval, dfe )
%vertices from the triangle (left side)

l=(dfe-1)*3;

e(m+1)=pfe+l+1;
e(m+2)=pfe+l+3*eval-4;
e(m+3)=pfe+l+3*eval-2;
e(m+4)=pfe+6*sz1t+l+1;
e(m+5)=pfe+6*sz1t+l+3*eval-4;
e(m+6)=pfe+6*sz1t+l+3*eval-2;

end


%ev2b.m
function [ e ] = ev2b( e, m, pfe, sz1t, eval, dfe )
%vertices from the triangle (right side)

l=(dfe-1)*3;

e(m+1)=pfe+5*sz1t-l+3*eval-8;
e(m+2)=pfe+5*sz1t-l+6*eval-11;
e(m+3)=pfe+5*sz1t-l+6*eval-13;
e(m+4)=pfe+11*sz1t-l+3*eval-8;
e(m+5)=pfe+11*sz1t-l+6*eval-11;
e(m+6)=pfe+11*sz1t-l+6*eval-13;

end


%pfer.m
function [ p ] = pfer( p, r, yv, zv, angle )
%perform a prism facet edge rotation

ca=cos(angle);
sa=sin(angle);
p(:,2)=p(:,2)-yv;
p(:,3)=p(:,3)-zv;
tp=p;

%perform the rotation
p(r(:),2)=ca*tp(r(:),2) - sa*tp(r(:),3);
p(r(:),3)=sa*tp(r(:),2) + ca*tp(r(:),3);

%shift the points back from the rotation axis
p(:,2)=p(:,2)+yv;
p(:,3)=p(:,3)+zv;

end
```

9.5 srfh.m

```
%plot a crystal file with any geometry

close all

%read in data
data=dlmread('fcc.crystal');
nfacets=data(1,1);
nvert=data(2:nfacets+1,1);
xyzp=data(nfacets+2:size(data(:,1)),:);

%move xyz points to zero position
xyzp(:,1)=xyzp(:,1)-mean(xyzp(:,1));
xyzp(:,2)=xyzp(:,2)-mean(xyzp(:,2));
xyzp(:,3)=xyzp(:,3)-mean(xyzp(:,3));

%plot the first facet
pts=zeros(nvert(1)+1,3);
for j2=1:nvert(1)
   if (j2==1)
      pts(1,:)=xyzp(1,:);
      pts(nvert(1)+1,:)=xyzp(1,:);
   else
      pts(j2,:)=xyzp(j2,:);
   end %if
end %for
figure
plot3(pts(:,1),pts(:,2),pts(:,3),'k');
hold on

%plot the rest
m=nvert(1);%+sum(nvert(2:2799));
for j1=2:size(nvert)
% for j1=2800:3072%3073:3328%size(nvert)
   pts=zeros(nvert(j1)+1,3);   %+1 to join the start & end points
   for j2=1:nvert(j1)
      m=m+1;
      if (j2==1)
         pts(1,:)=xyzp(m,:);
         pts(nvert(j1)+1,:)=xyzp(m,:);
      else
         pts(j2,:)=xyzp(m,:);
      end %if
   end %for
   plot3(pts(:,1),pts(:,2),pts(:,3),'k');
end %for

hold off

%set the axes limits
```

```
axval=6;
axis([-axval axval -axval axval -axval axval])

%label the axes
xlabel('X (\mum)','FontSize',10)
ylabel('Y (\mum)','FontSize',10)
zlabel('Z (\mum)','FontSize',10)
```