Developing Particulate Layers for a Light Scattering Model as Potential Means of Interpreting Remote Sensing Data of Planetary Surfaces

Mica Goldstone

Submitted to the University of Hertfordshire in partial fulfilment of the requirements of the Degree of MSc by research.

June 2013

Abstract

The study of terrestrial surfaces (regolith) of celestial bodies on a large scale is only realistic if conducted remotely. One method of achieving this is through investigating the directional properties of light scattered from them. This light contains information that can be used to interpret their origin, structure and taken as a whole, the evolution of the solar system.

Previous studies into photometric properties of light scattered as a function of phase angle, with particular attention to the feature known as surge is reviewed. Surge is where there is a sudden increase in brightness of an object when the backscattering angle approaches the source of the illumination. For celestial objects this is only seen when the celestial body in question is in opposition, i.e. the Earth is directly between the celestial object being studies and the Sun. It is therefore also known as the opposition effect. Features such as shadow hiding and coherent backscatter are explored and explained as are their individual contributions to surge.

There is some discussion given to the merits and inherent flaws taken by previous studies of this effect, namely the empirical best fit approach compared with various computational methods in determining probable surface features.

This thesis explores a method of simulating light scattering by particulate layers. This differs mostly from previous computational work concerning coherent backscatter in that this uses non-spherical particles as opposed to spheres (Mishchenko et al.) or aggregates of spheres (Litvinov et al.). The method is tested against backscattering measurements on snow layers by Kaasalainen et al.. For this purpose model layers have been constructed according to the snow layer descriptions (dominating crystal shape, mean snow/grain size, layer density). According to private communication with the authors, ice crystals are assumed to be randomly aligned. The method employed here is similar to the one described by Shkuratov at al.: Particles are randomly distributed in a cuboid, the lateral and bottom sides of which are cyclically closed.

Light scattering by these model layers and the resulting phase functions are computed by a ray tracing model in which incident rays have a given finite diameter and are associated with the electric field (not irradiance). Incident rays are arranged in a regular array covering the projected cross section of the layer. The sum of the cross sections of the normally incident rays is equal to the projected cross section of the layer. Diffraction is calculated for individual rays leaving the layer and the complex electric field values are added to the respective angular bins of the electric field distribution. In this way, the diffraction integral of a facet or sub-facet area is approximated by the sum of the diffraction integrals of the individual rays leaving the facet/ sub-facet area. When ray-tracing has finished the phase function is calculated.

Comparisons of modelling results with measurements have been carried out for two samples: one of compact hexagonal columns and one of thin plates. For comparison with the experimental results, the same four-parameter empirical fitting function as in Kaasalainen et al. was used. The phase functions of the compact particles can be compared directly. Similarities were found in the case of the improved seeding technique for needles and compact columns, though analysis of the field measurements and images presented in the Kaasalainen papers has called into question the similarities in the similarities between their samples and the simulations.

For non-compact particles such as hexagonal plates and needles the layer densities measured in the experiment could not be achieved for the model layer. This also enhances the problem of incomplete randomisation due to the limited layer cross section area (which is probably also the reason for the strong oscillations in the measured phase function of the compact hexagonal columns). Since particle size, shape and alignment distributions affect the width of the backscattering peak, it would have been helpful, if more information about these properties had been available.

Contents

1.0 Introduction

There is information within individual wavelengths of light scattered from regolith (surface soil) that if properly interpreted potentially reveals the structure of the surface such as size of the particles (grains of dust, ice, sand etc) that form the regolith, probable physical shape of these particles and their opacity and layer density.

How scattered light can contain this level of information relies on the fact that given a large enough sample (such as light being scattered from an asteroid), light undergoes scattering by multiple particles. This produces a scattering matrix which is an amalgamation of the individual interactions integrated over the probe's cross section. This cross section can be anything from a moving laser spot 4mm across used to study snow samples, up to the Hubble Space Telescope (ACS/HRC) (Showalter et al. 2008) for observing the rings of Saturn and Jupiter.

While it is not feasible to collect samples from remote celestial bodies so as to compare them with the data collected by telescopes, it is feasible to look at terrestrial samples and record the light scattering from them. This allows for the cataloguing of phase functions corresponding to the regolith. This phase function corresponds to normalised intensity of the scattered light as the angle between the light source, the regolith and the observer changes on account of the changing position of the observer while the angle between the light source and the plane of the regolith remains constant, i.e. its angular distribution and can therefore be described as its phase function. Where known phase functions from terrestrial samples are the same as those from remote objects, it is reasonable to presume that there are similarities between the terrestrial and celestial regolith.

The question arises as to the nature of surface structure of the remote celestial object when it does not match terrestrial samples from the catalogue. In these circumstances there needs to be a means of generating accurate phase functions for theoretical particles. This requires an understanding of the mechanism by which scattering curves are created and a means by which it can be recreated through simulations.

From basic principles, programs have been developed that use geometric optics to simulate the path light takes as it passes through a medium. Geometric optics achieves this by treating light as rays. As computer processing power has increased, these have become ever more powerful, increasing the quantity of light rays traced through a particle in order to produce a phase function and scattering pattern. Where a scattering pattern is simply a two dimensional representation of a phase function.

The natural progression of this development is the application of the program to multiple particles of varying sizes and packing densities in order to observe the scattering pattern. By choosing simulated particles that approximate naturally occurring material such as ice crystals of which the phase functions corresponding and ensemble/layer properties are already known, both the program can be improved and we move one step closer to being able to determine the probable structure of remote celestial bodies.

This research concerns itself with the modelling of a layer of regolith based on known samples of ice. Central to this is the work already carried out in the field by Kaasalainen et al. They sampled snow on various occasions and catalogued the phase functions against the structure of the snow. The structures included simple irregular particles, hexagonal columns, hexagonal plates and needle structure. They also recorded, density, average particle size, the depth at which the particles were collected and the degree of sintering (particles that are less regular and may even have fused into each other).

The primary issues with the project are attempting to create enough particles within a volume such that three aspects are satisfied:

- The particles do not intersect, i.e. they do not occupy the same space.
- The density of particles is consistent with known samples.
- The particles have the same ranges of structures as known samples (particle dimensions and shape).

The work has been carried out using MATLAB, a programming package well suited to dealing with mathematical problems. Use has been made of existing functions such as generating normalised random numbers and evaluating whether a line passes though a bounded plane. The language is also very good at dealing with arrays, negating the need for nested loops, i.e. do this process to that array rather than for each value on a line and for each line in the array, do this process.

An issue that does occur due to the nature of the problem is that arrays are continuously changing in size, e.g. an array of particle locations increases each time a new particle is added. This makes for code that is not optimized. Another feature of the code is that achieving higher densities requires ever more processing time as rejections due to overlapping occur more frequently.

While the bulk of the paper deals with the steps taken to generate the simulated regolith (referred to as a layer), the first part deals with a history of the subject and explaining the processes involved in detail.

Applications of this research lend themselves to remote observations of celestial bodies but also for terrestrial uses such as studying climate change. Satellite data of ice fields and deserts can in theory be analysed to determine how the surface changes over time, such as changes in average particle size corresponding to fresh snow verses old snow, removal of finer sand grains due to increased winds. By making assumptions about the refractive index of the layer particles, it may even be useful in determining the nature of chemicals detected through emission and absorption spectra. Other uses may include studies of the atmosphere looking at aerosols, levels of dust and types of ice crystals present.

2.0 Model Development Review

While theoretically, solutions to Maxwell's macroscopic equations are all that is required to describe all observed phenomena relating to light, achieving this for anything other than idealised situations is impractical due to the complexity of the calculations for even simple realistic circumstances.

Direct solutions when they are used invariably presume ideal conditions (e.g. homogenous media composed of particulate spheres) and are therefore unrealistic or require a phenomenal amount of computer processing. Geometric optics (and related methods) generally ignores certain phenomena such as interference or only produce reasonable results under specific conditions (particle to wavelength ratio: see 2.3). Another approach taken by Hapke looks at the empirical data in order to produce best-fit formula from laboratory and field observations and application of electromagnetic theory in order to interpret observed phenomena.

2.1 Light Scattering in Brief

Early work into the field of scattering made some simple assumptions such as considering the scattering material to be composed of spheres for which an analytical solution for Maxwell's equations can be produced (Mie theory). These models however cannot be used to reproduce phenomena such as halos and pillars.

Later work improved on this through the use of non-spherical particles (various techniques are summarised in Mishchenko – Light Scattering by Nonspherical Particles). This body of work derives methods of solving Maxwell's equations which are restricted to small size parameters ($k = 2\pi a/\lambda$ – where a is the characteristic particle size, i.e. the semi-major dimension of the surface or volume equivalent (in the case of spheres this will be its radius) and λ is the wavelength of light in surrounding medium), due to the high computational expense except when using the high symmetry case of spheres.

2.2 Backscattering Phenomena

2.2.1 Shadow-Hiding

This occurs where the surface being subjected to incident light is not perfectly smooth. The observer will see shadows created by the individual particles and objects with protruding features that form the surface. Only when the incident of light originates from the same direction as the observer with respect to the surface, will the shadows of the objects (and particles) be completely covered and therefore hidden from the observer, hence the term shadow-hiding.

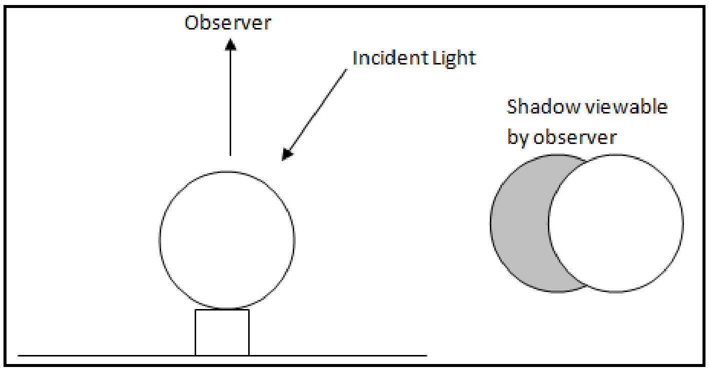


Fig.2.1. Shadow observed when there is an angle between the observer, the object and the incident light less than 180^{o}.

This corresponds to direct backscattering direction, i.e. angle of illumination is 0^{o} while the backscattering angle is 180^{o}. As the angle of observation changes from

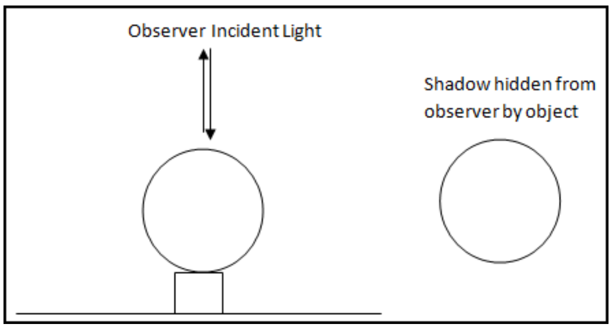


Fig.2.2. Shadow hiding occurs when the shadow is not observable, i.e. observation angle is 180^o.

the angle of direct backscattering on account, shadows become more pronounced and the backscattering brightness of the regolith diminishes.

This can be seen in fig.2.1 and fig.2.2. In fig.2.1, a crescent shadow is observed for the angle of approximately 135^o. In fig.2.2 with an observation angle of 180^o the entire shadow is hidden from the observer. As the angle between the direction of incidence and observation increased between the two diagrams the crescent became smaller and therefore the overall brightness increased.

Albedo is the ratio of reflected/scattered solar energy to incoming energy over wavelengths of approximately 0.3 to 2.5 micrometres. Values range from 0 (perfectly black) to 1 (perfect reflector/scatterer). It plays a role in shadow-hiding due to the opposition effect being dramatically larger for low albedo materials. This is because materials with low albedos absorb more light leading to less scattered light and therefore producing darker shadows.

Early models of inhomogeneous layers of particulate media (Stankevich, Shkuratov & Muinonen, 1999) used a geometric optics (GO) modelling approach to produce calculations of photometric characteristics with accuracies better than 1%. These indicated for statistically homogenous particulate media, packing density was the single parameter that characterised the shadow-hiding opposition effect.

It has been known for a long time that as the viewing angle of the rings of Saturn changed due to observing them from Earth, there is a sudden spike in the reflected brightness at specific times (Seeliger, 1895). This surge is due to backscatter but more importantly, it is not directly proportional to the viewing angle of the incident light reflected back towards Earth and the tilt of the rings. It was observed that the surge was confined to a small angle.

This opposition surge was also noted for the lunar surface. It has been observed that the brightness of the moon increased by nearly 40% between about one day before a full moon and the time of the full moon. The studies (Hapke, 1998) indicated that the reason could be explained by two potential mechanics: Shadow Hiding and Coherent Backscatter. Together these provide an important tool in remote sensing for determination of porosity and mean free path of photons in the regolith.

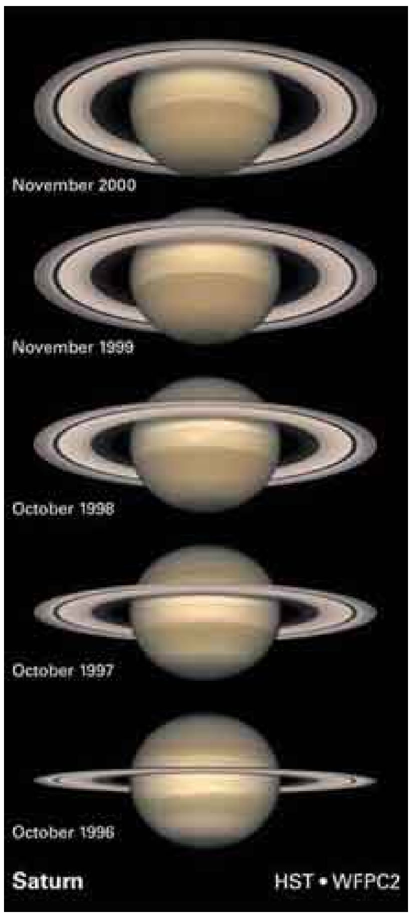


Fig.2.3.
http://spaceplace.nasa.gov/saturn-rings/

During the Galileo’s G7 orbit of Europa, similar discoveries were made (Helfenstein et al., 1998) though the brightness spike varied with the type of terrain observed.

2.2.2 Coherent Backscattering

Where a particle is non-absorbing, or at least only partially absorbing, part of the light will be reflected and another part of the light will be refracted into the particle and undergo further reflection/transmission events at the particle surfaces.

An interesting feature of this discussed below, is that given the ‘effectively infinite’ number of discrete layers that form a regolith, it can be proved that virtually all the light incident on the surface will be reflected or refracted out (in the case of ice or other optically transparent media). Of this light being scattered back towards the incident light, it is found that it generally becomes polarised to some degree (depending on the nature of the scattering material). At angles close to opposition, it is scattered and undergoes constructive interference producing coherent light and as a consequence there is a brightness surge.

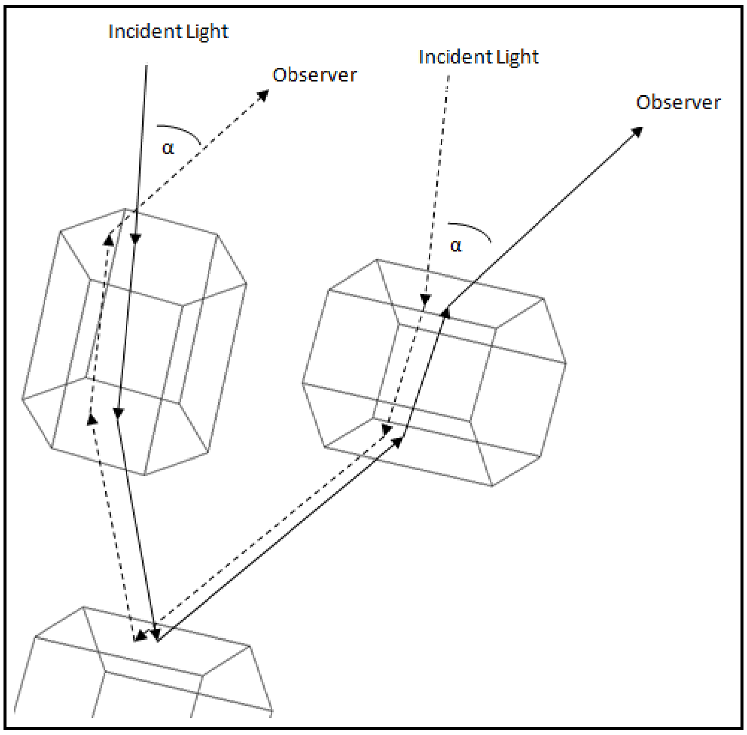


Fig.2.4. The dotted path is conjugate of the solid path. Shown in this diagram for three particles, this can be applied to any number of particles in the path.

In the diagram the two sets of arrows represent paths followed by incident light. The dotted arrows follow the conjugate of the solid wave path from the illuminator to the observer where the observer is a long distance from the scattering medium.

As the particles are randomly aligned, if the angle (α) between the light source, regolith (or scattering medium) and the observer is large, the effect of interference will average out between the pairs of conjugate scattered waves. Scattering will therefore be incoherent and there will not be a brightness surge due to coherent interference.

As the angle (α) tends to zero the phase difference between the conjugate paths though any sequences of particles also tend to zero, resulting in coherent interference.

2.2.3 Polarisation Opposition Effect

Light backscattering from a planetary surface has been discovered to be negatively polarised at small backscattering angles. It is a sharp asymmetric feature with a minimum at a phase angle comparable to the angular semi-width of the opposition brightness surge.

Unpolarised light can be considered to be both positively and negatively polarised by the same amount. As such as it passes through a medium it passes along conjugate pairs of paths as shown above. Positively polarised conjugate paths between particles in the scattering plane have their phase angles changed while those scattering through particles perpendicular to the plane, i.e. negatively polarised conjugate path do not. As a consequence, there is enhancement of the negatively polarised paths over a much wider range of phase angles.

While unpolarised light is both positively and negatively polarised as it passes through a medium, the phase angles of the positively polarised conjugate paths (between particles in the scattering plane) changes while for conjugate pairs negatively polarised rays (between particles perpendicular to the scattering plane) it remains zero. This results in enhancement of negatively polarised trajectories over a wider range of phase angles than positively polarised trajectories. Further to this, only certain particle configurations contribute to polarisation opposition effects.

The lunar surface for example exhibits prominent negative polarisation (Shkuratov, 2002).

2.2.4 Size Parameter

Kuga and Ishimaru (1988/9) studied backscatter as a function of density and spherical particle size and determined that backscatter enhancement could be achieved through different mechanisms. They termed the mechanisms Type I and II and defined them as:

Type I - coherent backscatter caused by light rays travelling along conjugate paths and is strongest for relatively small particles with size parameter $1<k<20$ and layer densities > 2.5%.

Type II - caused by the focusing effect of large particles in a disperse medium.

2.2.4.1Backscatter as a function of density for large size parameter

Their experiments with particles of very large size parameter *k*(>10,000) where $k=2\pi a/\lambda$, λ is the wavelength and *a* is the radius of the particle revealed a relationship between the density of the media and the intensity of the backscattering peak. As can be seen in the graph, as the density of the media decreased from 19.1% (*A*) down to 0.149%(*H*) the relative intensity of the backscatter peak increased.

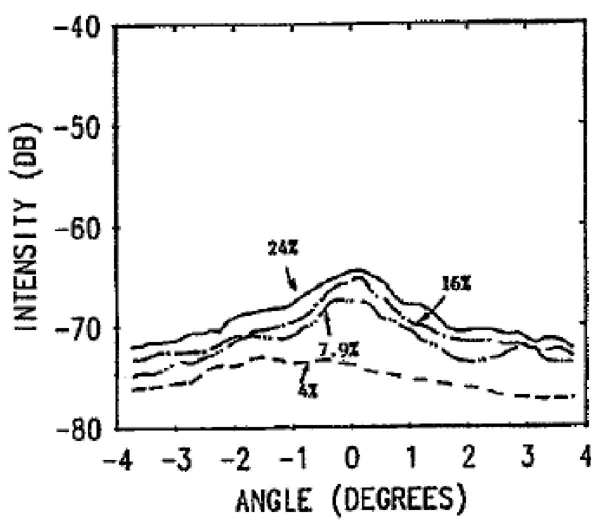


Fig.6 Backscattering enhancement by large particles. Particle size is 5 mm.

Fig.2.5. Kuga& Ishimaru. SPIE927(1988)33-37

Comparisons of experimental data were made with solutions for Second-Order multiple scattering (as this is suitable for large size parameters in the near backscatter direction). When compared the shapes of the predicted curves and those achieved experimentally were very close except for the back scatter peak. As coherent backscatter is not accounted for Second-Order multiple scattering theory and the peak obtained experimentally was the single feature missing from the solutions, they concluded that the peak must be due to coherent backscatter. The intensity at wider angles however is due to incoherent scattering.

2.2.4.2 Backscatter as a function of size

Kuga and Ishimaru (1988/9) looked at another type of backscatter they termed Type II, caused by the focusing effect of particles in a disperse media. They studied glass beads (5mm size) in a solution of water, varying the density of beads. In these circumstances they concluded that the focusing effect of the beads and its reflection was responsible for the increased intensity in the back direction. The key element in their studies was the average distance between the spheres suspended in the medium. The brightness surge (backscatter enhancement) was greatest where the average distance between the beads was equal to the focal length of the beads. This is related to the size parameter. In this case the density of the medium was important as the higher the density the greater the chance of two beads being within the focal distance of each other. This is indicated in the adjacent graph for various densities where the size parameter is greater than 10,000 and as such the brightness surge in the back direction cannot be attributed to coherent backscatter. Their experiments did not include samples of particles of different shapes. It cannot therefore be confirmed as to what effect this will have on Type II backscatter enhancement. It is however predicted that for non-spherical shapes such, there will be reduced enhancement simply on account of the lack of common alignments leading to focusing.

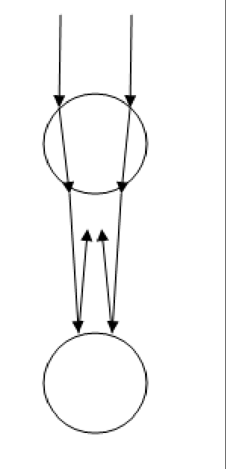


Fig.2.6. Type II Enhancement

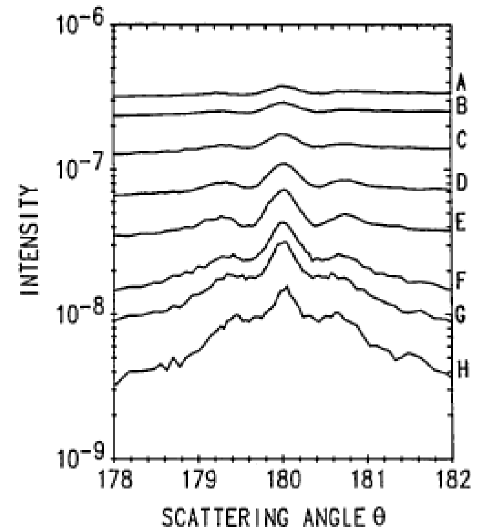


Fig. 3. Cross-polarized intensity for different volume densities: $A = 19.1\%$ ($\tau = 253$), $B = 9.55\%$ ($\tau = 126.5$), $C = 4.775\%$ ($\tau = 63.3$), $D = 2.39\%$ ($\tau = 31.6$), $E = 1.19\%$ ($\tau = 15.8$), $F = 0.597\%$ ($\tau = 7.9$), $G = 0.298\%$ ($\tau = 3.95$), $H = 0.149\%$ ($\tau = 1.98$).

Fig.2.7. For size factor 33,000. Kuga& Ishimaru. Applied Optics 28(1989)2165-2169

2.2.4.3 Summary of Kuga and Ishimaru Findings

For size parameters:

- 1<k<20 Strong Type I enhancement and most visible when the volume density becomes more than 2.5%.

- k≈50 Broad peak due to Mie scattering (single scattering) and sharp peak due to the backscattering enhancement Type I. Sharp peak appears on top of a broad peak when density greater than 10%
- k≈300 Mie region of scattering has a sharp peak and possibly create a focusing peak due to Type II enhancement. Difference between Mie (single) scattering and coherent backscattering is that Mie scattering decreases as volume density increases due to incoherent intensity increases due to multiple scattering.
- k > 10,000 Backscatter enhancement due to Type II enhancement (for spheres).

2.3 Geometric Optics

For particles large compared to the wavelength, one method by which backscatter can be modelled is by Geometric Optics (GO). This utilises rays, traced from a source to a screen or observer as a means of modelling the behaviour of light being emitted and detected. A light ray is a line drawn in space corresponding to the direction of flow of radiant energy. They propagate along rectilinear paths where the media is homogenous though can bend or be split in two at interfaces between two dissimilar media, obeying Snell's Law (derived from Fermat's principle of least time). Fundamentally, the rays passing into a media with a higher refractive index(*n*) will be bent towards the normal (and vice-versa) and the angle between incident ray and the normal is equal to angle of the reflected ray and the normal as shown in the diagram below.

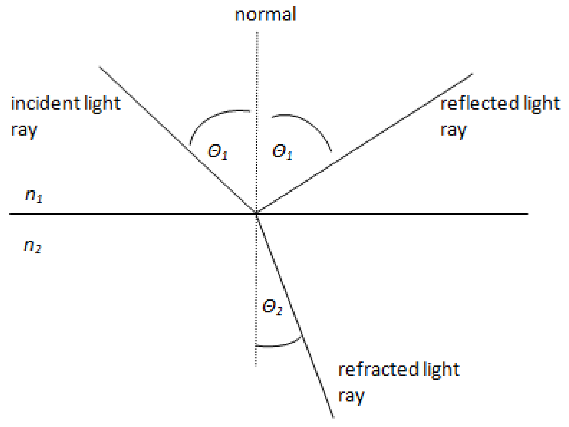


Fig.2.8. Refraction and reflection of light ray

Snell's law is expressed by the equation:

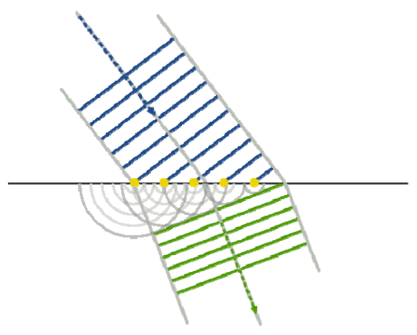


Fig.2.9. Light hitting an interface between two media http://en.wikipedia.org/wiki/File:Refraction_-_Huygens-Fresnel_principle.svg

This mechanic works very well for describing how lenses focus light and produce images, describing observed phenomena such as inversion of the image, magnification and diminution of the image and even virtual images.

Where the wave length of the light is small compared to the size of the objects, this mechanic is very good at describing how light is propagated and has been used in early models of media formed from particles (Stankevich, Shkuratov & Muinonen, 1999) This though was for homogenous particulate media.

In order to deal with other naturally occurring phenomena and situations where the wavelength of light is closer to the size of the particles it is interacting with it however needs refining.

In order to understand the phenomena, a plane light wave can be described as a series of point sources along the propagating wave front (Huygens-Fresnel principle) and that at any time after this the new wave front can be described as the product of the superposition of the individual wave fronts from these points. This demonstrates how light is bent as it passes through from media to another in the adjacent image.

The principle can be used to explain the phenomena of light and dark bands where light is diffracted around the edge of an object that is partially blocking the light or where light passes through an aperture. As can be seen from the image the new wave fronts are a formed from the wave fronts of the individual points. The points on the edge of the aperture are responsible for the light bending round the corner of the aperture. Where light is partially blocked by an object the points of the wave front on the edge of the object indicate how light is diffracted around the object.

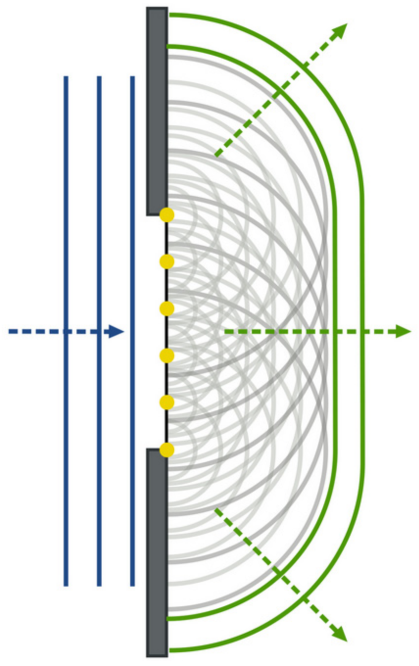


Fig.2.10. http://en.wikipedia.org/wiki/Huygens%E2%80%93Fresnel_principle

While this principle accounts for the bending of light at edges, it does not consider superposition, the interference of the light rays from the infinite quantity of point sources that theoretically lie along the wave front. The treatment of these both in the near field and the far field needs to be considered.

Near Field

Near field as specified by the Fresnel number $F\ (=a^2/L\lambda) \geq 1$, where a is the size of the aperture (or object around which light is being diffracted), L is the distance of the observer from the aperture and λ is the wavelength of light. Fresnel diffraction produces a series of light and dark bands in the near field.

Far Field

Far field diffraction is modelled using Fraunhofer diffraction equation. The far field is as the name suggests a long distance from the aperture (or partly obscuring object) and occurs when the distance is large enough such that the difference in phase between the light from the extremes of the aperture (or opposite edges of the obscuring object) is much less than the wavelength. As such individual contributions can be treated as though they are parallel. Numerically it is defined as $a^2/L\lambda \ll 1$. In the case of studies of ice crystals and particles only a few millimetres in length, the far field is effectively any distance greater than a metre.

2.3.1 Ray Tracing

As the ray hits each interface, in this case a facet, either into a particle or out of a particle, it will in reality be refracted and reflected. There are different approaches to how the rays are tracked through the medium and to the observer/collector.

2.3.1.1 Probabilistic Method

Rather than attempting to keep splitting the ray into two components representing the relative amplitudes of the reflected and refracted ray produced at each and every interface, a probabilistic approach was taken (Shkuratov & Grynko, 2005). This significantly cut down on computer overhead which would have required up to two rays to have been followed after every single interface where both refraction and reflection occurred. The probabilistic approach simply followed either the reflected or refracted path based on a random chance weighted by the relative amplitudes of the two possible rays. For example for light with intensity *I*, if the $I_{reflected} = 0.7\ I_{incident}$, there was a 70% chance that the program will follow the reflected path and 30% it would follow the refracted path.

2.3.1.2 Follow all Paths

Another approach is to simply follow each new ray as it is generated through refraction. This approach requires a lot more computing power as a consequence. Limits are set to the quantity of rays that will be generated based on the intensity of the ray compared to the intensity of the parent ray. It is generally found that there is a sharp increase in rays as the rays undergo first ray surface interaction but after a time as the intensity of the rays diminishes and the rays leave the layer, the quantity decreases. In this method there will always be energy left within the scattering medium as paths are terminated without leaving the layer. This is a much less simplistic approach than the probabilistic method.

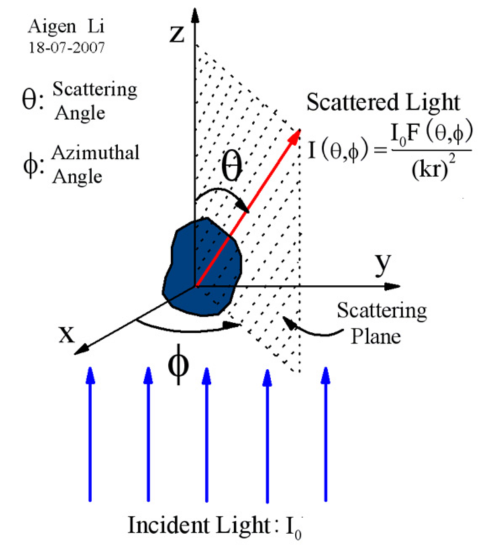


Fig.2.11. F(ϑ,φ) is the (dimensionless) angular scattering function, r >> λ/ 2π is the distance from the scatterer, and k = 2 π /λ is the wave number in vacuum

2.3.2 Diffraction on Facets

It was found that standard GO far field patterns *I*(θ,ϕ) (see diagram) produced from the passage of light through a single particle tended to consist of a series of dots, each representing where the ray finally ended. There is also a disagreement in phase functions of single, randomly aligned particles with planar facets (crystals): GO predicts sharp spikes in direct forward and backscattering and saw-tooth like halo-peaks. This though was often a poor comparison to both the scattering pattern produced through laboratory experiments and results generated by the exact but computationally expensive separation of variables method (SVM).

In order to improve the standard GO model, diffraction on facets was introduced (Hesse, Ulanowski, 2003). This worked on the principle that where the wavelength of light was comparable to the diameter of an aperture (in this case, a facet is the equivalent to an aperture), diffraction started to play a significant role in the transmission of light through a particle.

To accommodate it, the model took into account where on the facet the ray interacted with the particle. By considering a facet in two perpendicular dimensions as a slit, the new path of the ray following its refraction or reflection at a facet would then be deflected towards the nearest edge of the facet. The degree of deflection increases as the distance to the edge of the facet decreases.

The energy flow calculations were initially carried out for a half-plane, for which an exact solution of Maxwell's equations exists (Braunbek & Laukien). As this effectively mapped the energy flow through intersection points with the plane containing the half plane at any given distance from the slit edge, it could be applied to discrete quanta of energy, i.e. photons. By determining the propagation of energy the effective far-field deflection angle could be calculated with respect to the direction of incidence.

The formula below was constructed using the formula obtained for a half-plane.

$$\varphi(x) = \ arctan\left(\frac{\lambda}{4\pi^2}\left(\frac{1}{x} - \frac{1}{2a - x}\right)\right)$$

Where ϕ is the far-field deflection angle and x is the distance from the edge of the facet and $2a$ is the width of the facet (slit). In this case, where x = a (the middle of the facet), there was zero deflection, whereas at the edges ($x \ \rightarrow 0, x \ \rightarrow 2a$), deflection was maximum.

This effectively improved of the basic GO which ignores diffraction.

Instead of producing dots, the rays become more dispersed and the final scattering image produces results that were closer to experimental results and SVM model for phase function and than the standard GO method without unduly increasing the computational demands.

2.3.3 3D Implementation

As initially only the nearest facet edge had been taken into account and the degree of deflection calculated using an approximate relationship obtained from the exact half-plane diffraction theory as applied to a slit (with the same dimension as the facet), the above method produced only diffraction in two dimensions. This approach was expanded (Clarke et al. 2006) into three dimensions by applying two deflections each obeying the basis of the 2D rules, regardless of facet shape. The first deflection was always towards the nearest facet edge and the second was perpendicular to the first.

2.3.4 Further Improvements to the diffraction solution

A feature of the model developed this far was a singularity of the far-field deflection angle at the centre of the slit. This caused an overestimation of the number of raypaths contributing at very low deflection angles. To improve on this shortcoming, Hesse (2008) utilised the approximation developed by Prosser (1976) for Fraunhofer diffraction by a slit (which approximates forward scattering much better than the equation above) and derived best-fit formulas.

These produce depleted angular density of energy flow lines in the far field around the angular position of the first minimum for the Fraunhofer diffraction pattern as well as weaker reductions at higher-order diffraction minima in accordance with Prosser's statement regarding redistribution of energy.

The main advantage of this was to improve on standard GO when approximating the results of SVM over the whole angular range without significantly increasing computational overhead. The greatest improvements were at near-direct forward and backscatter, in the halo region and in the backscattering region between 142^{o} and 160^{o}.

2.3.5 Application of the RTDF to particles with curved surfaces

Up to this point the model was only effectively applicable to faceted particles. In many cases however ice melts then refreezes, creating irregular particles that have curved as well as faceted sides that are classified as 'hybrid' particles. Added to this, an aggregate may also contain dust grains that have curved surfaces.

The method used to simulate the curved surface (Hesse et al., 2009) was to treat them as having multiple faceted surfaces. In essence, a curved surface could be treated as having an infinite number of facets however for practical purposes the approach taken was consider a sphere for example as having a fixed number of trapezoidal facets with triangular ones at the 'poles'. Models were made and tested for spheres having 800, 1352 and 2592 facets, corresponding to 9^o, 7^o and 5^o angles between the normals of the adjacent facets respectively. Comparison of the results (especially for the 2592 facetted spheres) with T-matrix and Mie theory (for spheres) produced better results with respect to exact calculation than GO over the whole angular range and in particular in the near-exact forward and backscattering, and in the halo region.

2.4 Aggregate Model

Ray Tracing with Diffraction on Facets (RTDF) requires a medium to interact with therefore in order to apply RTDF particle models, virtual particles must exist.

Baran and Labonnote (2007) developed an ensemble model of ice crystals. The simplest and smallest crystal consisted of a single hexagonal ice column (these form the basis for the research detailed in this research). Larger aggregates were obtained by arbitrarily attaching other hexagonal columns or plates, becoming more complex. The largest in the ensemble was a chain-like crystal. The ensemble consisted of six ice-crystal members and was tested, generally predicting the ice water content and extinction measurements to within a factor of 2. It was also determined that more simple members of the ensemble were able to produce GO results consistent with measurements of polarised reflection taken by satellite instruments.

3. Physical Experimentation/Observation Review

3.1 Ice Analogues

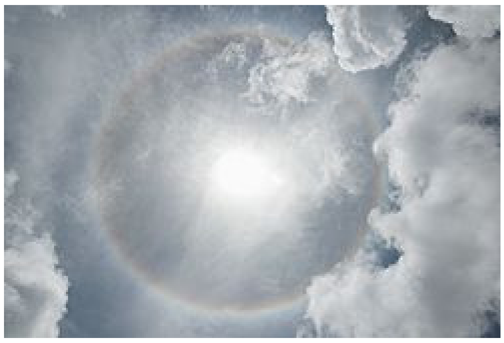
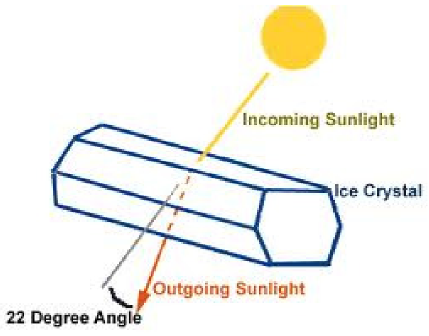


Fig.3.1. Images showing the 22 degree halo as light from the sun is scattered through cirrus clouds and corresponding ray tracing.



Ice analogues, grown in solution (Ulanowski et al., 2006) were shown to have close agreement between the measured functions and the analytic phase function for ice clouds on account of their near identical refractive index at optical wavelengths. The analytic phase function is a linear piecewise parameterization of the Henyey–Greenstein phase function generated by assuming some value of the asymmetry parameter. The experiments showed the 22^{o} halo peak for smooth rosettes and aggregates (but not for rough rosettes).

The results suggested that it would be possible to use such 2D scattering patterns to discriminate not only between crystals of different shape but also to obtain some information on surface properties and that 2D scattering could be used as a basis for determining classes of ice crystals. Supporting or alternative discrimination could also be provided by polarisation measurements.

Figs. 3.1 show a 22^{o} halo observed at cirrus clouds and a schematic demonstrating the formation of this halo, which corresponds to the minimum angle of deviation at a 60 degree prism.

3.2 Field Measurements of Ice Backscattering

Work by Kaasalainen (2006) established angular intensity distributions close to and including direct backscattering by snow. This determined key factors in the process, these being temperature (related to the changes in the grain structure) and grain shape and size. Interestingly enough increasing packing density of the snow only increased the surface brightness. The data produced is instrumental in determining whether the ice analogue described above to be used in the future modelling is sufficiently accurate enough to allow it to be used as a reference.

3.2.1 Kaasalainen Method of Data Collection

The goniometer employed by Kaasalainen et al. works through recording intensity against scattering (phase) angle for a sample. In the case of fresh snow needles, while snowing, surface snow was collected in a sample cup and snow was allowed to fall onto the surface of the sample. Other samples were collected by careful scooping. A series of 5 readings (shots) were collected using the charge-couple device (CCD) detector with the sample generally being rotated between series of shots. Once the shots were taken, the angle of the beam splitter was changed and the process was repeated. Instabilities in the apparatus along with speckle from the surface and other sources of noise meant that even when the sample was not rotated, the intensity from sequential shots would not be exactly the same. In cases where speckle (presumably reflection from a single facet from a surface crystal) significantly increased the intensity, the goniometer was rotated minimise it. "The sample was rotated so that a larger part (than just the laser spot size) of the sample would be represented, and also because that is one way to minimize the laser speckle effect" (Sanna Kaasalainen). Environmental conditions such as conducting the sampling while it was snowing played a role in increasing noise in the data.

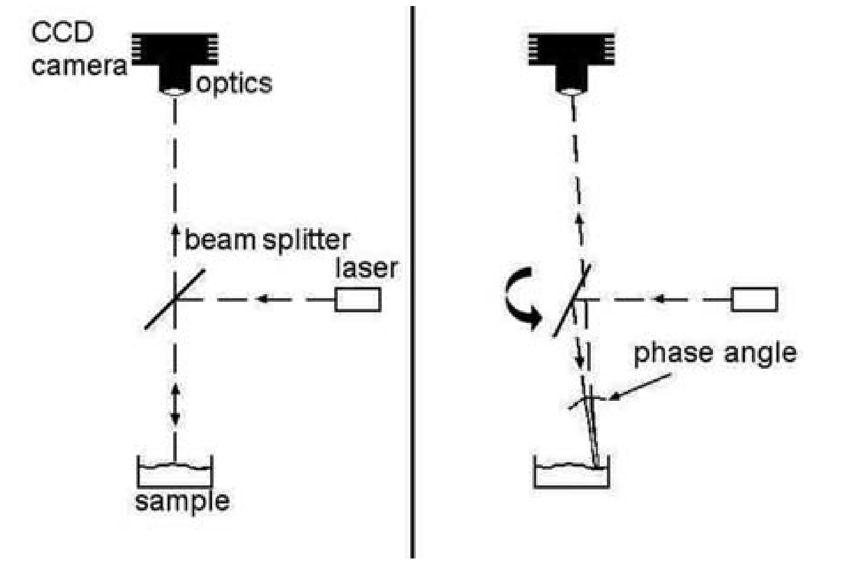


Fig.3.2. Small-angle goniometer employed by Kaasalainen (JoG vol52. 2006). The non-zero backscattering (phase) angles are obtained by rotating the beam-splitter.

The error bars in the data plots represent the standard deviation of the readings from the mean for the 5 shots. Increasing the number of shots to 10 did not improve errors and after testing by a team member the practise was abandoned.

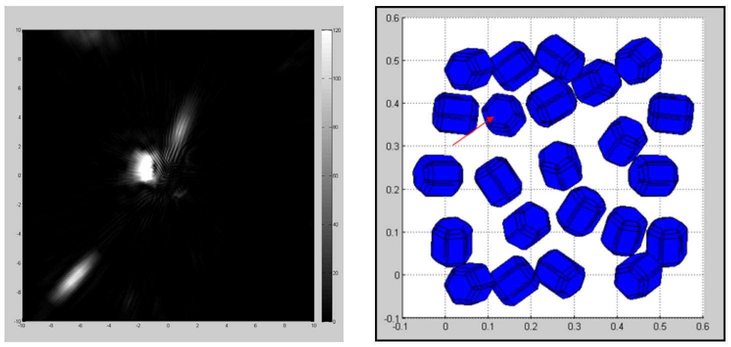


Fig.3.3. Glare in output is probably attributed to end facet (indicated in render with a red arrow) oriented towards observer.

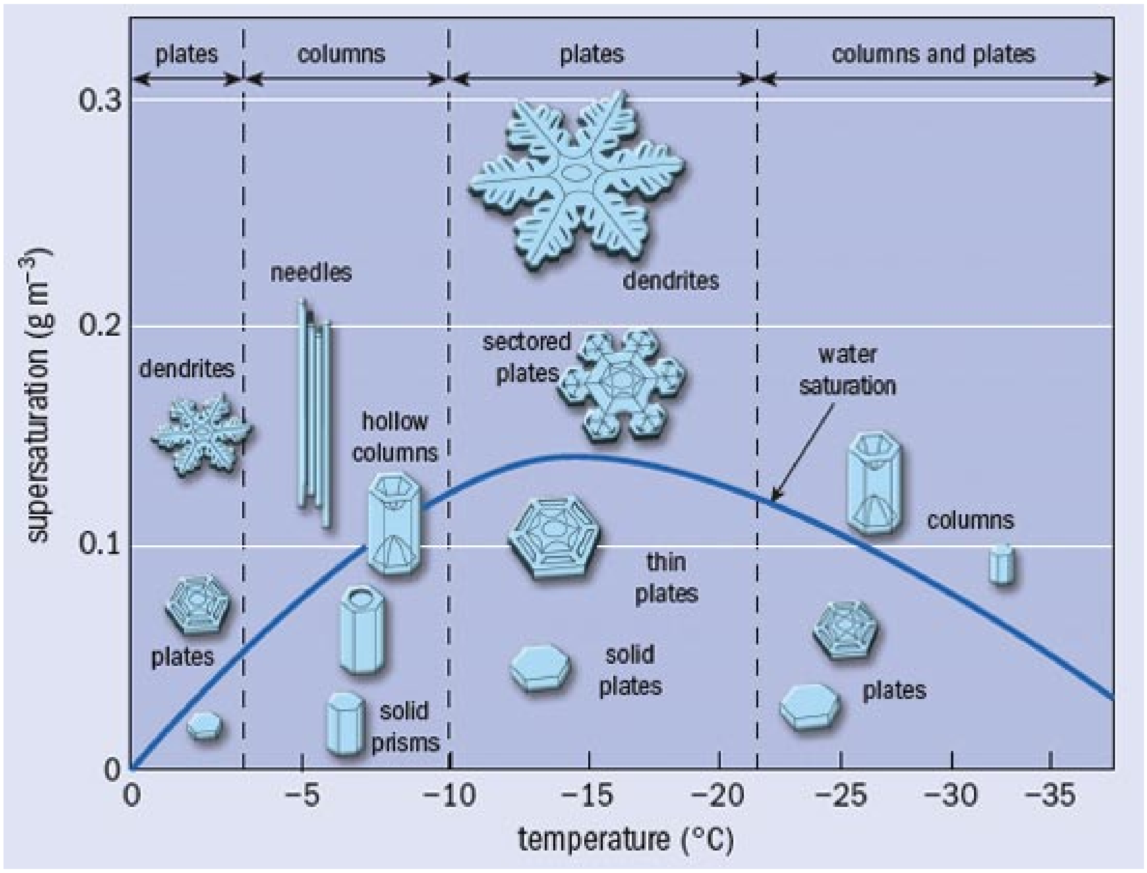
The beam splitter was adjusted so as to collect data from both sides of the central intensity peak, corresponding to backscatter (phase) angles greater than and less than 180^o (0^o). This was done to investigate symmetry about central intensity peak. In theory this is simply the equivalent to rotating the sample through 180^0 degrees while keeping the backscatter angle the same.

"[T]he fact that the error bars are different size is from the fact that specular reflections occur at some points, because of the random orientation of the grains" (Sanna Kaasalainen).This is in line with domination of the output by surface reflection from a few crystals such as in this case opposite.

Weather conditions such as extremely low temperatures along with precipitation and winds made both the collection of samples and data measurements difficult while the probability that more accurate sampling and data collection is not deemed likely in the future without a significant improvement in technology.

3.22 Images

The images presented in the 2006 paper were small and as a consequence it was not possible to determine differences in the classification of crystal snow used within the field. Access to the original images however have now provided considerably more information and when combined with the expanded details on the techniques used to collect data, highlights fundamental considerations both on the expectations of any model and even forming a consensus on how to classify snow sample from the diversity of crystal shapes that can be present. The diagram below indicates the type of snow crystals that can manifest based on the environmental conditions. Melting and refreezing of the crystals along with the presence of less than pristine crystals will all contribute to the phase function of the sample. Details of the images for samples corresponding to the simulated models investigated in this thesis are considered in 4.11.



Snowflake Morphology

Fig.3.4. Snowflakes can be characterized by the snow-crystal morphology diagram, which shows crystal shapes as a function of the temperature (x-axis) and humidity (y-axis) in which they grow. Here the humidity axis actually refers to the supersaturation — the excess water-vapour density above a humidity of 100%. The water saturation line (blue) shows the supersaturation that would be found in a dense cloud of water droplets. http://www.its.caltech.edu/~atomic/snowcrystals/primer/primer.htm

3.3 Astronomical observations of backscattering and how they apply to size, shape and refractive index of particles forming planetary regolith

In his work on the Opposition effect, Hapke (1998) takes the approach of separating the observed brightness surges caused by shadow-hiding and coherent backscatter and subjecting them to individual analysis. By the use of empirically derived formula, he achieved an analytical tool by which he could evaluate regolith in order to approximately determine porosity and transport mean-free paths of photons within it. In the paper there is what appears to be a developing difference of opinion between his body of work and that of Mishchenko, noting that in Mishchenko's work the mean-free path for photons was greater than the particle size, i.e. derived density was far too low, closer to that of a gas rather than an icy soil. Hapke admits that using either analysis is insufficient on its own to describe the observations made of the lunar surface. It was probable that opposition effects were due to a mixture of shadow-hiding and coherent backscatter.

3.3.1 Europa

Galileo's G7 orbit of Europa produced sufficient data that an initial evaluation of the regolith (Helfenstein et al. 1998) was able to indicate that shadow-hiding and coherent backscatter were responsible for the opposition surge. It was noted that stratigraphically young ridges with relatively pronounced topographic relief exhibited anomalously weak opposition surges in comparison to other Europan terrains that had similar spectral properties. From this it was concluded that surface probably consisted of course regolith grains rather than solid ice or compacted ice grains.

Data collected from the Cassini Visual and Infrared Mapping Spectrometer (VIMS) was analysed by a team led by Brown (2003). This revealed a surprisingly high opposition surge on Europa. Here they identify clear trends with albedos and surges such that higher albedos have lower opposition surges. An issue with this discovery was that under the applied mechanism, it would indicate a porosity of ~99%. This they reconcile with their inability to correct the data for rotational phase effects due to lack of published data. As such they concluded that neither coherent backscatter nor shadow-hiding provided a complete description of the Europa's opposition surge below 1^{o}.

This work was followed up by subjecting the data to the Hapke photometric function (Simonelli, 2004). Again it was largely concluded that surges at phase angles $< 1^{o}$ could not be explained by coherent backscatter and that shadow-hiding must play a significant component, even at wavelengths were Europa is bright.

Simultaneous photometric and polarimetric measurements of laboratory samples that simulate planetary regoliths were compared to computer models (Shkuratov et al., 2005). The research utilised a ray tracing model that had been previously developed (Stanevich et al., 1999). The two methods both produced negative polarisation and greater prominence of the opposition phenomena when the samples were compressed. Taken together they concluded that shadow-hiding was the main factor affecting backscatter at all phase angles though could not determine a direct contribution of the shadow-hiding effect to the formation of the negative polarisation branches. They supported the hypothesis that diffraction was a key factor in both polarisation and coherent backscatter. As they did not detect negative polarisation in fluffy fresh-fallen snow with course particles (Shkuratov et al., 2002) though did for metallic powders, it was implied that for metals there was an additional contributing mechanism.

Shkuratov's evaluation of Europa was that it was likely to consist of large icy regolith particles but with a subtle structure on micrometer-submicrometer scales.

An alternate method for modelling regolith and scattering has been to use direct solutions of the Maxwell equations (Mishchenko 2009). By numerically exact computation of electromagnetic scattering by media consisting of large numbers of randomly positioned spheres, demonstrations of the interference nature of specific backscattering effects could be made. This method it was claimed would eliminate any uncertainty associated with the use of an approximate theoretical approach; control precisely all physical parameters of the scattering medium, allowing one variable to be altered at a time; compute all relevant optical observables at once. For anything other than simple and ideal systems, this though is not technically possible. This could then be used to definitely answer the physical origin of brightness opposition effect and polarisation opposition

4. Method

The main objective of this thesis is to explore a method of simulating light scattering by particulate layers. Within this work, light scattering by snow layers will be modelled and compared with experimental results by Kaasalainen et al. In order to achieve this, a model of a virtual particulate layer has to be constructed that will simulate snow under various conditions. The model has to bear a resemblance to physical snow layers such as those studied by Kaasalainen et al.

The method taken to achieve this goal required a series of steps, building on simple models using MATLAB (Matrix Laboratory – a numerical computing environment) and adding sophistication. Essentially the creation of a virtual particle layer starts with defining the outer limits of the layer. The volume of this layer is then subjected to various methods of seeding bounding boxes. A bounding box is defined as a point within the layer surrounded by a volume such that no other bounding box shares the bounded volume. Once seeded, the total volume of boxes was compared to the volume of the layer in order to determine the density of the seeded mass (when applied to the density of ice).

Further work refined the process in the following ways:

- Replacing bounding boxes with bounding spheres
- Cyclically closing the sides
- Duplicating bounding spheres that crossed boundaries
- Inserting a crystal into the bounding sphere
- Randomly aligning crystal
- Removal of crystals beyond the layer limits
- Replacing bounding spheres with bounding cylinders (then reverting back)
- Replacing crystal with crystals with random parameters
- Testing crystals within overlapping bounding spheres
- Improving bounding spheres to test for convex hulls

4.1 Lattice Model

As a first step in the design, a 'wire frame' model was developed. This consisted of a three dimensional layer into which bounding boxes could be seeded at integer locations. For example, a layer, 3x3x3 has a potential of 64 locations. For these 'particles' could only be seeded into empty locations. The first image below shows the centres of two particles (red points) having been seeded while the second image shows the centres of 3 particles seeded in three dimensions.

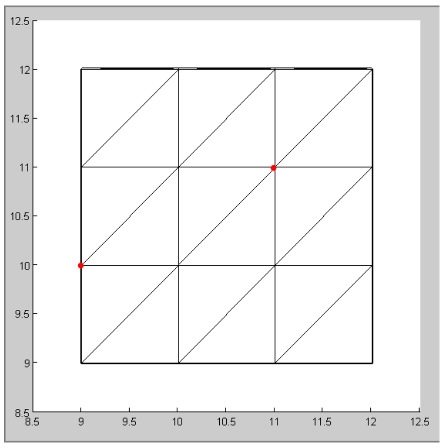


Fig.4.1. Wire frame model with two particles

Attempts at seeding a particle into a random location within the layer would then continue until a breakout-point was reached. The breakout-point occurred when the number of failures to seed a particle (rejections) due to the point already having a particle had been reached. Each successful seed reset the quantity of rejections back to zero.

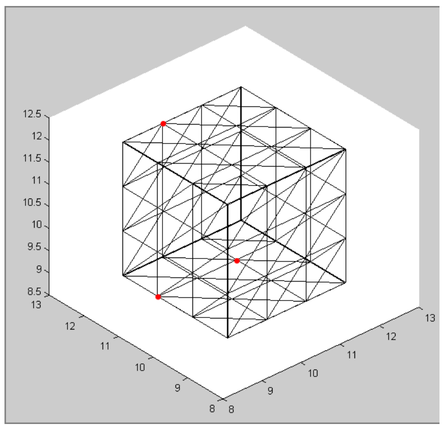


Fig.4.2. 3d view of wire frame model

The next step was to give 'volume' to the particles, turning them into bounding boxes. The bounding box dimensions were set to just greater than the distance between two neighbouring points on the lattice. The reason for being just greater was to ensure that they would impede on the surrounding space, i.e. when a bounding box was centred on a point, the edges of the box would be beyond the halfway point to all adjacent point (overlap marked in grey). This was achieved by allowing the particle to encompass all adjacent points and by checking at the point of seeding if there was a particle already seeded in an adjacent point in all three dimensions, i.e. the second one of the boxes to be seeded in the image would be rejected. Within the plane of the particle this would mean 8 locations. There would then also be 9 locations above the particle and 9 locations below the particle – a total of 26 locations (27 including the seed point).

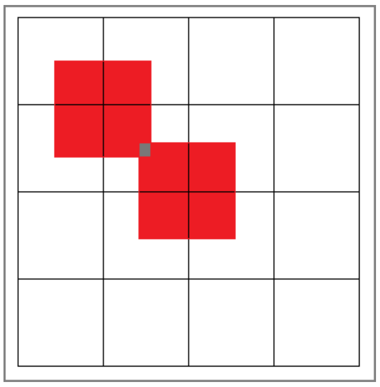


Fig.4.3. Overlap of particles (red) shown in grey

Preliminary Results

What was discovered was that irrespective of how large the breakout-point was set it was always impossible to achieve the potential density as cavities between the bounding boxes were created into which no particle could be seeded. By potential density it is meant the density achieved if all the boxes were assigned locations to minimise space between the boxes and the sides of the layer (like neatly stacked building blocks in a toy chest).

4.2 Development of MYSQL for storing data

Based on the original work plan MYSQL (a relational database management system) was integrated into MATLAB in order to create a flexible data store. This allowed for data to be spawned and saved, manipulated and recalled. Significant progress was achieved though this method had two fundamental issues:

- It meant that for the work to be continued by others following the completion of the MSc, it would require the installation of MYSQL in any workstation that would be using the code and have a working knowledge of MYSQL.
- It was essentially a waste of time as the output data had to be plain text in order to be used by the existing ray tracing model. This meant that all data spawned had to have an extra routine to extract it from the database then output as a text file.

The MYSQL phase was essentially redundant and as such was eventually removed from the core programs. While it is the author's opinion that including MYSQL is a sensible step forward both for ease of manipulation of data and speed of processing and ultimately it requires modification to the ray tracing model for acceptance of data straight from a MYSQL database.

4.3 Creating of bounding Sphere and Cyclically Closed Loops

Following on from a nodal array in which points could only exist at integer locations, the design was expanded to allow spheres to exist at floating point locations, i.e. at any location within the layer as produced by generating random floating point numbers for the X,Y and Z co-ordinates within the limits of the edge of the layer. The spheres were assigned along with a fourth parameter; radius.

A bounding sphere is essentially the smallest spherical volume that will completely envelope any object within it. An early approach was to determine the size of the bounding sphere based on the largest dimension of the crystal, i.e. if a hexagonal plate this would likely be radius dimension or length dimension for needles. The new approach determined the bounding sphere from both dimensions. This would only potentially increase the size of the bounding sphere and as a consequence increase the chances of using the convex hull test. This eliminates the small chance that there could be an overlap due to a crystal extending beyond its bounding sphere.

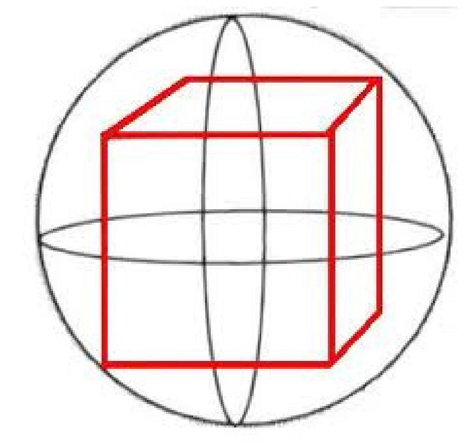


Fig.4.4. Bounding sphere around a cube

For a cube (adjacent diagram) of edge length 1 the circumscribing sphere will have a radius of 0.867 (=sqrt(3)/2), i.e. a diameter (dimensions) of 1.73 (d in Fig. 4.5). For hexagonal columns where the length is large compared with the diameter (needles) or the diameter is large compared with its length (plates), the surface radius will be much greater than its width (needles) or length (plates) and the resulting sphere's volume will be many times the volume of the hexagonal column.

This proved an impasse to creating layers with densities as high as those examined by Kaasalainen et al., (2006). This issue and its potential resolution are dealt with in section 4.6.

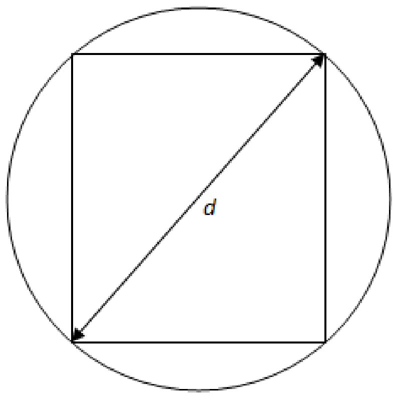


Fig.4.5. Bounding sphere has diameter d

Before the program committed to adding the sphere a check was made against all previously seeded spheres to ensure that the space was not already partly occupied by a previously seeded sphere.

The method of checking for overlap was a matter of calculating the separation between centre of the seeding sphere and all the particles already accepted into the permanent data array. If this separation was less than the combined radii of the sphere to be seeded and any of the previously seeded spheres, the seeding sphere would be rejected. A failure breakout was included. This was the quantity of sequential rejections required for the program to terminate.

An observed effect of failure breakout and random radius within assigned limits for seeding spheres was that towards the end of the program there was a notable decrease in the size of the spheres accepted. This became more pronounced with increasing value for failure breakout. This equated to filling in the cavities within the layer with smaller spheres. The greater the failure breakout, the higher the chance of a smaller sphere being randomly selected and assigned to ever smaller remaining cavities (see section on Increasing Breakout-Point). The samples studied by Kaasalainen et al., only give the range of snow crystal grain size rather than average grain size and size distribution. Some caution should therefore be taken in literal comparisons between the particle layer and the real snow samples.

If the radius of the sphere extended beyond the bounds of the layer, it would mean that there would be fewer spheres capable of overlapping. This would in practise result in a greater concentration of spheres being seeded along the edges of the layer and would not therefore be a true representation of snow as this would presume some degree of homogeneity in terms of density throughout the layer. To prevent this, cyclically closed loops were used, as in Stankevich (2007). This form of wrapping is the method by which one side of the layer is treated as being adjacent to the other. As such the seeding sphere was offset by the dimensions of the layer and re-evaluated for the purposes of overlapping with previously seeded particles.

While the density of a snow will increase with depth due to compression by weight of the snow, as the virtual layers are only one or two millimetres thick, it is can assumed that the change in density over this range will be negligible. This can be seen from the results obtained by Kassalainen (2006). Between the surface and 1cm there is only a minor increase in density.

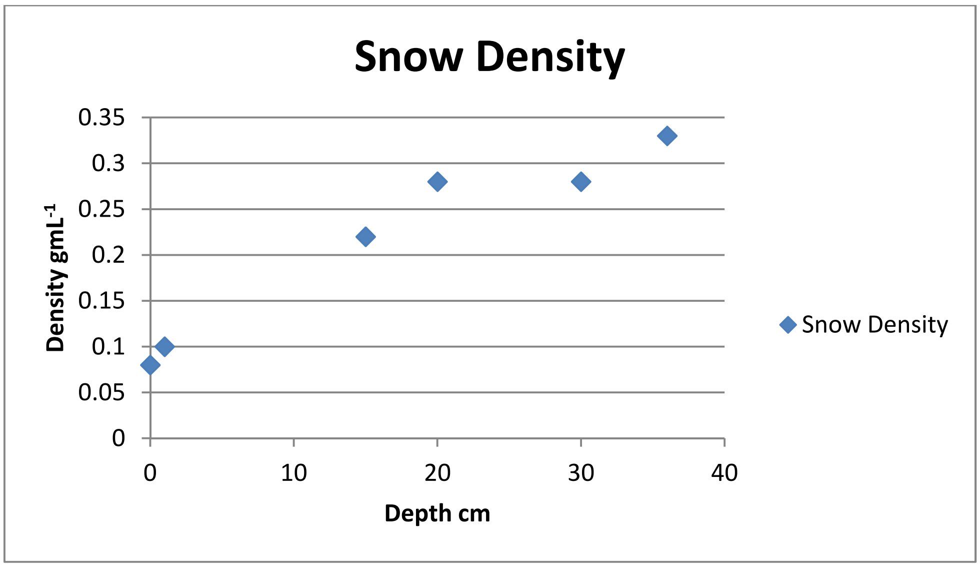


Fig.4.6. Graph indicating average snow density at depths. Kassalainen (2006)

4.4 Infinite Regolith

As a layer represented just a small sample of what would be considered an infinite region of regolith (or snow for the purposes of this work) it was important to account for rays passing through the sample and being scattered out of the side of the layer. Rays leaving one side are transposed to the opposite side of the layer and re-inserted. The reason for this approach is that the layer is considered an isolated sample of typical regolith within an 'infinite' plain. As such it can be presumed that for every light ray being scattered out of the sides of the sample layer, a light ray will be being scattered into it.

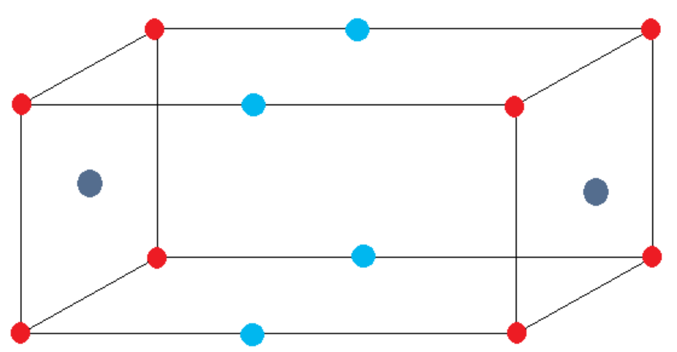
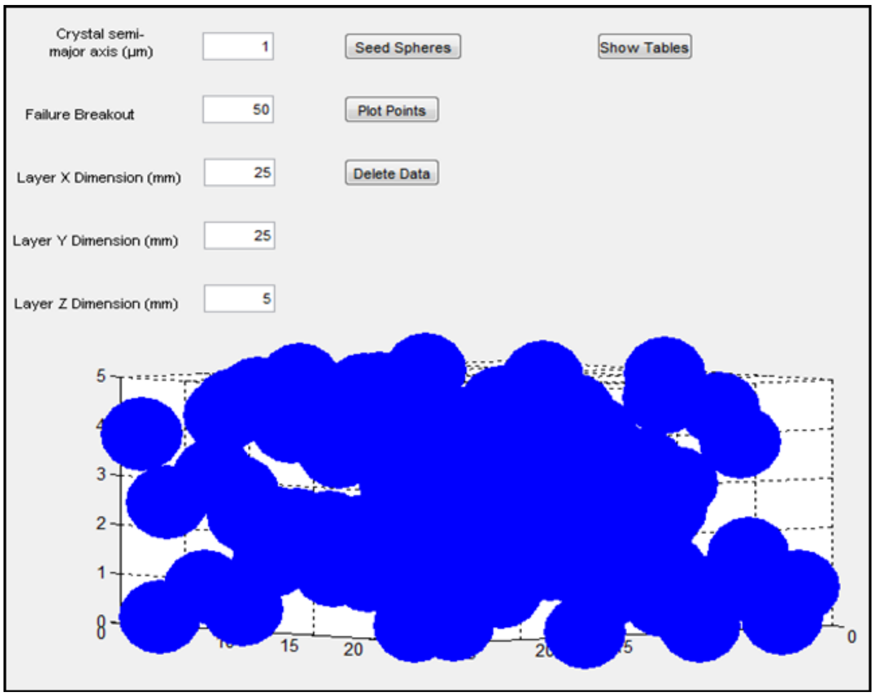


Fig.4.7. Coloured particles indicate where duplicates are generated as part of the seeding process.

To account for transposed rays maintaining their characteristics the entry point had to match the exit point. By this it is meant that if the ray was within a particle that crossed a side boundary, there would need to be an identical particle on the opposite side.

As a consequence, after wrapping, a second stage replaces the seeding sphere with an array of spheres depending on the number of boundaries being crossed. These spheres would be placed at the opposite boundaries. These are



Crystal semi-major axis (μm) 1

Failure Breakout 50

Layer X Dimension (mm) 25

Layer Y Dimension (mm) 25

Layer Z Dimension (mm) 5

Fig.4.8. As a first pass, a radius of unity was used and the dimensions of the box were set through iteration in order to generate approximately 100 spheres. This produced on average 100 successful seeds prior to failure breakout. This equated to a volume density 0.13, i.e. 0.13 of the volume contained a sphere (100x4π/3x25x25x5).In comparison, the maximum density if the spheres had been stacked is 0.74.

all seeded simultaneously and after checking for wrapping to prevent them detecting each other and causing the seeding to fail.

A feature of this is that a single sphere will be duplicated once, if crossing a single boundary (grey sphere in the diagram). It will be duplicated 3 times for a total of 4 spheres if crossing two boundaries (blue sphere in diagram) and if the initial seed is at a corner of the layer it will be duplicated 7 times for a total of 8 spheres (red sphere in the diagram).

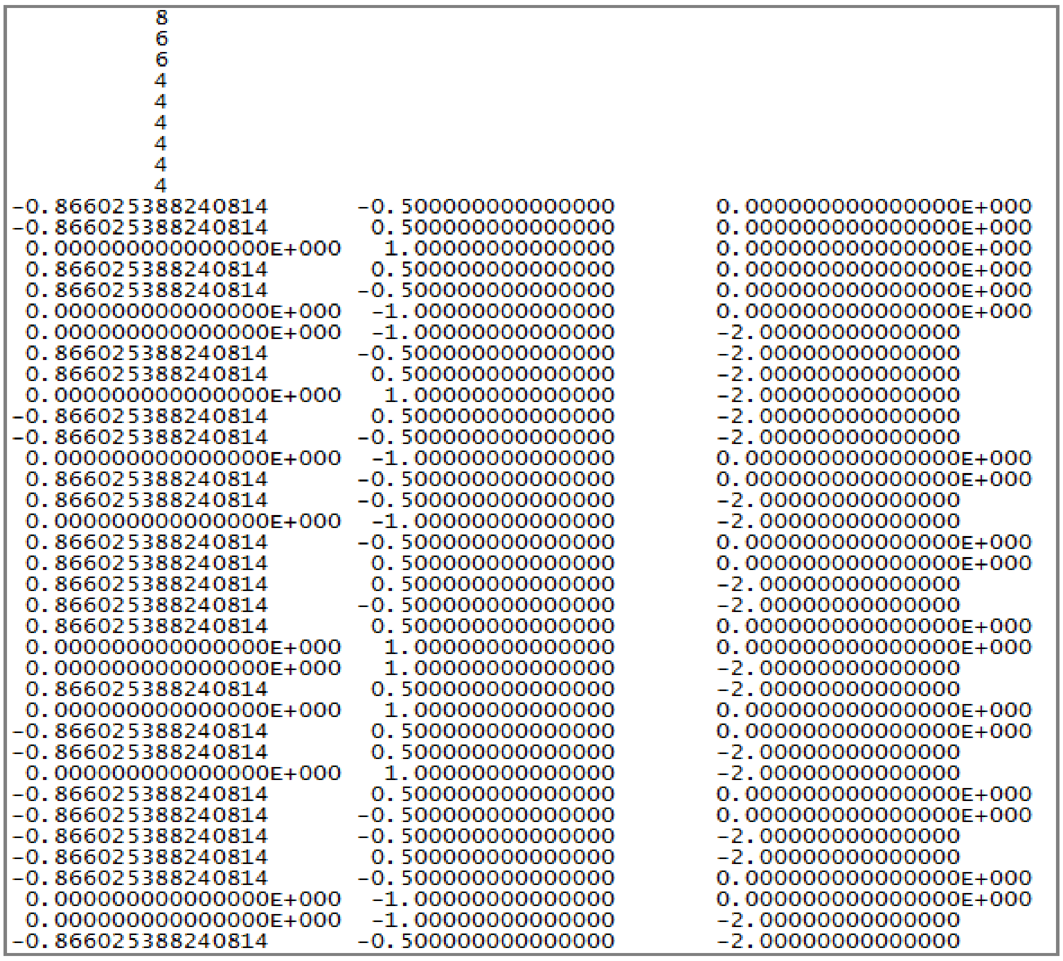
4.4.1 Replacement of bounding spheres with crystals

A crystal file is essentially nothing more than a means of describing the locations of the facets of a crystal (though by extension can also be used to describe the facets of multiple crystals). This is done by three sets of data.

- Quantity of facets
- Number of corners of each facet (Vertices)
- The X, Y and Z coordinate of the corners (Vertex Coordinates).

In order to create a layer of crystals, each of the spheres is replaced by the data of the crystal file then compiled into the appropriate format. First of all a crystal file of the dimensions appropriate to the bounding sphere needs to be generated.

For example, a hexagonal crystal has 8 facets: two hexagons and 6 rectangles. There are therefore 36(=6x2+6x4) vertices.
The coordinates of the vertices for a standard upright hexagonal column can easily be calculated through geometry.



```
           8
           6
           6
           4
           4
           4
           4
           4
           4
-0.866025388240814      -0.500000000000000        0.000000000000000E+000
-0.866025388240814       0.500000000000000        0.000000000000000E+000
 0.000000000000000E+000   1.00000000000000        0.000000000000000E+000
 0.866025388240814       0.500000000000000        0.000000000000000E+000
 0.866025388240814      -0.500000000000000        0.000000000000000E+000
 0.000000000000000E+000  -1.00000000000000        0.000000000000000E+000
 0.000000000000000E+000  -1.00000000000000       -2.00000000000000
 0.866025388240814      -0.500000000000000       -2.00000000000000
 0.866025388240814       0.500000000000000       -2.00000000000000
 0.000000000000000E+000   1.00000000000000       -2.00000000000000
-0.866025388240814       0.500000000000000       -2.00000000000000
-0.866025388240814      -0.500000000000000       -2.00000000000000
 0.000000000000000E+000  -1.00000000000000        0.000000000000000E+000
 0.866025388240814      -0.500000000000000        0.000000000000000E+000
 0.866025388240814      -0.500000000000000       -2.00000000000000
 0.000000000000000E+000  -1.00000000000000       -2.00000000000000
 0.866025388240814      -0.500000000000000        0.000000000000000E+000
 0.866025388240814       0.500000000000000        0.000000000000000E+000
 0.866025388240814       0.500000000000000       -2.00000000000000
 0.866025388240814      -0.500000000000000       -2.00000000000000
 0.866025388240814       0.500000000000000        0.000000000000000E+000
 0.000000000000000E+000   1.00000000000000        0.000000000000000E+000
 0.000000000000000E+000   1.00000000000000       -2.00000000000000
 0.866025388240814       0.500000000000000       -2.00000000000000
 0.000000000000000E+000   1.00000000000000        0.000000000000000E+000
-0.866025388240814       0.500000000000000        0.000000000000000E+000
-0.866025388240814       0.500000000000000       -2.00000000000000
 0.000000000000000E+000   1.00000000000000       -2.00000000000000
-0.866025388240814       0.500000000000000        0.000000000000000E+000
-0.866025388240814      -0.500000000000000        0.000000000000000E+000
-0.866025388240814      -0.500000000000000       -2.00000000000000
-0.866025388240814       0.500000000000000       -2.00000000000000
-0.866025388240814      -0.500000000000000        0.000000000000000E+000
 0.000000000000000E+000  -1.00000000000000        0.000000000000000E+000
 0.000000000000000E+000  -1.00000000000000       -2.00000000000000
-0.866025388240814      -0.500000000000000       -2.00000000000000
```

Fig.4.9. Data for a hexagonal column (1x1) prior to being rotated. Note that the centre is offset such that the column is entirely below the z-axis (last column of co-ordinates)

4.4.2 Centring origin within Crystal

The code to generate crystals was adapted from FORTRAN code, translating it into MATLAB (as detailed below). This produced crystals in the positive X, Y and Z axis. When these were implemented into the main code, off-sets were used to adjust bounding spheres. While having no obvious detrimental effect on the code, modifying the original crystal file structure, generating the crystal around the origin rather than from a corner has simplified various areas of the code by eliminating the need for off-sets and simplifies improvements to the seeding routines.

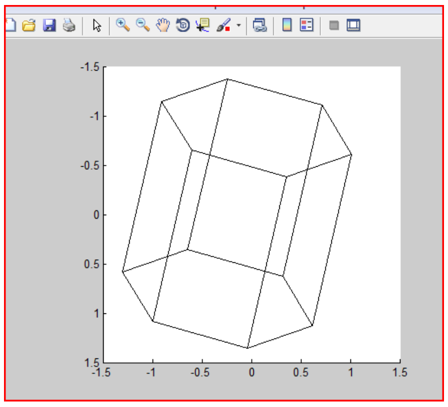


Fig.4.10. Hexagonal column (1x1) after re-orientation

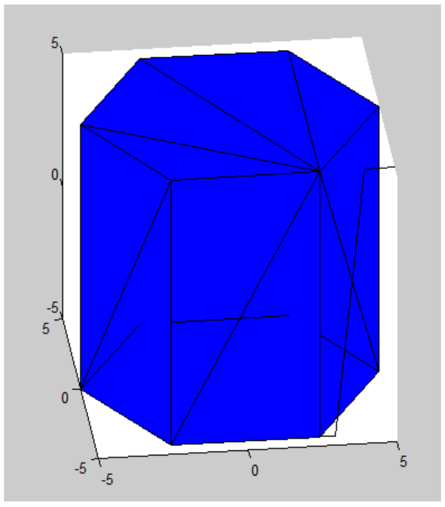


Fig.4.11. Hexagonal column (1x1) after re-orientation and centring code improvement

4.5 Conversion of FORTRAN Crystal Generation Code

The ray tracing program was designed to use data for a single crystal. A FORTRAN program existed that generated and rotated a single crystal based on parameters supplied through prompts in a DOS window. As this project is designed to use a range of crystals within a layer it is unrealistic to either use this method or create many pre-generated crystals from which to select from randomly. A new program based on the FORTRAN code has therefore been written for MATLAB. This has the advantage of generating and rotating crystals as they are seeded.

The program generates hexagonal prisms. Choosing this shape has advantages over other shapes in that by having a long column length with respect to radius, the program can generate needles. While by having short lengths and large radii, plates, another common ice crystal, could be generated. This means that 3 common types of ice crystal could be generated from the one program. The program also include

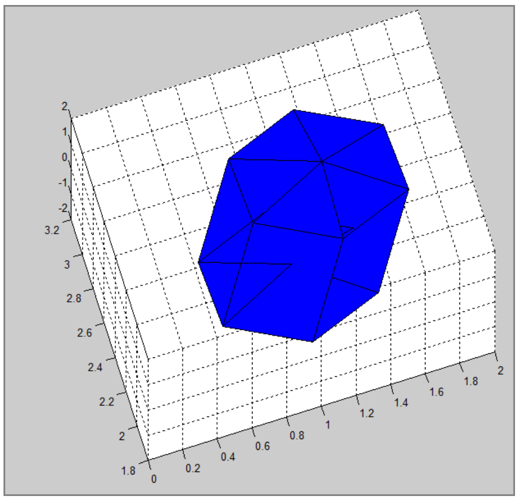
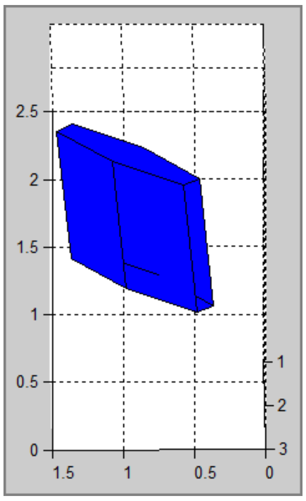


Fig.4.12. Reoriented hexagonal column with non-planar basal facets



Hexagonal column with oblique planar basal facets non-planar basal facets, i.e. 6 facets at the ends of the crystal that could either be pointing outwards or inwards as well as oblique ends.

4.5.1 Data Output

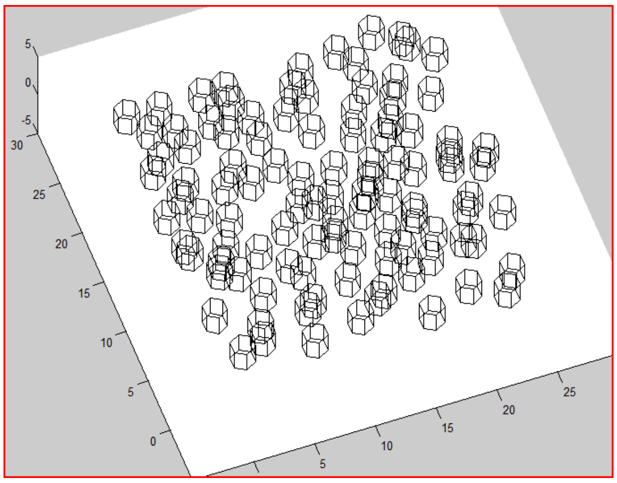


Fig.4.13. Layer of hexagonal columns – pre-reorientation

The output data for the layer is assembled from the accumulation of the three sources of data; total facets, vertices and vertex coordinates.

Example: For hexagonal crystals (with non-planar basal facets), a box of twenty-five randomly positioned crystals would have the format 200 (25x8), representing the total number of facets, followed by 25 strings of 6,6,4,4,4,4 representing the vertices of the individual facets and finally an 900x3 array detailing the coordinates of the 900 vertices of the facets in 3 dimensions.

As the vertex coordinates generated by the crystal creation code are relative to the first vertex coordinate, seeding the crystals into the layer is a case of offsetting all the vertex coordinates of the seeding particle by the coordinates of the bounding sphere, taking care to account for the difference in the centre of the bounding sphere and the coordinate of the first vertex.

4.5.2 Orientation

Clearly crystals are not uniformly lined up within a snow sample and as such each crystal has to be oriented to some degree (as explained below) at the point of seeding. As the bounding sphere covers the major axis of the crystal, it is possible to rotate the crystal within the three dimensions without the possibility of it then overlapping with another crystal.

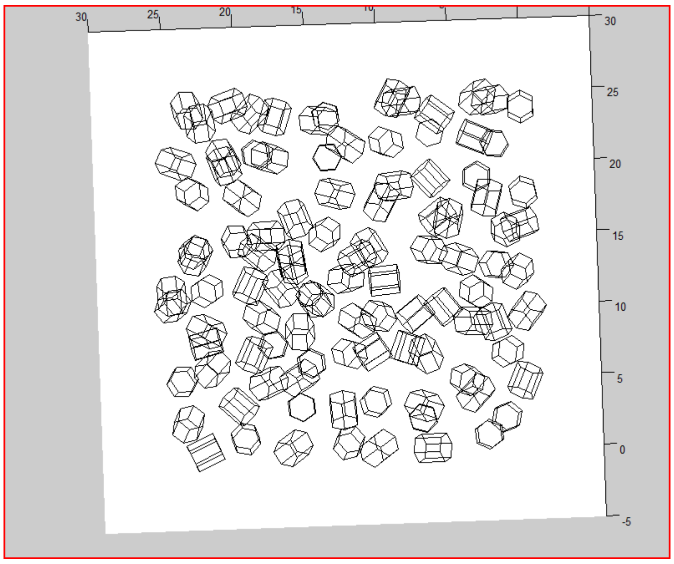


Fig.4.14. Layer of hexagonal columns – post-reorientation

This has to be done at the point of seeding rather than to each crystal in the final array. The reason for this is that where seeds are duplicated during the seeding process due to crossing a layer boundary each duplicate crystal has to match the orientation of the original seed.

Degree of randomness

As some images appeared to show that needle crystals could settle, forming layers (like the game jackstraws), during the coding for random orientation an option was included that restricted the degree of freedom of the prism in the vertical plane such as in Fig. 4.15.

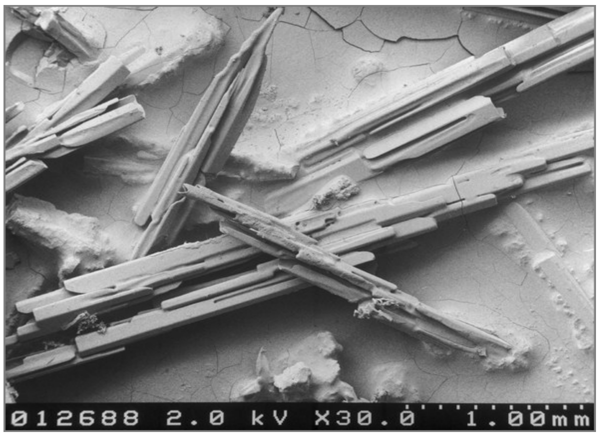


Fig.4.15. Image suggesting settled orientation *http://forum.xcitefun.net/microscopic-images-snowflakes-t49288.html*

Three options were given that set the range for the beta-euler angle.

Random (β= acos(1-2ø))
Settled (β= øπ/10+π/2)
Flat (β= π/2)

Where ø is a random floating point number between 0 and 1.

Acos(1-2ø) in the random setting gives a fully distributed range of orientations. Using øπ for example, gives an even distribution across the limits of π, this does not equate to a fully distributed range of orientations.

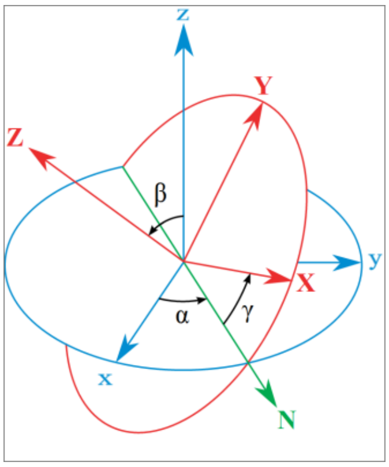


Fig.4.16. Euler Angles *http://en.wikipedia.org/wiki/File:Eulerangles.svg*

One advantage in defining the beta-euler angle was that it allowed the bounding sphere to be replaced with bounding cylinders that could be used to reduce the height component of a bounding cylinder. Later however this was reviewed in light of further images of snow surfaces.

A request for information on the orientation of needles in fresh snow to Jouni Peltoniemi appears to confirm what was already suspected: the crystal orientation can be assumed to be random. Fig.4.18 clearly shows that many of the surface needles are oriented close to perpendicularly to the surface. While this ultimately required a rewrite, removing bounding cylinders, the orientation function was left in (vestigial remain).

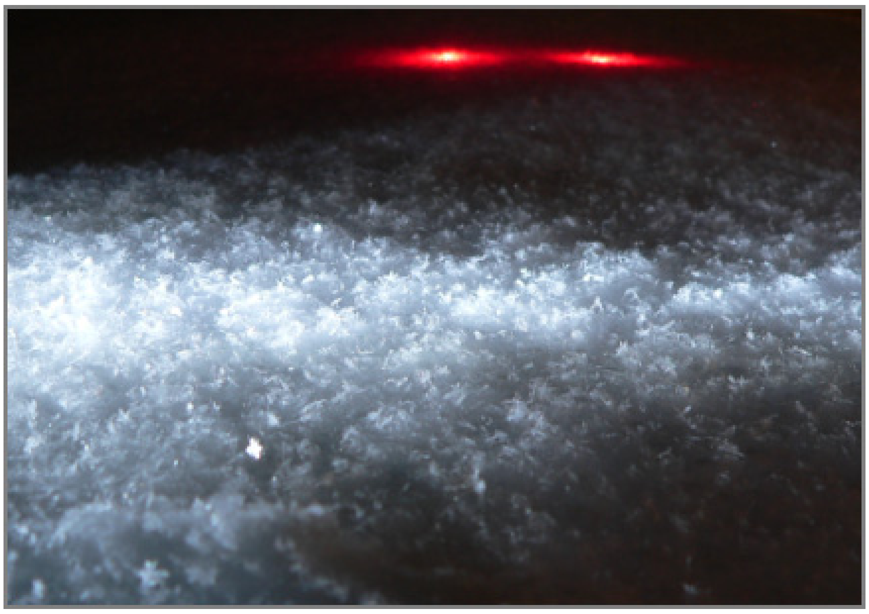


Fig. 4.17. Jouni Peltoniemi image - many of the surface needles are oriented close to perpendicularly to the surface

It is however noted that this vestigial remains may still come in useful on account of results being dominated by a bright spikes (when the sample size is a problem – see below). As the primary purpose of the method is to explore coherent backscatter, glare can easily swamp the data for samples with a small horizontal surface area. Glare occurs when one or more of the facets happens to be aligned such that a large facet (with respect to the overall dimensions of the layer) happens to be effectively perpendicular to the incident rays. These can be seen in nature when looking at snow. Despite a wide expanse of white, the snow appears to twinkle as the head is moved. A single crystal of snow can suddenly 'outshine' the surrounding snow.

Fig.4.18. Enlargement of above image

The glare can be seen in the Fig.4.19 for a sample with layer dimensions of 0.5mmx0.5mmx0.2mm and rounded hexagonal columns with dimensions 0.1mmx0.05mm. The horizontal line *y*=0 corresponds to the scattering angle range of 170 to 180 degrees. The black & white scattering image is purely external reflection. This image is dominated by the white spot just off-centre produced by a facet aligned similarly to the one indicated by the red arrow in the render of the crystal file.

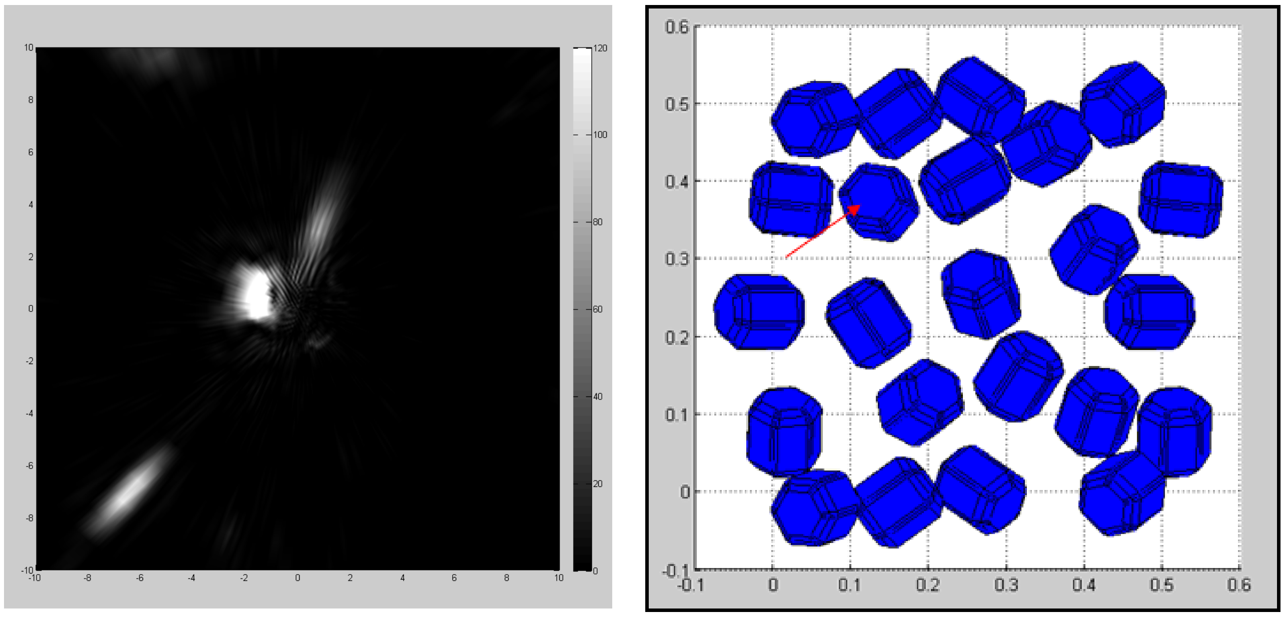


Fig.4.19. Glare in output is probably attributed to end facet (indicated in render with a red arrow) oriented towards observer

While it is understood that limiting the rotation in the Z axis ultimately means that the layer is not a true representation of a snow layer, it will reduce the potential for this sort of glare and theoretically allowing for better observation of the coherent backscatter phenomena.

The size of the sample however poses problems in its own right. With relatively few surface crystals (representing certain crystal orientations)contributing to the reflection, it means that the pattern will be dominated by individual reflections rather than smoothing out as each reflection contributes only a small amount to the overall backscatter pattern.

4.5.3 Removal of Seeds outside a Layer

As indicated above, a bounding sphere that extended beyond a layer boundary automatically generated the corresponding duplicate spheres (and these in turn had to be tested in order to determine if further spheres were duplicated). As part of the rotation function, a feature had to be added that allowed for the removal of crystals that had become positioned outside of the layer boundary. While the centre of the initial bounding sphere would be within the limits of the layer, any duplicate will have their centre outside of the layer boundary. As a consequence, the orientation of a crystal could result in all points of a duplicate crystal existing outside of the layer boundaries.

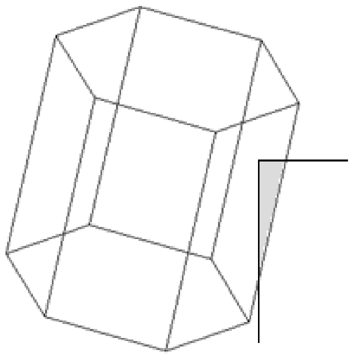


Fig.4.20. Intersection of corner of layer with hexagonal column indicated in grey

The removal however is not perfect, in that it is based on the vertices of the duplicates rather than checking if any of the planes of the facets intersect the bounding box formed from the layer boundaries (this would then require a second check to determine if any of the edges of either the facet or the layer bounding box intersected). This approach however was expedient as the only case where it is possible to fail is where a facet formed from the vertices crossed a boundary of the layer even though all the vertices lay outside of the layer such as over a corner. It is however something to be aware off and an area to improve along with the use of convex hulls to solve the density issue (see below).

4.5.4 Random Sized Spheres

Based on Baran and Labonnote (2007), ice crystals of a range of sizes are typical within any sample of snow. As such the single sized seed sphere had to be replaced by a randomly generated size. Initially a simple random generation between a maximum and minimum sphere size was used. This however meant that there were definite cut-off points in the size and there was an even distribution of crystal dimensions. As such this was deemed to be highly unrealistic. This was demonstrated by One (1968) in field studies of ice crystals in clouds.

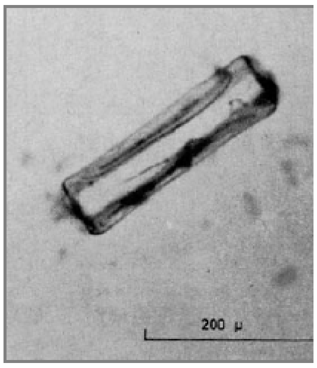


Fig.4.21. Hexagonal Column, Ono (1968)

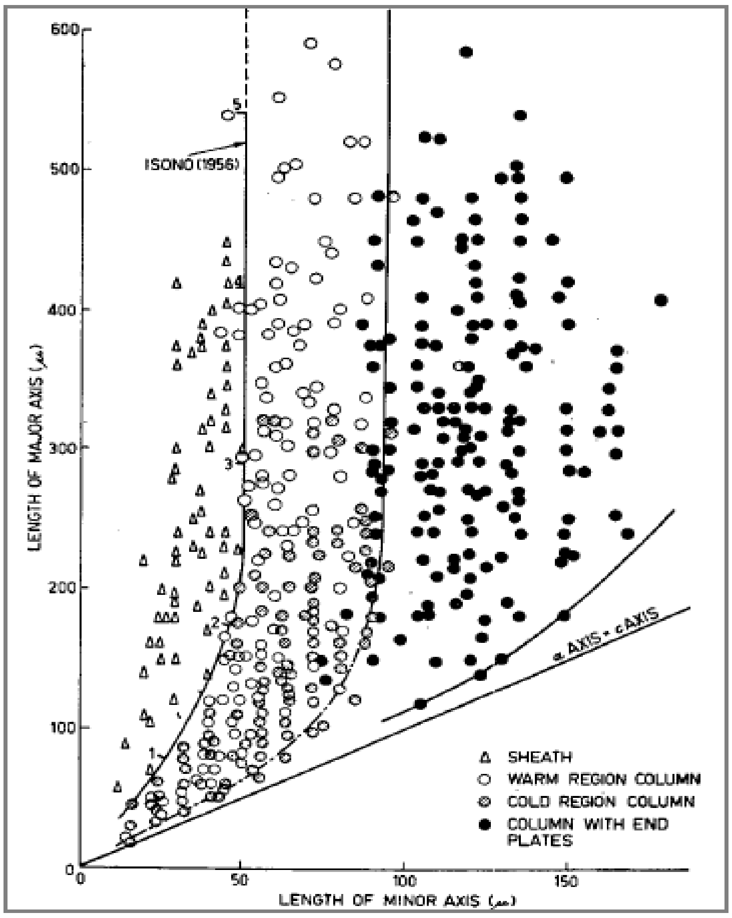


Fig.4.22. Ono (1968) – showing the relationship of length along the major and minor axis in various columnar ice crystals (unfilled circles represent hexagonal columns)

While a range of ice crystals were evident it was not an even distribution between the limits. In the graph below (looking at the unfilled circles), it is clear that the greatest concentration of crystals is around major axis 130μm and minor axis 50μm. While not being able to expressly confirm the true distribution profile of the snow samples of Kaasalainen, a dispersion profile was applied to the seeding process.

As such the program was modified in order to include an average sphere size and a 3sigma parameter. This allowed the spheres to have a random radius but within a distribution bell-curve peaking at most common radii. 3sigma means that 99.7% of all randomly generated ranges will be within this range.

The average length and the 3sigma point for both semi-major and semi-minor axis, corresponding to length and radius of hexagonal columns could be set manually in the user interface.

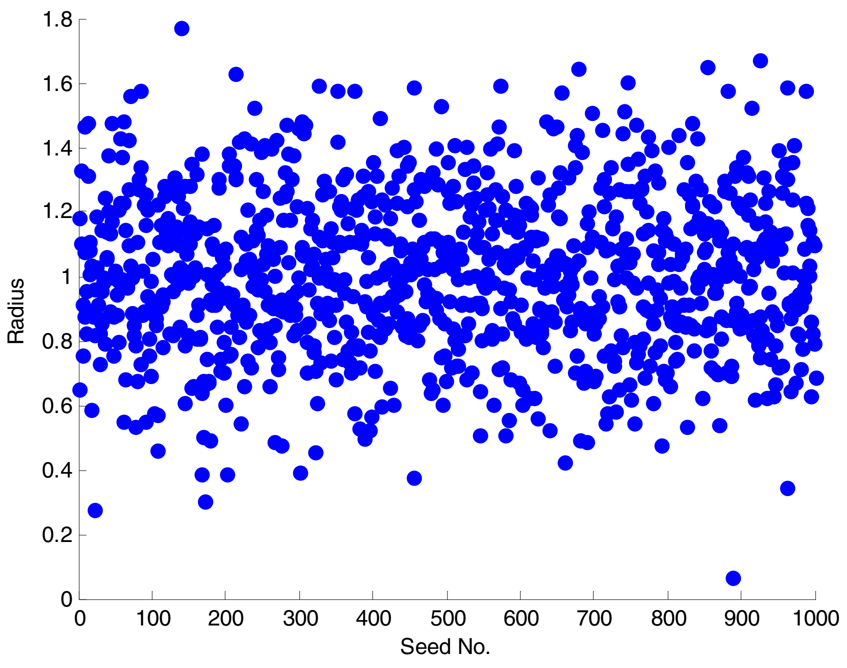


Fig.4.23. T he plot is a plot of 1,000 normally distributed points(seeds) where 1 is the mean and 0.25 is the 3sigma standard distribution. In this case 99.7% of all points will be within 0.25 and 1.75. In the plot it can be seen that only a few points are obviously outside of this range.

4.6 Density

A density parameter was included in the output file. This was initially determined as the approximate total volume of crystals against the volume of the layer accounting for the density of crystals (the value of the crystal density can be set in the interface and is reported in the output file).

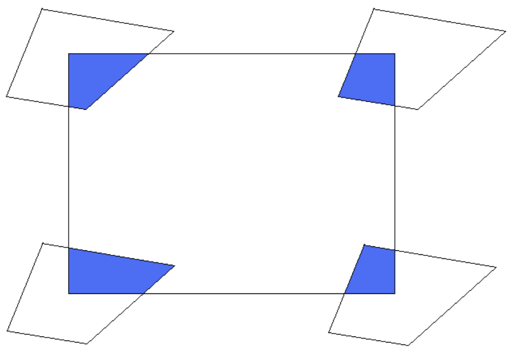


Fig.4.24. 1 seed and 3 duplicates only accounts for 1 full area, as indicated in blue shading

Where crystals crossed a boundary, duplicates are off-set by the dimensions of the layer and seeded across the opposite boundary. A crystal for example that had a third of its volume outside a boundary would be duplicated. The duplicate would be off-set and seeded on the other boundary so that only a third of its volume lay within the boundary. A feature of this is that though a crystal may end up being seeded 8 times, its total contribution to the volume within the boundaries of the layer would be only one crystal. The 2d example below shows the coloured parts of the duplicated crystal that effectively contribute to the layer density.

In the case of the hexagonal columns having non-planar basal facets, no account of this addition or subtraction to the overall density has been made. The reason for this is that as the non-planar facets end may be indented or filled it is assumed that their contribution to the volume will be negligible.

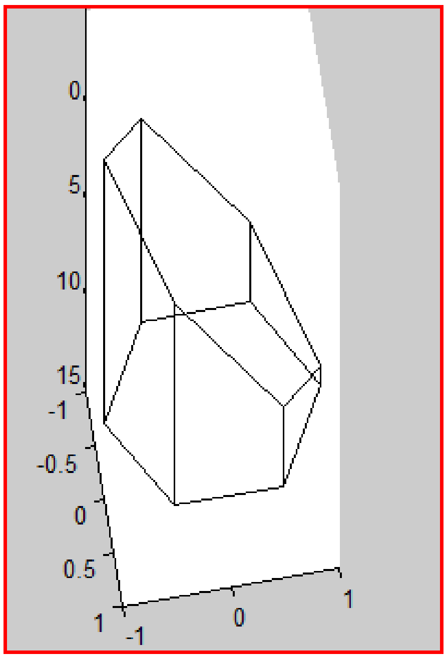


Fig.4.25. Basal obliques account for true density less than reported density

Even taking into account both the indents, boundaries and the density there is however a feature that has not been taken into consideration, this being the basal obliques.

These will ultimately mean that the density reported will always be slightly greater than the true density if these have been included in the column generation program.

For example, if the length(L) to radius (r) is 20:1 and maximum angle of an oblique is 45^0, a third of the crystals will have no oblique ends, a third will have one and a third will have 2. On average, a crystal will have 1. The average angle of the oblique will be 22.5^0. Applying geometry to the part of cross section area containing the cylinder axis (removed and remaining) and the remaining area generates the following:

$$deltaL = 2x \tan 22.5$$

Where $deltaL$ is the part of L covered by the oblique end.

$$where\ x\ varies\ between \frac{\sqrt{3}r}{2} and\ r$$

$$V_{Remaining} = \mathrm{V_o}\left[\frac{1 - \mathrm{deltaL}}{2L}\right]$$

Approximating for tan22.5 produces and substituting for L and r

$$V_{Remaining} \approx 0.99\mathrm{V_o}$$

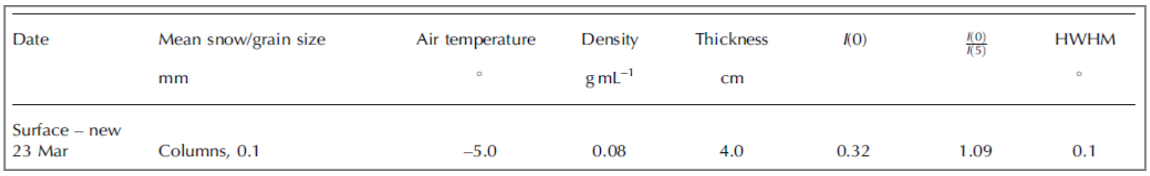
Even when columns with diameters equal to their heights, the fraction remaining is 0.8.

Oblique basal facets were not used in generating hexagonal plates.

4.6.1 Low Density

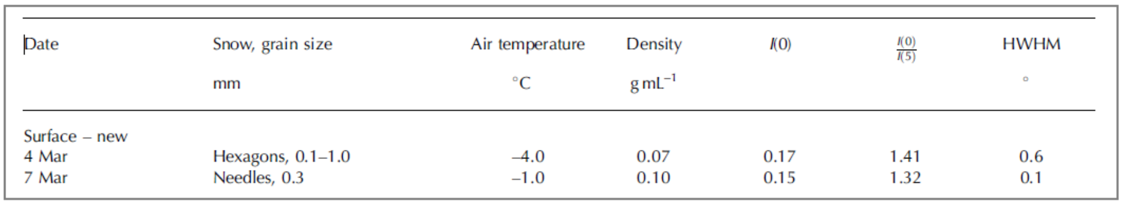
Initially, the premise was to model hexagonal columns as per the simplest ice crystals studied by Baran and Labonnote (2007) where the length of the column was approximately equal to the diameter, i.e. L ≈ 2r. This would mean that the semi-major axis of the resulting crystal would be approximately equal to the semi-minor axis. As a consequence, using bounding spheres would be a reasonable means of achieving desired densities for the layer.

As development of the code progressed, it became clear that for the layer model to have any real use in modelling planetary regolith, it would have to be able to generate layers consistent with field observations of real snow layers. Generation of layers consistent with the observations of Kaasalainen (2006) was undertaken. Three samples in particular were chosen on account of them being fresh surface snow.



Date	Mean snow/grain size	Air temperature	Density	Thickness	$I(0)$	$\frac{I(0)}{I(5)}$	HWHM
	mm	°	g mL^{-1}	cm			°
Surface – new							
23 Mar	Columns, 0.1	–5.0	0.08	4.0	0.32	1.09	0.1

Fig.4.26(a). 2004 samples



Date	Snow, grain size	Air temperature	Density	$I(0)$	$\frac{I(0)}{I(5)}$	HWHM
	mm	°C	g mL^{-1}			°
Surface – new						
4 Mar	Hexagons, 0.1–1.0	–4.0	0.07	0.17	1.41	0.6
7 Mar	Needles, 0.3	–1.0	0.10	0.15	1.32	0.1

Fig.4.26(b). 2005 samples

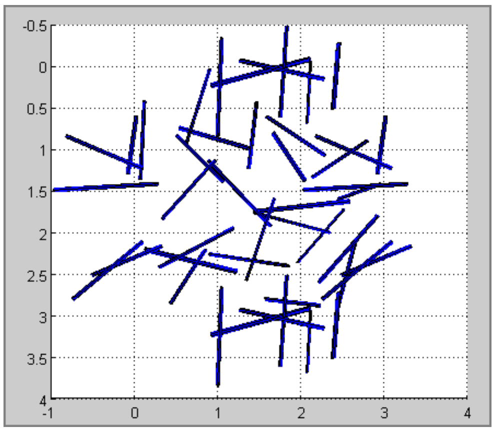


Fig.4.27.Needle layer

Layer Parameters
Hexagonal crystals
Rejections 500
Crystals Seeded 47
Crystal length (100±15)µm
Crystal radius (20±3)µm
Max indent 10%
Total Crystal Volume
0.04469mm^3
Density 0.00248g/ml

This is important as crystal structure is most likely to be retained at the surface while the snow is fresh. In terms of the layer samples, it meant that pristine crystals such as those represented in the adjacent image could be generated.

Compression and aging of snow is likely to corrupt the individual ice crystal structures through three processes – *simple breaking of the delicate crystals*, *partial melting and refreezing* and *sintering*. Sintering is where the crystals amalgamate with other crystals. In all cases the crystal shape will be modified. This however is where the inherent weakness in bounding spheres became apparent. Where one axis was large compared to the other (long, thin needles and wide thin plates), the actual mass of the crystal was effectively small compared with the volume of the bounding sphere. As a consequence, the density of the layers produced was much lower than that of the samples.

Bounding Cylinders

Efforts to decrease bounding volume included replacing the sphere with a cylinder. This was based on the assumption that settling crystals would be closer to horizontal. A short cylinder would therefore have lower volume than a sphere of diameter equal to the long axis of the crystal. Though increasing density marginally, they were eventually rejected as the snow samples examined reveals random crystal orientation.

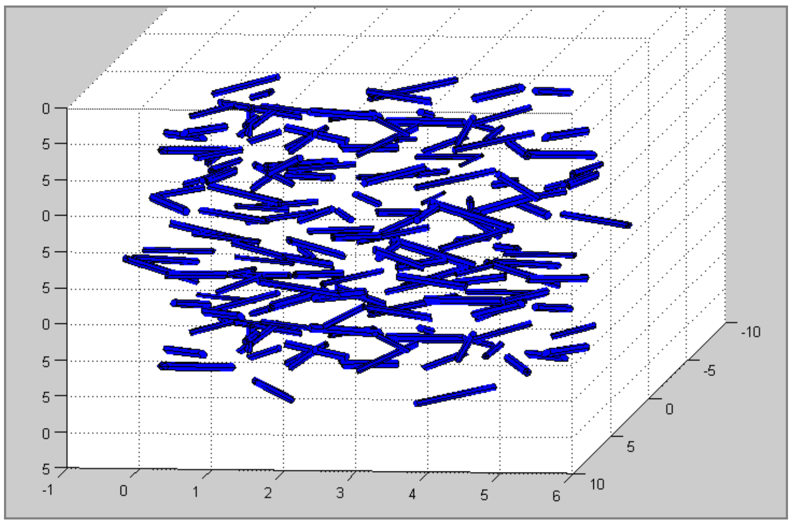
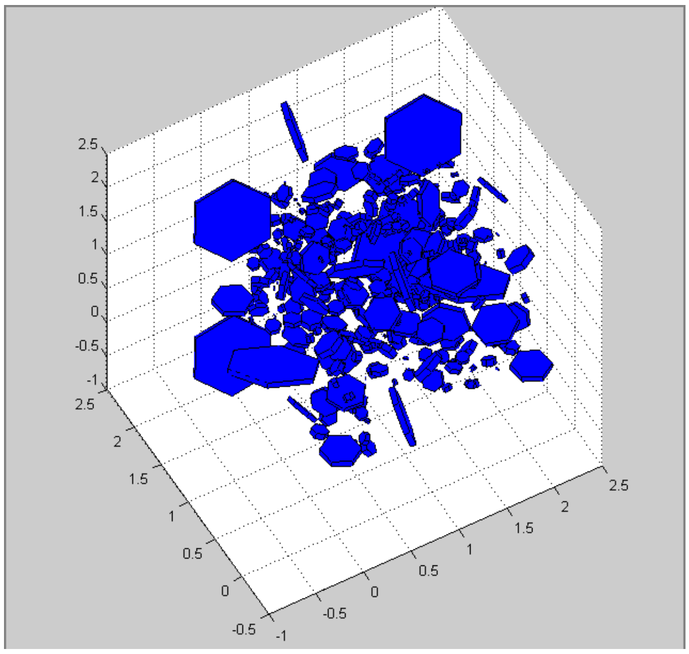


Fig.4.28. Bounding cylinders used to increase density based on theory that needles will settle rather than maintain random orientation

Increasing Breakout-point

A second method was to simply increase the breakout-point for the seeding process. This meant that there was a greater chance of generating a bounding sphere capable of fitting within the potential volume at a random location within a layer. An interesting consequence of this is that while the overall density does not significantly increase, the quantity of seeded crystals before rejections attain breakout-point is significantly higher. The reason for this is that as seeding continues, there is a higher likelihood that a sphere at the larger end of the random distribution radii will be rejected. Towards the end of the process, virtually all spheres and therefore crystals will be at the shorter end of the normal distribution.



Layer Parameters
Hexagonal crystals
Rejections 5000
Crystals Seeded 305
Crystal length (5±1)µm
Crystal radius (30±15)µm
Max indent 0%
Total Crystal Volume 0.39974
Crystal density $0.91670mm^3$
Layer Density 0.04581g/ml

Fig.4.29.Hexagonal Plates with high rejectionparameter

This can be seen from the data:

Crystal Seeded	Bounding Sphere Radius	Crystal Seeded	Bounding Sphere Radius
1	0.45433	296	0.02849
2	0.45433	297	0.02417
3	0.13589	298	0.01966
4	0.32149	299	0.03084
5	0.18063	300	0.09864
6	0.03832	301	0.02312
7	0.28602	302	0.11493
8	0.28602	303	0.02282
9	0.28602	304	0.06993
10	0.28602	305	0.07031

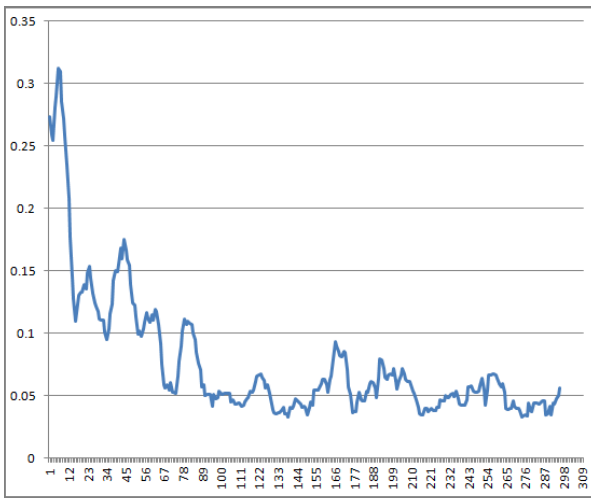


Fig.4.30. The graph shows the weighted average of the radius (over 10 seeds) in mm against seeding

This means that the actual distribution deviates from a normal distribution. The reason for no significant increase in the density is that the overall mass contributed by the smaller crystals is relatively small compared to the mass of even a single large crystal seeded early on in the process.

Presuming plates:

$$V = \frac{3}{2}\sqrt{3}r^2 L$$

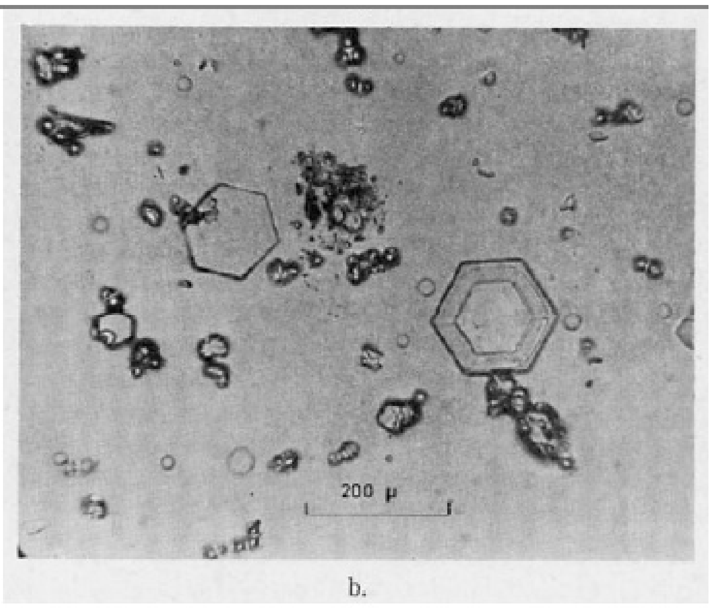
Volume of Seed1 =0.022328mm^3 (where r=0.45433mm & L=0.05mm)

Volume of Seed350 = 0.000535mm^3 (where r=0.07031mm & L=0.05mm)

This meant that the first crystal seeded was over 40 times the volume of the final crystal seeded (and this was by no means the smallest).

The example of the seeding process shows that even after 50 crystals (and their duplicates) have been seeded, the average radius being accepted has fallen by 50%.

While on the surface this could appear to be a major failing in the seeding process, it has to be taken in line with the samples being examined. The photograph (Fig.4.31) of hexagonal plate crystals (Ono, 1969) is a good example. While two distinctive hexagonal plates are present with diameters close to 200µm, there are many irregular crystals in the sample. The authors state that the method of collection meant that 'breakage of the ice crystals was reduced, though not entirely eliminated'. It cannot be determined as to what degree the irregular crystals are a function of the collection method rather than naturally occurring. While the samples studied by Kaasalainen were surface samples and therefore presumably less likely to suffer damage due to collection, there is no indication in the data as to the relative abundance of broken and irregular crystals present in the samples.



b.

FIG. 7. Hexagonal plate ice crystals, a., which had begun to sprout dendritic growths at the corners and, b., without sprouts at corners. Observed simultaneously at about −15C, 15 October 1966.

Fig.4.31.Ono 1969.

4.7 Rounded Edges

As has been commented on above, an issue with using precise hexagonal columns (as needles, plates and columns) is they are pristine unlike crystals that have undergone sintering, dendritic

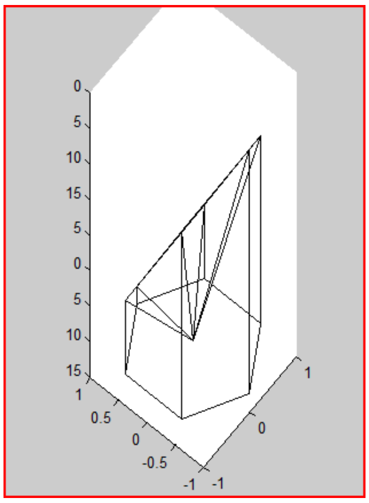


Fig.4.32. Pristine model of hexagonal column with indented oblique basal planar facets

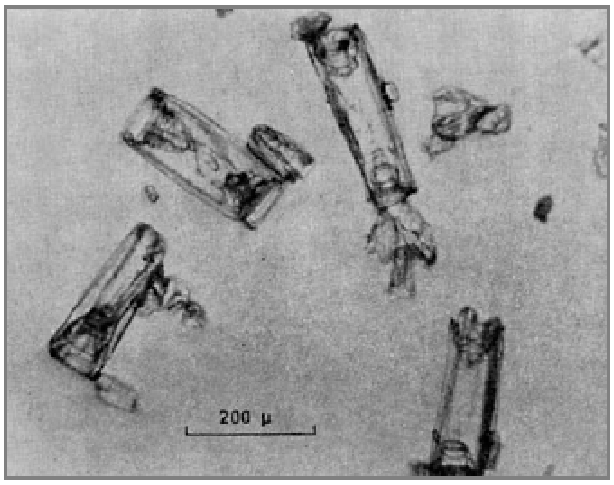


Fig.4.33. Riming and dendritic growth evident on ice columns (Ono, 1969)

growth, riming or have been broken. As can be seen Fig.4.33. even these hexagonal crystals with indents are far from pristine compared with the model generated version.

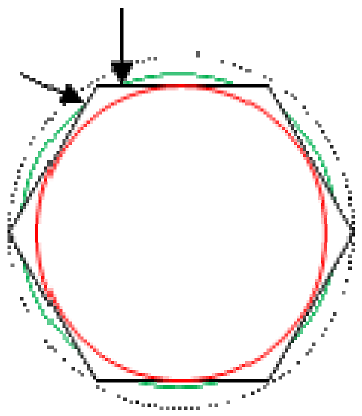


Fig.4.34. Green circle indicates rounding factor (arrows indicate intersection points)

Recent developments have produced hexagonal columns with rounded edges. This works by assigning a rounding factor between 1 and sqrt(3)/2, indicated by the green circle in Fig.4.34.

A rounding factor of 1 corresponds to the circumcircle (dotted line), i.e. no rounding while, sqrt(3)/2 corresponds to the inner circle (red), i.e. complete rounding. Next, the intersections points between rounding factor circle and hexagon are calculated (arrows). Then a number of new segments are chosen and the corresponding number of points on the green circle between the two arrows are calculated.

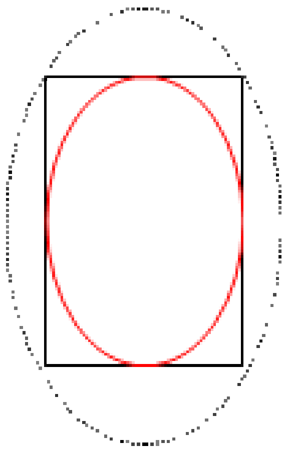


Fig.4.35. Ellipses used when dealing with perpendicular cross sections

The same method is applied for perpendicular cross section apart from using ellipses instead of circles (Fig.4.35).

By using rounded edges, it was possible to simulate more samples collected by Kaasalainen, specifically samples taken from snow that had partially melted and been refrozen.

As the program only generates a crystal file when there is a successful incorporation of a bounding sphere into the layer, advantage was taken of this recent development. Seeding a layer in which the limits of the hexagonal columns were set and had no random distribution, allowed a pre-generated crystal file of a hexagonal column with rounded edges to be substituted. The same process of rotating and duplicating was still applied.

By introducing a large number of comparatively small facets, features such as coherent backscatter are expected to become more readily observable.

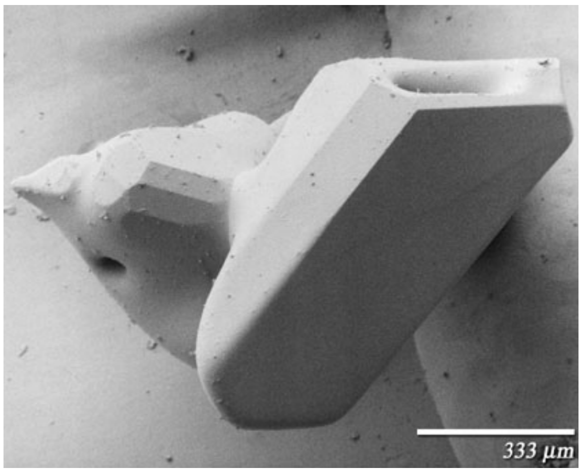


Fig.4.36. Image indicating rounded ends http://emu.arsusda.gov/snowsite/selected/9206.jpg

In the raytracing program individual rays have a finite diameter which is set as an input parameter. As these individual rays are traced, any child ray of the ray leaving the layer has its E-field calculated. This E-field corresponds to the far field diffraction pattern at the circular aperture appropriate to its diameter. The square of the sum of spatial E-field contributions of all rays passing through the same facets is an approximation of the far field diffraction pattern of a larger beam including all the individual rays. Ray tracing needs to be completed before the diffraction pattern is calculated, because all E-field contributions at a particular scattering angle need to be added. This includes rays leaving the layer from different crystals. By having more facets, there are a greater number of child rays contributing to the far field diffraction pattern, thereby reducing the dominance from individual facets and also increasing the amount of coherent backscatter, i.e. a greater number of spatial E-fields.

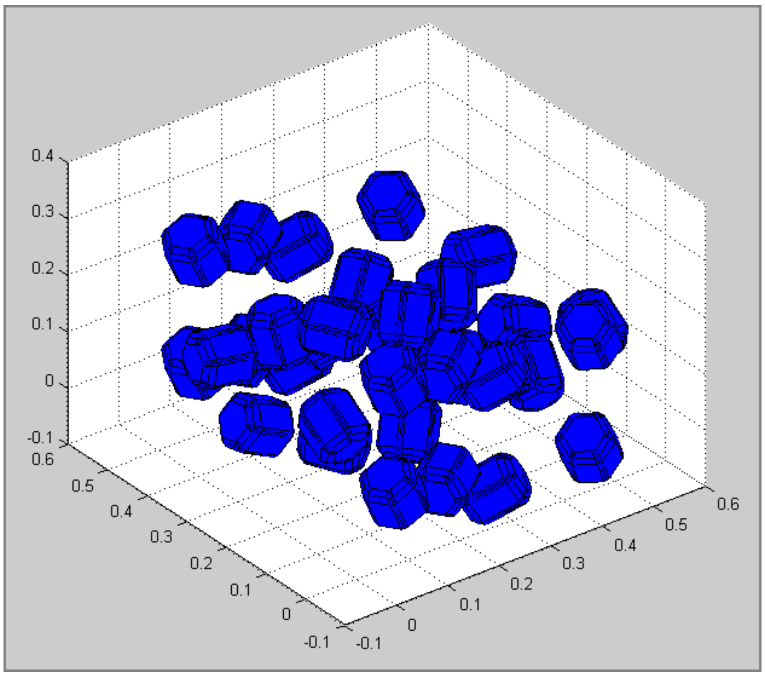


Fig.4.37. Model layer of hexagonal columns with rounded edges

4.7.1 Influence of the size parameter on coherent backscattering

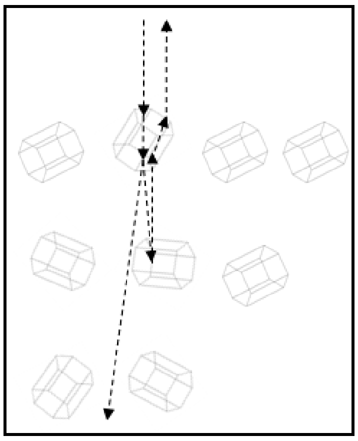


Fig.4.38. Normal limits of diffraction.

As smaller apertures have wider diffraction patterns (stronger E-fields at larger scattering angles), an approach to increase coherent backscatter was to increase the wavelength of the rays and in this way decrease the size parameter $2\pi a/\lambda$, where a is a characteristic particle radius (see section 2.1). As no change to refractive index, absorption by the crystals is made, this equates to changing the size of the crystals with respect to the wavelength of the rays without rewriting the crystal files and generating new layers.

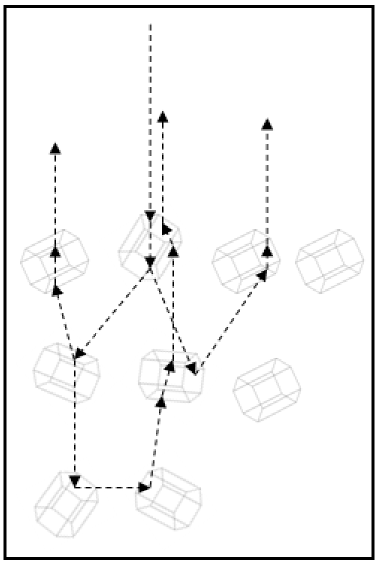


Fig.4.39. Increased diffraction increases possible paths contributing to coherent backscatter.

The approach increases the number of ray-paths that can contribute to coherent backscatter (see 2.2.2 for explanation of coherent backscatter). The two diagrams indicate how increasing diffraction (between Fig.4.38 and Fig.4.39) increase the possible range of paths a ray can follow. As only some of these have the potential for producing coherent backscatter by conjugate paths, (rays can follow the path in both directions and still reach the observer) increasing diffraction produces more paths with the potential to contribute.

4.8 Testing crystals within overlapping bounding spheres

While bounding spheres allow for quick testing for potential overlap before seeding a crystal, when the crystals being seeded were needles or hexagonal plates, crystal volume to bounding sphere volume was very small resulting in very low layer density. Rather than rejecting a bounding sphere at the time of seeding, should it overlap with any previously seeded spheres, a method was devised to test the crystals within the bounding spheres for overlap prior to seeding.

The test to determine whether two crystals intersect requires that each facet of the crystal to be seeded is treated as a polygon in 3D space. The lines forming the contours (boundaries) of the facets forming crystals within the bounding spheres that overlap the seeding sphere are tested against these polygons in order to determine whether there is an intersect. If they intersect, the crystal being seeded is rejected and its bounding sphere is removed from the bounding sphere array. If the crystal does not intersect with previously seeded crystals, it is added.

As the bounding sphere function occurs before the crystal generation and replacement function it is impossible to reject the sphere at this stage based on the crystals. As such an array of overlapping spheres is generated to be used in the crystal generation and replacement function. By the same token, it would be computationally expensive to always proceed with an array of overlapping spheres irrespective of the quantity of overlapping crystals. As a compromise based on empirical testing of processing time against size of overlapping array, the quantity was set to a maximum of 5 overlapping spheres.

As a sphere could overlap 5 others at a corner of the layer, this equates to 40(=5x8) facets to test for intersection.

When a crystal is added (as the first crystal to be seeded in a layer always is), as well as storing its vertex and facet data, it is also added to the line equation matrix. This is a matrix consisting of the equations for every edge that forms the crystal (along with its associated bounding sphere), i.e. 3 equations of a straight line in the x, y and z axis (parametric representation).

At the time of generation of the crystal to be seeded, the equation of the plane corresponding to each facet is generated (for the crystal and all its duplicates) from three points.

While methods could be employed to reduce this based on the quantity of duplicate lines within a hexagonal column and associating in the case of duplicates treating bounding spheres individually, the extra checks to separate them are likely to be as computationally expensive as dealing with the entire matrix in a single pass.

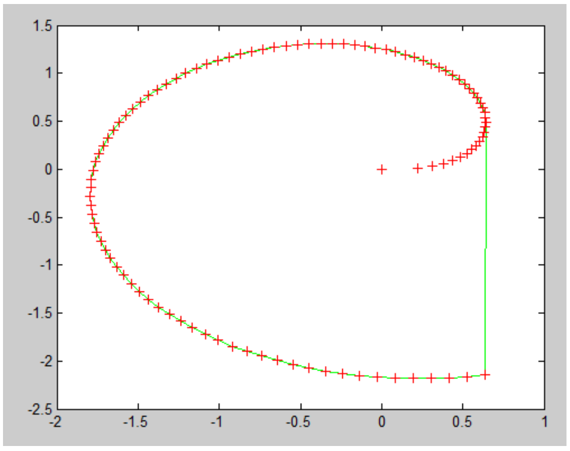


Fig.4.40.Green line forming convex hull around shape formed from red crosses

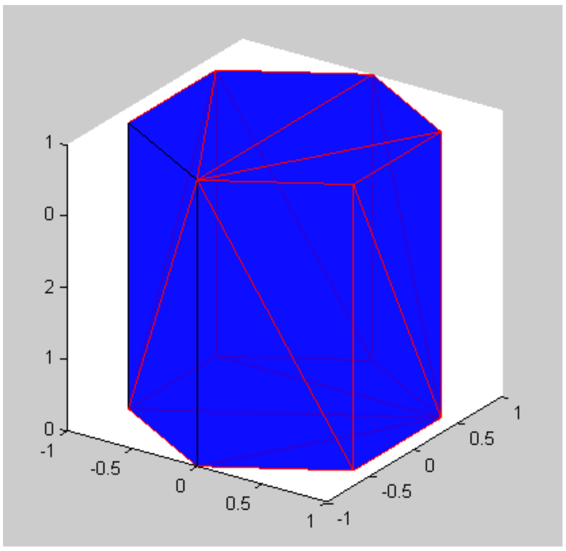


Fig.4.41. Delaunay triangular surface indicated in red forming the convex hull for a hexagonal column.

Each of these triangles is initially treated as an infinite plane in order to determine where on the plane each and every line forming a seeding crystal crosses it, i.e. the cross product of the plane and the line is determined (or ignored if the line and plane are parallel). This generates an array of intersection points between the lines and the planes.

A convex hull is generated for the seeding crystal (and its duplicates). A convex hull in this case is effectively a bounding polygon circumscribing all the points that form the object.

In the case of a three dimensional object this is formed through Delaunay triangulation.

The next step is then to determine if the point of intersection lie within the bounding limits of the Delaunay triangles *(see below).*

The operation is however speeded up as the intersection points for all lines and planes are tested against the entire convex hull formed from the entire array of Delaunay triangles in a single function.

This function returns an array of the lines that intersect the convex hull of the crystal. If the array is empty, then it is confirmed that there are no intersection points within the crystal.

Note that for crystals with large facets such as plates, the process has to be two way, insomuch as the planes of the seeding crystal have to be tested against all the lines of the previously seeded crystal (this is a step that can largely be omitted for needles and columns). The reason for this is that there is a strong possibility that a previously seeded crystal may have a facet through which a line passes even though no lines of the already seeded crystal pass through a facet of the seeding crystal.

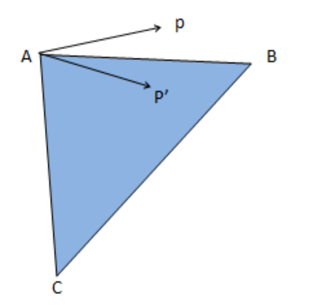


Fig.4.42. Determining if a point is within a triangle

Determining if a point is within a bounding triangle

A triangle is defined by the points ABC and therefore lines AB, BC and CA. Each of these lines splits space such that one half is entirely outside the triangle. A point inside the triangle must be below AB, left of BC and right of AC. Further, determining if the point is on the correct side of the line is achieved by determining the cross products. In the case of the cross product of **(OB-OA) x (OP-OA)**, the vector will be pointing out of the page, whereas the cross product **(OB-OA) x (OP'-OA)** will be a vector pointing into the page. In the above example the vector from **(OB-OA) x (OC-OA)** also points into the page. As such it can be seen that p' is below the line potentially within the triangle while p is above the line and therefore outside the triangle. In all cases **(OB-OA) x (Op'-OA)** has the same direction as **(OB-OA) x (OC-OA)**, and therefore is potentially within the triangle. It is just a case of doing the same analysis from the perspective of the other edges of the triangle. If all results prove positive then the point p' is within the triangle.

4.8.1 Code Speed

As detailed above, the test for convex hulls was a feature added to the code close towards the end of the course as a direct result of the inability of the bounding sphere model to achieve high densities equating to those studied by Kaasalainen without compromising other aspects such as average grain size. The improvement relied on not immediately rejecting a bounding sphere if it overlapped previously accepted and seeded bounding spheres. It would instead create an array of bounding spheres being overlapped. A crystal would still be created for the potential bounding sphere and from this the equations for its edges and facets would be determined.

For needles (crystals with relatively small facet dimensions to surface area), the equations for all the facets in all the crystals (where a bounding sphere produced duplicates due to crossing a layer boundary) were used to determine an array of intersect points against all the edge equations for all the crystals (and appropriate duplicates) belonging to all the previously seeded bounding spheres with which there was an overlap.

This method can only be described as 'quick and dirty' relying on processing power to generate a very large array of intersection points.

Consider a bounding sphere that due to crossing a layer boundary has produced 4 duplicates. For a hexagonal column, there are 8 facets per crystal which means 32 facet equations. As each facet is treated individually (so that it is known which vertex is connected to which), each crystal also has 36(=2x6+6x4) edges (equating to 108 equations for the lines in three dimensions). In this example there are therefore 144 edges.

If the bounding sphere(s) overlap 8 other spheres (which is not unusual), it can be seen that to determine the intercepts of all the facets of the seeding bounding sphere against all the edges of the previously seeded 8 crystals within the bounding sphere requires: 9,216(=32x36x8) solutions.

The reason why this was implemented in such a fashion was that at this time vertex data was stored independently of bounding sphere and there was no simple means of distinguishing between the duplicates. This was because up until this point, this data was simply not necessary as part of the bounding sphere design as once a sphere had been accepted, the crystal data was simply added to a growing array.

The first objective of speeding up the code was therefore an overhaul of how the data was stored. Each bounding sphere was updated with its duplicate number (set to 1 for the original crystal). When determining overlaps between seeding bounding spheres and previously seeded bounding spheres, an array could therefore be generated specifically determining which bounding sphere, duplicate and which (if more than one bounding sphere being seeded) was being overlapped.

overlapCrystalArray

```
  1  1  1
  1  1  2
  1  2  3
  1  2  5
  1  3  4
  1  3  6
  1  4  7
  1  4  8
  5  1  3
  5  2  1
```

Column1 – overlapping bounding sphere

Column2 – duplicate of the bounding sphere being overlapped with

Column3 – duplicate of the seeding bounding sphere responsible for the overlap

Fig.4.43.

In this case there are 10 overlapping situations, which means that for hexagonal columns, there are 2,880(=10x36x8) interception points to test compared with 160,512(=32x8x36x6) under the previous method.

4.8.2 Methods Pursued in Optimisation

MATLAB is a designed to deal with arrays in preference to loops. In the following example (Fig.4.44), two sets of code that generate the same results are presented. The first sample uses a standard 'for loop' while (though the data is pre-allocated to an array) the second uses built-in MATLAB functionality to process the entire array of entries (effectively) simultaneously:

```
tic                             %tic-toc starts and stops the clock
Bnv = zeros(size(A));            % We pre-allocate to level the playing field
for i=1:size(A,1)
    for j=1:size(A,2);
        Bnv(i,j) = log(A(i,j));
    end
end
nonvec = toc;
tic
Bv = log(A);
vec = toc;
assert(isequal(Bnv,Bv));
ratio = nonvec / vec

ratio =
   33.0086
```

Fig.4.44.

As can be seen, making use of MATLAB's ability to process arrays of data can lead to significant optimisation. The approach was used to replace this code:

```
for q=(1:size(overlapCrystalArray))
    for p=(1:size(planeEquation))
        for j=(1:size(potentialOverlapLineEquationMatrix))
            if          (overlapCrystalArray(q,1)==potentialOverlapLineEquationMatrix(j,1))...
                    &&  (overlapCrystalArray(q,2)==potentialOverlapLineEquationMatrix(j,2))...
                    &&  (overlapCrystalArray(q,3)==planeEquation(p,2))
             newfunction=subs(planeEquation(p,3),P,potentialOverlapLineEquationMatrix(j,3:5));
                t0=solve(newfunction);
                point1=subs(potentialOverlapLineEquationMatrix(j,3:5),t,t0);
                point1=double(point1);
                intersectArray=vertcat(intersectArray,point1);
            end
        end
    end
end
```

Fig.4.45.

with this:

```
bigArrayI=repmat(potentialOverlapLineEquationMatrix,size(planeEquation));
disp(' big arrayI')
bigArrayII=repmat(planeEquation,size(potentialOverlapLineEquationMatrix));
disp(' big arrayII')
bigArrayIII=[bigArrayI(:,3:5) bigArrayII(:,3)]
disp(' big arrayIII')
disp('new function')
[newfunctionII]=subs(bigArrayIII(:,4),P,bigArrayIII(:,1:3))
 [t0]=solve(newfunctionII)

for t=(1:size(bigArrayIII))
    point1=subs(potentialOverlapLineEquationMatrix(:,3:5),t,t0);
end
point1=double(point1)
disp(' big arrayIII function solving')
```

Fig.4.46.

Despite the benefits of using arrays, it was found that this method did not improve the speed of the code. Further investigative work in this field is therefore necessary.

4.8.3 Reciprocation of Convex Hulls

For needles and even columns, individual facets are relatively small with respect to the overall volume. Where one crystal intersects another during the seeding process they invariably do in such a way that at least two facets are overlapping, i.e. in the case of the seeding crystal at least one of its facets is intercepted by the edge of a previously seeded crystal. In the case of a hexagonal plate, where its end facets are account for a very large proportion of the overall surface area, it can easily be the case that the edges of the seeded crystal all lie outside of the facet of a previously seeded crystal even though the facet itself is intercepted by the lines. In these situations the test for interception has to be both ways, i.e. testing all the edges of one crystal against all the facets of the other and vice versa.

It was discovered post submission that there was in fact an error in the code which meant that this test (though commented out for speed) was in fact faulty due to incorrectly determining the array of vertices from the previously seeded crystals.

The reciprocation convex hull code is only used in the successful pass through the convex hull test for obvious reasons.

The other issue with hexagonal plates concerns their individual surface areas with respect to their volume. Once a density of 0.015g/ml is reached there is a notable increase in overlapping and therefore convex hull tests which often lead to rejection.

4.9 Improvements and Alternative Approaches

A considerable number of minor modifications to the code were made. Many of these simply replaced older routines and functions written in the early stages of development due to improved understanding of MATLAB and clearer objectives. They also fixed potential errors. The more relevant ones are noted below.

4.9.1 Improved Layer Data Record

The quantity of partial and whole crystals within a layer was recorded as the quantity of crystals seeded. This data was refined to include the quantity of crystals and the number of unique crystals. This makes for better comparison of data when considering the relative surface area of the layer compared with the volume, i.e. where the this is large there will be a much greater proportion of duplicate crystals.

The data record was also upgraded to record information about its corresponding bounding sphere and whether it was a duplicate. These are reported as seed and particle number.

4.9.2 Output Irrespective of Termination

An option that will undoubtedly be added to this is to generate and a new layer file after a set quantity of crystals are successfully seeded. This will cost insignificant extra time though will ensure that even the termination of the program due to time limits, the layer generated up to this point will at least be saved.

4.9.3 Tetris Model

Tests were carried out after modifying the code to generate layers following the 'Tetris' principle. Bounding spheres are initially set with a displacement from the bottom of the layer equal to the thickness of the layer. This displacement along the Z axis is then reduced by a set amount. At this new Z displacement, there is a bounding sphere test against all previously seeded bounding spheres. If there isn't an overlap with any previously seeded spheres, the Z displacement is again reduced and the process is repeated until either the Z displacement is zero (the centre of the bounding sphere is at the bottom of the layer, or there is an overlap.

Where there is an overlap, the process moves onto the convex hull test to determine of the crystals within the bounding box overlap. Where this is the case the crystal is then seeded at the previous step. Where the bounding sphere reaches zero displacement without overlapping another sphere, the crystal is then seeded at the bottom of the layer. In theory, the primary benefit of this model is to ensure that every bounding sphere adds to the layer whereas in the previous method, the convex hull test can lead to the seed being rejected.

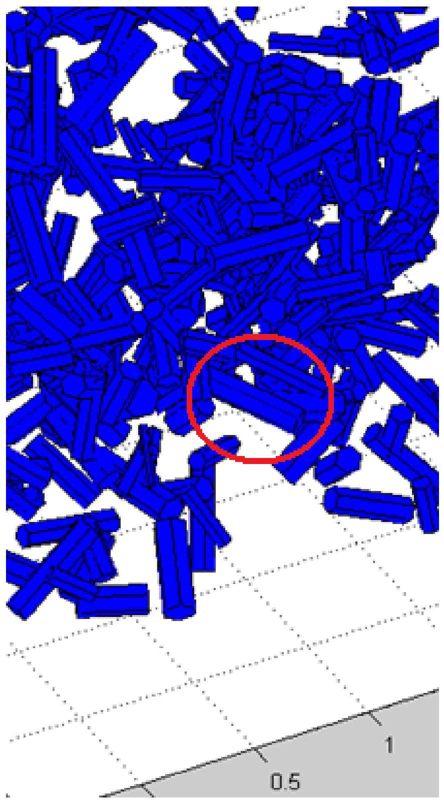


Fig.4.47. Void between 1 and 1.5 generated due to the bounding sphere of the circled crystal.

4.9.3.1 Issues
There were two issues with this approach:

- Modification of original code.
- Generation of voids.

4.9.3.1.1 Modification of Code
In the original code, once a bounding sphere had been accepted, the next stage of the process was to generate all the necessary facet plane and line equations for the crystals along with all the vertex data. This was initially generated as a single pass into temporary arrays. As a consequence of the changes, it was not sufficient to merely change the bounding sphere locations. The line and facet plane equations along with the vertex data had to be re-derived (this method was preferred over a second temporary array for the data). This caused issues initially on account of resetting various arrays within loops while previously these had simply been dropped and called each pass.

4.9.3.1.2 Generation of Voids
Testing this method resulted in the creation of large voids within the layer. These are generated when a large horizontal crystal 'rests' on top of a vertical crystal or group of crystals that have themselves stacked up, thereby 'roofing' them.

A method of solving this is to continue to the downward steps of the crystal, retesting the convex hull, while reserving the last point at which it passed the convex hull test and therefore be seeded. This will be explored further. If this method is employed along with seeding until a specific density has been reached, it is expected that:

- Large crystals may still be rejected as they cannot be seeded at any point in their descent.
- Smaller crystals will eventually dominate, thereby decreasing the average crystal dimensions as per early approaches based on 'termination after a specific number of rejections'.

The decision is therefore whether to terminate the seeding process at densities lower than specified if the average crystal size drops below a given threshold. The simplest mechanism for achieving this is through the reintroduction of the maximum number of rejections before the code is terminated. This is based on the probability that seeds rejected will lie within the upper radius range of the standard deviation curve.

4.10 Ray Tracing Program

The program works by tracing the paths of beamlets directed perpendicularly at the surface of the layer, through a series of refractions and reflections. Fraunhofer diffraction is accounted for only when the ray leaves the layer and, subject to the ray trace not being terminated for a beamlet due to limits placed on the starting parameters, the scattered ray will end up exiting at an angle to the incoming ray. The energy of the ray is distributed over all angular bins according to the far field diffraction pattern. The final energy in each of these bins is determined in order to generate graphs representing the scatting pattern generated by the layer.

The scattering program has input parameters corresponding to:

- Layer File Name
- Number of Layers
- Wavelength in microns
- Real part of the refractive index of the crystals
- Imaginary part of the refractive index of the crystals
- Maximum number of facet interactions per incoming ray
- Maximum number of total internal reflections
- Crystal distortion
- Edge length of square

Layer File Name
This simply refers to the file generated by the layer program. The scattering program imports this file, extracting the data it needs.

Number of Layers
As the crystals are permeable to light and as not all the rays will be scattered back out of the surface, increasing the number of layers above 1 means that rays that pass through the first layer are then inserted into a second layer (actually they are re-entered into the first layer with their vertical co-ordinate offset by the thickness of the layer). This increases the degree of scattering.

Wavelength in Microns
This is wavelength of the rays being used by the scattering program. While approximately the same wavelength as that used by Kaasalainen (they used 632.8nm He-Ne laser)was appropriate for developing virtual analogues, a range of wavelengths were employed for purposes simulating smaller crystals and for testing if coherent backscattering increased due to stronger diffraction.
As diffraction is based on the ratio of wavelength of the incident light to the size of the facet (acting as an aperture) and particle it belongs to (the diffraction patterns of the individual facets add up to the diffraction pattern of the particle), modifying the wavelength was a simply means of modifying the size of the crystals without having to generate a new layer.

In the case of the improved seeding sample, the wavelength was increased in order to widen the backscatter peaks and observe the increased coherent backscatter.

Real Part of the Refractive Index

As the layers were simulating water ice, the real part of the refractive index was set to 1.31.

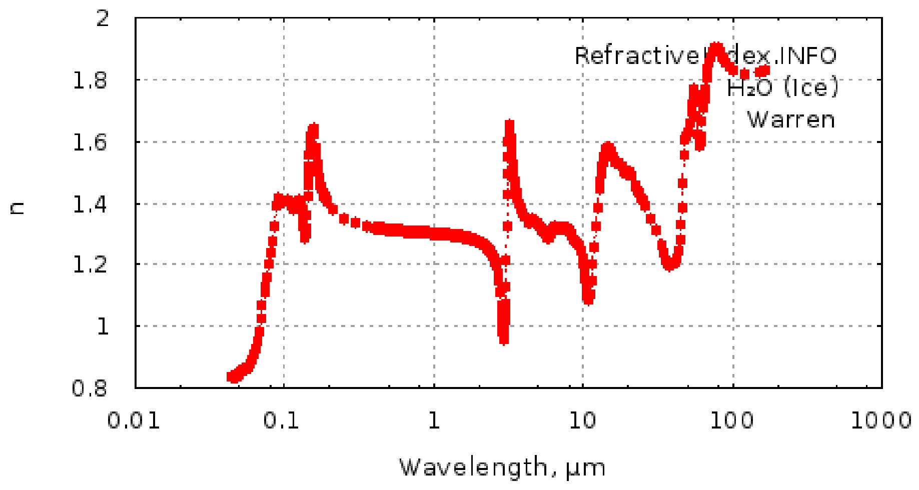


Fig.4.48.Refractive Index of Water – Real Part

Imaginary Part of the Refractive Index

Also known as the extinction coefficient, representing how much energy of the ray is absorbed as it

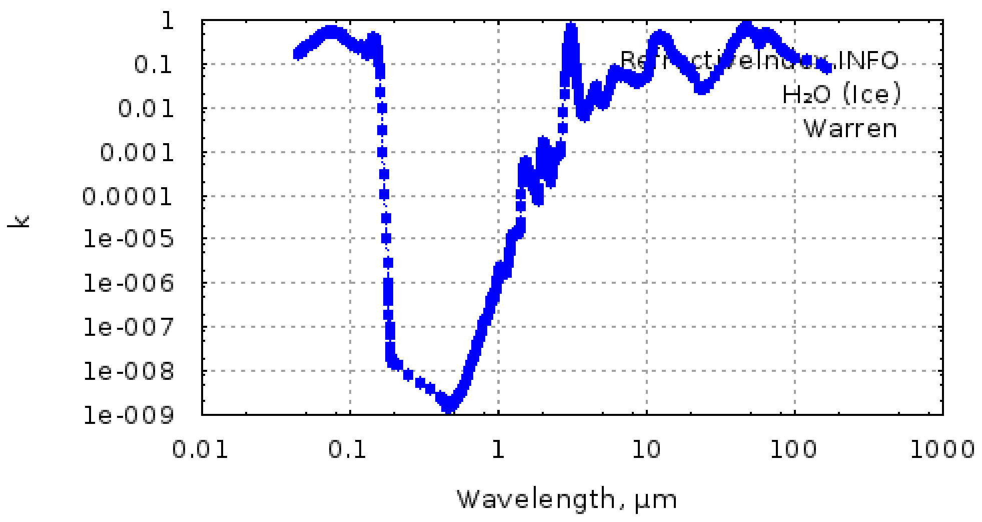


Fig.4.49. Refractive Index of Water – Imaginary Part

passed through ice. This is defaults to $1.04\text{x}10^{-8}$, for 630nm.

Maximum Number of Ray-Facet Interactions

This sets the maximum number of interactions with crystal facets calculated for the incoming ray and its descendents. This is used to determine the raytree. The raytree contains the incident ray and all rays descending by reflection and refraction at crystal facets up to the maximum number of ray-facet interactions providing that the energy of the descendent ray is greater than 10^{-10} times that of the incident ray, at which point the computation stops. If the number of interactions is set to 1 then there are 2 paths (the reflected and refracted path), following the interaction. Only external reflection is accounted for (as the refracted path never reaches a 'detector'. If the number of interactions is set to 2, external reflection and transmission at the next facet are accounted for. The number of potential paths is therefore 2^(maximum number of facet interactions). A standard setting of 10 means that a raytree will have a potential of 1024 (2^10) termination points. In all probability the number of termination points will be much lower as for a start, reflection from the first interface will in all likelihood never enter the layer (unless the facet happened to be at a very oblique angle to the incident ray) and will instead undergo diffraction (and therefore cover the complete range of angular bins).

Once the paths have been determined, the E-field amplitude matrix at all interactions is registered down the raytree until the end points. Rays exiting the layers are registered in bins distributed around the layer. These record the amplitude matrix of the E-field in the far field. Irradiance is then calculated from the contributions of all E-field amplitudes (accounting for phase).

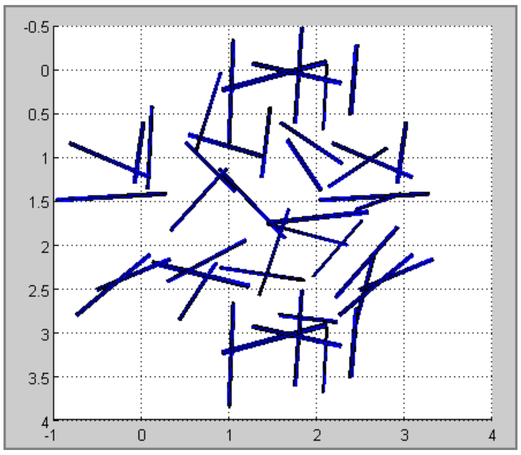


Fig.4.50.The low density of this needle crystal layer will account for few interactions by rays before leaving the layer.

For each incident ray, descendent rays are only calculated up to the set number of interactions. If for example initially there was 500,000 rays and none left the layer there would be approximately 5 million raypaths after the first pass. As each of these would then form new branches, it is theoretically possible for there to be 5 billion rays to trace.

In theory this will eventually decrease as the rays leave the layer. In the adjacent image of needle crystals, it can be seen that from the top view, the crystals only account for a modest amount of the projected surface area. The vast majority of incident rays will therefore pass through the layer. Even stacking multiple layers will not alter this as the layers are directly above each other. The alternative is to increase the depth of the layer, thereby increasing the likelihood of a ray striking a crystal rather than passing through the layer. Increased depth to the point where a ray is incapable of passing through a layer without

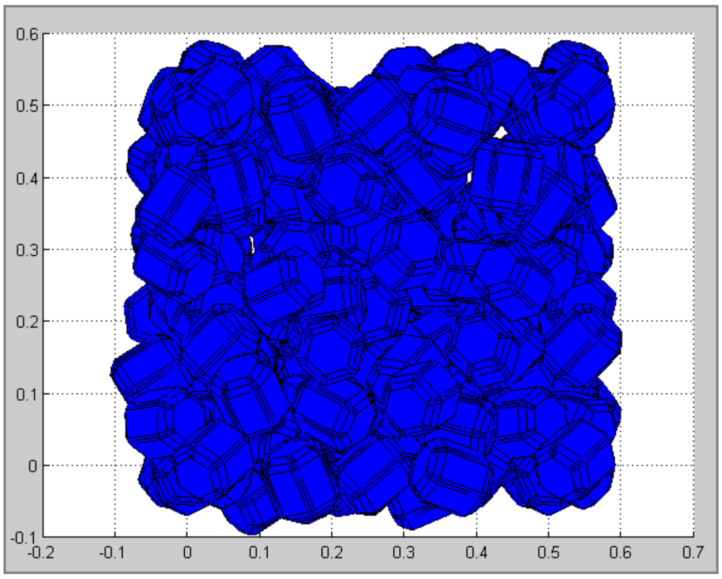


Fig.4.50. A dense layer of rounded hexagonal columns will lead to very large raytrees.

encountering a crystal has its issues. Under these circumstances, the number of rays in a raytree starts to increase significantly to the point where the program terminates due to processing time constraints before completing the ray trace. This occurred for the round crystals where the depth was sufficiently deep so as to ensure that virtually all rays generated their complete raytrees.

Maximum Number of Total Internal Reflections
This is the maximum number of total internal reflections that will be followed before the program terminates the ray. Termination will essentially increase the lost energy, energy unaccounted for by the program. If set too low then it will leave too much energy within the layer as the program terminates. This can lead to an unrealistic result. If set too high and there are crystals that generate a large amount of total internal reflections due to the angle between facets, this can cause increases in the time requirement to run the program.
The default is 100. This should give sufficiently high enough opportunity for the ray to leave the crystal while keeping running times as efficient as possible.

One of the features of the program is to record the missing energy. This is the energy that has not left the layer and has not been absorbed (as rays leaving the sides of the layer are re-entered into the layer). Where absorption by crystals equals zero, this is the energy lost through termination of the ray after 10 ray-crystal surface interactions (and any, if any rays that undergo more than 100 total internal reflections).

Crystal Distortion
This represents the degree of irregularity to the surface of a facet. It corresponds to random chance that the facet will be treated as off its designated alignment for the purposes of the reflection/refraction of an incident ray. For the purposes of this thesis it is been set to 0.0, representing pristine crystals. It is mentioned here purely for completeness in the description of the ray tracing program.

Edge Length of Square
The ray tracing program uses the dimensions of the layer perpendicular to the incident ray. From this, the number of rays of cross-section = (edge length of square)^2 and their start positions are calculated. The ray diameter needs to be small enough to reach conversion of the diffraction pattern. The exact value depends on the facet areas and gap areas between them. For all child rays of these rays leaving the layer, Fraunhofer diffraction on a circular projected cross-section is calculated.
Determining the value of this is a matter of estimation for best results against time. The smaller the value, the greater the quantity of rays and the greater the computational overhead, while the larger the number the fewer the rays but also the lower the accuracy when accounting for diffraction. The weighting of the typical facet (aperture) size against the cross sectional area of the ray is therefore important.

The diameter of the ray needs to be small enough compared to the facet, so that details of the facet shape are considered appropriately when calculating the surface integral for diffraction (eqs.5a&b in Hesse et al. JQSRT 113(2012)342).

$$\overrightarrow{E_{\parallel x'z'}}(\vec{x}) = \frac{ie^{ikr}E_{0\parallel}cos\alpha}{2\pi r}\left(\vec{k}\ \mathrm{x}\ \overrightarrow{\epsilon_2}\right)\mathrm{F}(\mathrm{S}, \alpha; \vartheta', \varphi')$$

Where:

k=2π/λ is the wave number

$\overrightarrow{E_{\parallel x'z'}}(\vec{x})$ is the component of the incident wave polarised parallel to the x'z'- plane at arbitrary direction of observation $\vec{x}$.

$\vec{k}$ is the normalised wave vector in the direction of observation

r is the length of the vector $\vec{x}$ from origin to the observation point

α is the angle of incidence of the plane wave with the wave vector $\vec{k}_0$

$\overrightarrow{\epsilon_2}$ is the unit vector in the y' direction

$\mathrm{F}(\mathrm{S}, \alpha; \vartheta', \varphi')$ is the surface integral of the exact field with respect to the polar coordinates of $\vec{k}$

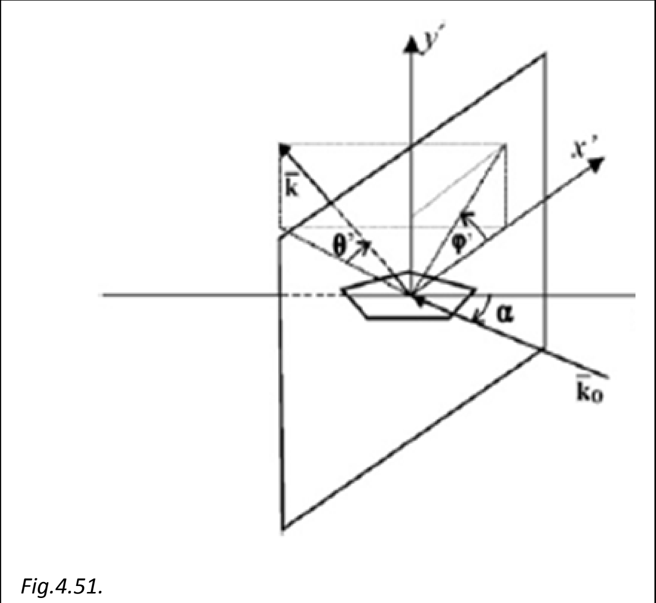


Fig.4.51.

In the samples tested, the integral is the sum of the integrals over the ray cross sections of all rays passing through the same set of facets before leaving the layer. For example a raydiam =0.0025, equated to approximately 1/40 times the mean hexagon diameter of 0.1.

Similar extrapolations are used to determine the perpendicular polarised component of the wave.

Scattering Image
As rays leave the layer they are registered in a bin corresponding to their zenithal and azimuthal scattering angle. Each bin is a complex 2x2 matrix that records the E-field of the rays entering them. After the raytrace has been completed the irradiance distribution is calculated from the E-field. This step can only be completed after all the contributions to the E-field have been generated on account of the irradiance being calculated from the square of the sum of the individual contributions. As some of these may be negative, the sum needs determining first.

Samples Simulated
Four samples were simulated using the program, corresponding, as closely as was possible to the data collected by Kaasalainen. Three of the samples represented surface samples composed of short columns, plates and needles. The fourth sample generated used the rounding edges crystal. This layer differed in that the generated crystals were uniform in dimension.
The samples were used in the scattering program to generate graphs of intensity against scattering angle.

4.11 Sample Sizes

While the program is capable of generating samples of unlimited volumes, this is not practical in terms of processing time both for the layer generation and the ray tracing program. As can be seen, even a small sample can easily have 640k initial rays to trace. In practice the samples have to be severely limited in order to make efficient use of available computing resources.

While the samples examined by Kaasalainen had a diameter of 4mm corresponding to 12.57mm^2 the samples for the rounded hexagonal columns were 1mm^2 though they had a depth of 2mm. The purpose of having a significantly higher depth was to ensure that all rays interacted with a crystal at some point during the initial pass through the layer.

For each layer, a suitable ray diameter is chosen along with a number of layers. The output is then examined on a scatter graph and analysed.

Crystal Type	**Needle**	**Improved Needle**	**Plate**	**Compact Column**	**Rounded Column**
Layer Density	0.0169g/ml	0.01662g/ml	0.04581 g/ml	0.08050 g/ml	0.12564 g/ml
Layer Dimensions	3x3x2mm^3	4x4x2mm^3	2x2x2 mm^3	1x1x1 mm^3	1x1x2 mm^3
Ray Diameter	0.0025	0.0025	0.0025	0.0025	0.0025
QTY Crystals	97	252	305	143	445
Crystal Radius	(50±8)μm	(50±8)μm	(300±150)μm	(50±7)μm	50μm
Crystal Length	(1,000±150)μm	(300±50)μm	(50±10)μm	(100±15)μm	100μm

Kaasalainen Samples

Crystal Type	**Needle**	**Plate**	**Compact Column**	**Refrozen (Rounded)**
Layer Density	0.10/ml	0.07 g/ml	0.08 g/ml	0.12 g/ml
Crystal Size	300μm	500μm	100μm	200μm

4.11.1 Needle Layer

The original image reveals that classifying the

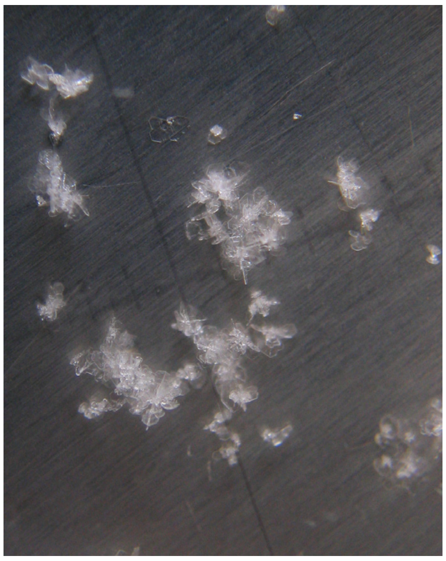


Fig.4.52.Fresh show sample 7th March 2005 – Log states there are no hexagons, just the needle-like small grains from an ongoing snowfall (-2C temperature).

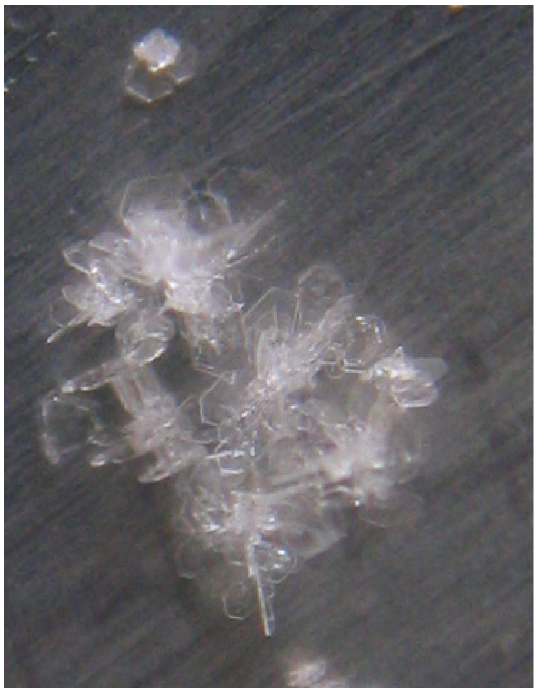


Fig.4.53. Zoomed in section indicates the presence of hexagonal plates in large amounts.

crystal shape of this sample of fresh snow is somewhat subjective. The presence of hexagonal plates within the same sample as the needles may have influenced the phase function. This cannot be confirmed as data was collected from specific areas of the sample rather than from the sample as a whole. It is entirely possible that the needle layer data is a product of instances where the needle to hexagonal plate ratio is high.

The enlarged image (fig.4.53) also reveals that the hexagonal plates are not always separate crystals. It is very common for hexagonal plates to be attached to a column (CP1a in the Magono-Lee classifications, appendix E) or in the case above form into radiating assemblages of plates (P7a). It can clearly be seen that there are distinct differences between the model and the image of the sample.

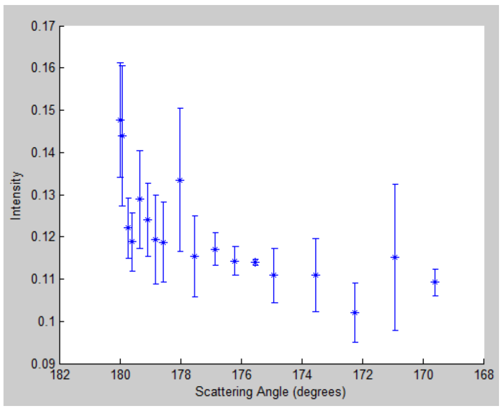


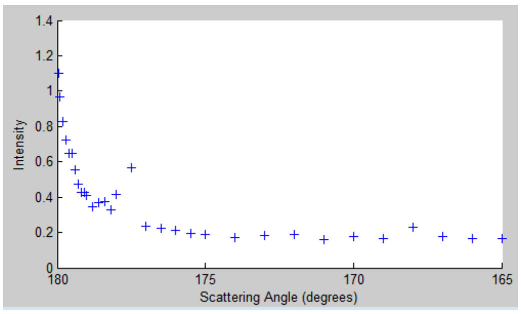
Fig.4.54. Scatter plot generated from Kaasalainen data 7th March 2005. Density 10%, grain size ≈300µm

Kaasalainen data has been plotted and compared with the phase functions for both the needle layer (density 1.81%) and hexagonal plate layer (density 5%). The crystal (grain) sizes are approximately of equal size. Note that the plate layer though being seeded using the radius parameter 600µm will be much smaller due to high rejection parameter, leading to the acceptance of smaller crystals with increasing time.

In case of the Kaasalainen data the error bars are large corresponding to noisy data, though a backscatter peak is present and calculated as I(180)/I(175) = 0.15

In both model cases, the scattering peak has a large incoherent component to the backscatter (due to the coherent backscatter to background intensity factor limit of 2).

Fig.4.55. Comparison with layer model phase functions



Needle layer. Density 1.81%, grain size ≈300 µm

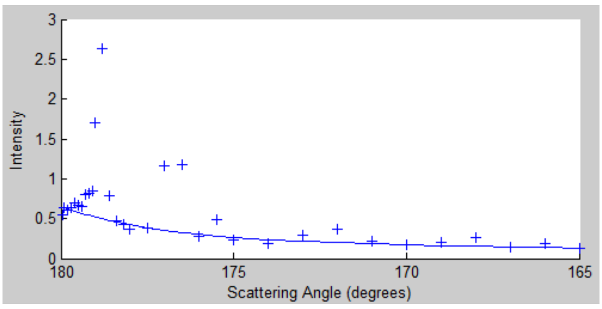


Plate layer. Density 5%, grain size <600 µm

Layer	*I(180°)/I(175°)*	HWHM	Density
Needles	3.38	0.39	1.81%
Hexagonal Plates	2.46	1.8	5%
Kaasalainen	1.32	0.1	10%

From a qualitative perspective the phase function for needles is a closer match to snow sample classified as needles though from the data neither layer appears particularly close in terms of both relative intensity and half-width half-maximum. In terms of density however neither model is close.

For the samples modelled, the density for the snow samples was achieved by weighing a filled container, the surface of which was levelled off with a spade (to remove excess snow). Presumably this will have altered the sample density to some degree by creating voids, or compressing the sample. As presumably this sample was not the one used on the goniometer though have to presume that all possible efforts were made to ensure that the sample measured was as close to that weighed as possible.

4.11.2 Hexagon Sample

The image is a cropped version of the original, highlighting important features and raising three important points:

- While classified as hexagons by Kaasalainen, this is classified as stellar crystals (P1d) and ordinary dendrites (P1e) by Magono-Lee.
- The layer model for hexagons is inappropriate for modelling this sample.
- There are large voids in the sample, calling into question density measurement (as noted in section above) and sample size, in particular as to whether there was complete randomisation of crystal orientation for the purpose of measurements.

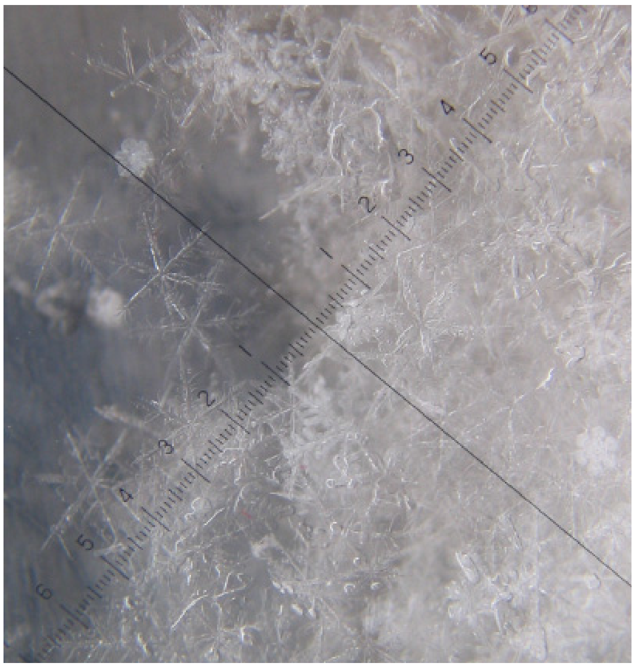


Fig.4.56. Hexagons. Scale is in millimetres

At present the modelling of layers only extends generating random crystals that are effectively hexagonal columns. These can be plates, compact columns and needles through defining the radius and length component input parameters. Features such as indents and bullets can be included through replacing either or both basal facets with 6 triangular facets. Further, the column can be classed as broken through giving an inclination to the basal facet.

There are options in the code to replace the crystal generation function with a call to a pre-generated crystal, allowing a single crystal to be randomly orientated and seeded into the layer. This has been used in the creation of the rounded column layer described in the thesis. During early development it was also used to generate layers of combination of bullets (C2a). The benefit of this is that these compact arrangements will both generate high density layers through bounding spheres, though the benefit of convex hull test if applied is negligible as can be seen in Fig.4.57(b). As such it is likely that models of stellar crystals will in fact be models formed using hexagonal plates and replacing the plates with stellar crystals prior to creating the layer. As modelling hexagonal plates has a high convex hull failure rate there are clearly issues to be solved.

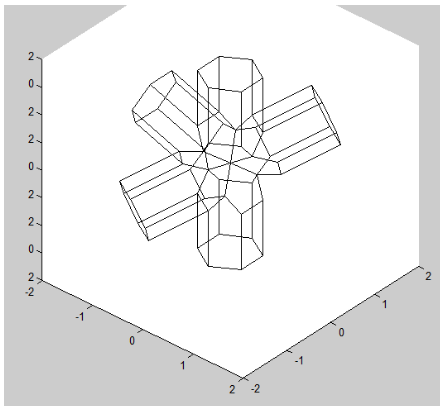


Fig.4.57(a). A pent-bullet rosette (C2a)

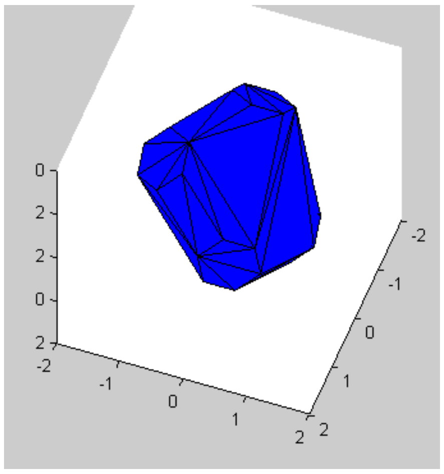


Fig.4.57(b). Convex hull of a pent-bullet rosette (C2a)

5. Results

In this section the data from the four types of samples is set out. In the case of the first three, Needles, Plates and Columns, the seeding process used an on-the-fly method of generating the individual crystals. This allowed for generation of crystals of differing size. In the case of the Rounded Column, a single crystal was generated. The code for generating Rounded Columns had been written in Fortran and was not therefore compatible with the Matlab code used for generating the layers. There was insufficient time to convert the code into Matlab.

In each of the cases the layer generated through the seeding process was subjected to the RTDF program and the data was recorded. In most cases the overall phase function was produced and this was compared to the findings by Kaasalainen (2006). In a few cases two-dimensional scattering data is presented. This is useful in identifying artefact bright patches that are likely to correspond to external reflection from individual facets. Domination of the overall scattering results by few large reflecting facets is an unfortunate side effect of having small sample sizes. Even though the re-entry of rays leaving via the sides of the layer, the overall structure is a repeating pattern and as a consequence, it is reasonable to assume that a few large facets can end up accounting for a large amount of the intensity on account of the relatively low quantity of seeded crystals and therefore the incomplete randomisation of the orientations present. Where this has occurred, it equates to a bright patch in the 2D-scattering image such as in this example:

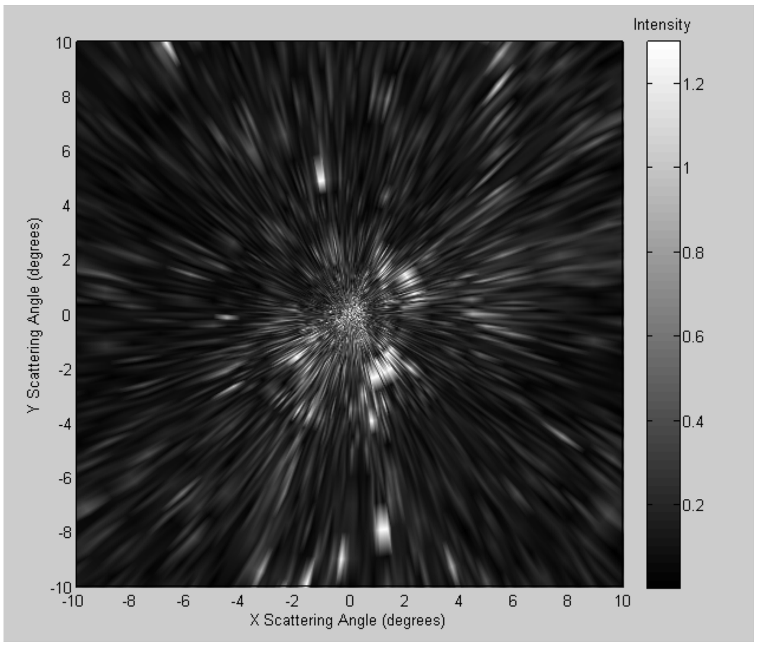


Fig.5.1. Polar Plot of Intensity against scattering angle for needles L0.3 x R0.05

Where the phase function bears a passing resemblance to the findings of Kaasalainen's or there is at least an indication of a backscatter peak, analysis of the peak is conducted using a similar method to that used by Kaasalainen. This is done to produce two figures, the half-width-half-maximum (HWHM) of the backscatter peak and *$I(180^o)/I(175^o)$*. For ease of determining these, a best fit function is used to give the relative intensity at scattering angles. This function takes the form *$I(\alpha)=a\ exp(-\alpha/d) + b + k\alpha$*, where *a, d, k* and *b* are empirical parameters defining the backscatter peak and linear part of the intensity (Piironen, 2000).

As the samples produced by the program were treated as perfectly transparent (zero absorption), shadow hiding effect plays less of a part in the results (it will still prevent direct illumination of particles that do not form the surface of the layer) as the rays are not absorbed after a few ray surface interactions. As such the backscatter peak relative to baseline (or linear part of the scatter intensity curve) is generally expected to be small. Shadow hiding still plays a role as confirmed by

Kaasalainen. Compressing samples increased the intensity of reflected light by reducing the quantity of rays entering the layer. A possible reason is demonstrated as is shown in the adjacent diagram. It can be seen that were there are larger gaps rays are able to penetrate deeper into the layer before encountering the first surface. The reflected ray may therefore encounter another surface rather than leaving the layer.

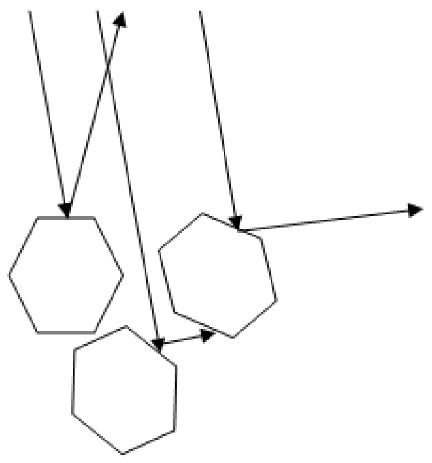


Fig.5.2(a). Normal scattering by irregular uncompressed surface

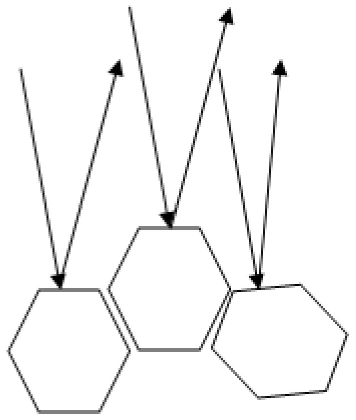


Fig.5.2(b). Compressed layer leads to fewer rays reflecting into the layer

Significant Features

As well as the principal backscatter peak, other features in the results are important as a means of determining if the layer model is simulating real situations. One feature in particular is identification of the 22^o halo in the forward phase function presuming that the rays passing through are not completely absorbed (for perfectly transparent material this is residual energy in the layer when the program completes) or backscattered. This will be apparent in the azimuthally resolved scattering results either as a partial or full bright circle. As it is a feature of single scattering, as the density of the layer increases, multiple scattering will increase and as a consequence, the halo will disappear.

5.1 Needles

Layer Data
Hexagonal crystals
Layer $(3x3x2)mm^3$
Rejections 25,000
Crystals Seeded 97
Crystal length (1,000±150)µm
Crystal radius (50±8)µm
Max indent 10%
Total Crystal Volume 0.304 mm^3 (1.7%)
Crystal density 0.91670g/ml
Density 0.017g/ml
Size Parameter 492

Due to the bounding sphere method of seeding the crystals, the needle layer suffered from very low densities. Even though this layer was generated through the bounding cylinder method which constrained rotation in the z-axis, there are large voids between the crystals on account of the large diameter of the bounding cylinders. The density of the sample examined by Kaasalainen was 5 times denser than this layer.

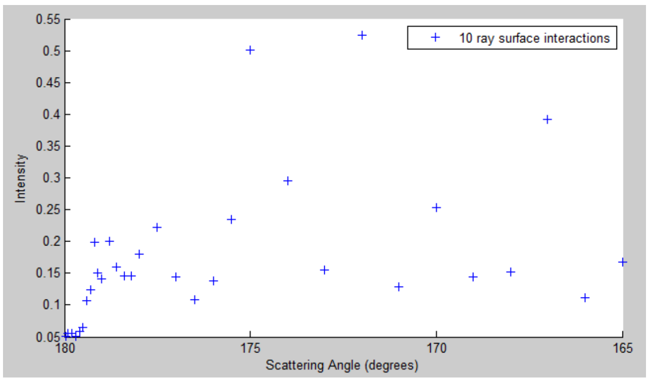


Fig.5.3(a). 10 ray-surface interactions

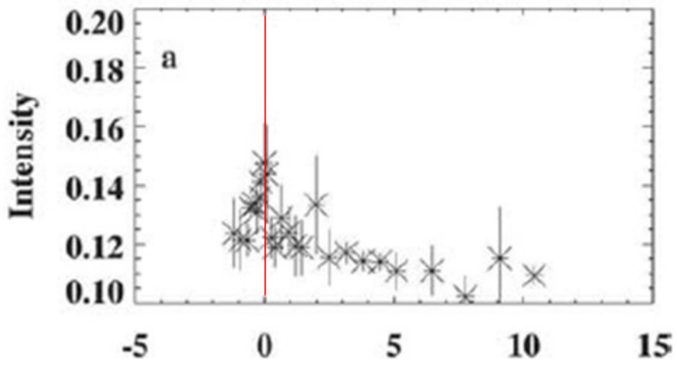


Fig.5.3(b). Needles (Kaasalainen, 2006 – Fig 7a, 7March 2005)

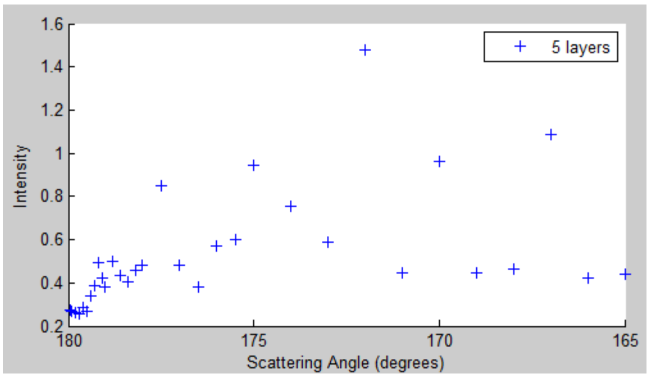


Fig.5.3(c). Needles – increasing layers to 5

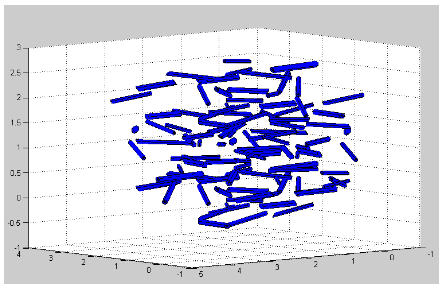


Fig.5.4(a).Layer model

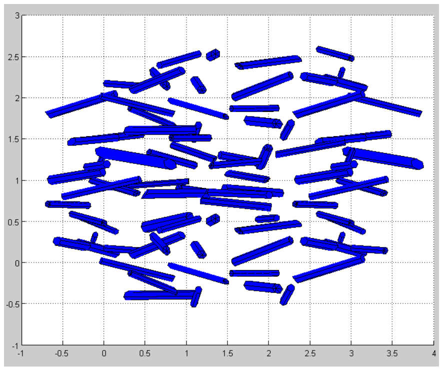


Fig.5.4(b). x-z orientation revealing lack of randomisation

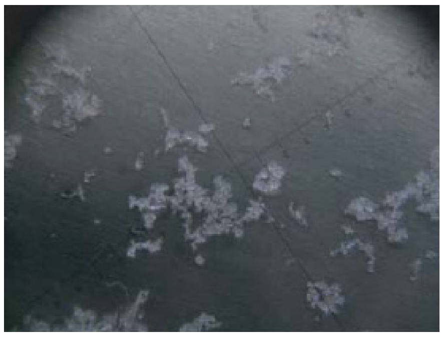


Fig.5.4(c).Needle sample (Kaasalainen, 2006)

The scatter graph Fig.5.3(c) is for 10 ray surface interactions. There is no indication of a backscatter peak. There are also little obvious qualitative similarities to the findings of

Kaasalainen (centre scatter graph for needles of similar size).

The large peaks (178^{o}, 175^{o} & 172^{o}) are likely caused by surface reflection.

It is probable that the layer was dominated by reflection from individual facets of the crystal due to only partial randomisation of the crystal orientations and on account of the few crystals present.

Further, the use of 25,000 as the rejection parameter (program terminates after it has failed to seed a crystal after 25,000 attempts since successfully seeding the last one) meant that by the time the program terminated, typical bounding spheres had diameters of 700µm's.

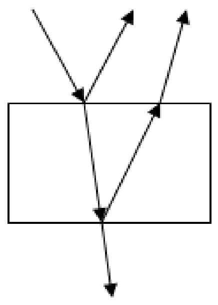


Fig.5.5(a). Single layer with rays leaving through bottom

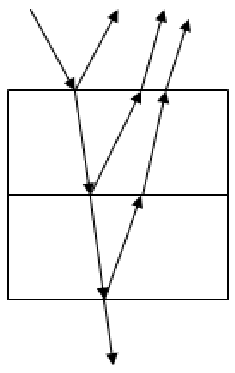


Fig.5.5(b). Multiple layers result in more rays backscattering

A means to simulate scattering by a thicker layer of snow was achieved through the use of stacking layers. It was anticipated that this would lead to more light returning to the surface rather than being lost through the bottom of the layer and therefore an increase in overall intensity (see adjacent diagram).

It can be seen that the only significant differences between the two scatter plots is an increase in intensity. There is very little change in the overall shape of the scattering curve though with increasing lays the intensity increases. This is expected as some of the light that was previously leaving the layer through the bottom will instead leave through the top contributing to backscatter.

5.1.1 Improved Seeding Technique

Layer Data
Hexagonal crystals
Layer (4x4x2) mm^3
Rejections 100
Crystals Seeded 481 (318 after convex hulls)
Crystal length (300±50)µm
Crystal radius (50±8)µm
Max indent 0%
Total Crystal Volume 0.58017 mm^3 (1.81%)
Crystal density 0.91670g/ml
Layer Density 0.01662g/ml
Size Parameter 492

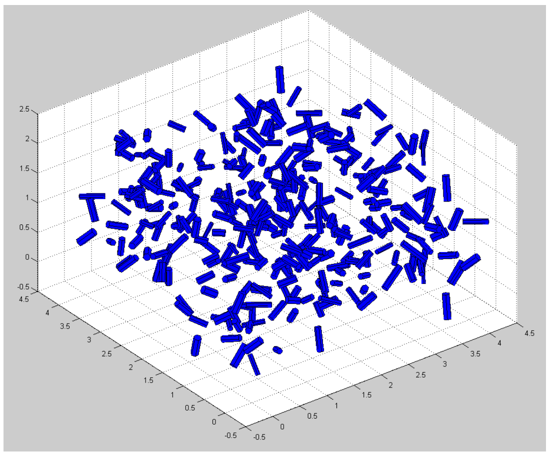


Fig.5.6. Increased layer density due to testing overlap between individual crystals

While bounding spheres overlap during the seeding process, there is a very strong chance especially in the case of needles that the needle being seeded does not share space with already seeded needles. Improvements to the seeding routine used convex hulls to determine if the needle to be seeded and all existing needles within overlapping spheres actually intersected.

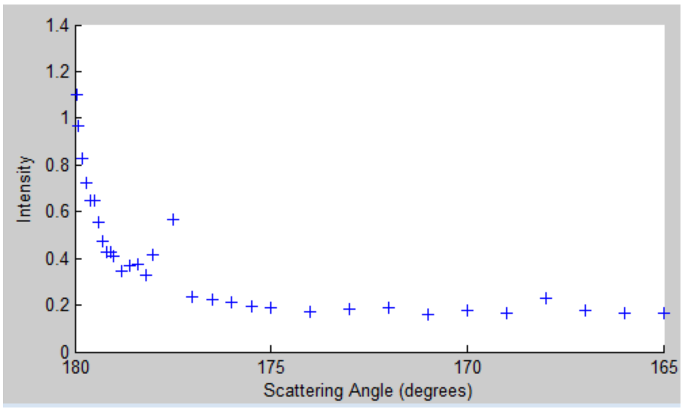


Fig.5.7(a). Layer of needles revealing similarity to Kaasalainen's results with backscatter peak and peak at 177.5° (2.5°) despite still having a lower density

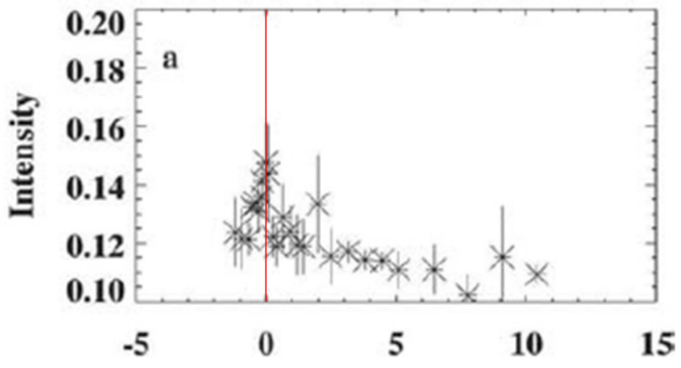


Fig.5.7(b). Needles (Kaasalainen, 2006 – Fig 7a, 7March 2005)

The process proved very slow on account of the processing time for the function. The density for the layer was increased six times that of the above layer though was still a sixth of the density of needles in the Kaasalainen samples.

With 252 unique crystals equating to 318 when accounting for layer edge duplicates forming the layer, i.e. 2544 facets, layer provided data that was qualitatively comparable to the phase function recorded by Kaasalainen (see adjacent image). The results for the single layer bear a similarity to the findings of Kaasalainen with a sharp peak at 180° (0) and 177.5° corresponding respectively to 0° and 2.5° in Kaasalainen's results.

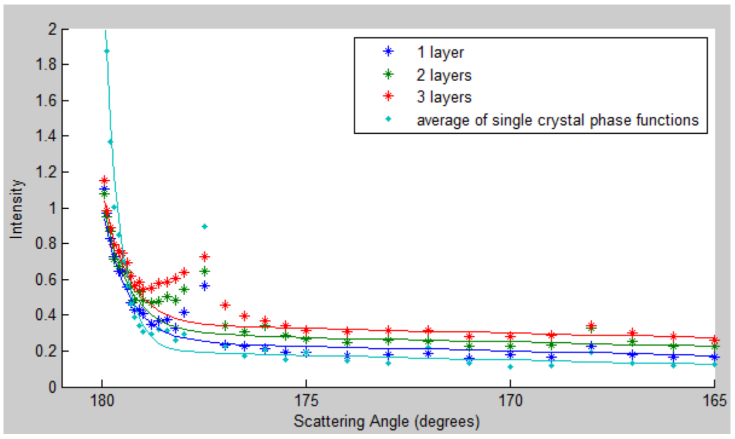


Fig.5.8. For size parameter 492

Increasing layers in order to simulate deeper samples of snow give similar curves though the overall intensity increases. A feature is a reduction in intensity and a broadening of the backscatter peak compared with the phase function for a normalised averaged single crystal.

This is in agreement with Kuga and Ishimaru's 1988) observations for large size parameter and low layer density. For a higher size parameter, the phase function for the media should have a similarity to that of a single particle. In the case of spherical particles this is the Mie-scattering region.

The results are given the table below.

Layer	*I(180°)/I(175°)*	HWHM
1	9.50	0.42
2	3.81	0.39
3	3.38	0.39
Average of single crystal	13.71	0.25
Kaasalainen	1.32	0.1

5.1.2 Modifying Ray Diameter

Ray diameter was initially to 2.5 μm corresponding to approximately 1/40th of a 100 μm diameter hexagonal facet. Halving the ray diameter increased the quantity of rays and therefore the accuracy of the phase function, in this instance resulting in slightly increased backscatter. The improved accuracy however was not sufficiently significant to justify increasing the processing time from 8hr40 minutes to 32hr31 (effectively the inverse square of the diameter change factor). It was deemed that the keeping a ray diameter of 2.5 μm under virtually all circumstances was acceptable.

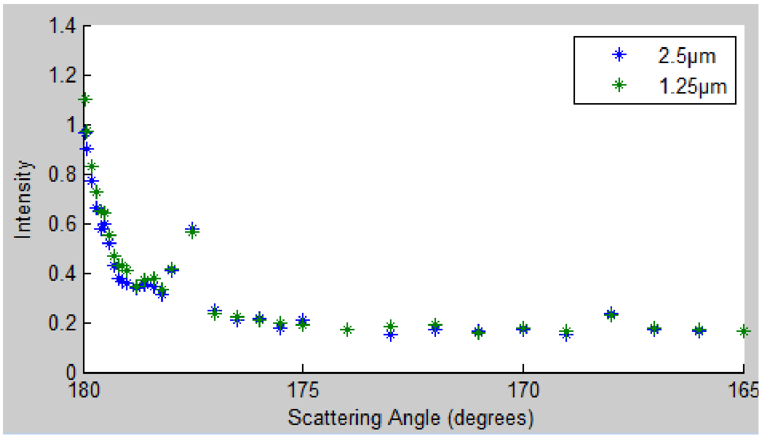


Fig.5.9.

5.1.3 Needle Layer with Convex Hull Test

There are still issues with the seeding routine though density has increased dramatically. Memory failure eventually terminated the program before it achieved target density. At this point it is uncertain as to whether this can be solved through pre-allocation of specific matrices or whether this is simply due to how MATLAB handles very large data arrays and as such requires a higher computing specifications.

Previous Improved Seeding
Layer Data
Hexagonal crystals
Layer (4x4x2) mm^3
Crystals Seeded 481 (318 after convex hulls)
Crystal length (300±50)µm
Crystal radius (50±8)µm
Max indent 0%
Total Crystal Volume 0.58017 mm^3 (1.81%)
Crystal density 0.91670g/ml
Layer Density 0.01662g/ml
Size Parameter 492

Post Submission
Layer Data
Hexagonal crystals
Layer (4x4x2) mm^3
Crystals Seeded 1311 (1203 unique)
Crystal length (300±50)µm
Crystal radius (50±8)µm
Max indent 0%
Total Crystal Volume 2.26535 mm^3 (7.1%)
Crystal density 0.91670g/ml
Layer Density 0.06490g/ml
Size Parameter 492

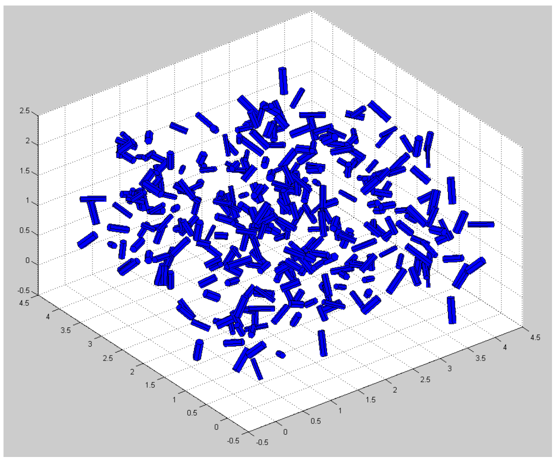


Fig.5.10(a). Pre-convex, density 1.81%

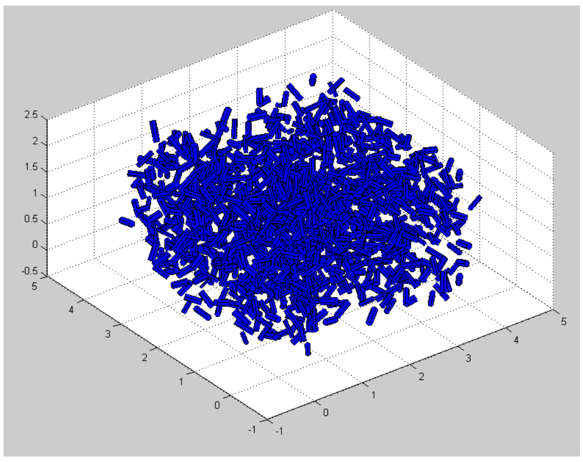


Fig.5.10(b).Post–convex update, density 7.1%

The target density (Kaasalainen, 2005) was 0.10g/ml as such the layer achieved 65% of the required density.

RTDF was processed on this layer for 10 ray-surface interactions. The data for 1 layer, 2 layers and average of single crystal phase function were collected for analysis.

In the data produced for the low density layer a feature is a reduction in intensity and a broadening of the backscatter peak compared with the normalised average of all single crystal phase functions.

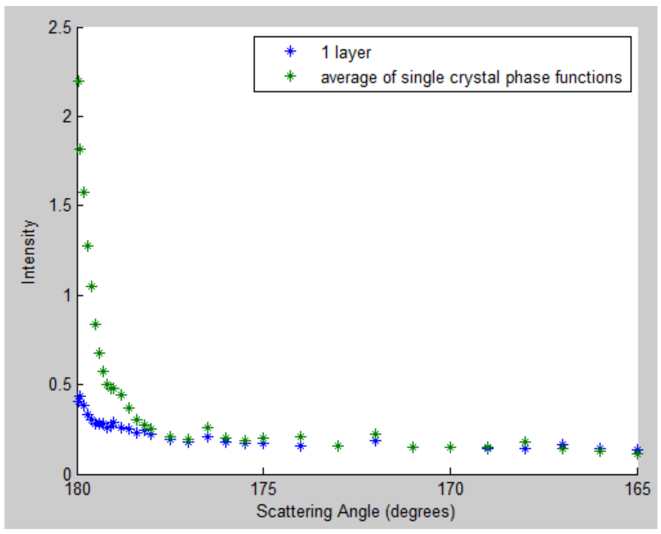


Fig.5.11. Comparison of 1 layer with average of single crystal phase function

The anticipated broadening of the central peak is not in evidence for the higher density layer.

The decreasing overall brightness with increased density diverges from the observations by Kaasalainen.

"We also attempted to determine the effects of packing density by mechanically compressing some samples or allowing the sample to flatten out overnight. The only observed effects were an increase in surface brightness..."

The backscatter peak for the 1.81% is significantly higher than the background intensity and as such cannot be wholly attributed to coherent backscatter. As such it is probable that the increase in particles has reduced incoherent backscatter caused by surface reflection and factors such as small sample size and lack of complete randomisation of particle orientations. Further, while a low density layer is comparable to a cloud of independently scattering particles whereas as the density increases to 7.1% it is closer to multiple particle scattering layer insomuchas the rays leaving one particle are more likely to enter another crystal and leave in a different direction to direct backscatter.

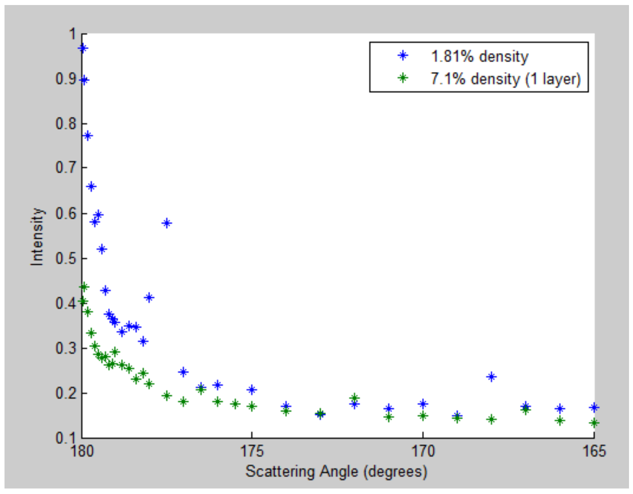


Fig.5.12(a).Comparison of phase functions for different density needle layers.

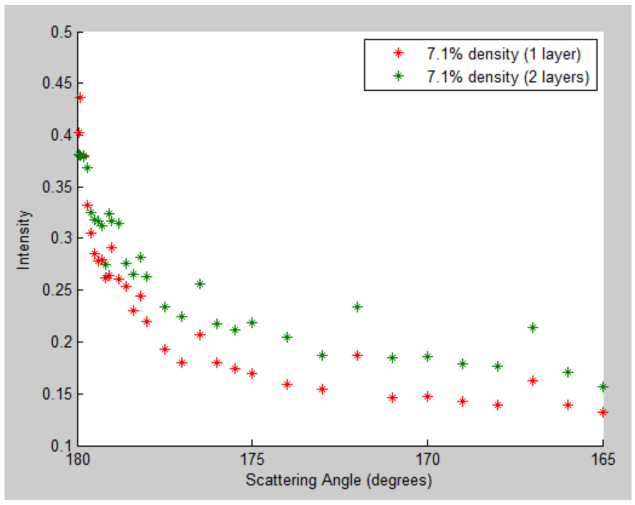


Fig.5.12(b). Increasing layers reduced peak intensity, increased half width and overall intensity.

5.1.4 Increasing Layers

Increasing layers, corresponding to an increase in the depth of the snow layer does appear to lead to a broadening of the central peak and an overal increase in the background intensity as expected. There is also a decrease in the peak intensity corresponding to direct backscatter which as with the difference between the 1.81% and 7.1% can be attributed to a decrease in single particle scattering and an overall increase in the amount of ray-surface interactions and therefore increasing random phase distribution of the the rays.

5.1.5 12 Ray-surface Interactions

The RTDF model terminated prior to completing the ray trace for 12 interactions for this layer. No further data is available.

5.1.6 Without External Reflection

The work carried out by Kaasalainen reduced noise by taking shots where speckle was low. A major contribution to speckle is surface reflection from individual snow crystals. As contributions to the phase function of the sample by reflection from individual surface facets is inversely proportional to the sample size, it is probable that in the models generated, external reflection will be high. External reflection contribution was eliminated from the phase function for the 1.81% and 7.1% density models and the resulting phase function was compared with those including external reflection.

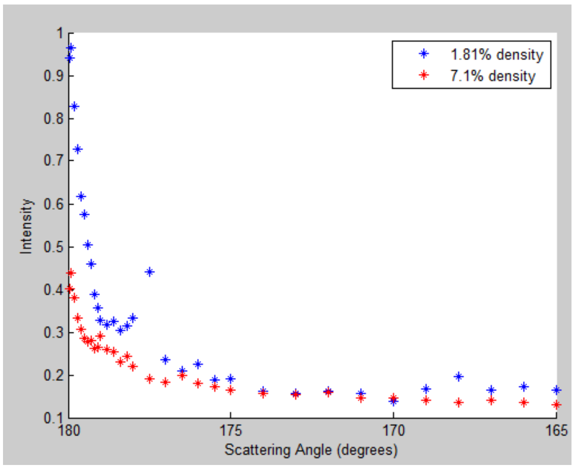


Fig.5.13. No external reflection contribution reduces peaks and produces smoother phase functions

While the scatter plots looks superficially the same there is a noticeable drop in the 178^o peak for 1.81% density and the overall phase function is much smoother.

That the peak at 178^0 for the 1.81% density has not been wholly eliminated can be explained by the structure of the crystals used in the model. First of all, the facets likely to contribute significantly to any peak outside of direct backscatter angle is one that is close to perpendicular to the incoming rays.

The symmetrical structure of the crystals used in the generation of the model however will mean that even though external reflection is not contributing, first order refraction needs to be accounted for. In these cases there will be an identical facet parallel to the surface facet which will reflect the first order refracted ray back to the surface facet. In many cases the exiting rays will traverse parallel and nearly paths to the externally reflected ray as demonstrated in the diagram.

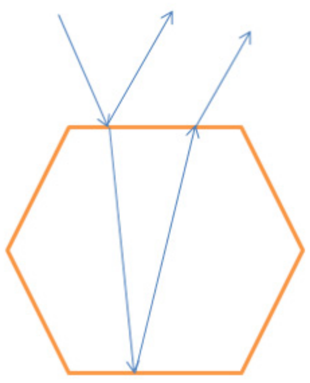


Fig.5.14. Exiting first order refraction rays may follow similar paths to external reflection

5.1.7 Forward Scatter

Analysis of the forward scatter pattern produced by the layer reveals the possibility of a partial 22^{o} halo. It may be an artefact due to the relatively low density and small sample size, i.e. there is a lack of fully optimised random orientation distribution of the crystals. This will only be confirmed when either sample sizes can be increased without hitting the computational time limit or the density can be increased, thereby increasing the range of orientations.

The upper image also has four bright spots. These were eliminated (middle image) by changing the ray diameter. Ray diameter is not something touched on by this thesis though essentially the bright spots are a result of the large ray-diameter/wavelength ratio. Adjusting the ray-diameter proved sufficient at identifying them as artefacts.

Comments on it have been included here purely for completeness as forward scatter was not the focus of this thesis. Obtaining backscatter data under laboratory conditions (as opposed to observing the effect on celestial bodies) is hampered by the source and collector needing to be in effectively the same location. Further, forward scatter is of little use for astronomical studies of celestial bodies except possibly where observing light passing through nebulas.

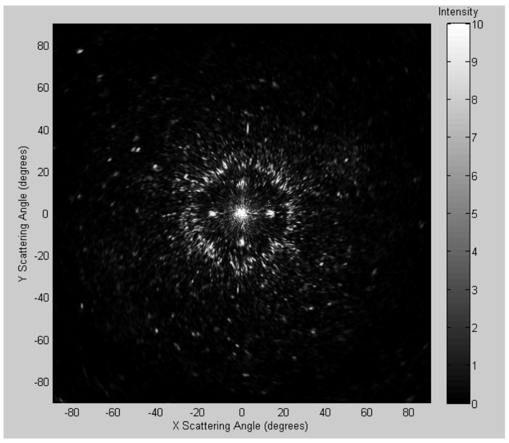


Fig.5.15(a). The four peaks close to 14 degree scattering angle are artefacts. Halving the ray diameter eliminates them.

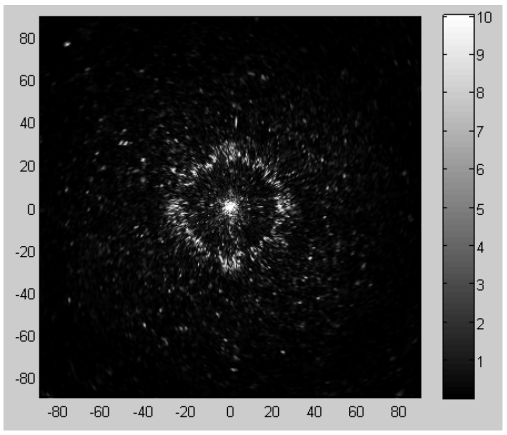


Fig.5.15(b). The presence of the 22 degree halo however indicates that randomisation of the particle orientation is sufficient.

5.2 Plates

Layer Data
Hexagonal crystals
Layer (2x2x2) mm^3
Rejections 5000
Crystals Seeded 305
Crystal length (50±10)µm
Crystal radius (300±150)µm
Max indent 0%
Total Crystal Volume 0.39974 mm^3 (5%)
Crystal density 0.91670g/ml
Layer Density 0.04581g/ml
Size Parameter 3000

Hexagonal plates achieved higher densities than needles simply on account of their dimensions. The primary issue however was that the layer was easily dominated by a single crystal (and its duplicates where it crossed a layer boundary). Due to the high rejection parameter the vast majority of the particles were very small, filling in the gaps between the bounding spheres of the earlier seeds.

A consequence of this is the large spikes in the scatter curve especially for one interaction (probably reflection from the large facets). Once the quantity of interactions increase however the dominance of the spikes is to some degree reduced allowing for slightly improved interpretation of the data.

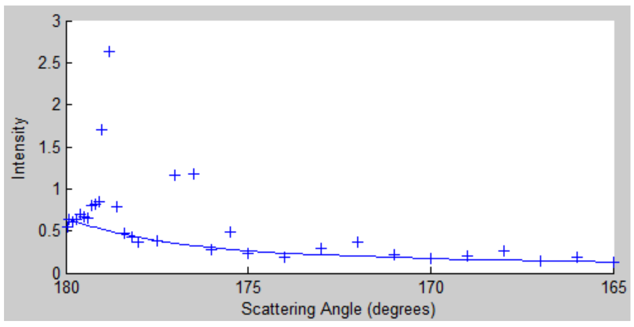


Fig.5.16. Scatter plot for 10 ray-surface interactions and 1 layer along with best fit curve

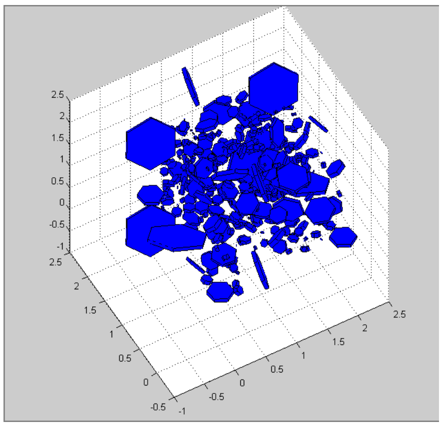


Fig.5.17. The layer is dominated by a few large plates. Further, the high rejection level meant that there were a lot of small crystals, skewing the average crystal away from the stated layer parameter.

Analysis is carried on the 10 ray-surface interactions data for 1 layer. A best fit curve is applied to this in order to produce Half-width-half-maximum and *I(180^o)/I(175^o)* which are then compared to those recorded by Kaasalainen.

Layer	*I(180^o)/I(175^o)*	HWHM
Hexagonal Plates	2.46	1.8
Kaasalainen	1.41	0.6

Considerable scepticism regarding the best fit curve has to be included in the analysis. The large peaks may well be obscuring real trends and as such the HWHM may be much smaller. A much larger layer surface is required in order to increase random orientation distribution.

5.3 Compact Hexagonal Column

Layer Data
Hexagonal Columns with end facets
Layer (1x1x1)mm^3
Rejections 100
Crystals Seeded 143
Crystal length (100±15)µm
Crystal radius (50±7)µm
Max indent 10%
Total Crystal Volume 0.08782 mm^3 (8.8%)
Crystal density 0.91670g/ml
Layer Density 0.08050g/ml
Size Parameter 492

Hexagonal columns have a height roughly equal to their widest diameter, in this case 0.1mm by 0.1mm. They therefore have dimensions most suited to bounding spheres for the purpose of seeding, leading to the highest densities out of all the layers where the particle is created on the fly. This is the only layer that achieved densities comparable with the samples studied by Kaasalainen.

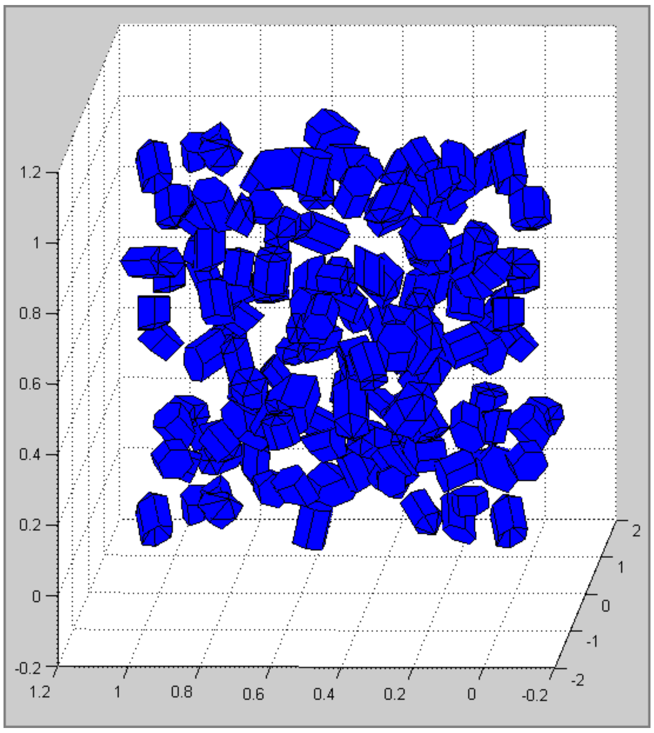


Fig.5.18.

Further, this layer included hexagonal crystals with indents and oblique end facets. The scatter curve for this is considered for 12 ray interactions and 2 layers. The difficulty here is that due to the amount of noise caused by incomplete randomisation of particle orientations, determining the best fit curve is not obvious *(an attempt though later rejected is included in the appendices)*. Only a relatively smooth line could be generated and as such no obvious peak could be determined.

A closer examination of the scatter curve, using a larger scale is conducted in order to qualitatively compare the scatter plot with the findings by Kaasalainen.

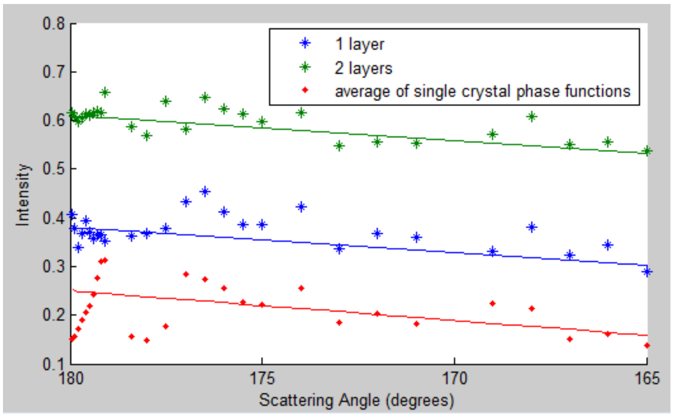


Fig.5.19(a).

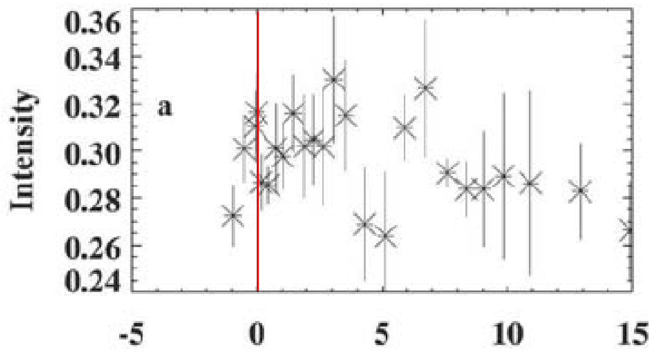


Fig.5.19(b). Increasing both layers and interactions failed to produce curves better suited to assigning best fit curves.

Layer	*I(180°)/I(175°)*	HWHM
1 layer	1.061	-
2 layers	1.048	-
Kaasalainen	1.09	0.1

On the larger scale there are similarities in the two scatter plots. This may prove to be nothing more than artefacts on account of small quantity of crystals forming the layer.

As has been previously pointed out there is considerable background noise which is being attributed to the surface reflection from a few crystals.

5.3.1 Modified Size Parameter

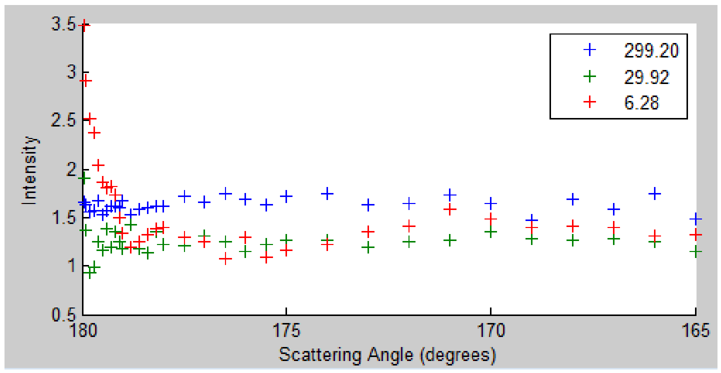


Fig.5.20(a). Scatter plots for differing size parameters

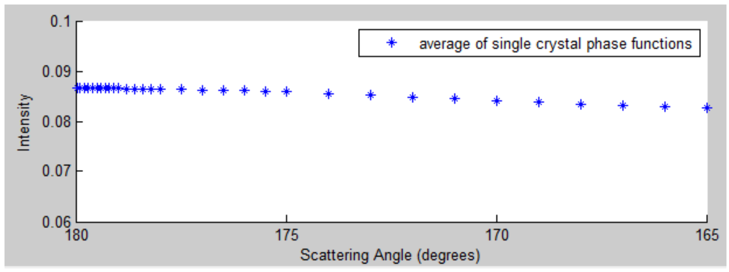


Fig.5.20(b).

Altering the wavelength of the rays while maintaining the same refractive index had the effect of changing the size parameter as the size parameter = *2πa/λ, where λ* is the wavelength and *a* is the radius of the particle. The scatter plot of phase curves is for different wavelengths equating to the size factors indicated.

The scatter plot for the average of single crystal phase function does not show a backscatter peak. The distinct peak for the size parameter 6.28 must be attributed to coherent backscatter. The results are line with Kuga that predicts a sharp backscatter peak due to Type I scattering where the size parameter is between 1 and 20 and layer density greater than 2.5% (this layer has a density of 8.8%).

Layer Size Parameter	*I(180°)/I(175°)*	HWHM
6.28	1.84	0.29°

It is also noted that the backscatter intensity is greater than for the improved seeding technique (fig. 5.7(a)). This is presumable attributed to the increased density of the layer.

5.3.1.2 Size parameter 20 for compact columns

Layer Data
Hexagonal Columns with end facets
Layer (1x1x1)mm^3
Rejections 100
Crystals Seeded 143
Crystal length (100±15)µm
Crystal radius (50±7)µm
Max indent 10%
Total Crystal Volume 0.08782 mm^3 (8.8%)
Crystal density 0.91670g/ml
Layer Density 0.08050g/ml
Size Parameter 20

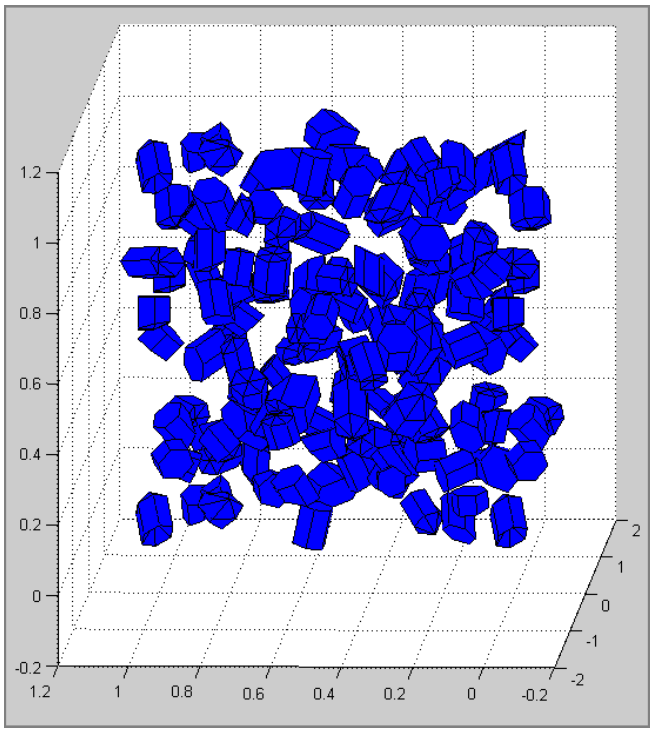


Fig.5.21.

In order for two partial waves traversing time-reversed (conjugate) paths to add constructively, their phase difference must be small. This coherence condition may be stated as ΔΦ/2π «1 where ΔΦ is the phase difference in the waves. As such there will be a critical angle θ_{crit} below which condition will be satisfied and phase coherence maintained. This critical angle is

$$\theta_{crit} \approx \frac{\lambda}{\sqrt{2\ell s}}$$

Where λ is the wavelength of the waves, *s* is the total scattering path length (which includes path inside the particle and therefore particle size) and ℓ is the transport mean-free-path length (average distance between the surfaces). Increasing density and therefore decreasing the transport mean-free-path length will increase θ_{crit} which was one of the primary objectives of modelling layers. The alternative however is to increase the wavelength.

A wavelength was 15.7µm was used in the RTDF program. As there is no change to the refractive index of the crystals, this simply equates to changing the size parameter of the crystals in the layer as size parameter = *2πa/λ, where λ* is the wavelength and *a* is the radius of the particle.

Therefore modifying the size parameter will effectively increase θ_{crit} and therefore increase the scattering angle at which coherent backscatter occurs.

This is not purely for investigative purposes as other factors play a role in the modelling as the photographs show ice crystals with very small components which could be translated into smaller 'local size parameter'. As backscatter peak is undoubtedly related to particle size and therefore the contributions by these small particles is worth investigating.

As coherent backscatter cannot exceed a factor of 2 (due to it being based on the square of the sum of the conjugate waves) with respect to background intensity, any enhancement above this cannot be attributed to coherent backscatter and must be attributed to noise from sources such as dominating surface reflection. This highlights the need for a large sample area so as to limit the contribution from any one reflecting surface. It also means that the peak is proportional to the overall background intensity. Primary interest is in observing the shape of the phase function and comparing it to that of a single crystal as there will be negligible coherent backscatter from a single crystal though there may be a strong reflection and incoherent backscatter (which is the sum of the square of the conjugate waves).

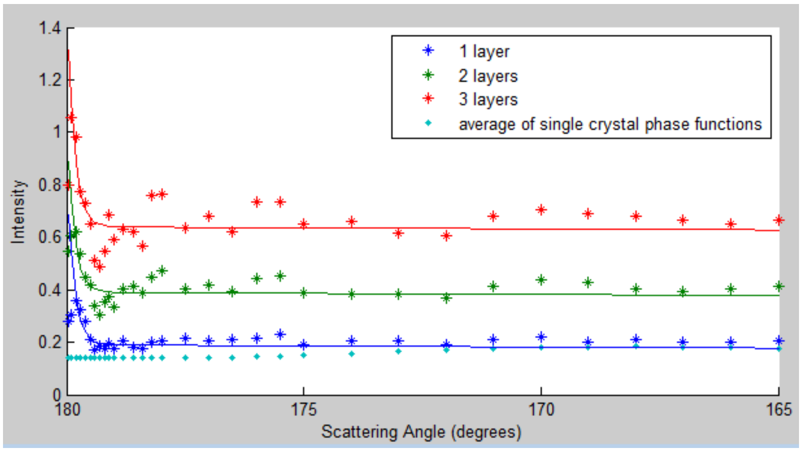


Fig.5.22. Scatter plot of size parameter 20 compact columns

The scatter plot graph is for multiple layers. Data for more than one layer was achieved by re-entering any rays that exited the first layer at the bottom back into the top of the layer. As random-walk means that the more times this occurs, the greater the proportion of any rays entering the layer will leave back through the surface (where the medium is perfectly transparent, i.e. non-absorbing), increasing layers equates to increased background intensity. In all three layer cases there is a backscatter peak which is not present in the case of the average of the single crystal phase function. As these peak intensities do not surpass a factor of 2 and are dominate in direct backscatter regions, it is a reasonable assumption to attribute them to coherent backscatter.

Layer Size Parameter	*I(180^o)/I(175^o)*	HWHM
20	2.5	0.15^o

It is therefore reasonable to state that the model layer is capable of generating coherent backscatter at low densities, in line with Kuga& Ishimaru (1988).

5.4 Rounded Columns

Layer Data
Rounded hexagonal crystals
Layer (0.5x0.5x0.2)mm^3
Rejections 500
Crystals Seeded 24
Crystal length 100µm
Crystal radius 50µm
Max indent 0%
Total Crystal Volume 0.01257 mm^3 (25%)
Crystal density 0.91670g/ml
Layer Density 0.23039g/ml
Size Parameter 492

As indicated earlier, modelling of this layer required the generation of a particle to be seeded into the bounding spheres (after randomisation of alignment). The relatively large volume of the rounded columns compared with their bounding sphere generated high densities comparable with Kaasalainen samples for rounded grains.

As this was a very small sample, in terms of only having 24 crystals, the large facets such as those indicated in the above diagram along with incomplete randomised orientation contributed to noise despite the relatively high quantity of facets.

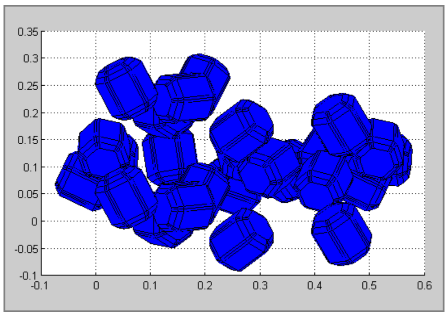


Fig.5.23(a).

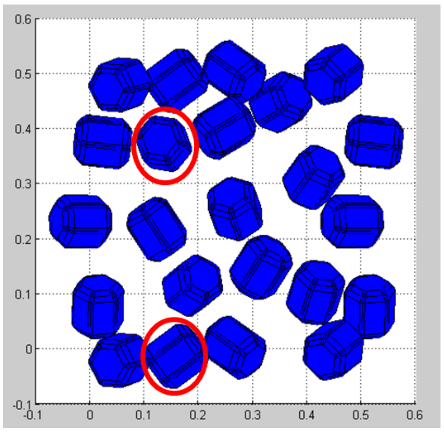


Fig.5.23(b). Probable facets responsible for scattering peak circled in red.

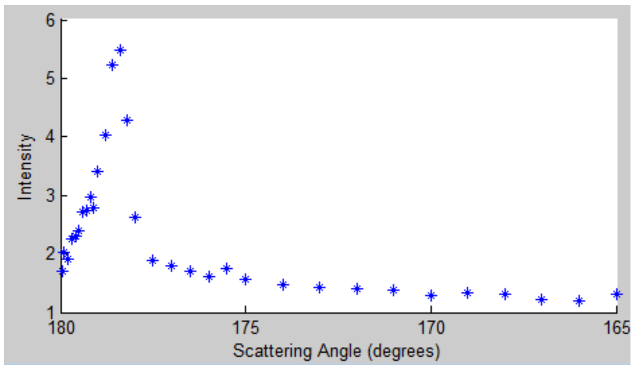


Fig.5.24(a). Strong external reflections make determining best fit phase function impossible

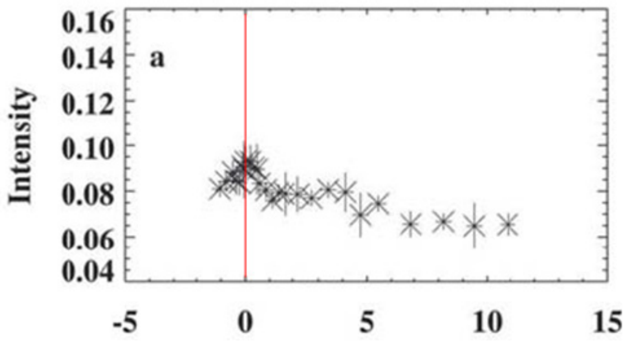


Fig.5.24(b). Refrozen (rounded) surface grains

The spike in the phase function is likely due to the circled facets and prevented any useful analysis.

Kaasalainen's samples included surface grains that had melted and been refrozen producing rounding effects, grains that had been rounded through natural erosion (day old) and grains taken from deep into the snow layer, again having become rounded. The layer data was therefore compared to a few samples.

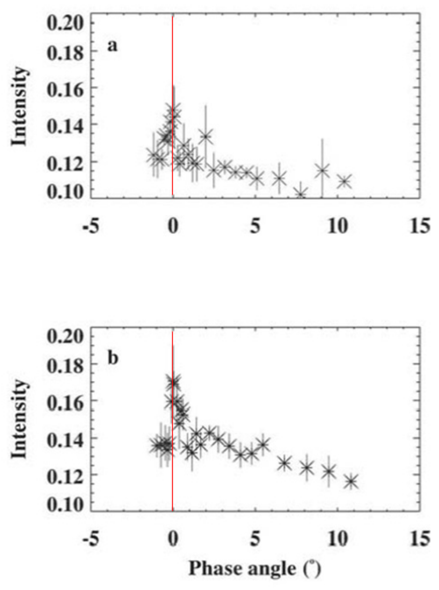


Fig.5.25. Compressing and refreezing grains increased the backscatter peak

The findings of Kaasalainen et al. (2006) indicate that where there rounded grains such as occurs upon melting and refreezing, the curves are much smoother, compared with fresh course snow. They also found that the rounded grains also produced much stronger backscatter peak.

While Kaasalainen discovered that compressing the snow, thereby increasing packing density with a shovel had little effect on the curve other than to increase overall brightness. It did not significantly alter the shape of the curve.

This said, the graphs are those of a density of 0.08g/mL and 0.10g/mL following compression.

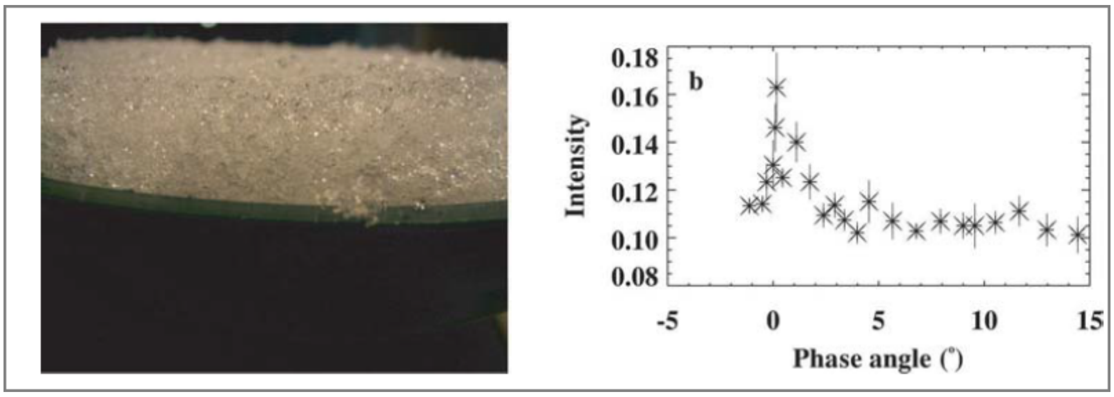
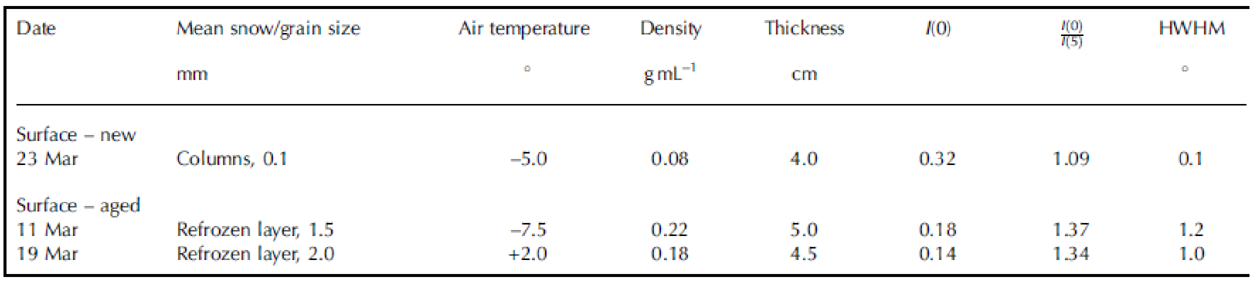


Fig.5.26. Refrozen layer 19th March



Date	Mean snow/grain size mm	Air temperature °	Density g mL^{-1}	Thickness cm	$I(0)$	$\frac{I(0)}{I(5)}$	HWHM °
Surface – new							
23 Mar	Columns, 0.1	−5.0	0.08	4.0	0.32	1.09	0.1
Surface – aged							
11 Mar	Refrozen layer, 1.5	−7.5	0.22	5.0	0.18	1.37	1.2
19 Mar	Refrozen layer, 2.0	+2.0	0.18	4.5	0.14	1.34	1.0

5.4.1 Increased Rounding

Layer Data
Rounded hexagonal crystals
Layer $(1x1x2)mm^3$
Rejections 500
Crystals Seeded 445
Crystal length 100µm
Crystal radius 50µm
Max indent 0%
Total Crystal Volume 0.27410 mm^3 (13.7%)
Crystal density 0.91670g/ml
Layer Density 0.12564g/ml
Size Parameter 492

In an attempt to decrease the contribution by large surface facets, the degree of rounding was increased. This was justified by equating the crystals to ice crystals that has slightly melted before refreezing, thereby causing them to lose their sharp edges. While this process also undoubtedly led to sintering and amalgamation of the crystals, it is possible that sufficient quantities of crystals existed as individuals and were as a consequence not too dissimilar to rounded hexagonal columns.

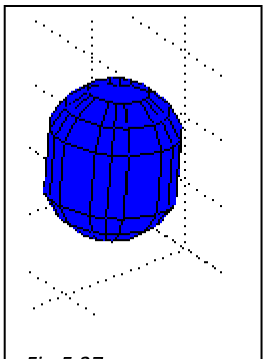


Fig.5.27.

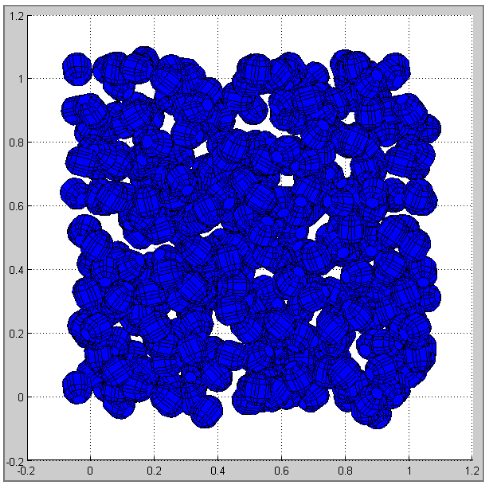


Fig.5.28(a).

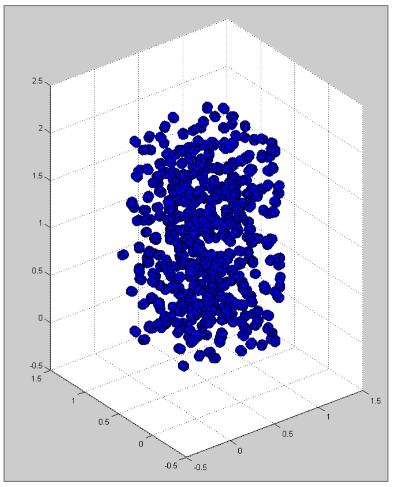


Fig.5.28(b).

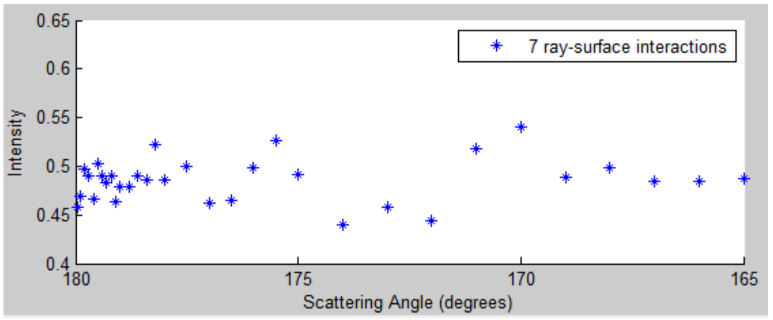


Fig.5.29(a). 7 ray surface interactions for 1 layer compared with rounded refrozen grains.

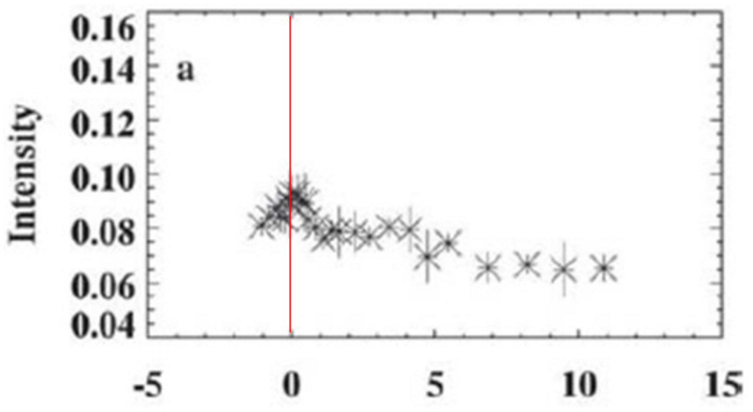


Fig.5.29(b).

The depth of the layer was increased in order to ensure that virtually all rays entering the layer would interact with a crystal rather than pass through the bottom of the layer (an issue with stacking layers is that rays that miss all particles in the first layer will also miss all particles in all layers). As it turned out using multiple layers was not possible with this sample on account of processing time limits. As such the data collected was for multiple ray interactions for a single layer.

Layer	Size mm	Density g/ml	*I(180°)/I(175°)*	HWHM
7 ray-surface Highly Rounded Column	0.1	0.23	-	-
Surface - refrozen	0.1-0.4	0.12	1.3	0.6
Melting > Refrozen	0.5-1.0	0.15	1.4	0.2
Deep Rounded	0.5-1.5	0.32	1.35	0.2

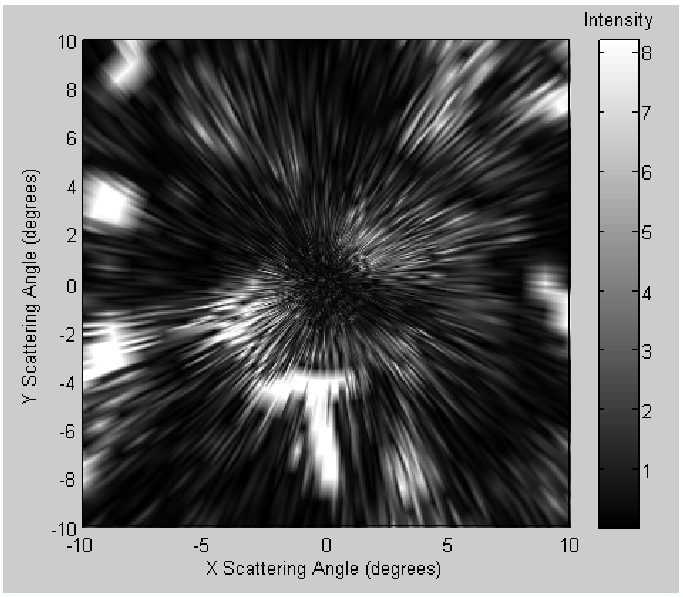


Fig.5.30(a). External reflection

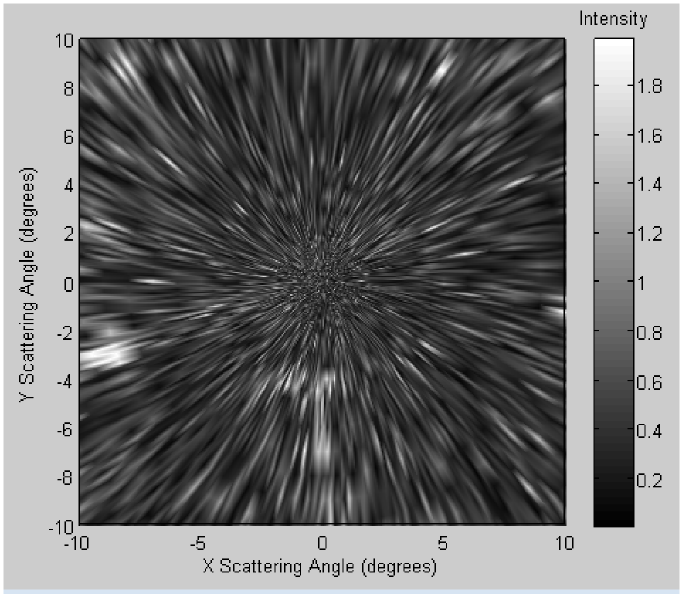


Fig.5.30(b). 7 ray surface interaction

The 2-D scatter plots however give little supporting evidence to the presence of the backscatter peak.

6.0 Conclusions and Discussion

From a qualitative stance, the research has been successful. The phase functions generated by some of the layers reveal strong similarities to those produced by the field work Kaasalainen. The compact hexagonal layer is very interesting as the backscatter peak is clearly related to the size parameter and is therefore in agreement with Kuga (1988). This would indicate that for the size parameter of6 the backscatter is coherent as by 492 it has effectively disappeared.

Another test that may prove interesting is to investigate the focusing effect using the highly rounded crystals (5.4.1). The density of the layer is 13.7% may generate Type II backscatter. This will require decreasing the wavelength in order to achieve a size factor of 33,000 (Kuga 1988). Of course even the highly rounded crystals are not spherical and as such the intensity increase is expected to be lower than that produced by spheres, presuming any enhancement is created at all.

Overall there is however a long way to go to achieving the goal of a catalogue of phase functions corresponding to a range of both crystal shapes and densities to the point of being able to accurately predict the structure of regolith from remote observation.

6.1 Bounding Volumes

A major obstacle to generating layers with density profiles consistent with the observations of Kaasalainen has been the use of bounding spheres as a means of seeding crystals. While the method is reasonable where the semi-major and semi-minor axis of a crystal are similar such as in the case of the compact column, where they diverge significantly such as in the case of needles and hexagonal plates, the resulting empty volume within the bounding sphere severely reduces the potential density of the resulting layer. Increasing the quantity of rejections to very high numbers proved not to be a solution on account of reducing the average size of the crystal as the quantity of crystals seeded increased. This was a consequence of the decreasing volume between the bounding spheres decreased.

The solution using a more exact treatment of the volume of the seeded crystal through the use of convex hulls was only completed towards the end of the research when it became apparent that previous methods to increase layer density proved inadequate. The seeding process replaced the bounding sphere with a convex hull formed from Delaunay triangulations and determined if any of the lines forming the frame of the seeding crystal passed through the convex hull.

The 30% increase in crystals seeded for the $h/2a \approx 3$ layer through the use of convex hull testing was a modest improvement though limited due to the relatively low aspect ratio of the crystals. This meant that where the spheres overlapped there was a high degree of probability that the crystals would also overlap.

Seeding improvement through convex hulls is related to the aspect ratio. The smaller the volume within a bounding sphere occupied by the crystal, the greater the chance that two crystals in overlapping bounding spheres will not share the same volume. It is therefore reasonable to assume that the convex hull intersection test will give better results with increasing aspect ratios. The issue however is that as bounding sphere density increases, the average quantity of potential overlaps also increases resulting in more convex hull intersection tests.

In practise the issue however with this approach was simply that it had a massive computing overhead and where it was tried for high aspect ratio crystals for even modest sized layers simply failed to generate a layer after running for days due to being terminated by the time limit.

The approach to speed up the process set a maximum quantity of overlaps between the seeding bounding sphere and previously seeded bounding spheres. The reason was that where there were numerous overlaps, it was likely that the convex hull test would confirm overlap with at least one previously seeded crystal. This compromise ultimately limited the density of the layer.

Further, crystals such as the hexagonal plates required a second test as the first test only checked for the framework of the seeding crystal against all the convex hulls of the overlapping spheres. For hexagonal plates, the convex hull of the crystal being seeded also needed to be checked against all the lines forming the framework of the previously seeded and potentially overlapping plates.

As this is clearly the way forward for further research, methods will have to be implemented to optimise this code. The first step would be to look at how data is generated, stored and used while processing. As code is the product of development steps, data is generated that was needed in earlier versions and may now prove redundant though has not yet been pruned. It is also certain that some methods of manipulating the data are not optimised. These include adding to arrays while running checks on the arrays or processing the data of an entire array rather than narrowing down the data that is selected prior to manipulating it. In the case of the line array (the matrix containing the functions for the lines forming the framework of the crystals), there is a selection process which has to loop through the array twice each time there is a convex hull check. Replacing this with a more sophisticated look-up function will speed up the process as each new hexagonal crystal (without indented/filled basal facets) added to the layer effectively increases this array by 24 entries.

Another optimisation is to determine if there are methods of avoiding the convex hull test without significantly undermining the seeding processes. One that has recently been considered is rejection based on the separation between the seeding sphere and those it overlapped with. As the crystals within the bounding sphere pass through the centre of the sphere, two spheres whose origins are within the average radius of the crystal will generally have overlapping crystals. As such, where the separation is less than the sum of the crystal radius (columns) or half-lengths (hexagonal plates), the seeding bounding sphere can be rejected without testing for convex hull intersection. Such a test would speed up the process significantly for crystals with low aspect ratios such as compact columns.

Primary candidate for scrutiny however lies with the convex hull test and a means of either replacing it with a cross product test (looking at each of the facets individually) or determining a means of speeding up the process. Up until the convex hull test, even very large layers were generally generated in a few minutes. After this function, even small layers could easily take many hours.

The convex hull test can take anything up to 3 seconds per test. A hexagonal column has 24 lines forming its framework and each of these were tested against anything up to 20 different convex hulls as this was the limit placed on the maximum quantity of bounding spheres to test against overlap. It can therefore be seen that just these rejections alone added up to 92 hours to the run time of the code. It is actually significantly less than this for two reasons. First it is unlikely that the there were 20 overlaps in each case. Second, some degree of optimisation was included insofar as the function was run using a matrix that included all the lines forming the framework. This meant that each test took significantly less than 72(=3x24)seconds.

As the final density was around double the density of samples without the function, around 100 crystals were also subjected to the test and passed. This is anything up to 40 hours though again, significantly less for the reasons described above.

Despite this, it was effectively found that generating even this relatively small layer took days and layers were often generated over the weekend.

In the case of hexagonal plates, the process requires reciprocation due to the significant chance of a facet of the seeding crystal intercepting a line forming part of the framework of the previously seeded crystal. As such this effectively doubles the time to generate a layer. It is hardly surprising therefore that for anything other than very small layers, the programs terminated prior to producing a result.

6.2 Improvement Options

There is always room for improvement beyond optimizations of the code discussed above.

6.2.1 Pre-allocation of volume

As snow falls such that the ice crystals rest against other crystals, a similar approach could be used as part of the seeding process to ensure that space is filled with crystals. This can be achieved by choosing a vertical position and systematically moving the seeding crystal down in steps until its convex hull intersects a convex hull of a pre-seeded crystal, at which point the downward movement is terminated at the previous step. It is suspected however that this approach will suffer the same computing overheads as the convex hull approach.

6.3 Sintering

Further development can be made using convex hulls to include sintering – the process by which one or more crystals amalgamate into a single crystal, in the case of ice crystals through partially melting and refreezing. The process can theoretically be achieved by replacing a crystal that overlaps with another crystal with the convex hull of the sum of the two crystals. There would have to be some determination of the degree of sintering allowed. Further, rather than outputting the seeded crystals to an emerging array of data for the facets, vertexes and their coordinates the data will need to be outputted only after the crystals have been seeded as the sintering process will cause one or more crystals to be converted into one.

6.3.1 Dendrites

When hexagonal plates grow, they have a steadily increasing chance of growing dendrites. As these become more complex they eventually achieve the common conception of a six sided snowflake. A possible development of the seeding process could be to generate structures at the intersection of vertexes as seen in the below photo (Ono, 1969).

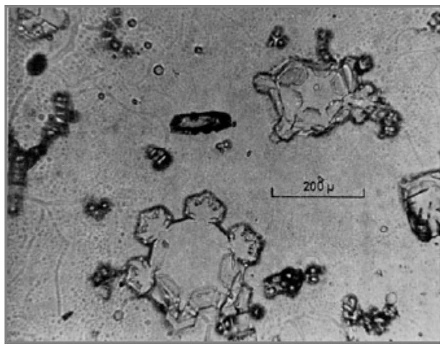


Fig.6.1. Dendritic growths leading towards classical snowflakes.

The sintering approach could be used as a standalone function to create crystal file of commonly occurring shapes such as hexagonal plates with smaller dendritic hexagonal plates at the corners. The growths could be grown on any point and at any size, though for this approach to be useful will require studies of snow samples to determine the proportion hexagon plates having dendritic growths and the relative proportions of the growths compared to the plate for snow samples.

A less complex approach is to treat the surfaces of the crystals as rough rather than perfectly smooth facets. This changes the angle of incidence between the ray and the surface, thereby changing the angles of reflection and refraction. This will result in a greater degree of scattering and may compensate for relatively small layer samples.

6.4 Range of Crystals

This initial development uses simple relatively homogenous crystals. It is a relatively simple matter to extend this to seeding layers with a range of crystal types. Snow for example may consist of a mixture of sheaths (hexagonal columns with indented basal facets) and plates. Regolith consisting of dust and ice are likely to contain cuboid sand grains along with ice crystals. In these cases however the extra data such as the refractive index and absorption coefficient of individual grains will have to be included in the layer data.

6.5 More Detailed Characterization of Snow Data

The data provided by Kaasalainen does not provide sufficient data regarding the limits of the snow crystals present in the samples. Only the average size parameter is indicated. As the samples were characterised by a common crystal shape, it is assumed that the average size applied to said crystal. Along with the pristine crystals will be a proportion of broken crystals and irregular fragments. It is unknown as to what degree these exist within the sample. It is however noted that compressing (and therefore presumably further damaging the ice crystals) did not significantly affect the phase function of the snow sample. This is to some degree encouraging as it indicates that while the crystal shapes are identifiable they by and large dictate the phase function of the layer. This will have to be confirmed through the creation of layers with differing levels of arbitrarily shaped particles alongside the common particles forming the layer.

6.6 Summary

The thesis explored a method of simulating light scattering by layers consisting of non-spherical particles. The particles were chosen to simulate pristine ice crystals such as occurring in relatively fresh snow. Four types of crystals were investigated:

- Hexagonal columns, aspect ratios 10:1 and 3:1
- Compact hexagonal columns, aspect ratio 1:1
- Hexagonal plates, aspect ratio 1:12
- Rounded columns, aspect ratios 1:1

The primary programming aim was to generate a layer such that it could be used to simulate an infinite plain of regolith. It achieved this by seeding crystals into a layer volume through a cyclically closed loop such that when light rays left the layer along the sides of the layer volume they could be reinserted back into the layer, though offset by the appropriate layer dimension.

Once the layer generating program was functional, the objective was to determine if the layer would generate a backscatter surge identified with coherent backscatter as a consequence of light undergoing constructive interference as they passed along conjugate ray paths.

Due to the limits of bounding spheres and computational overhead incurred through the use of convex hull interception tests, consideration of density was included.

A second objective was to determine if the qualitative phase function of the particulate layers agreed with the field studies of Kaasalainen.

The layers were tested using a ray tracing program which included diffraction on facets. Parameters altered or identified with data collected for the investigations included:

- Wavelength
- Quantity of Layers
- Ray-surface Interactions
- Ray diameter

6.61 The results show a backscattering surge for:

- Small particle size parameter and layer volume density 8.78% (5.3.1) and 1.81% (5.1.1). The higher surge was observed for the larger volume density. This is in agreement with Kuga and Ishimaru.
- Higher size parameter and relatively low layer density (1.81%, 5.1.1). This agrees with Kuga and Ishimaru's observation that tenuous media the phase function should have similarity to single particle phase function.
- Qualitative agreement with measurements at snow layers (Kaasalainen et al.) for fresh snow consisting of hexagonal columns (5.1.1 & 5.3).

6.7 Further work:

- Optimisation of the convex hull intersection code to increase layer density.
- Test model against measurements taken from well characterised samples.
- Investigate particle roughness, sintering and inclusion of fragments.

6.8 Methods of Improving Sampling Data

A major part of the project was achieving layer samples with densities consistent with the samples collected by Kaasalainen and achieving complete randomisation through large surface areas. This process however appears to have surpassed the actual sampling methods employed by Kaasalainen. Hostile sampling conditions such as weather combined with both the delicate nature of the snow crystals and the small area scanned out by the probe meant that data was very noisy. Further, the samples were for a relatively small number of crystals and that the samples were not of pristine crystals or even of a single type. It would have aided the modelling process if Kaasalainen had used specific classifications for the samples collected such as those used by Magono-Lee along with enlarged sections from the photos.

The next step in sampling therefore should include a method by which an area can be examined in order to generate 2D plots. This could be achieved by taking shots of a revolving sample in order to generate time-resolved data. This may be sufficient to indentify features within the plot that cannot be observed within a one dimensional phase function. It is possible that there is data unique to a crystal shape embedded in the data that is not necessarily the product of dominating surface reflection such as in the case when a small sample (and therefore incomplete randomisation) is examined.

Output from the RTDF program for needle grains of dimension ratio 3:1 produced two-dimensional scattering functions. These revealed the presence of bright and dark patches at specific polar coordinates. When treated as one dimensional scattering function these would equate to angles with larger error regions.

It is the estimation of Sanna Kaasalainen that achieving field data to obtain 2D plots is not likely to be forthcoming in the near future without necessary advances in technology.

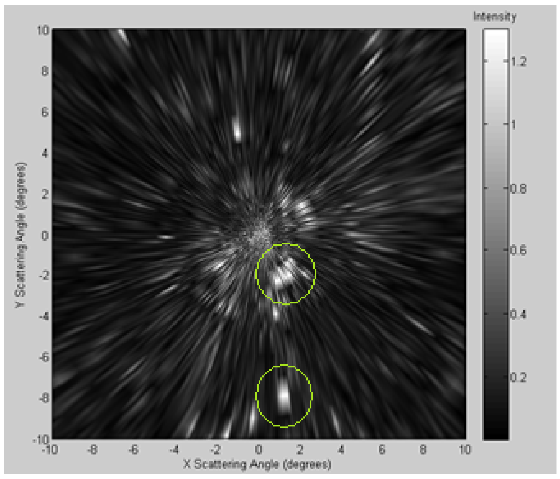


Fig.6.2. Polar Plot of Intensity against scattering angle for Needle grains L0.3 x R0.05

7.0 Acknowledgements

A special thanks to Sanna Kaasalainen for supplying data and images and also explaining the specifics of data collection. This corrected a number of assumptions including a continuously rotating sample and near continuous recording of intensity. When working with virtual samples it is easy to overlook the sheer difficulty in collecting pristine snow samples and subjecting them to scrutiny in often less than perfect conditions.
Also to Evelyn Hesse for going above and beyond the call of mentoring, often proofreading work at nights and weekends on account of looming deadlines.

References

Baran, A. J., Labonnote, L.-C. A self-consistent scattering model for cirrus. I: The solar region. Quarterly Journal of the Royal Meteorological Society, 133(2007)1899-1912.

Braunbeck, W. and Laukien, G. Optik, 9, 174 (1952).

Brown, R.H., Baines, K.H., Bellucci, G., et al. Observations with the Visual and Infrared Mapping Spectrometer (VIMS) during Cassini's flyby of Jupiter. Icarus, 164(2003)461-470.

Corey, R., Kissner, M., et al. Coherent backscattering of light, http://patarnott.com/pdf/cbsweb.pdf

Clarke, A.J.M., Hesse, E., Ulanowski, U. and Kaye, P.H. A 3D implementation of ray tracing combined with diffraction on facets: Verification and a potential application. Journal of Quantitative Spectroscopy & Radiative Transfer, 100(2006)103-114.

Hapke, B. The Opposition Effect of the Moon Coherent Backscatter and Shadow Hiding. Icarus 133(1998)89-97.

Helfenstein, P., Currier, N., Clark, B. E., et al. Galileo Observations of Europa's Opposition Effect. Icarus, 135(1998)41-63.

Hesse, E., Ulanowski, Z. Scattering from long prism computed using ray tracing combined with diffraction on facets. Journal of Quantitative Spectroscopy & Radiative Transfer, 79-80(2003)721-732.

Hesse, E. Modelling diffraction during ray-tracing using the concept of energy flow lines. Journal of Quantitative Spectroscopy & Radiative Transfer, 109(2008)1374-1383.

Hesse, E., Mc Call, D.S., Ulanowski, Z., Stopford, C., Kaye, P.H. Application of RTDF to particles with curved surfaces. Journal of Quantitative Spectroscopy & Radiative Transfer, 110(2009)1599-1603.

Hesse, E., Macke, A., Havemann, S., Baran, A.J., Ulanowski, Z., Kaye, P.H. Modelling diffraction by facetted particles. Journal of Quantitative Spectroscopy & Radiative Transfer, 113(2012)342-347.

Kaasalainen, S., et al. Optical properties of snow in backscatter. Journal of Glaciology, 52(2006)179.

Kaasalainen, S., et al. Small-angle goniometry for background measurements in the broadband spectrum. Applied Optics, 44(2005)1485.

Kuga, Y., Ishimaru, A., Backscattering by Randomly Distributed Particles of Different Sizes. SPIE, 927 (1988) 33-37.

Kuga, Y., Ishimaru, A., Backscattering enhancement by randomly distributed very large particles. Applied Optics, 28(1989) 2165-2169.

Mishchenko, M., et al. Direct solutions of the Maxwell equations explain opposition phenomena observed for high-albedo solar system objects. The Astrophysical Journal, 705(2009)L118-L122.

Mishchenko, M. Overview of scattering by nonspherical particles. In: Light Scattering by Nonspherical Particles: Theory, Measurement and Applications. Academic Press, San Diego. (2000)30-48.

Ono, A. The shape and riming properties of ice crystals in natural clouds. Journal of the Atmospheric Sciences, 26 (1968) 138-147.

Piironen, J., Muinonen, K., Keranen, S., Karttunen, H., Peltoniemi, J. Backscattering of Light by Snow. Observing Land from Space (2000), 219-228.

Prosser, R.D. The interpretation of diffraction and interference in terms of energy flow. International Journal of Theoretical Physics, 15(1976)169-80.

Shkuratov, Y., Grynko, Y. Light scattering by media composed of semitransparent particles of different shapes in ray optics approximation: consequence for spectroscopy, photometry and polarimetry of planetary regoliths. Icarus 173(2005)16-28.

Showalter, M., Pater, I., Verbanac, G., et al. Properties and dynamics of Jupiter's gossamer rings from Galileo, Voyager. Icarus 195(2008)361-377.

Hubble and Keck images

Simonelli, D., Buratti B., Europa's Opposition in the near-infrared: interpreting disk-integrated observations by Cassini VIMS. Icarus 172(2004)149-162.

Stankevich, D., Shkturatov, Y., Muinonen, K. Shadow-hiding effect in inhomogeneous layered particulate media. Journal of Quantitative Spectroscopy & Radiative Transfer, 63(1999)445-458.

Stankevich, D., Istomina, L., Shkuratov, Y., Videen G. The scattering matrix of random media consisting of large opaque spheres calculated using ray tracing and accounting for coherent backscattering enhancement. Journal of Quantitative Spectroscopy and Radiative Transfer, 106 (2007)509-519.

Ulanowski, Z., Hesse, E., Kaye P., Baran, A. Light scattering by complex ice-analogue crystals. Journal of Quantitative Spectroscopy and Radiative Transfer, 100(2006)382-392.

Appendix

A. Sample line equation matrix

Single hexagonal column crystal

Form – bounding sphere number and equations for the x,y, z coordinates for any given value of t.

```
[ 202,   (37726843968657*t)/4503599627370496 + 2366325988986969/2251799813685248,   (1370224362431*t)/70368744177664 + 6362875491685209/281474976710656,
(109406936949779*t)/2251799813685248 + 6171296937557361/9007199254740992]
[ 202,   (95716183761297*t)/4503599627370496 + 4770378821942595/4503599627370496, (13569969234053*t)/281474976710656 + 6368356389134933/281474976710656,
(13358769786049*t)/2251799813685248 + 6608924685356477/9007199254740992]
[ 202,   (28976617051401*t)/2251799813685248 + 1216523751425973/1125899906842624,  (4041976486425*t)/140737488355328 + 3190963179184493/140737488355328,
6662359764500673/9007199254740992 - (192106412614737*t)/4503599627370496]
[ 202,   2462024119903347/2251799813685248 - (37726843968657*t)/4503599627370496,    1597502577835459/70368744177664 - (1370224362431*t)/70368744177664,
6278146939271199/9007199254740992 - (109406936949779*t)/2251799813685248]
[ 202,   4886321395838037/4503599627370496 - (95716183761297*t)/4503599627370496,    399033088368257/17592186044416 - (6784984617027*t)/140737488355328,
5840519191472083/9007199254740992 - (13358769786049*t)/2251799813685248]
[ 202,   1197651303019185/1125899906842624 - (28976617051401*t)/2251799813685248,  3185479722329029/140737488355328 - (8083952972849*t)/281474976710656,
(192106412614737*t)/4503599627370496 + 5787084112327887/9007199254740992]
[ 202,  (191432367522595*t)/9007199254740992 + 7209048977209083/9007199254740992, (13569969234053*t)/281474976710656 + 3201840128004547/140737488355328,
(13358769786049*t)/2251799813685248 + 5776446697695905/9007199254740992]
[ 202,   (75453687937315*t)/9007199254740992 + 3700240672365839/4503599627370496,   (1370224362431*t)/70368744177664 + 6417250225243147/281474976710656,
(109406936949779*t)/2251799813685248 + 5829881776840101/9007199254740992]
[ 202,   7475935032668993/9007199254740992 - (28976617051401*t)/2251799813685248,  6422731122692871/281474976710656 - (8083952972849*t)/281474976710656,
(192106412614737*t)/4503599627370496 + 6267509524639217/9007199254740992]
[ 202,  7360028564463389/9007199254740992 - (191432367522595*t)/9007199254740992,  3207323584860011/140737488355328 - (6784984617027*t)/140737488355328,
6651722349868691/9007199254740992 - (13358769786049*t)/2251799813685248]
[ 202,   3584298098470397/4503599627370496 - (75453687937315*t)/9007199254740992,     400067325030373/17592186044416 - (1370224362431*t)/70368744177664,
6598287270724495/9007199254740992 - (109406936949779*t)/2251799813685248]
[ 202,   (28976617051401*t)/2251799813685248 + 7093142509003479/9007199254740992,   (4041976486425*t)/140737488355328 + 1598899075759061/70368744177664,
6160659522925379/9007199254740992 - (192106412614737*t)/4503599627370496]
[ 202, 2366325988986969/2251799813685248 - (2372161446944397*t)/9007199254740992, (32720811351035*t)/281474976710656 + 6362875491685209/281474976710656,
6171296937557361/9007199254740992 - (5318707315991*t)/4503599627370496]
[ 202,   (75453687937315*t)/9007199254740992 + 7093142509003479/9007199254740992,    (1370224362431*t)/70368744177664 + 1598899075759061/70368744177664,
(109406936949779*t)/2251799813685248 + 6160659522925379/9007199254740992]
[ 202,  (593040361736099*t)/2251799813685248 + 3584298098470397/4503599627370496,   400067325030373/17592186044416 - (32720811351035*t)/281474976710656,
(5318707315991*t)/4503599627370496 + 6598287270724495/9007199254740992]
[ 202,   4770378821942595/4503599627370496 - (37726843968657*t)/4503599627370496,   6368356389134933/281474976710656 - (1370224362431*t)/70368744177664,
6608924685356477/9007199254740992 - (109406936949779*t)/2251799813685248]
[ 202,  4770378821942595/4503599627370496 - (593040361736099*t)/2251799813685248, (32720811351035*t)/281474976710656 + 6368356389134933/281474976710656,
6608924685356477/9007199254740992 - (5318707315991*t)/4503599627370496]
[ 202,  (191432367522595*t)/9007199254740992 + 3584298098470397/4503599627370496,    (6784984617027*t)/140737488355328 + 400067325030373/17592186044416,
(13358769786049*t)/2251799813685248 + 6598287270724495/9007199254740992]
[ 202, (2372161446944395*t)/9007199254740992 + 7360028564463389/9007199254740992,   3207323584860011/140737488355328 - (8180202837759*t)/70368744177664,
(5318707315991*t)/4503599627370496 + 6651722349868691/9007199254740992]
[ 202,   1216523751425973/1125899906842624 - (95716183761297*t)/4503599627370496, 3190963179184493/140737488355328 - (13569969234053*t)/281474976710656,
6662359764500673/9007199254740992 - (13358769786049*t)/2251799813685248]
[ 202, 1216523751425973/1125899906842624 - (2372161446944395*t)/9007199254740992,   (8180202837759*t)/70368744177664 + 3190963179184493/140737488355328,
6662359764500673/9007199254740992 - (5318707315991*t)/4503599627370496]
[ 202,   (28976617051401*t)/2251799813685248 + 7360028564463389/9007199254740992,  (8083952972849*t)/281474976710656 + 3207323584860011/140737488355328,
6651722349868691/9007199254740992 - (192106412614737*t)/4503599627370496]
[ 202, (2372161446944395*t)/9007199254740992 + 7475935032668993/9007199254740992, 6422731122692871/281474976710656 - (32720811351035*t)/281474976710656,
(5318707315991*t)/4503599627370496 + 6267509524639217/9007199254740992]
[ 202,   2462024119903347/2251799813685248 - (28976617051401*t)/2251799813685248,   1597502577835459/70368744177664 - (4041976486425*t)/140737488355328,
(192106412614737*t)/4503599627370496 + 6278146939271199/9007199254740992]
[ 202, 2462024119903347/2251799813685248 - (2372161446944395*t)/9007199254740992,  (32720811351035*t)/281474976710656 + 1597502577835459/70368744177664,
6278146939271199/9007199254740992 - (5318707315991*t)/4503599627370496]
[ 202,   7475935032668993/9007199254740992 - (75453687937315*t)/9007199254740992,   6422731122692871/281474976710656 - (1370224362431*t)/70368744177664,
6267509524639217/9007199254740992 - (109406936949779*t)/2251799813685248]
[ 202,  (593040361736099*t)/2251799813685248 + 3700240672365839/4503599627370496, 6417250225243147/281474976710656 - (32720811351035*t)/281474976710656,
(5318707315991*t)/4503599627370496 + 5829881776840101/9007199254740992]
[ 202,   (37726843968657*t)/4503599627370496 + 4886321395838037/4503599627370496,     (1370224362431*t)/70368744177664 + 399033088368257/17592186044416,
(109406936949779*t)/2251799813685248 + 5840519191472083/9007199254740992]
[ 202,  4886321395838037/4503599627370496 - (593040361736099*t)/2251799813685248,   (32720811351035*t)/281474976710656 + 399033088368257/17592186044416,
5840519191472083/9007199254740992 - (5318707315991*t)/4503599627370496]
[ 202,  3700240672365839/4503599627370496 - (191432367522595*t)/9007199254740992, 6417250225243147/281474976710656 - (13569969234053*t)/281474976710656,
5829881776840101/9007199254740992 - (13358769786049*t)/2251799813685248]
[ 202, (2372161446944397*t)/9007199254740992 + 7209048977209083/9007199254740992,   3201840128004547/140737488355328 - (8180202837759*t)/70368744177664,
(5318707315991*t)/4503599627370496 + 5776446697695905/9007199254740992]
[ 202,   (95716183761297*t)/4503599627370496 + 1197651303019185/1125899906842624,  (6784984617027*t)/140737488355328 + 3185479722329029/140737488355328,
(13358769786049*t)/2251799813685248 + 5787084112327887/9007199254740992]
[ 202, 1197651303019185/1125899906842624 - (2372161446944397*t)/9007199254740992,   (8180202837759*t)/70368744177664 + 3185479722329029/140737488355328,
5787084112327887/9007199254740992 - (5318707315991*t)/4503599627370496]
[ 202,   7209048977209083/9007199254740992 - (28976617051401*t)/2251799813685248,  3201840128004547/140737488355328 - (4041976486425*t)/140737488355328,
(192106412614737*t)/4503599627370496 + 5776446697695905/9007199254740992]
[ 202, (2372161446944397*t)/9007199254740992 + 7093142509003479/9007199254740992,  1598899075759061/70368744177664 - (32720811351035*t)/281474976710656,
(5318707315991*t)/4503599627370496 + 6160659522925379/9007199254740992]
[ 202,   (28976617051401*t)/2251799813685248 + 2366325988986969/2251799813685248,  (8083952972849*t)/281474976710656 + 6362875491685209/281474976710656,
6171296937557361/9007199254740992 - (192106412614737*t)/4503599627370496]
```

B. Best fit analysis

Takes the form I(α)=a exp(-α/d) + b + kα, where *a, d, k* and *b* are empirical parameters defining the backscatter peak and linear part of the intensity (Piironen, 2000)

Equivalent to:

$$I(\alpha) = I_s \exp(-0.5\alpha / l) + I_b + k\alpha$$

Piironen, (2000)

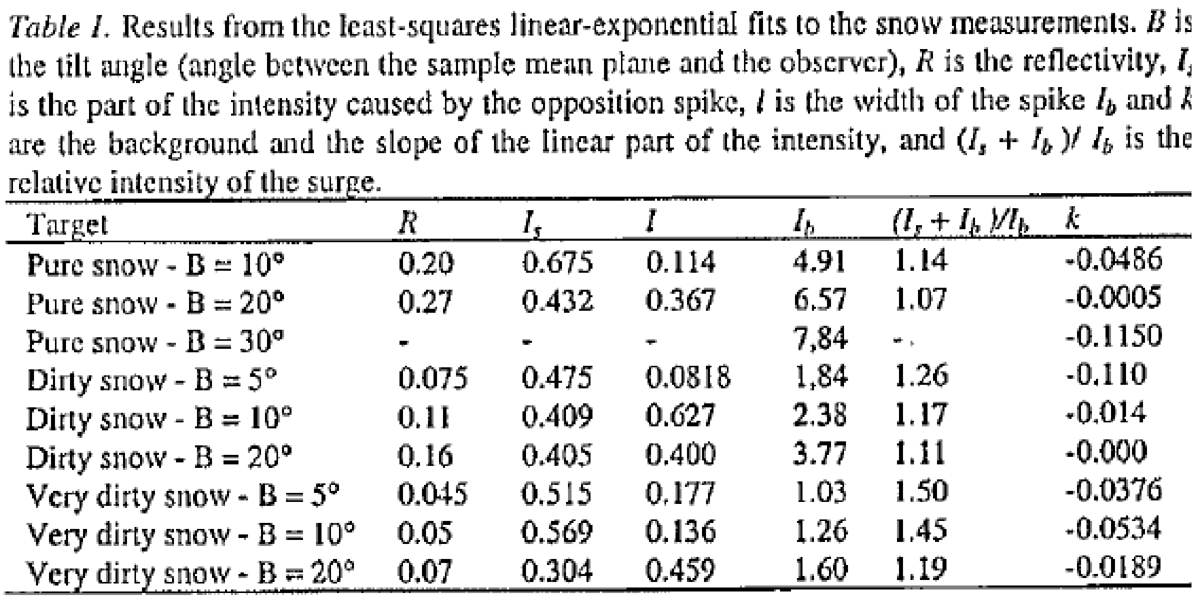


Table I. Results from the least-squares linear-exponential fits to the snow measurements. B is the tilt angle (angle between the sample mean plane and the observer), R is the reflectivity, I_s is the part of the intensity caused by the opposition spike, l is the width of the spike I_b and k are the background and the slope of the linear part of the intensity, and $(I_s + I_b)/I_b$ is the relative intensity of the surge.

Target	R	I_s	l	I_b	$(I_s + I_b)/I_b$	k
Pure snow - B = 10°	0.20	0.675	0.114	4.91	1.14	-0.0486
Pure snow - B = 20°	0.27	0.432	0.367	6.57	1.07	-0.0005
Pure snow - B = 30°	-	-	-	7,84	-	-0.1150
Dirty snow - B = 5°	0.075	0.475	0.0818	1,84	1.26	-0.110
Dirty snow - B = 10°	0.11	0.409	0.627	2.38	1.17	-0.014
Dirty snow - B = 20°	0.16	0.405	0.400	3.77	1.11	-0.000
Very dirty snow - B = 5°	0.045	0.515	0.177	1.03	1.50	-0.0376
Very dirty snow - B = 10°	0.05	0.569	0.136	1.26	1.45	-0.0534
Very dirty snow - B = 20°	0.07	0.304	0.459	1.60	1.19	-0.0189

Piironen, (2000)

Sample data:

Improved seeding technique (5.1.1)

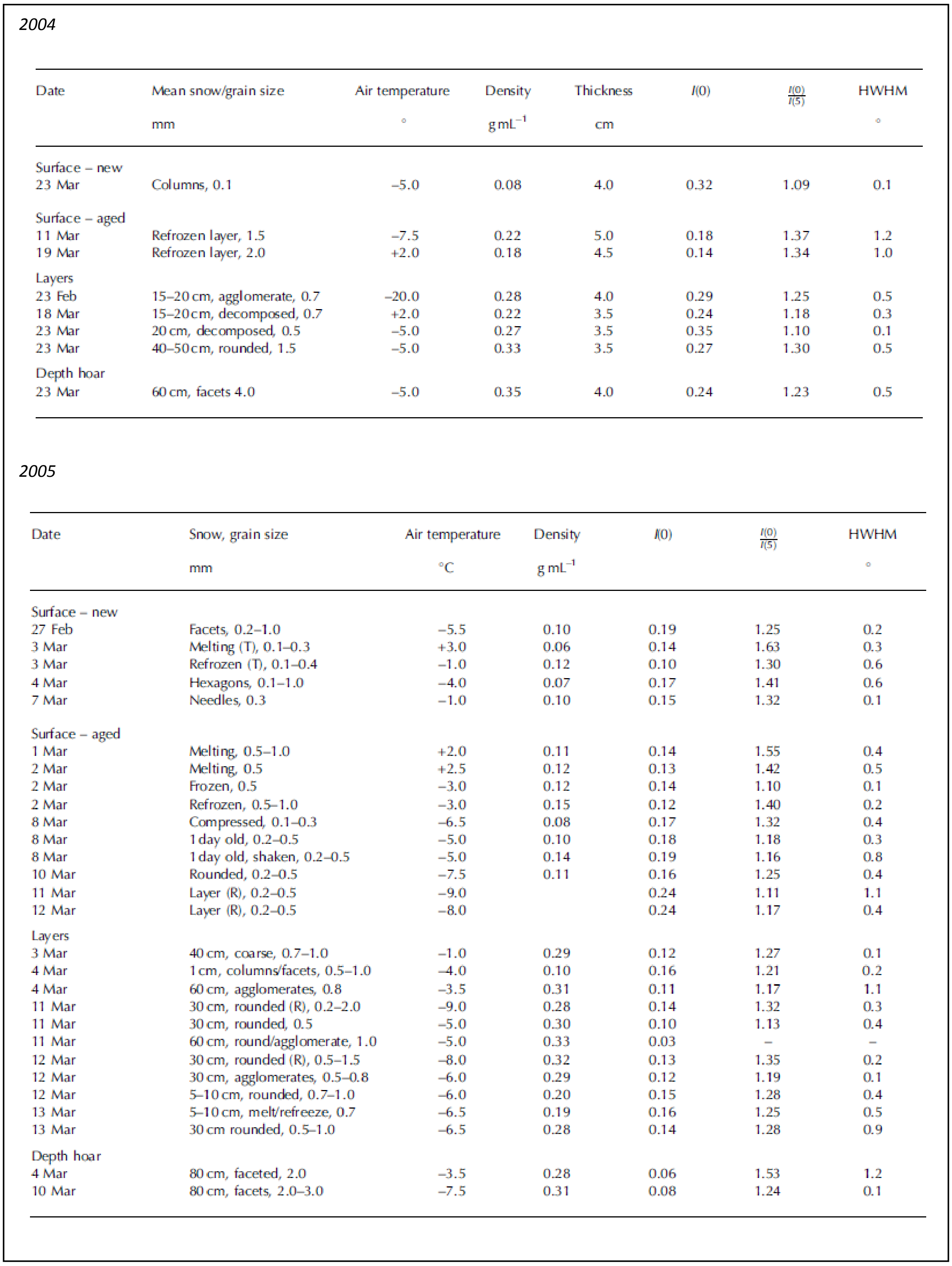
	Data			Best Fit		
Angle	1 layer	2 layers	3 layers	1 layer	2 layers	3 layers
165	0.165041	0.226132326	0.260221424	0.175	0.225	0.275
166	0.168413	0.229248158	0.282460294	0.18	0.23	0.28
167	0.179381	0.251271625	0.298895185	0.185000001	0.235	0.285
168	0.229346	0.331010885	0.343220533	0.190000003	0.24	0.29
169	0.16615	0.233718038	0.286287078	0.195000017	0.245	0.295
170	0.177226	0.227335567	0.28310788	0.200000084	0.25	0.3
171	0.161264	0.227537898	0.279955278	0.205000418	0.255	0.305
172	0.189109	0.252142611	0.310951856	0.210002071	0.260002	0.310002
173	0.182247	0.263075019	0.316712568	0.215010256	0.26501	0.31501
174	0.173479	0.245169754	0.309346177	0.220050797	0.270051	0.320051
175	0.190841	0.265663337	0.313420459	0.225251597	0.275252	0.325252
175.5	0.194624	0.290171627	0.338148319	0.228059939	0.27806	0.32806
176	0.209859	0.33932778	0.369011099	0.231246168	0.281246	0.331246

176.5	0.225786	0.308356755	0.394951249	0.235273398	0.285273	0.335273
177	0.235256	0.342592859	0.454347477	0.24117231	0.291172	0.341172
177.5	0.567313	0.644076598	0.728483264	0.251236729	0.301237	0.351237
178	0.417472	0.543243995	0.636561515	0.270571653	0.320572	0.370572
178.2	0.329314	0.483191959	0.605665165	0.283100851	0.333101	0.383101
178.4	0.376775	0.501963877	0.584958879	0.299977959	0.349978	0.399978
178.6	0.368606	0.476497169	0.575896268	0.322844689	0.372845	0.422845
178.8	0.345412	0.467084133	0.552038106	0.353955774	0.403956	0.453956
179	0.410834	0.480439588	0.544635421	0.396422388	0.446422	0.496422
179.1	0.428741	0.532337358	0.587599927	0.423197585	0.473198	0.523198
179.2	0.429459	0.482142487	0.565736319	0.454526942	0.504527	0.554527
179.3	0.470555	0.558674578	0.616396107	0.491211056	0.541211	0.591211
179.4	0.554514	0.635553387	0.690645963	0.53416683	0.584167	0.634167
179.5	0.64514	0.694312906	0.747667582	0.584496723	0.634497	0.684497
179.6	0.647654	0.669711796	0.758982787	0.643473211	0.693473	0.743473
179.7	0.725075	0.711959686	0.792137311	0.712585263	0.762585	0.812585
179.8	0.828947	0.87103316	0.887919066	0.793614452	0.843614	0.893614
179.9	0.968378	0.946642851	0.979970182	0.88860157	0.938602	0.988602
179.95	1.102519	1.075589698	1.152597816	0.942083864	0.992084	1.042084
180				0.95	1.05	1.1
175				0.100008104	0.275252	0.325252

Compact columns (best fit curve rejected)

x (Scattering Angle)	y (Intensity)	I(x) =0.10*exp((x-180)/0.085)+ .3505 + (0.0005*(x-180))	I(x′)= 0.10*exp((x-180)/0.075)+ .3385 + (0.0003*(x-180))
170	0.349108322237321	0.345500000000000	0.335500000000000
171	0.360786353305948	0.346000000000000	0.335800000000000
172	0.367419531790180	0.346500000000000	0.336100000000000
173	0.336157720316112	0.347000000000000	0.336400000000000
174	0.421332989263057	0.347500000000000	0.336700000000000
175	0.387067063511777	0.348000000000000	0.337000000000000
175.500000000000	0.385406764790405	0.348250000000000	0.337150000000000
176	0.411830141863650	0.348500000000000	0.337300000000000
176.500000000000	0.454332405258556	0.348750000000000	0.337450000000000
177	0.432217481019587	0.349000000000000	0.337600000000000
177.500000000000	0.378514150085313	0.349250000000017	0.337750000000000
178	0.368426713082422	0.349500000006044	0.337900000000262
178.500000000000	0.361299774529905	0.349750002167603	0.338050000206115
179.100006103516	0.351637740906730	0.350052524319827	0.338230616302294
179.199996948242	0.364725957564123	0.350108173834967	0.338262329899744
179.300003051758	0.365320550500794	0.350176515376207	0.338298843974211
179.399993896484	0.358330408607446	0.350285969676225	0.338353541701845
179.500000000000	0.370441204754143	0.350528821704005	0.338477263380134
179.600006103516	0.394207667387503	0.351204261232787	0.338862836121997
179.699996948242	0.368110164649066	0.353282109112948	0.340241488448332
179.800003051758	0.338512485535985	0.359909250607846	0.345388628772465
179.899993896484	0.378247300282281	0.381284299568297	0.364827566908671
179.949996948242	0.405963555033843	0.406003642091668	0.389824621930492
		+/- 0.0136	+/- 0.0121
		I(175) = 0.3480	I(175) = 0.3370
		I(180) =0.4505	I(180) = 0.4385
		I(180/175) = 1.2945	((180/175) = 1.2945

C. Kaasalainen snow data



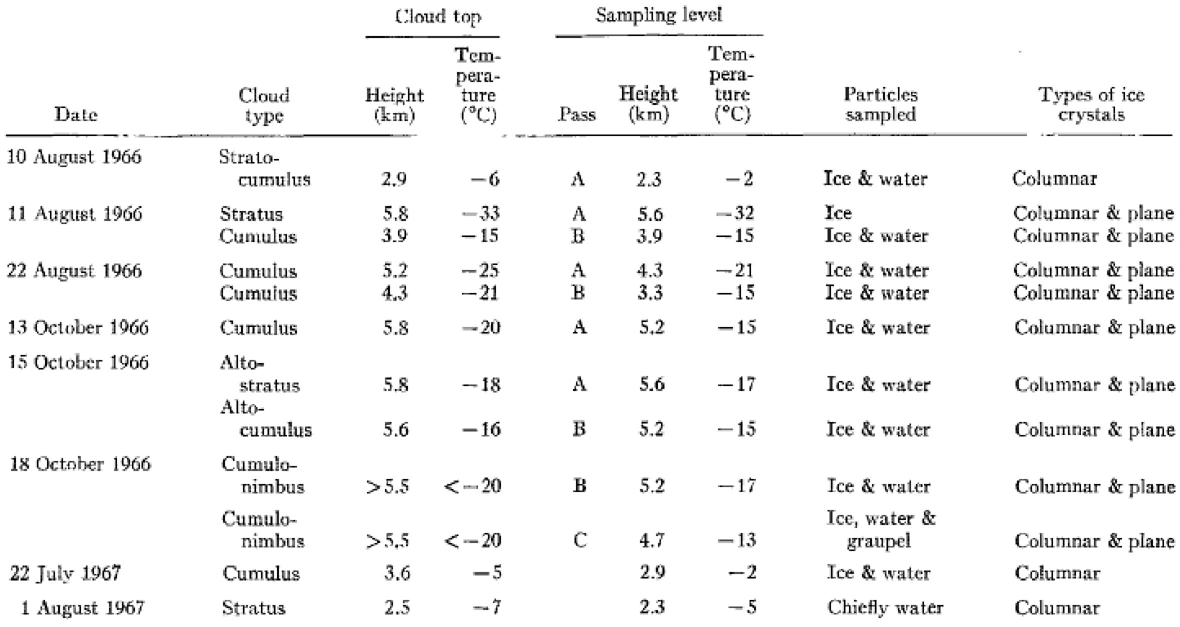
2004

Date	Mean snow/grain size	Air temperature	Density	Thickness	$I(0)$	$\frac{I(0)}{I(5)}$	HWHM
	mm	°	g mL^{-1}	cm			°
Surface – new							
23 Mar	Columns, 0.1	−5.0	0.08	4.0	0.32	1.09	0.1
Surface – aged							
11 Mar	Refrozen layer, 1.5	−7.5	0.22	5.0	0.18	1.37	1.2
19 Mar	Refrozen layer, 2.0	+2.0	0.18	4.5	0.14	1.34	1.0
Layers							
23 Feb	15–20 cm, agglomerate, 0.7	−20.0	0.28	4.0	0.29	1.25	0.5
18 Mar	15–20 cm, decomposed, 0.7	+2.0	0.22	3.5	0.24	1.18	0.3
23 Mar	20 cm, decomposed, 0.5	−5.0	0.27	3.5	0.35	1.10	0.1
23 Mar	40–50 cm, rounded, 1.5	−5.0	0.33	3.5	0.27	1.30	0.5
Depth hoar							
23 Mar	60 cm, facets 4.0	−5.0	0.35	4.0	0.24	1.23	0.5

2005

Date	Snow, grain size	Air temperature	Density	$I(0)$	$\frac{I(0)}{I(5)}$	HWHM
	mm	°C	g mL^{-1}			°
Surface – new						
27 Feb	Facets, 0.2–1.0	−5.5	0.10	0.19	1.25	0.2
3 Mar	Melting (T), 0.1–0.3	+3.0	0.06	0.14	1.63	0.3
3 Mar	Refrozen (T), 0.1–0.4	−1.0	0.12	0.10	1.30	0.6
4 Mar	Hexagons, 0.1–1.0	−4.0	0.07	0.17	1.41	0.6
7 Mar	Needles, 0.3	−1.0	0.10	0.15	1.32	0.1
Surface – aged						
1 Mar	Melting, 0.5–1.0	+2.0	0.11	0.14	1.55	0.4
2 Mar	Melting, 0.5	+2.5	0.12	0.13	1.42	0.5
2 Mar	Frozen, 0.5	−3.0	0.12	0.14	1.10	0.1
2 Mar	Refrozen, 0.5–1.0	−3.0	0.15	0.12	1.40	0.2
8 Mar	Compressed, 0.1–0.3	−6.5	0.08	0.17	1.32	0.4
8 Mar	1 day old, 0.2–0.5	−5.0	0.10	0.18	1.18	0.3
8 Mar	1 day old, shaken, 0.2–0.5	−5.0	0.14	0.19	1.16	0.8
10 Mar	Rounded, 0.2–0.5	−7.5	0.11	0.16	1.25	0.4
11 Mar	Layer (R), 0.2–0.5	−9.0		0.24	1.11	1.1
12 Mar	Layer (R), 0.2–0.5	−8.0		0.24	1.17	0.4
Layers						
3 Mar	40 cm, coarse, 0.7–1.0	−1.0	0.29	0.12	1.27	0.1
4 Mar	1 cm, columns/facets, 0.5–1.0	−4.0	0.10	0.16	1.21	0.2
4 Mar	60 cm, agglomerates, 0.8	−3.5	0.31	0.11	1.17	1.1
11 Mar	30 cm, rounded (R), 0.2–2.0	−9.0	0.28	0.14	1.32	0.3
11 Mar	30 cm, rounded, 0.5	−5.0	0.30	0.10	1.13	0.4
11 Mar	60 cm, round/agglomerate, 1.0	−5.0	0.33	0.03	–	–
12 Mar	30 cm, rounded (R), 0.5–1.5	−8.0	0.32	0.13	1.35	0.2
12 Mar	30 cm, agglomerates, 0.5–0.8	−6.0	0.29	0.12	1.19	0.1
12 Mar	5–10 cm, rounded, 0.7–1.0	−6.0	0.20	0.15	1.28	0.4
13 Mar	5–10 cm, melt/refreeze, 0.7	−6.5	0.19	0.16	1.25	0.5
13 Mar	30 cm rounded, 0.5–1.0	−6.5	0.28	0.14	1.28	0.9
Depth hoar						
4 Mar	80 cm, faceted, 2.0	−3.5	0.28	0.06	1.53	1.2
10 Mar	80 cm, facets, 2.0–3.0	−7.5	0.31	0.08	1.24	0.1

D. Ono ice crystals data and sample images



		Cloud top		Sampling level				
Date	Cloud type	Height (km)	Temperature (°C)	Pass	Height (km)	Temperature (°C)	Particles sampled	Types of ice crystals
10 August 1966	Strato-cumulus	2.9	−6	A	2.3	−2	Ice & water	Columnar
11 August 1966	Stratus	5.8	−33	A	5.6	−32	Ice	Columnar & plane
	Cumulus	3.9	−15	B	3.9	−15	Ice & water	Columnar & plane
22 August 1966	Cumulus	5.2	−25	A	4.3	−21	Ice & water	Columnar & plane
	Cumulus	4.3	−21	B	3.3	−15	Ice & water	Columnar & plane
13 October 1966	Cumulus	5.8	−20	A	5.2	−15	Ice & water	Columnar & plane
15 October 1966	Alto-stratus	5.8	−18	A	5.6	−17	Ice & water	Columnar & plane
	Alto-cumulus	5.6	−16	B	5.2	−15	Ice & water	Columnar & plane
18 October 1966	Cumulo-nimbus	>5.5	<−20	B	5.2	−17	Ice & water	Columnar & plane
	Cumulo-nimbus	>5.5	<−20	C	4.7	−13	Ice, water & graupel	Columnar & plane
22 July 1967	Cumulus	3.6	−5		2.9	−2	Ice & water	Columnar
1 August 1967	Stratus	2.5	−7		2.3	−5	Chiefly water	Columnar

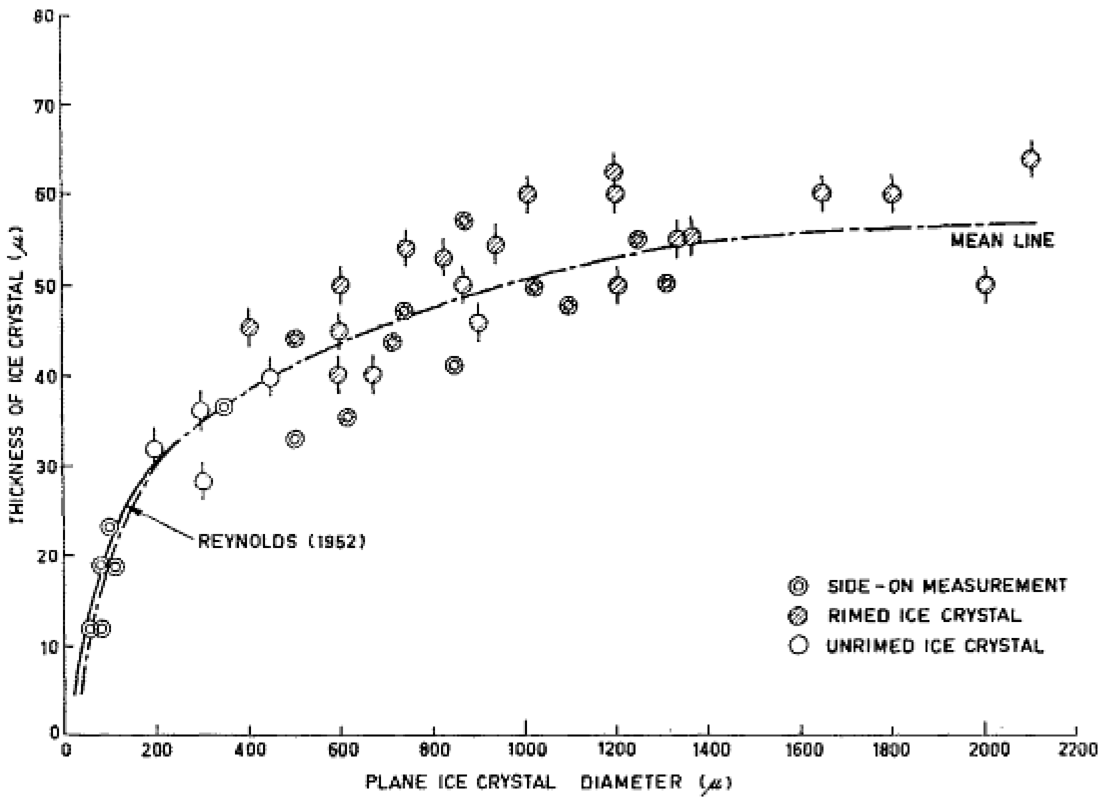


FIG. 6. Relation between the thickness and diameter of plane ice crystals. The double circles indicate side-on measurement of crystal thickness; hatched circles, rimed crystals; open circles, ice crystals without riming. The experimental results of Reynolds (1952) are plotted for comparison.

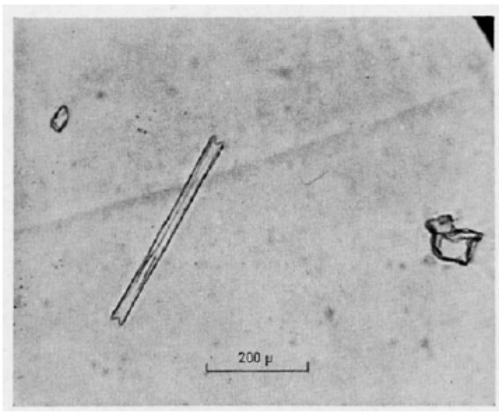


FIG. 1. Sheath ice crystal observed at −5C, 1 August 1967.

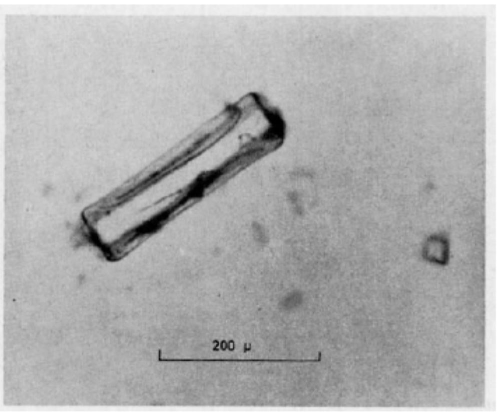


FIG. 2. Warm-region column observed at −2C, 10 August 1966.

Fig1. Needle aspect ratio 10:1(≈400μm)

Fig2. Column aspect ratio 3:1((≈250μm)

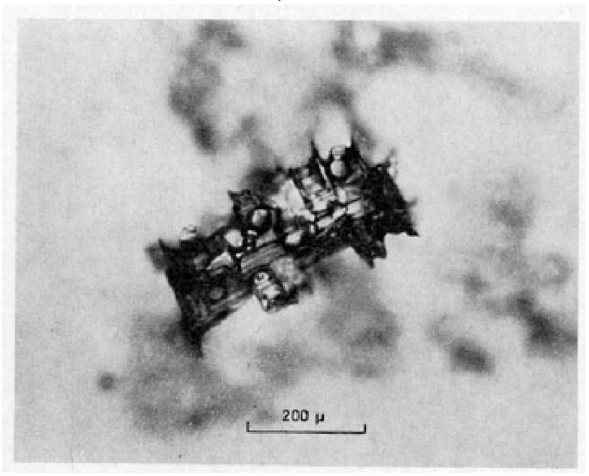
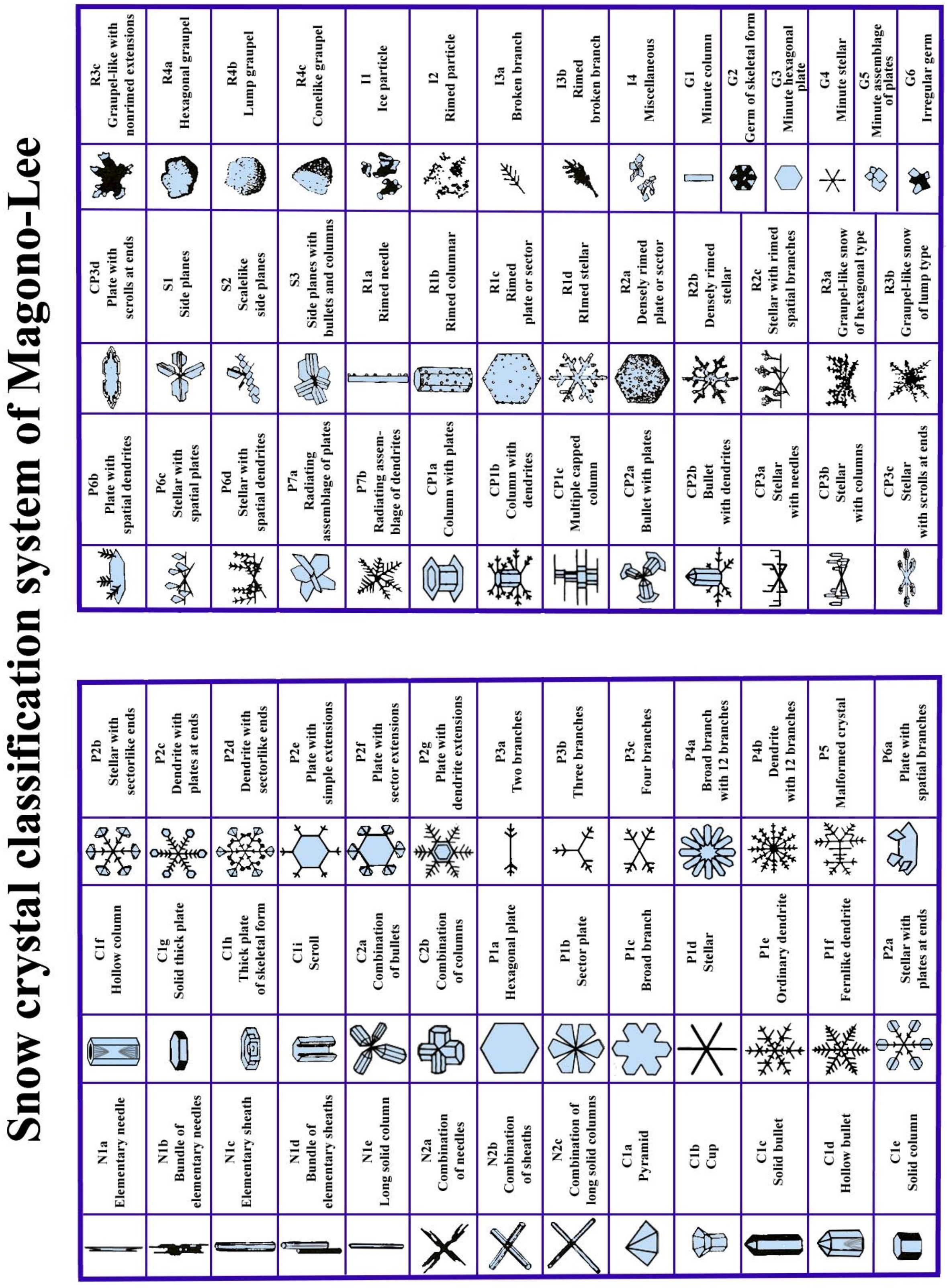


FIG. 4. Column with end plates observed at −15C, 22 August 1966. Accreted cloud droplets froze with the same "c" axis orientation as that of the substrate column crystal.

Fig4. Column with heavy dendritic growths

E. Magono-Lee Snowflake classifications



Snow crystal classification system of Magono-Lee

N1a Elementary needle	C1f Hollow column	P2b Stellar with sectorlike ends
N1b Bundle of elementary needles	C1g Solid thick plate	P2c Dendrite with plates at ends
N1c Elementary sheath	C1h Thick plate of skeletal form	P2d Dendrite with sectorlike ends
N1d Bundle of elementary sheaths	C1i Scroll	P2e Plate with simple extensions
N1e Long solid column	C2a Combination of bullets	P2f Plate with sector extensions
N2a Combination of needles	C2b Combination of columns	P2g Plate with dendrite extensions
N2b Combination of sheaths	P1a Hexagonal plate	P3a Two branches
N2c Combination of long solid columns	P1b Sector plate	P3b Three branches
C1a Pyramid	P1c Broad branch	P3c Four branches
C1b Cup	P1d Stellar	P4a Broad branch with 12 branches
C1c Solid bullet	P1e Ordinary dendrite	P4b Dendrite with 12 branches
C1d Hollow bullet	P1f Fernlike dendrite	P5 Malformed crystal
C1e Solid column	P2a Stellar with plates at ends	P6a Plate with spatial branches

P6b Plate with spatial dendrites	CP3d Plate with scrolls at ends	R3c Graupel-like with nonrimed extensions
P6c Stellar with spatial plates	S1 Side planes	R4a Hexagonal graupel
P6d Stellar with spatial dendrites	S2 Scalelike side planes	R4b Lump graupel
P7a Radiating assemblage of plates	S3 Side planes with bullets and columns	R4c Conelike graupel
P7b Radiating assemblage of dendrites	R1a Rimed needle	I1 Ice particle
CP1a Column with plates	R1b Rimed columnar	I2 Rimed particle
CP1b Column with dendrites	R1c Rimed plate or sector	I3a Broken branch
CP1c Multiple capped column	R1d Rimed stellar	I3b Rimed broken branch
CP2a Bullet with plates	R2a Densely rimed plate or sector	I4 Miscellaneous
CP2b Bullet with dendrites	R2b Densely rimed stellar	G1 Minute column G2 Germ of skeletal form
CP3a Stellar with needles	R2c Stellar with rimed spatial branches	G3 Minute hexagonal plate
CP3b Stellar with columns	R3a Graupel-like snow of hexagonal type	G4 Minute stellar G5 Minute assemblage of plates
CP3c Stellar with scrolls at ends	R3b Graupel-like snow of lump type	G6 Irregular germ

F. Code

Basic code consists of five programs:

Sandbox – Main code that calls functions and generates files.

Assemble Layer – Uses generated parameters to create bounding spheres and duplicates where they cross boundaries. Also tests for simple rejection based on degree of overlapping.

Hexcolumn – Generates the vertex data for a hexagonal column with options for multiple basal facets (indents and bullets) and oblique basal facets.

Add Crystal to Layer – Replaces bounding spheres with crystals after rotation.

Show Crystals – Used for generating the images of the layer.

Tetris Variant:

Sandbox , Assemble Layer, Add Crystal to Layer

Sandbox

```
function varargout=Sandbox4(varargin)


handles.length=10;
handles.width=10;
handles.thickness=2;
handles.failureBreakOut=10;
handles.outputFileName='seedCrystal.txt';
handles.L_mean=0.1;
handles.L_deviation=0.03;
handles.R_mean=0.5;
handles.R_deviation=0.1;
handles.aveIndent=0.0;
handles.crystalDensity=0.9167;
% case 'Hex Columns Only'
    handles.crystalSeeds=1;
% case 'Hex/Single Indents'
%    handles.crystalSeeds=2;
% case 'Hex/Sing/Doub'
%   handles.crystalSeeds=3;
handles.obliqueBasalFacet=0;

rejections=0;
uniqueCrystals=0;
totalCrystals=0;
layer=zeros;
seeds=0;
crystalNo=0;
emergingArrayFacets=0;
emergingArrayVertex=zeros(3,0);
emergingArrayVertexNewStructure=zeros(3,0);
emergingArrayVertexLocation=zeros(3,0);
totalVolume=0;
density=0;
planeMatrix=zeros(0,0);
```

```
lineMatrix=zeros(0,0);
lineEquationMatrix=sym([]);
%convrejectiions=0
doprinting=0
%while(rejections<handles.failureBreakOut);
%replaced failure breakout with density - this will be modified so that
%this can be an input. This is only reasonable once seeding can regularly
%get below the densities that are present in ice (see Kaasalanian).
while (density < .08)
%while (seeds < 30)
    %spawn random crystal dimensions - choosing the longest for overall
    %diameter
    crystalLength=normrnd(handles.L_mean,handles.L_deviation);
    crystalRadius=normrnd(handles.R_mean,handles.R_deviation);

    %Set bounding sphere limits
    seedRadius=0.5*((crystalLength^2+(2*crystalRadius)^2)^0.5 );

    %Attempt to seed this diameter object into the layer also seed
    %duplicates where the object crosses a boundary


[layer,failed,ParticleNo,particle,overlapCrystalArray]=assembleLayer4(handl
es.length,handles.width,handles.thickness,seedRadius,layer,crystalRadius,se
eds);

   if failed==1
       rejections=rejections+1;
   else
       %create crystal
       createHexCrystal(crystalLength,crystalRadius,handles);
       %rotate crystal and its duplicates and create their absolute
       %position

[NewNumberFacets,newlayerData,newlayerData2,newlayerVertexData,crystalVerte
xData,lineEquation,planeEquation,boundingSphereNumber,qtyCrystalsSeeded]=ad
dCrystalToLayer4(particle,handles.length,handles.width,handles.thickness,se
eds,overlapCrystalArray);
       %Convex hull check. This creates a matrix of line equations
       %corresponding to the overlapping bounding spheres. These are then
       %converted into an array of intersection points against all the
       %facets of the the seeding crystal. Finally there is a check to
       %determine if any of these are within a bounding convex hull of the
       %seeding crystals
       hullRejection=0;
       if size(overlapCrystalArray)>0
           overlapCrystalArray=unique(overlapCrystalArray,'rows')
           intersect=zeros(0,0);
           intersectArray=zeros(0,0);
           syms x y z t;
           P = [x,y,z];
           tic;
           fprintf('\n\nQverlap Equation Matrix: ')

vi=ismember(lineEquationMatrix(:,1:2),overlapCrystalArray(:,1:2),'rows');
           potentialOverlapLineEquationMatrix=lineEquationMatrix(vi,:);
           toc
           planeEquation;
           tic;
           fprintf('\nPlane Equation: ')
```

```
            for q=(1:size(overlapCrystalArray))
                for p=(1:size(planeEquation))
                    for j=(1:size(potentialOverlapLineEquationMatrix))
                        %fprintf('do they equal? %s,
%s',potentialOverlapLineEquationMatrix(j,3),planeEquation(p,2));
                        if
(overlapCrystalArray(q,1)==potentialOverlapLineEquationMatrix(j,1))...
                                &&
(overlapCrystalArray(q,2)==potentialOverlapLineEquationMatrix(j,2))...
                                &&
(overlapCrystalArray(q,3)==planeEquation(p,2))
                            %fprintf('\n yeah, got here')

newfunction=subs(planeEquation(p,3),P,potentialOverlapLineEquationMatrix(j,
3:5));
                            t0=solve(newfunction);

point1=subs(potentialOverlapLineEquationMatrix(j,3:5),t,t0);
                            point1=double(point1);
                            intersectArray=vertcat(intersectArray,point1);
                        end
                        if size(intersectArray)>0
                            [d1,d2,d3]=size(crystalVertexData);
                            for i=(1:d3)

intersect=inhull(intersectArray,crystalVertexData(:,:,i));
                                if any(intersect)==1
                                    hullRejection=1;
                                    disp('   Rejected by Convex Hull Test')
                                    break
                                end
                            end
                        end
                        if hullRejection==1;
                            break
                        end
                    end
                    if hullRejection==1;
                        break
                    end
                end
                if hullRejection==1;
                    break
                end
            end
            toc
                 tic
% %                 disp('   Revised plane equation')
%
bigArrayI=repmat(potentialOverlapLineEquationMatrix,size(planeEquation));
%                   disp(' big arrayI')
%                   toc
%                   tic
%
bigArrayII=repmat(planeEquation,size(potentialOverlapLineEquationMatrix));
%                   disp(' big arrayII')
%                   toc
%                   tic
%                   bigArrayIII=[bigArrayI(:,3:5) bigArrayII(:,3)]
%                   disp(' big arrayIII')
%                   toc
%                   tic
%                   disp('new function')
%                   %for t=(1:size(bigArrayIII))
%
[newfunctionII]=subs(bigArrayIII(:,4),P,bigArrayIII(:,1:3))
%                   %end
%
%                   toc
%                   tic
%
%                       [t0]=solve(newfunctionII)
%
%                   for t=(1:size(bigArrayIII))
%
point1=subs(potentialOverlapLineEquationMatrix(:,3:5),t,t0);
%                   end
%                   point1=double(point1)
%                   disp(' big arrayIII function solving')
%                   toc

%             size(intersectArray)
%             toc
%              tic
%              fprintf('\nConvex hull: ')
%
%             toc
%As above though now lines and facets are reversed (needed for large
facets)
%Test all the lines generated by the add crystal to layer against all
%facets already seeded Might not need this step...

% Creates a facets corresponding to the overlapping bounding spheres. These
are then
% tested against the line matrix of the seeds to create intersection
points.
% The points are then tested against the convex hulls of the previously
seeded crystals
       if hullRejection==0
           intersect=zeros(0,0);
           intersectArray=zeros(0,0);
           tic;
           fprintf('\n\nOverlap Plane Matrix (Reciprocation): ')

vi=ismember(planeMatrix(:,1:2),overlapCrystalArray(:,1:2),'rows');
           potentialOverlapPlaneMatrix=planeMatrix(vi,:);
           toc
           tic;
           fprintf('\nLine Equation (Reciprocation): ')
           for q=(1:size(overlapCrystalArray))
               for j=(1:size(potentialOverlapPlaneMatrix))
                   for p=(1:size(lineEquation))
                       if
(overlapCrystalArray(q,1)==potentialOverlapPlaneMatrix(j,1))...
                             &&
(overlapCrystalArray(q,2)==potentialOverlapPlaneMatrix(j,2))...
                             &&
(overlapCrystalArray(q,3)==lineEquation(p,2))
                           %fprintf('\n yeah, got here')

newfunction=subs(potentialOverlapPlaneMatrix(j,3),P,lineEquation(p,3:5));
                           t0=solve(newfunction);
                           point1=subs(lineEquation(p,3:5),t,t0);
                           point1=double(point1);
                           intersectArray=vertcat(intersectArray,point1);
                       end
                   end
               end
           end
           size(intersectArray)
           toc
           %Extract all the vertex data for the seeded crystals to test
against these
           %intersect points.
           tic
           fprintf('\nAssemble array of vertices from existing vertice
matrix: ')

vi=ismember(emergingArrayVertexLocation(:,1:2),overlapCrystalArray(:,1:2),'
rows');

potentialOverlapArrayVertexLocation=emergingArrayVertexLocation(vi,:);
           c=1;
           b=1;
           testArrayOfCrystals=zeros;
           for y=1:size(potentialOverlapArrayVertexLocation)
              if y>1
                  if
potentialOverlapArrayVertexLocation(y,1)~=potentialOverlapArrayVertexLocati
on(y-1,1)...
                  ||
potentialOverlapArrayVertexLocation(y,2)~=potentialOverlapArrayVertexLocati
on(y-1,2);
                    c=1;
                    b=b+1;
                  end
              end

testArrayOfCrystals(c,1:3,b)=potentialOverlapArrayVertexLocation(y,3:5);
              c=c+1;
           end
           toc
           tic
           fprintf('\nConvex Hull (Reciprocation): ')
           testArrayOfCrystals;
           if size(intersectArray)>0
               [d1,d2,d3]=size(testArrayOfCrystals);
               for i=(1:d3)

intersect=inhull(intersectArray,testArrayOfCrystals(:,:,i));
                   if any(intersect)==1
                       hullRejection=1;
                       disp('   Rejected by Convex Hull Test
(Reciprocation): ')
                       break
                   end
               end
           end
           toc
       end


       end
```

```
%        %just for reference to see how many were rejected
%        if (intersect>0)&&(overlapping==1)
%            convrejectiions=convrejectiions+1;
%
%        end
       %if any(intersect)==0 > replaced with line below
       if hullRejection==0;

           %from assemble layer
            j=seeds;
            for i=1:ParticleNo
                for q=1:3
                layer(i+j,q)=particle(i,q);
                end
                layer(i+j,4)=crystalRadius;
                %layer(i+j,5)=boundingCylinderHeight; oldcrap
                layer(i+j,5)=crystalRadius;
                layer(i+j,6)=boundingSphereNumber;
                layer(i+j,7)=i;
                seeds=seeds+1;
            end



           %Update volume - only seed once for duplicate particles as the
parts
           %within the layer boundaries will only account for 1 complete
           %crystal
           %crystalVolume=pi*crystalRadius^2*crystalLength; %* ParticleNo
           crystalVolume=3/2*3^.5*crystalRadius^2*crystalLength;
           totalVolume=totalVolume+crystalVolume;
           totalCrystals=totalCrystals+qtyCrystalsSeeded;
           uniqueCrystals=uniqueCrystals+1;

density=totalVolume*handles.crystalDensity/(handles.length*handles.width*ha
ndles.thickness);
           %add these duplicates to the emerging array
           emergingArrayFacets=emergingArrayFacets+NewNumberFacets;
           emergingArrayVertex=vertcat(emergingArrayVertex,newlayerData);
           planeMatrix=vertcat(planeMatrix,planeEquation);
           %lineMatrix=vertcat(lineMatrix,line);
           lineEquationMatrix=vertcat(lineEquationMatrix,lineEquation);
           %add new structure to create a carriage return

emergingArrayVertexNewStructure=vertcat(emergingArrayVertexNewStructure,new
layerData2);

emergingArrayVertexNewStructure=vertcat(emergingArrayVertexNewStructure,0);

emergingArrayVertexLocation=vertcat(emergingArrayVertexLocation,newlayerVer
texData);
       else
           rejections=rejections+1;
           %eh - what's the next line trying to do???
           %layer(line(1,1)==layer(:,6),:)=[]
       end
   end
%move this down so it does the below step every time
%end
```

```
doprinting=doprinting+1;
if doprinting==5;
    doprinting=0;
end
if doprinting==0

%output all Emerging Array information to a file
output= fopen('crystalLayerFile_ReadMe.txt','w');
fprintf(output,'Layer dimensions (%dx%dx%d)mm^3
\r\n',handles.length,handles.width,handles.thickness);
%fprintf(output,'Layer width %d \r\n',handles.width);
%fprintf(output,'Layer thickness %d \r\n',handles.thickness);
fprintf(output,'rejections %d \r\n',handles.failureBreakOut);
fprintf(output,'Crystals Seeded %d\r\n',totalCrystals);
fprintf(output,'of which Unique %d\r\n',uniqueCrystals);
fprintf(output,'Hex crystals \r\n');
fprintf(output,'Crystal length %3.3f +/- %3.4f
\r\n',handles.L_mean,handles.L_deviation);
fprintf(output,'Crystal radius %3.3f +/- %3.4f
\r\n',handles.R_mean,handles.R_deviation);
fprintf(output,'Average indent(of length) %d%% \r\n',handles.aveIndent);
fprintf(output,'Total Crystal Volume %5.5f \r\n',totalVolume);
fprintf(output,'Crystal density %5.5f \r\n',handles.crystalDensity);
fprintf(output,'Layer Density %5.5f \r\n',density);
fprintf(output,'Location (x \t y \t\t z) \t\t Sphere rad \t Seed \t
Particle no. \r\n');
for i=(1:size(layer))
    fprintf(output,'%5.5f \t %5.5f \t %5.5f \t %5.5f \t %d \t %d
\r\n',layer(i,1),layer(i,2),layer(i,3),layer(i,4),layer(i,6),layer(i,7));
end
fclose(output);
%whos
%save bobsybob.mat

%output all Emerging Array to a file
output=fopen('crystalLayerFile.txt','w');
fprintf(output,'%d \r\n',emergingArrayFacets);
for i=(1:size(emergingArrayVertex))
    fprintf(output,'%d\r\n ',emergingArrayVertex(i));
end
for i=(1:size(emergingArrayVertexLocation))
    fprintf(output,'%5.5f \t %5.5f \t %5.5f
\r\n',emergingArrayVertexLocation(i,3),emergingArrayVertexLocation(i,4),eme
rgingArrayVertexLocation(i,5));
end
fclose(output);



%output all Emerging Array to a file - with new structure
output=fopen('crystalLayerFile2.txt','w');
fprintf(output,'%d \r\n',emergingArrayFacets);
fprintf(output,'%d \r\n',totalCrystals);
insertReturn=1;
for i=(1:size(emergingArrayVertexNewStructure))
    if insertReturn==1
        fprintf(output,'%d \r\n',emergingArrayVertexNewStructure(i));
        insertReturn=0;
    elseif emergingArrayVertexNewStructure(i)==0
        fprintf(output,'\r\n');
        insertReturn=1;
    else
        fprintf(output,'%d ',emergingArrayVertexNewStructure(i));
    end
end
for i=(1:size(emergingArrayVertexLocation))
    fprintf(output,'%5.5f \t %5.5f \t %5.5f
\r\n',emergingArrayVertexLocation(i,3),emergingArrayVertexLocation(i,4),eme
rgingArrayVertexLocation(i,5));
end
fclose(output);
%output all Emerging Array to a file - with new structure
output=fopen('crystalLayerFile3.txt','w');
fprintf(output,'%d \r\n',emergingArrayFacets);
fprintf(output,'%d \r\n',totalCrystals);
insertReturn=1;
for i=(1:size(emergingArrayVertexNewStructure))
    if insertReturn==1
        fprintf(output,'%d \r\n',emergingArrayVertexNewStructure(i));
        insertReturn=0;
    elseif emergingArrayVertexNewStructure(i)==0
        fprintf(output,'\r\n');
        insertReturn=1;
    else
        fprintf(output,'%d ',emergingArrayVertexNewStructure(i));
    end
end

for i=(1:size(emergingArrayVertexLocation))
    fprintf(output,'%5.5f \t %5.5f \t %5.5f \t %5.5f \t %5.5f
\r\n',emergingArrayVertexLocation(i,1),emergingArrayVertexLocation(i,2),eme
rgingArrayVertexLocation(i,3),emergingArrayVertexLocation(i,4),emergingArra
yVertexLocation(i,5));
end
fclose(output);



%output all FacetMatrix to a file
% output=fopen('planeMatrix.txt','w');
% for i=(1:size(planeMatrix))
%     fprintf(output,'%5.5f \t %5.5f \t %5.5f \t %5.5f \t %5.5f \t %5.5f \t
%5.5f \t %5.5f \t %5.5f \t %5.5f \t %5.5f \t %5.5f \t %5.5f \t %5.5f \t
%5.5f \t %5.5f \t %5.5f \t %5.5f \t %5.5f \t %5.5f \r\n',
planeMatrix(i,1:20));
% end
% fclose(output);

%output all lineMatrix to a file
output=fopen('lineEquationMatrix.txt','w');
outputLineEquation=char(lineEquationMatrix);
for i=(1:size(outputLineEquation))
    fprintf(output,'%s \t %s \t %s \t %s \r\n',outputLineEquation(i,1:4));
end
fclose(output);

%end doprinting
end
%end moved to here
end
```

```
function createHexCrystal(crystalLength,crystalRadius,handles,hObject,
eventdata)
%indent=handles.maxIndent;
indent =
normrnd((handles.aveIndent/2),(handles.aveIndent/6))*crystalLength/100;
crystalType = randi(handles.crystalSeeds);
if crystalType==1
    crystalType='standardcolumn';
elseif crystalType==2
    crystalType='indentsinglecolumn';
else crystalType='indentdoublecolumn';
end
obliqueEnds=0;%randi(1,1,2)+1;

obliqueBasalFacet=0;%randi(1,1,handles.obliqueBasalFacet);
hexcolumn( crystalLength,
crystalRadius,indent,crystalType,obliqueEnds,obliqueBasalFacet,handles.outp
utFileName );
```

Assemble Layer

```
function
[layer,failed,ParticleNo,particle,overlapCrystalArray]=assembleLayer4(lengt
h,width,thickness,crystalRadius,layer,trueRadius,seeds)

%[layer,failed,ParticleNo,particle,overlapCrystalArray]=assembleLayer3(hand
les.length,handles.width,handles.thickness,seedRadius,layer,crystalRadius);

       %Set random location of particle to be seeded
       overlapCrystals=zeros(0,0);
       overlapCrystalArray=zeros(0,0);
       failed=0;
       ParticleNo=1;
       overlap=0;
        particle(1,1)=(rand*length+eps); %-(crystalRadius/2));
        particle(1,2)=(rand*width+eps); %+(crystalRadius/2));
        particle(1,3)=(rand*thickness+eps-(crystalRadius/2));

        %Determine if the crystal overlaps the edges, if so, spawn another
        %crystal shifted over to the opposite side.
        if ((particle(1,1)+crystalRadius)>length);
            ParticleNo=ParticleNo+1;
            particle(ParticleNo,1)=particle(1,1)-length;
            particle(ParticleNo,2)=particle(1,2);
            particle(ParticleNo,3)=particle(1,3);
        end
       if ((particle(1,1)-crystalRadius)<0);
            ParticleNo=ParticleNo+1;
            particle(ParticleNo,1)=particle(1,1)+length;
            particle(ParticleNo,2)=particle(1,2);
            particle(ParticleNo,3)=particle(1,3);
       end
       if ((particle(1,2)+crystalRadius)>width);
            ParticleNo=ParticleNo+1;
            particle(ParticleNo,1)=particle(1,1);
            particle(ParticleNo,2)=particle(1,2)-width;
            particle(ParticleNo,3)=particle(1,3);
       end
       if ((particle(1,2)-crystalRadius)<0);
            ParticleNo=ParticleNo+1;
            particle(ParticleNo,1)=particle(1,1);
            particle(ParticleNo,2)=particle(1,2)+width;
            particle(ParticleNo,3)=particle(1,3);
       end

       if ((particle(1,3)+crystalRadius)>thickness);
            ParticleNo=ParticleNo+1;
            particle(ParticleNo,1)=particle(1,1);
            particle(ParticleNo,2)=particle(1,2);
            particle(ParticleNo,3)=particle(1,3)-thickness;
       end
       if ((particle(1,3)-crystalRadius)<0);
            ParticleNo=ParticleNo+1;
            particle(ParticleNo,1)=particle(1,1);
            particle(ParticleNo,2)=particle(1,2);
            particle(ParticleNo,3)=particle(1,3)+thickness;
       end
      %Where it overlaps two edges, another diagonal particle needs to be
      %added

      %length and width
        if (((particle(1,1)+crystalRadius)>length) &&
((particle(1,2)+crystalRadius)>width));
            ParticleNo=ParticleNo+1;
            particle(ParticleNo,1)=particle(1,1)-length;
            particle(ParticleNo,2)=particle(1,2)-width;
            particle(ParticleNo,3)=particle(1,3);
        end
       if (((particle(1,1)-crystalRadius)<0) && ((particle(1,2)-
crystalRadius)<0));
            ParticleNo=ParticleNo+1;
            particle(ParticleNo,1)=particle(1,1)+length;
            particle(ParticleNo,2)=particle(1,2)+width;
            particle(ParticleNo,3)=particle(1,3);
       end
       if (((particle(1,1)-crystalRadius)<0) &&
((particle(1,2)+crystalRadius)>width));
            ParticleNo=ParticleNo+1;
            particle(ParticleNo,1)=particle(1,1)+length;
            particle(ParticleNo,2)=particle(1,2)-width;
            particle(ParticleNo,3)=particle(1,3);
       end

        if (((particle(1,1)+crystalRadius)>length) && ((particle(1,2)-
crystalRadius)<0));
            ParticleNo=ParticleNo+1;
            particle(ParticleNo,1)=particle(1,1)-length;
            particle(ParticleNo,2)=particle(1,2)+width;
            particle(ParticleNo,3)=particle(1,3);
        end



       %length and thickness
        if (((particle(1,1)+crystalRadius)>length) &&
((particle(1,3)+crystalRadius)>thickness));
```

```
            ParticleNo=ParticleNo+1;
            particle(ParticleNo,1)=particle(1,1)-length;
            particle(ParticleNo,2)=particle(1,2);
            particle(ParticleNo,3)=particle(1,3)-thickness;
        end
       if (((particle(1,1)-crystalRadius)<0) && ((particle(1,3)-
crystalRadius)<0));
            ParticleNo=ParticleNo+1;
            particle(ParticleNo,1)=particle(1,1)+length;
            particle(ParticleNo,2)=particle(1,2);
            particle(ParticleNo,3)=particle(1,3)+thickness;
       end

       if (((particle(1,1)+crystalRadius)>length) && ((particle(1,3)-
crystalRadius)<0));
            ParticleNo=ParticleNo+1;
            particle(ParticleNo,1)=particle(1,1)-length;
            particle(ParticleNo,2)=particle(1,2);
            particle(ParticleNo,3)=particle(1,3)+thickness;
        end
       if (((particle(1,1)-crystalRadius)<0) &&
((particle(1,3)+crystalRadius)>thickness));
            ParticleNo=ParticleNo+1;
            particle(ParticleNo,1)=particle(1,1)+length;
            particle(ParticleNo,2)=particle(1,2);
            particle(ParticleNo,3)=particle(1,3)-thickness;
       end
       %width and thickness

       if (((particle(1,2)+crystalRadius)>width) &&
((particle(1,3)+crystalRadius)>thickness));
            ParticleNo=ParticleNo+1;
            particle(ParticleNo,1)=particle(1,1);
            particle(ParticleNo,2)=particle(1,2)-width;
            particle(ParticleNo,3)=particle(1,3)-thickness;
       end
       if (((particle(1,2)-crystalRadius)<0) && ((particle(1,3)-
crystalRadius)<0));
            ParticleNo=ParticleNo+1;
            particle(ParticleNo,1)=particle(1,1);
            particle(ParticleNo,2)=particle(1,2)+width;
            particle(ParticleNo,3)=particle(1,3)+thickness;
       end
       if (((particle(1,2)+crystalRadius)>width) && ((particle(1,3)-
crystalRadius)<0));
            ParticleNo=ParticleNo+1;
            particle(ParticleNo,1)=particle(1,1);
            particle(ParticleNo,2)=particle(1,2)-width;
            particle(ParticleNo,3)=particle(1,3)+thickness;
       end
       if (((particle(1,2)-crystalRadius)<0) &&
((particle(1,3)+crystalRadius)>thickness));
            ParticleNo=ParticleNo+1;
            particle(ParticleNo,1)=particle(1,1);
            particle(ParticleNo,2)=particle(1,2)+width;
            particle(ParticleNo,3)=particle(1,3)-thickness;
       end
       %length, width and thickness
        if
(((particle(1,1)+crystalRadius)>length)&&((particle(1,2)+crystalRadius)>wid
th) && ((particle(1,3)+crystalRadius)>thickness));
```

```
            ParticleNo=ParticleNo+1;
            particle(ParticleNo,1)=particle(1,1)-length;
            particle(ParticleNo,2)=particle(1,2)-width;
            particle(ParticleNo,3)=particle(1,3)-thickness;
        end
       if (((particle(1,1)-crystalRadius)<0)&& ((particle(1,2)-
crystalRadius)<0)&& ((particle(1,3)-crystalRadius)<0));
            ParticleNo=ParticleNo+1;
            particle(ParticleNo,1)=particle(1,1)+length;
            particle(ParticleNo,2)=particle(1,2)+width;
            particle(ParticleNo,3)=particle(1,3)+thickness;
       end

        if (((particle(1,1)+crystalRadius)>length)&&((particle(1,2)-
crystalRadius)<0) && ((particle(1,3)+crystalRadius)>thickness));
            ParticleNo=ParticleNo+1;
            particle(ParticleNo,1)=particle(1,1)-length;
            particle(ParticleNo,2)=particle(1,2)+width;
            particle(ParticleNo,3)=particle(1,3)-thickness;
        end
       if (((particle(1,1)-crystalRadius)<0)&&
((particle(1,2)+crystalRadius)>width)&& ((particle(1,3)-crystalRadius)<0));
            ParticleNo=ParticleNo+1;
            particle(ParticleNo,1)=particle(1,1)+length;
            particle(ParticleNo,2)=particle(1,2)-width;
            particle(ParticleNo,3)=particle(1,3)+thickness;
       end

        if
(((particle(1,1)+crystalRadius)>length)&&((particle(1,2)+crystalRadius)>wid
th) && ((particle(1,3)-crystalRadius)<0));
            ParticleNo=ParticleNo+1;
            particle(ParticleNo,1)=particle(1,1)-length;
            particle(ParticleNo,2)=particle(1,2)-width;
            particle(ParticleNo,3)=particle(1,3)+thickness;
        end
       if (((particle(1,1)-crystalRadius)<0)&& ((particle(1,2)-
crystalRadius)<0)&& ((particle(1,3)+crystalRadius)>thickness));
            ParticleNo=ParticleNo+1;
            particle(ParticleNo,1)=particle(1,1)+length;
            particle(ParticleNo,2)=particle(1,2)+width;
            particle(ParticleNo,3)=particle(1,3)-thickness;
       end
       if (((particle(1,1)+crystalRadius)>length)&&((particle(1,2)-
crystalRadius)<0) && ((particle(1,3)-crystalRadius)<0));
            ParticleNo=ParticleNo+1;
            particle(ParticleNo,1)=particle(1,1)-length;
            particle(ParticleNo,2)=particle(1,2)+width;
            particle(ParticleNo,3)=particle(1,3)+thickness;
       end
      if (((particle(1,1)-crystalRadius)<0)&&
((particle(1,2)+crystalRadius)>width)&&
((particle(1,3)+crystalRadius)>thickness));
            ParticleNo=ParticleNo+1;
            particle(ParticleNo,1)=particle(1,1)+length;
            particle(ParticleNo,2)=particle(1,2)-width;
            particle(ParticleNo,3)=particle(1,3)-thickness;
       end

        %Next check if the crystal(s) overlaps with an existing crystal. If
        %so reject.
        %for each particle, if x, y and z locations of the two particles
        %overlaps
        %particle();
        if seeds>0
            for i=1:ParticleNo
                a=particle(i,:);
                b=layer(:,1:3);
                %determine the separation of this particle with all others
                %in the layer
                separation=bsxfun(@minus,a,b);
                %convert this into a two scalar gaps as as crystals tend to
                %be horizontal
                a=separation(:,1).^2;
                b=separation (:,2).^2;
                c=separation(:,3).^2;
                minSeparation=(a+b+c).^.5;

                %determine if any of these gaps is less than the sum of the
                %radii
                combinedRadii=(bsxfun(@plus,layer(:,4),crystalRadius));
                testForGap=minSeparation-combinedRadii;
                %get the crystals that overlap, 6th column in the layer
                %array
                layer;
                overlapCrystals=layer((testForGap<0),6:7);
                overlapCrystals(:,3)=i;

overlapCrystalArray=vertcat(overlapCrystalArray,overlapCrystals);

                %this pulls out an array of negatives. If there is a
                %negative, then there is overlap.
                %testForGap=size(testForGap(find(testForGap<0)),1)

                Gap=0;

                for p=1:size(testForGap)
                    if testForGap(p)<0

                        Gap=Gap+1;
                    end
                end
                %allow for 20 overlaps before rejecting, the overlaps will
                %be tested in more detail when replacing spheres with
                %crystals
                if Gap>3
                    overlap=1;
                end
            end
        end
        if overlap==1
            failed=1;
%         else   %Else seed.
%             particle()
%             j=seeds;
%             for i=1:ParticleNo
%                 for q=1:3
%                 layer(i+j,q)=particle(i,q);
%                 end
%                 layer(i+j,4)=crystalRadius;
%                  layer(i+j,5)=boundingCylinderHeight;
%                  layer(i+j,6)=BoundingSphereNumber+ParticleNo;
%                  seeds=seeds+1;
%              end
        end
end
```

Hex Column

```
function [ crystalFile ] = hexcolumn( l, r, indent, crystalSelection,
obliqueEnds, obliqueBasalFacet,outputfile )
%l
%r
%indent
%create crystal data for a hexagonal colum

%Test input parameters
%crystalSelection = 'indentdoublecolumn';
%obliqueEnds =2;
%obliqueBasalFacet = 20;
%indent=.5;
%l=-1;
%r=1;
%outputfile='temptcrystal.txt'

crystalPoints = textread('allcolumns.points','%s', 'delimiter',
'\n','whitespace', '');

rtt = (3^.5)/2;
totalVertexes = 0;
vertexPerSide = 0;
totalFaces = 0;
quitOut=0;
v=0;
k=0;



%Read the file from the name given to the end point of the crystal. Convert
%the data into an array of faces.
for i=1:size(crystalPoints)
         if quitOut == 1;
            break
         end

    if strcmp(crystalPoints(i),crystalSelection)==1

        for (q=i:size(crystalPoints))
           if strcmp(crystalPoints(q),'n') == 1
            totalFaces = totalFaces+1;
            v=v+1;
            vertexPerSide(v)= str2double(crystalPoints(q+1));
            totalVertexes = totalVertexes + vertexPerSide(v);

            for (j = (q+2):(q+1+str2double(crystalPoints(q+1))))
                k=k+1;
                crystalVertexData(k,1:3) = str2double(crystalPoints(j));
            end
           end
            if strcmp(crystalPoints(q),'e') == 1
```

```
                quitOut=1;
                break
            end

        end
    end

end

%Set the locations of the various corners(points) of a hex column based on
the
%dimensions provided.



VertexData(13,3)= -indent;
VertexData(14,3) = -l + indent;

VertexData(1,1) = r;
VertexData(1,2) = 0;

VertexData(2,1) = r/2;
VertexData(2,2) = -rtt*r;

VertexData(3,1) = -r/2;
VertexData(3,2) = -rtt*r;

VertexData(4,1) = -r;
VertexData(4,2) = 0;

VertexData(5,1) = -r/2;
VertexData(5,2) = rtt*r;

VertexData(6,1) = r/2;
VertexData(6,2) = rtt*r;

VertexData(13,1) = 0;
VertexData(13,2) = 0;



for i = (1:6)

    VertexData(i + 6,1) = VertexData(i,1);

    VertexData(i + 6,2) = VertexData(i,2);

end


%data for the 13th and 14th corners are only needed where there are points
%or indents in the hex column ends
VertexData(14,1) = VertexData(13,1);

VertexData(14,2) = VertexData(13,2);
```

```
if obliqueEnds > 0

    for i = 1:6
        %tand as angle is in degrees
        VertexData(i,3) = 0 - ((VertexData(1,1) - 
VertexData(i,1))*tand(obliqueBasalFacet));

    end

    if obliqueEnds == 2

        for i = 7:12

            VertexData(i,3) = -l + ((VertexData(4,1) + 
VertexData(i,1))*tand(obliqueBasalFacet));

        end

    else

        for i = 7:12

            VertexData(i,3) = -l;

        end

    end

else

    for i = 1:6

        VertexData(i,3) = 0;

    end

    for i = 7:12

        VertexData(i,3) = -l;

    end

end

%Apply the vertex data(corners/points) for the crystal to the crystal
%points in order to generate the various faces and output to the file
%indicated. This generates a file of the individual corner points of the
%facets in the file.
output= fopen(outputfile,'w');
fprintf(output,'%d \r\n',totalFaces);
for i=(1:totalFaces)
    fprintf(output,'%d\r\n ',vertexPerSide(i));
end

for i=(1:size(crystalVertexData))
```

```
    fprintf(output,'%5.5f \t %5.5f \t %5.5f
\r\n',VertexData(crystalVertexData(i,1),1),VertexData(crystalVertexData(i,2
),2),VertexData(crystalVertexData(i,3),3));
    i = i+3;
end
fclose(output);
```

Add Crystal to Layer

```
 %This is a modified version of makelayer2. It takes the accepted crystal,
%rotates it and adds it to the layer as it is being assembLed. It may have
%to add multiple crystals as where a crystal crosses a boundary, a second
%crystal is added on the opposite side. As multiple boundaries can be
%crossed, multiple crystals can be seeded.
function
[NewNumberFacets,newlayerData,newlayerData2,newlayerVertexData,crystalVerte
xData,lineEquation,planeEquation,boundingSphereNumber,qtyCrystalsSeeded]=ad
dCrystaltoLayer4(particle,length,width,thickness,seeds,overlapCrystalArray)
%
[NewNumberFacets,newlayerData,newlayerData2,newlayerVertexData,crystalVerte
xData,lineEquation,planeEquation,boundingSphereNumber]=addCrystalToLayer3(p
article,handles.length,handles.width,handles.thickness,seeds,overlapCrystal
Array,boundingSphereNumber);

qtyCrystalsSeeded=0;
boundingSphereNumber=seeds+1;
newlayerData=zeros;
newlayerData2=zeros;
newlayerVertexData=zeros;
%this is the basic crystal to be seeded.
crystal=dlmread('seedCrystal.txt');
%crystal=dlmread('roundedHexD005H01f09f12.dat');


noFacets=crystal(1,1); %first numer in crystal file is number of facets
% each facet will have a number of vericies associated with it.  Number of
% verticies for nth facet given in row n+1
noVerticies=crystal(2:noFacets+1,1);
% Each vertex has x,y and z data in 3 colums after the list containing the
% number of verticies per facet
vertexData=crystal(noFacets+2:end,1:3);
%Centre of Gravity for the crystal
sp1=0;
sp2=0;
sp3=0;
n=0;
for r=(1:noFacets)
    for s=(1:noVerticies(r))
        n=n+1;
        sp1=vertexData(n,1)+sp1;
        sp2=vertexData(n,2)+sp2;
        sp3=vertexData(n,3)+sp3;
    end
end
sp1=sp1/n;
```

```
sp2=sp2/n;
sp3=sp3/n;

%get data on the seeding points for each crystal
%old method
    %layerfile=dlmread('mica.txt');
    %seedPointData=layerfile(1:end,1:3);
%new method
    seedPointData=particle;

%Determine total number of facets
%size(seedPointData,1) is the number of rows, i.e. the number of crystals
in
%the layer
% remove the; to generate the data to the command screen-note that only
% so much will be shown


%old method, now added during seeding

%NewNumberFacets=size(seedPointData,1)*noFacets;
% j=0;
% k=1;
% newlayerData2(1,1)=noFacets;
% for i=(1:NewNumberFacets)
%     k=k+1;
%     j=j+1;
%     if j>noFacets
%         j=1;
%         newlayerData2(k,1)=0;
%         k=k+1;
%         newlayerData2(k,1)=noFacets;
%         k=k+1;
%     end
%     newlayerData(i,1)=noVerticies(j);
%     %for layout2
%     newlayerData2(k,1)=noVerticies(j);
% end
% % remove the; to generate the data to the command screen-note that only
% % so much will be shown
% newlayerData2;
%


%take the x,y,z coordinates of seed point data and off-set each of the
%points for the crystal file by this amount and place the whole set in the
%new layer vertex data
for p=(1:size(seedPointData,1))
    vertexData=crystal(noFacets+2:end,1:3); %refresh vertex data each pass

    %it is at this point that we need to adjust the vertex data from its
    %original to a rotated crystal before updating the crystal.
    %Crystal Rotation
    alpha_euler=pi*rand*2;
    beta_euler=acos(1.0-2.0*rand);
    gamma_euler=pi*rand*2;

    s1=sin(alpha_euler);
    s2=sin(beta_euler);
    s3=sin(gamma_euler);
    c1=cos(alpha_euler);
    c2=cos(beta_euler);
    c3=cos(gamma_euler);

    r11=-c2*s1*s3+c1*c3;
    r12=-c2*s1*c3-c1*s3;
    r13=s2*s1;
    r21=c2*c1*s3+s1*c3;
    r22=c2*c1*c3-s1*s3;
    r23=-s2*c1;
    r31=s2*s3;
    r32=s2*c3;
    r33=c2;
    z=0;
    for r=(1:noFacets)
        for s=(1:noVerticies(r))
            z=z+1;
            tempVertexData(z,1)=vertexData(z,1)-sp1;
            tempVertexData(z,2)=vertexData(z,2)-sp2;
            tempVertexData(z,3)=vertexData(z,3)-sp3;
        end
    end
    z=0;
    for r=(1:noFacets)
        for s=(1:noVerticies(r))
            z=z+1;

vertexData(z,1)=tempVertexData(z,1)*r11+tempVertexData(z,2)*r12+tempVertexD
ata(z,3)*r13;

vertexData(z,2)=tempVertexData(z,1)*r21+tempVertexData(z,2)*r22+tempVertexD
ata(z,3)*r23;

vertexData(z,3)=tempVertexData(z,1)*r31+tempVertexData(z,2)*r32+tempVertexD
ata(z,3)*r33;
            %vertexData(z,4)=p;
        end
    end

end

%finish buggering about with the crystal file vertex data and apply
       %new crystal data to seedpoints for the crystal and store data
%newlayerVertexData=zeros;
%below needs to be updated so that particles which completely lie
       %outside the bounds of the box are completely removed. This means
       %writing to a temp array before then adding to the full array...
       %if all points are>max z or<min z reject
       %if all points are>max y or<min y reject
       %if all points are>max x or<min x reject


j=0;
k=1;
m=0;
n=0;
NewNumberFacets=0;
%newlayerData=zeros;
for p=(1:size(seedPointData,1))
    maxX=0;
    minX=width;
    maxY=0;
    minY=length;
    maxZ=0;
    minZ=thickness;

    for q=(1:size(vertexData,1));

        tempVertexData(q,1:3)=vertexData(q,1:3)+seedPointData(p,1:3);

        if maxX<tempVertexData(q,1)
            maxX=tempVertexData(q,1);
        end
        if minX>tempVertexData(q,1)
            minX=tempVertexData(q,1);
        end
        if maxY<tempVertexData(q,2)
            maxY=tempVertexData(q,2);
        end
        if minY>tempVertexData(q,2)
            minY=tempVertexData(q,2);
        end
        if maxZ<tempVertexData(q,3)
            maxZ=tempVertexData(q,3);
        end
        if minZ>tempVertexData(q,3)
            minZ=tempVertexData(q,3);
        end
    end

    %This bit removes any crystals that lie outside the layer
    if (maxX<=0) || ( maxY<=0) || (maxZ<=0) || (minX>=length) ||
(minY>=width) || (minZ>=thickness)
        %maxX           minX              maxY              minY
maxZ              minZ
        %maybe not remove this next line as in truth, the seeds
        %seeds=seeds-1
    else
        qtyCrystalsSeeded=qtyCrystalsSeeded+1;
        %adding facets
        NewNumberFacets=NewNumberFacets+noFacets;
        %adding vertex data
        %set below outside the loop
        %j=0;
        %k=1;
        if k==1
            newlayerData2(1,1)=noFacets;
        end
        for i=(1:noFacets)
            k=k+1;
            j=j+1;
            m=m+1;
            if j>noFacets
                j=1;
                newlayerData2(k,1)=0;
                k=k+1;
                newlayerData2(k,1)=noFacets;
                k=k+1;
            end
            newlayerData(m,1)=noVerticies(j);
            %for layout2
            newlayerData2(k,1)=noVerticies(j);
        end
        % remove the; to generate the data to the command screen-note that
only
        % so much will be shown
        newlayerData2;

        for q=(1:size(vertexData,1));

newlayerVertexData(((n)*(size(vertexData,1)))+q,1)=boundingSphereNumber;
            newlayerVertexData(((n)*(size(vertexData,1)))+q,2)=p;

newlayerVertexData(((n)*(size(vertexData,1)))+q,3:5)=vertexData(q,1:3)+seed
PointData(p,1:3);
        end

        n=n+1;
        for q=(1:size(vertexData,1));

crystalVertexData(q,1:3,n)=vertexData(q,1:3)+seedPointData(p,1:3);
        end
        % remove the; to generate the data to the command screen-note that
only
        % so much will be shown
        newlayerVertexData;

    end

end



%             for p=(1:size(seedPointData,1))
%                 for q=(1:size(vertexData,1));
%                     newlayerVertexData( ((p-
1)*(size(vertexData,1)))+q,1:3)=vertexData(q,1:3)+seedPointData(p,1:3);
%                 end
%             end




 %This section generates the crystal layer file. The first line is the
 %number of facets, then it is a loop for each facet and then the array of
 %vertex data. Currently being stored as crystalfile.txt

output=fopen('crystalfile.txt','w');
fprintf(output,'%d \r\n',NewNumberFacets);
for i=(1:size(newlayerData))
    fprintf(output,'%d\r\n ', newlayerData(i));
end

for i=(1:size(newlayerVertexData))
    fprintf(output,'%d\t %d \t %5.5f \t %5.5f \t %5.5f
\r\n',newlayerVertexData(i,1),newlayerVertexData(i,2),newlayerVertexData(i,
3:5));
end

fclose(output);
```

```
%generate data for the facet matrix-this is used to determine overlaps
%between crystals
% currentPositionInVertexData=0;
% for i=(1:NewNumberFacets)
%     facet(i,1)=boundingSphereNumber;
%     facet(i,2)=newlayerData(i);
%
%     for j=(1:newlayerData(i))
%         p=(j-1)*3;
%         for k=(1:3)
%
facet(i,2+p+k)=newlayerVertexData(currentPositionInVertexData+j,k);
%         end
%     end
%     currentPositionInVertexData=currentPositionInVertexData+j;
% end


%generate data for the line matrix-again, used for overlaps between
%crystals
% currentPositionInVertexData=0;
% q=1;
% for i=(1:NewNumberFacets)
%
%
%     for j=(1:newlayerData(i))
%         if j==newlayerData(i)
%             endpoint=currentPositionInVertexData+1;
%         else endpoint=currentPositionInVertexData+j+1;
%         end
%         for k=(1:3)
%             line(q,1)=boundingSphereNumber;
%
line(q,k+1)=newlayerVertexData(currentPositionInVertexData+j,k);
%             line(q,k+4)=newlayerVertexData(endpoint,k);
%         end
%         q=q+1;
%     end
%     currentPositionInVertexData=currentPositionInVertexData+j;
%end


%This is new stuff, essentially generating a matrix of the line functions
%and the facet planes rather than calculating these on the fly every time.

%generate data for the plane matrix-this is used to determine overlaps
%between crystals
currentPositionInVertexData=1;
planeEquation=sym([]);
size(overlapCrystalArray);
%if size(overlapCrystalArray)>0
    for i=(1:NewNumberFacets)
        planeEquation(i,1)=boundingSphereNumber;

planeEquation(i,2)=newlayerVertexData(currentPositionInVertexData,2);
        %get plane points for the facet from 3 points in the plane
        %P1 P2 P3
        P1=newlayerVertexData(currentPositionInVertexData,3:5);
        P2=newlayerVertexData(currentPositionInVertexData+1,3:5);
        P3=newlayerVertexData(currentPositionInVertexData+2,3:5);

        normal=cross(P1-P2, P1-P3);
        syms x y z;
        P=[x,y,z];
        planefunction=dot(normal, P-P1);
        planeEquation(i,3)=planefunction;
        %dot(P-P1, normal);
        %realdot=@(u, v) u*transpose(v);
        %realdot(P-P1,normal);


currentPositionInVertexData=currentPositionInVertexData+newlayerData(i);
    end
%end
planeEquation;
%generate data for the line Equation matrix-again, used for overlaps
between
%crystals. Maybe replace the above line matrix?

%Despite being only 8 facets to a hex column,this means that there are 36
%lines (6x4 + 8x2), even though some of the lines are effectively the same.

currentPositionInVertexData=0;
q=1;
lineEquation=sym([0,0,0,0]);
for i=(1:NewNumberFacets)

    for j=(1:newlayerData(i))
        if j==newlayerData(i)
            endpoint=currentPositionInVertexData+1;
        else endpoint=currentPositionInVertexData+j+1;
        end

        P4=newlayerVertexData(currentPositionInVertexData+j,3:5);
        P5=newlayerVertexData(endpoint,3:5);
        syms t;
        line2=P4+t*(P5-P4);
        line2(1,1);
        line2(1,2);
        line2(1,3);
        lineEquation(q,3:5)=line2(1,1:3);
        lineEquation(q,1)=boundingSphereNumber;

lineEquation(q,2)=newlayerVertexData(currentPositionInVertexData+j,2);
        q=q+1;
    end
    currentPositionInVertexData=currentPositionInVertexData+j;
end

%save crystal.mat
```

Show Crystals

```
 function h=showcrystals(fname,b,g)
```

```
%% Load and display a .crystal file in a MATLAB figure window

% h = figure handle

% fname = full path to .crystal file

% b = beta euler angle

% g = gamma euler angle

%

% Function written by Chris Stopford July 2009.  c.stopford@herts.ac.uk


% import data from .crystal file

if 1==2 % test!

    crystal=dlmread('l1_7_d13_7_flat.crystal');

    b=10;

    g=10;

else


crystal=dlmread(fname);

end

noFacets=crystal(1,1); %first numer in crystal file is number of facets

noPrismFacets=6;  % this is 6 for hexagonal columns

noBasalFacets=noFacets-noPrismFacets;

% each facet will have a number of vericies associated with it.  Number of
% verticies for nth facet given in row n+1

noVerticies=crystal(2:noFacets+1,1);

% Each vertex has x,y and z data in 3 colums after the list containing the

% number of verticies per facet

vertexData=crystal(noFacets+2:end,1:3);
```

```
% consider facet 1 first (for easy programming!)

individualFacet_noVerticies=noVerticies(1);

individualFacet_vertexStartPoint=1;

individualFacet_vertexEndPoint=noVerticies(1);

% select relevant vertex coordinate data:

individualFacet_vertexData=vertexData(individualFacet_vertexStartPoint:indi
vidualFacet_vertexEndPoint,:);



% plot facet

tri=delaunay(individualFacet_vertexData(:,1),individualFacet_vertexData(:,2
));

trisurf(tri,individualFacet_vertexData(:,1),individualFacet_vertexData(:,2)
,individualFacet_vertexData(:,3),'facecolor','blue','edgecolor','blue')

hold on

% add black wireframe - need to add first facet to the end to close edges

individualFacet_vertexData=[individualFacet_vertexData;individualFacet_vert
exData(1,:)];

plot3(individualFacet_vertexData(:,1),individualFacet_vertexData(:,2),indiv
idualFacet_vertexData(:,3),'black')

set(gca,'DataAspectRatio',[1 1 1]);



%%

for facet=2:noFacets

    % collect relevent data:

    individualFacet_noVerticies=noVerticies(facet);

    individualFacet_vertexStartPoint=sum(noVerticies(1:facet-1))+1;

    individualFacet_vertexEndPoint=sum(noVerticies(1:facet));



    % select relevant vertex coordinate data:
```

```
individualFacet_vertexData=vertexData(individualFacet_vertexStartPoint:indi
vidualFacet_vertexEndPoint,:);



    % plot facet

    x=vertexData(:,1);

    y=vertexData(:,2);

    z=vertexData(:,3);



    K = convhulln([x y z]);

    %Mica Comment - the line below colours everything blue!!!

%trisurf(K,x,y,z,'edgecolor','none','facecolor','blue','facealpha',0.1);

    %[X,Y]=meshgrid(x,y);

    %Z=griddata(x,y,z,X,Y);

    %surf(X,Y,Z)


tri=delaunay(individualFacet_vertexData(:,1),individualFacet_vertexData(:,2
));


trisurf(tri,individualFacet_vertexData(:,1),individualFacet_vertexData(:,2)
,individualFacet_vertexData(:,3),'facecolor','blue','edgecolor','blue')


    % add black wireframe - need to add first facet to the end to close
edges


individualFacet_vertexData=[individualFacet_vertexData;individualFacet_vert
exData(1,:)];

h=plot3(individualFacet_vertexData(:,1),individualFacet_vertexData(:,2),ind
ividualFacet_vertexData(:,3),'black');
end
```

Sandbox (Tetris)

```
function varargout=Sandbox_Tetris(varargin)


handles.length=2;
handles.width=2;
handles.thickness=2;
handles.failureBreakOut=10;
handles.outputFileName='seedCrystal.txt';
handles.L_mean=0.3;
handles.L_deviation=0.1;
handles.R_mean=0.05;
handles.R_deviation=0.01;
handles.aveIndent=0.0;
handles.crystalDensity=0.9167;
% case 'Hex Columns Only'
    handles.crystalSeeds=1;
% case 'Hex/Single Indents'
%     handles.crystalSeeds=2;
% case 'Hex/Sing/Doub'
%    handles.crystalSeeds=3;
handles.obliqueBasalFacet=0;

rejections=0;
uniqueCrystals=0;
totalCrystals=0;
layer=zeros;
seeds=0;
crystalNo=0;
emergingArrayFacets=0;
emergingArrayVertex=zeros(3,0);
emergingArrayVertexNewStructure=zeros(3,0);
emergingArrayVertexLocation=zeros(3,0);
totalVolume=0;
density=0;
planeMatrix=zeros(0,0);
lineMatrix=zeros(0,0);
lineEquationMatrix=sym([]);
%convrejectiions=0
doprinting=0;
%while(rejections<handles.failureBreakOut);
%replaced failure breakout with density - this will be modified so that
%this can be an input. This is only reasonable once seeding can regularly
%get below the densities that are present in ice (see Kaasalanian).
while (density < .08)
%while (seeds < 30)
    %spawn random crystal dimensions - choosing the longest for overall
    %diameter
    crystalLength=normrnd(handles.L_mean,handles.L_deviation);
    crystalRadius=normrnd(handles.R_mean,handles.R_deviation);

    %Set bounding sphere limits - hypotenuse of half length and radius
    seedRadius=( (crystalLength/2)^2 + crystalRadius^2 )^0.5 ;

    %Attempt to seed this diameter object into the layer also seed
    %duplicates where the object crosses a boundary


[layer,failed,ParticleNo,particle,overlapCrystalArray]=assembleLayer_Tetris
(handles.length,handles.width,handles.thickness,seedRadius,layer,crystalRad
ius,seeds);

   if failed==1
       rejections=rejections+1;
   else
       %create crystal
       createHexCrystal(crystalLength,crystalRadius,handles);
       %rotate crystal and its duplicates and create their absolute
       %position
```

```
[NewNumberFacets,newlayerData,newlayerData2,newlayerVertexData,crystalVerte
xData,lineEquation,planeEquation,boundingSphereNumber,qtyCrystalsSeeded]=ad
dCrystalToLayer_Tetris(particle,handles.length,handles.width,handles.thickn
ess,seeds,overlapCrystalArray);
        %Convex hull check. This creates a matrix of line equations
        %corresponding to the overlapping bounding spheres. These are then
        %converted into an array of intersection points against all the
        %facets of the the seeding crystal. Finally there is a check to
        %determine if any of these are within a bounding convex hull of the
        %seeding crystals
        hullRejection=0;
        if size(overlapCrystalArray)>0
            overlapCrystalArray=unique(overlapCrystalArray,'rows')
            intersect=zeros(0,0);
            intersectArray=zeros(0,0);
            syms x y z t;
            P = [x,y,z];
            tic;
            fprintf('\n\nOverlap Equation Matrix: ')

vi=ismember(lineEquationMatrix(:,1:2),overlapCrystalArray(:,1:2),'rows');
            potentialOverlapLineEquationMatrix=lineEquationMatrix(vi,:);
            toc
            planeEquation;
            tic;
            fprintf('\nPlane Equation: ')
            for q=(1:size(overlapCrystalArray))
                for p=(1:size(planeEquation))
                    for j=(1:size(potentialOverlapLineEquationMatrix))
                        %fprintf('do they equal? %s,
%s',potentialOverlapLineEquationMatrix(j,3),planeEquation(p,2));
                        if
(overlapCrystalArray(q,1)==potentialOverlapLineEquationMatrix(j,1))...
                                &&
(overlapCrystalArray(q,2)==potentialOverlapLineEquationMatrix(j,2))...
                                &&
(overlapCrystalArray(q,3)==planeEquation(p,2))
                            %fprintf('\n yeah, got here')

newfunction=subs(planeEquation(p,3),P,potentialOverlapLineEquationMatrix(j,
3:5));
                            t0=solve(newfunction);

point1=subs(potentialOverlapLineEquationMatrix(j,3:5),t,t0);
                            point1=double(point1);
                            intersectArray=vertcat(intersectArray,point1);
                        end
                        if size(intersectArray)>0
                            [d1,d2,d3]=size(crystalVertexData);
                            for i=(1:d3)

intersect=inhull(intersectArray,crystalVertexData(:,:,i));
                                if any(intersect)==1
                                    hullRejection=1;
                                    disp('   Rejected by Convex Hull Test')
                                    break
                                end
                            end
                        end
                        if hullRejection==1;
                            break
                        end
                    end
                    if hullRejection==1;
                        break
                    end
                end
                if hullRejection==1;
                    break
                end
            end
            toc

%As above though now lines and facets are reversed (needed for large
facets)
%Test all the lines generated by the add crystal to layer against all
%facets already seeded Might not need this step...

% Creates a facets corresponding to the overlapping bounding spheres. These
are then
% tested against the line matrix of the seeds to create intersection
points.
% The points are then tested against the convex hulls of the previously
seeded crystals
       if hullRejection==0
            intersect=zeros(0,0);
            intersectArray=zeros(0,0);
            tic;
            fprintf('\n\nOverlap Plane Matrix (Reciprocation): ')

vi=ismember(planeMatrix(:,1:2),overlapCrystalArray(:,1:2),'rows');
            potentialOverlapPlaneMatrix=planeMatrix(vi,:);
            toc
            tic;
            fprintf('\nLine Equation (Reciprocation): ')
            for q=(1:size(overlapCrystalArray))
                for j=(1:size(potentialOverlapPlaneMatrix))
                    for p=(1:size(lineEquation))
                        if
(overlapCrystalArray(q,1)==potentialOverlapPlaneMatrix(j,1))...
                                &&
(overlapCrystalArray(q,2)==potentialOverlapPlaneMatrix(j,2))...
                                &&
(overlapCrystalArray(q,3)==lineEquation(p,2))
                            %fprintf('\n yeah, got here')

newfunction=subs(potentialOverlapPlaneMatrix(j,3),P,lineEquation(p,3:5));
                            t0=solve(newfunction);
                            point1=subs(lineEquation(p,3:5),t,t0);
                            point1=double(point1);
                            intersectArray=vertcat(intersectArray,point1);
                        end
                    end
                end
            end
            size(intersectArray)
            toc
            %Extract all the vertex data for the seeded crystals to test
against these
            %intersect points.
            tic
           fprintf('\nAssemble array of vertices from existing vertice
matrix: ')

vi=ismember(emergingArrayVertexLocation(:,1:2),overlapCrystalArray(:,1:2),'
rows');

potentialOverlapArrayVertexLocation=emergingArrayVertexLocation(vi,:);
           c=1;
           b=1;
           testArrayOfCrystals=zeros;
           for y=1:size(potentialOverlapArrayVertexLocation)
              if y>1
                  if
potentialOverlapArrayVertexLocation(y,1)~=potentialOverlapArrayVertexLocati
on(y-1,1)...
                  ||
potentialOverlapArrayVertexLocation(y,2)~=potentialOverlapArrayVertexLocati
on(y-1,2);
                    c=1;
                    b=b+1;
                  end
              end

testArrayOfCrystals(c,1:3,b)=potentialOverlapArrayVertexLocation(y,3:5);
              c=c+1;
           end
           toc
           tic
           fprintf('\nConvex Hull (Reciprocation): ')
           testArrayOfCrystals;
           if size(intersectArray)>0
               [d1,d2,d3]=size(testArrayOfCrystals);
               for i=(1:d3)

intersect=inhull(intersectArray,testArrayOfCrystals(:,:,i));
                   if any(intersect)==1
                       hullRejection=1;
                       disp('   Rejected by Convex Hull Test
(Reciprocation): ')
                       break
                   end
               end
           end
           toc
       end
       end

       if hullRejection==0;

           %from assemble layer
            j=seeds;
            for i=1:ParticleNo
                for q=1:3
                layer(i+j,q)=particle(i,q);
                end
                layer(i+j,4)=crystalRadius;
                %layer(i+j,5)=boundingCylinderHeight; oldcrap
                layer(i+j,5)=crystalRadius;
                layer(i+j,6)=boundingSphereNumber;
                layer(i+j,7)=i;
                seeds=seeds+1;
            end



           %Update volume - only seed once for duplicate particles as the
parts
           %within the layer boundaries will only account for 1 complete
           %crystal
           crystalVolume=3/2*3^.5*crystalRadius^2*crystalLength;
           totalVolume=totalVolume+crystalVolume;
           totalCrystals=totalCrystals+qtyCrystalsSeeded;
           uniqueCrystals=uniqueCrystals+1;

density=totalVolume*handles.crystalDensity/(handles.length*handles.width*ha
ndles.thickness);
           %add these duplicates to the emerging array
           emergingArrayFacets=emergingArrayFacets+NewNumberFacets;
           emergingArrayVertex=vertcat(emergingArrayVertex,newlayerData);
           planeMatrix=vertcat(planeMatrix,planeEquation);
           %lineMatrix=vertcat(lineMatrix,line);
           lineEquationMatrix=vertcat(lineEquationMatrix,lineEquation);
           %add new structure to create a carriage return

emergingArrayVertexNewStructure=vertcat(emergingArrayVertexNewStructure,new
layerData2);

emergingArrayVertexNewStructure=vertcat(emergingArrayVertexNewStructure,0);

emergingArrayVertexLocation=vertcat(emergingArrayVertexLocation,newlayerVer
texData);
       else
           rejections=rejections+1;
       end
   end
%move this down so it does the below step every time
%end

doprinting=doprinting+1;
if doprinting==5;
    doprinting=0;
end
if doprinting==0

%output all Emerging Array information to a file
output= fopen('crystalLayerFile_ReadMe.txt','w');
fprintf(output,'Layer dimensions (%dx%dx%d)mm^3
\r\n',handles.length,handles.width,handles.thickness);
%fprintf(output,'Layer width %d \r\n',handles.width);
%fprintf(output,'Layer thickness %d \r\n',handles.thickness);
fprintf(output,'rejections %d \r\n',handles.failureBreakOut);
fprintf(output,'Crystals Seeded %d\r\n',totalCrystals);
fprintf(output,'of which Unique %d\r\n',uniqueCrystals);
fprintf(output,'Hex crystals \r\n');
fprintf(output,'Crystal length %3.3f +/- %3.4f
\r\n',handles.L_mean,handles.L_deviation);
fprintf(output,'Crystal radius %3.3f +/- %3.4f
\r\n',handles.R_mean,handles.R_deviation);
fprintf(output,'Average indent(of length) %d%% \r\n',handles.aveIndent);
fprintf(output,'Total Crystal Volume %5.5f \r\n',totalVolume);
fprintf(output,'Crystal density %5.5f \r\n',handles.crystalDensity);
fprintf(output,'Layer Density %5.5f \r\n',density);
fprintf(output,'Location (x \t y \t\t z) \t\t Sphere rad \t Seed \t
Particle no. \r\n');
for i=(1:size(layer))
    fprintf(output,'%5.5f \t %5.5f \t %5.5f \t %5.5f \t %d \t %d
\r\n',layer(i,1),layer(i,2),layer(i,3),layer(i,4),layer(i,6),layer(i,7));
end
fclose(output);
%whos
%save bobsybob.mat

%output all Emerging Array to a file
output=fopen('crystalLayerFile.txt','w');
fprintf(output,'%d \r\n',emergingArrayFacets);
for i=(1:size(emergingArrayVertex))
    fprintf(output,'%d\r\n ',emergingArrayVertex(i));
end
for i=(1:size(emergingArrayVertexLocation))
    fprintf(output,'%5.5f \t %5.5f \t %5.5f
\r\n',emergingArrayVertexLocation(i,3),emergingArrayVertexLocation(i,4),eme
rgingArrayVertexLocation(i,5));
end
fclose(output);



%output all Emerging Array to a file - with new structure
output=fopen('crystalLayerFile2.txt','w');
fprintf(output,'%d \r\n',emergingArrayFacets);
fprintf(output,'%d \r\n',totalCrystals);
insertReturn=1;
for i=(1:size(emergingArrayVertexNewStructure))
    if insertReturn==1
        fprintf(output,'%d \r\n',emergingArrayVertexNewStructure(i));
        insertReturn=0;
    elseif emergingArrayVertexNewStructure(i)==0
        fprintf(output,'\r\n');
        insertReturn=1;
    else
        fprintf(output,'%d ',emergingArrayVertexNewStructure(i));
    end
end
for i=(1:size(emergingArrayVertexLocation))
    fprintf(output,'%5.5f \t %5.5f \t %5.5f
\r\n',emergingArrayVertexLocation(i,3),emergingArrayVertexLocation(i,4),eme
rgingArrayVertexLocation(i,5));
end
fclose(output);
%output all Emerging Array to a file - with new structure
output=fopen('crystalLayerFile3.txt','w');
fprintf(output,'%d \r\n',emergingArrayFacets);
fprintf(output,'%d \r\n',totalCrystals);
insertReturn=1;
for i=(1:size(emergingArrayVertexNewStructure))
    if insertReturn==1
        fprintf(output,'%d \r\n',emergingArrayVertexNewStructure(i));
        insertReturn=0;
    elseif emergingArrayVertexNewStructure(i)==0
        fprintf(output,'\r\n');
        insertReturn=1;
    else
        fprintf(output,'%d ',emergingArrayVertexNewStructure(i));
    end
end

for i=(1:size(emergingArrayVertexLocation))
    fprintf(output,'%5.5f \t %5.5f \t %5.5f \t %5.5f \t %5.5f
\r\n',emergingArrayVertexLocation(i,1),emergingArrayVertexLocation(i,2),eme
rgingArrayVertexLocation(i,3),emergingArrayVertexLocation(i,4),emergingArra
yVertexLocation(i,5));
end
fclose(output);



%output all FacetMatrix to a file
% output=fopen('planeMatrix.txt','w');
% for i=(1:size(planeMatrix))
%     fprintf(output,'%5.5f \t %5.5f \t %5.5f \t %5.5f \t %5.5f \t %5.5f \t
%5.5f \t %5.5f \t %5.5f \t %5.5f \t %5.5f \t %5.5f \t %5.5f \t %5.5f \t
%5.5f \t %5.5f \t %5.5f \t %5.5f \t %5.5f \t %5.5f \r\n',
planeMatrix(i,1:20));
% end
% fclose(output);

%output all lineMatrix to a file
output=fopen('lineEquationMatrix.txt','w');
outputLineEquation=char(lineEquationMatrix);
for i=(1:size(outputLineEquation))
    fprintf(output,'%s \t %s \t %s \t %s \r\n',outputLineEquation(i,1:4));
end
fclose(output);

%end doprinting
end
%end moved to here
end

function createHexCrystal(crystalLength,crystalRadius,handles,hObject,
eventdata)
%indent=handles.maxIndent;
indent =
normrnd((handles.aveIndent/2),(handles.aveIndent/6))*crystalLength/100;
crystalType = randi(handles.crystalSeeds);
if crystalType==1
    crystalType='standardcolumn';
elseif crystalType==2
    crystalType='indentsinglecolumn';
else crystalType='indentdoublecolumn';
end
obliqueEnds=0;%randi(1,1,2)+1;

obliqueBasalFacet=0;%randi(1,1,handles.obliqueBasalFacet);
hexcolumn_tetris( crystalLength,
crystalRadius,indent,crystalType,obliqueEnds,obliqueBasalFacet,handles.outp
utFileName );
```

Assemble Layer (Tetris)

```
 function
[layer,failed,ParticleNo,particle,overlapCrystalArray]=assembleLayer_Tetris
(length,width,thickness,crystalRadius,layer,trueRadius,seeds)

%[layer,failed,ParticleNo,particle,overlapCrystalArray]=assembleLayer3(hand
les.length,handles.width,handles.thickness,seedRadius,layer,crystalRadius);

       failed=0;
       failed_due_to_overlap=1;

       %What we need to do for Tetris keep moving the particle up until it
no longer overlaps.
       %This is done by increasing the z value of the crystal until it no
       %longer causes an overlap. The upward movement is 1/20th of the
       %thickness

       %Set random location of particle to be seeded
        particle(1,1)=(rand*length+eps); %-(crystalRadius/2));
        particle(1,2)=(rand*width+eps); %+(crystalRadius/2));
        %particle(1,3)=(rand*thickness+eps-(crystalRadius/2));
        particle(1,3)=0

    while failed_due_to_overlap==1 %&& particle(1,3)<thickness)
        failed_due_to_overlap=0
        overlapCrystals=zeros(0,0);
        overlapCrystalArray=zeros(0,0);
        ParticleNo=1;
        particle=particle(1,1:3)
        if particle(1,3)==0
            particle(1,3)=eps
        else
            particle(1,3)=particle(1,3)+(thickness/20)
        end
        %Determine if the crystal overlaps the edges, if so, spawn another
        %crystal shifted over to the opposite side.
        if ((particle(1,1)+crystalRadius)>length);
            ParticleNo=ParticleNo+1;
            particle(ParticleNo,1)=particle(1,1)-length;
            particle(ParticleNo,2)=particle(1,2);
            particle(ParticleNo,3)=particle(1,3);
        end
       if ((particle(1,1)-crystalRadius)<0);
            ParticleNo=ParticleNo+1;
            particle(ParticleNo,1)=particle(1,1)+length;
            particle(ParticleNo,2)=particle(1,2);
            particle(ParticleNo,3)=particle(1,3);
       end
       if ((particle(1,2)+crystalRadius)>width);
            ParticleNo=ParticleNo+1;
            particle(ParticleNo,1)=particle(1,1);
            particle(ParticleNo,2)=particle(1,2)-width;
            particle(ParticleNo,3)=particle(1,3);
       end
       if ((particle(1,2)-crystalRadius)<0);
            ParticleNo=ParticleNo+1;
            particle(ParticleNo,1)=particle(1,1);
            particle(ParticleNo,2)=particle(1,2)+width;
            particle(ParticleNo,3)=particle(1,3);
       end
```

```
%        if ((particle(1,3)+crystalRadius)>thickness);
%             ParticleNo=ParticleNo+1;
%             particle(ParticleNo,1)=particle(1,1);
%             particle(ParticleNo,2)=particle(1,2);
%             particle(ParticleNo,3)=particle(1,3)-thickness;
%        end
%        if ((particle(1,3)-crystalRadius)<0);
%             ParticleNo=ParticleNo+1;
%             particle(ParticleNo,1)=particle(1,1);
%             particle(ParticleNo,2)=particle(1,2);
%             particle(ParticleNo,3)=particle(1,3)+thickness;
%        end
      %Where it overlaps two edges, another diagonal particle needs to be
      %added

      %length and width
        if (((particle(1,1)+crystalRadius)>length) &&
((particle(1,2)+crystalRadius)>width));
            ParticleNo=ParticleNo+1;
            particle(ParticleNo,1)=particle(1,1)-length;
            particle(ParticleNo,2)=particle(1,2)-width;
            particle(ParticleNo,3)=particle(1,3);
        end
       if (((particle(1,1)-crystalRadius)<0) && ((particle(1,2)-
crystalRadius)<0));
            ParticleNo=ParticleNo+1;
            particle(ParticleNo,1)=particle(1,1)+length;
            particle(ParticleNo,2)=particle(1,2)+width;
            particle(ParticleNo,3)=particle(1,3);
       end
       if (((particle(1,1)-crystalRadius)<0) &&
((particle(1,2)+crystalRadius)>width));
            ParticleNo=ParticleNo+1;
            particle(ParticleNo,1)=particle(1,1)+length;
            particle(ParticleNo,2)=particle(1,2)-width;
            particle(ParticleNo,3)=particle(1,3);
       end

        if (((particle(1,1)+crystalRadius)>length) && ((particle(1,2)-
crystalRadius)<0));
            ParticleNo=ParticleNo+1;
            particle(ParticleNo,1)=particle(1,1)-length;
            particle(ParticleNo,2)=particle(1,2)+width;
            particle(ParticleNo,3)=particle(1,3);
        end



       %length and thickness
%         if (((particle(1,1)+crystalRadius)>length) &&
((particle(1,3)+crystalRadius)>thickness));
%             ParticleNo=ParticleNo+1;
%             particle(ParticleNo,1)=particle(1,1)-length;
%             particle(ParticleNo,2)=particle(1,2);
%             particle(ParticleNo,3)=particle(1,3)-thickness;
%         end
%        if (((particle(1,1)-crystalRadius)<0) && ((particle(1,3)-
crystalRadius)<0));
%             ParticleNo=ParticleNo+1;
%             particle(ParticleNo,1)=particle(1,1)+length;
%             particle(ParticleNo,2)=particle(1,2);
%             particle(ParticleNo,3)=particle(1,3)+thickness;
%         end
%
%         if (((particle(1,1)+crystalRadius)>length) && ((particle(1,3)-
crystalRadius)<0));
%             ParticleNo=ParticleNo+1;
%             particle(ParticleNo,1)=particle(1,1)-length;
%             particle(ParticleNo,2)=particle(1,2);
%             particle(ParticleNo,3)=particle(1,3)+thickness;
%          end
%         if (((particle(1,1)-crystalRadius)<0) &&
((particle(1,3)+crystalRadius)>thickness));
%             ParticleNo=ParticleNo+1;
%             particle(ParticleNo,1)=particle(1,1)+length;
%             particle(ParticleNo,2)=particle(1,2);
%             particle(ParticleNo,3)=particle(1,3)-thickness;
%         end
%         %width and thickness
%
%         if (((particle(1,2)+crystalRadius)>width) &&
((particle(1,3)+crystalRadius)>thickness));
%             ParticleNo=ParticleNo+1;
%             particle(ParticleNo,1)=particle(1,1);
%             particle(ParticleNo,2)=particle(1,2)-width;
%             particle(ParticleNo,3)=particle(1,3)-thickness;
%         end
%         if (((particle(1,2)-crystalRadius)<0) && ((particle(1,3)-
crystalRadius)<0));
%             ParticleNo=ParticleNo+1;
%             particle(ParticleNo,1)=particle(1,1);
%             particle(ParticleNo,2)=particle(1,2)+width;
%             particle(ParticleNo,3)=particle(1,3)+thickness;
%         end
%         if (((particle(1,2)+crystalRadius)>width) && ((particle(1,3)-
crystalRadius)<0));
%             ParticleNo=ParticleNo+1;
%             particle(ParticleNo,1)=particle(1,1);
%             particle(ParticleNo,2)=particle(1,2)-width;
%             particle(ParticleNo,3)=particle(1,3)+thickness;
%         end
%         if (((particle(1,2)-crystalRadius)<0) &&
((particle(1,3)+crystalRadius)>thickness));
%             ParticleNo=ParticleNo+1;
%             particle(ParticleNo,1)=particle(1,1);
%             particle(ParticleNo,2)=particle(1,2)+width;
%             particle(ParticleNo,3)=particle(1,3)-thickness;
%         end
%         %length, width and thickness
%          if
(((particle(1,1)+crystalRadius)>length)&&((particle(1,2)+crystalRadius)>wid
th) && ((particle(1,3)+crystalRadius)>thickness));
%             ParticleNo=ParticleNo+1;
%             particle(ParticleNo,1)=particle(1,1)-length;
%             particle(ParticleNo,2)=particle(1,2)-width;
%             particle(ParticleNo,3)=particle(1,3)-thickness;
%          end
%         if (((particle(1,1)-crystalRadius)<0)&& ((particle(1,2)-
crystalRadius)<0)&& ((particle(1,3)-crystalRadius)<0));
%             ParticleNo=ParticleNo+1;
%             particle(ParticleNo,1)=particle(1,1)+length;
%             particle(ParticleNo,2)=particle(1,2)+width;
%             particle(ParticleNo,3)=particle(1,3)+thickness;
%        end
%
%         if (((particle(1,1)+crystalRadius)>length)&&((particle(1,2)-
crystalRadius)<0) && ((particle(1,3)+crystalRadius)>thickness));
%             ParticleNo=ParticleNo+1;
%             particle(ParticleNo,1)=particle(1,1)-length;
%             particle(ParticleNo,2)=particle(1,2)+width;
%             particle(ParticleNo,3)=particle(1,3)-thickness;
%         end
%        if (((particle(1,1)-crystalRadius)<0)&&
((particle(1,2)+crystalRadius)>width)&& ((particle(1,3)-crystalRadius)<0));
%             ParticleNo=ParticleNo+1;
%             particle(ParticleNo,1)=particle(1,1)+length;
%             particle(ParticleNo,2)=particle(1,2)-width;
%             particle(ParticleNo,3)=particle(1,3)+thickness;
%        end
%
%         if
(((particle(1,1)+crystalRadius)>length)&&((particle(1,2)+crystalRadius)>wid
th) && ((particle(1,3)-crystalRadius)<0));
%             ParticleNo=ParticleNo+1;
%             particle(ParticleNo,1)=particle(1,1)-length;
%             particle(ParticleNo,2)=particle(1,2)-width;
%             particle(ParticleNo,3)=particle(1,3)+thickness;
%         end
%        if (((particle(1,1)-crystalRadius)<0)&& ((particle(1,2)-
crystalRadius)<0)&& ((particle(1,3)+crystalRadius)>thickness));
%             ParticleNo=ParticleNo+1;
%             particle(ParticleNo,1)=particle(1,1)+length;
%             particle(ParticleNo,2)=particle(1,2)+width;
%             particle(ParticleNo,3)=particle(1,3)-thickness;
%        end
%        if (((particle(1,1)+crystalRadius)>length)&&((particle(1,2)-
crystalRadius)<0) && ((particle(1,3)-crystalRadius)<0));
%             ParticleNo=ParticleNo+1;
%             particle(ParticleNo,1)=particle(1,1)-length;
%             particle(ParticleNo,2)=particle(1,2)+width;
%             particle(ParticleNo,3)=particle(1,3)+thickness;
%        end
%       if (((particle(1,1)-crystalRadius)<0)&&
((particle(1,2)+crystalRadius)>width)&&
((particle(1,3)+crystalRadius)>thickness));
%             ParticleNo=ParticleNo+1;
%             particle(ParticleNo,1)=particle(1,1)+length;
%             particle(ParticleNo,2)=particle(1,2)-width;
%             particle(ParticleNo,3)=particle(1,3)-thickness;
%        end

        %Next check if the crystal(s) overlaps with an existing crystal. If
        %so reject.

        %for each particle, if x, y and z locations of the two particles
        %overlaps
        %particle();
        if seeds>0
            Overlapping=0;
            for i=1:ParticleNo
                if Overlapping>0
                    a=particle(i,:);
                    b=layer(:,1:3);
                    %determine the separation of this particle with all
others
                    %in the layer
                    separation=bsxfun(@minus,a,b);
                    %convert this into a two scalar gaps as as crystals
tend to
                    %be horizontal
                    a=separation(:,1).^2;
                    b=separation (:,2).^2;
                    c=separation(:,3).^2;
                    minSeparation=(a+b+c).^.5;

                    %determine if any of these gaps is less than the sum of
the
                    %radii
                    combinedRadii=(bsxfun(@plus,layer(:,4),crystalRadius));
                    testForGap=minSeparation-combinedRadii;
                    %get the crystals that overlap, 6th column in the layer
                    %array
                    layer;
                    overlapCrystals=layer((testForGap<0),6:7);
                    overlapCrystals(:,3)=i;

overlapCrystalArray=vertcat(overlapCrystalArray,overlapCrystals);

                    %this pulls out an array of negatives. If there is a
                    %negative, then there is overlap.
                    %testForGap=size(testForGap(find(testForGap<0)),1)
                    for p=1:size(testForGap)
                        if testForGap(p)<0,Overlapping=Overlapping+1;end
                    end
                else
                    failed_due_to_overlap=1
                end
            end
           %if failed_due_to_overlap==1,break,end
        end
    end
         %if failed_due_to_overlap==1,failed=1,end
end
```

Add Crystal to Layer (Tetris)

```
%This is a modified version of makelayer2. It takes the accepted crystal,
%rotates it and adds it to the layer as it is being assembLed. It may have
%to add multiple crystals as where a crystal crosses a boundary, a second
%crystal is added on the opposite side. As multiple boundaries can be
%crossed, multiple crystals can be seeded.
function
[NewNumberFacets,newlayerData,newlayerData2,newlayerVertexData,crystalVerte
xData,lineEquation,planeEquation,boundingSphereNumber,qtyCrystalsSeeded]=ad
dCrystaltoLayer4(particle,length,width,thickness,seeds,overlapCrystalArray)
%
[NewNumberFacets,newlayerData,newlayerData2,newlayerVertexData,crystalVerte
xData,lineEquation,planeEquation,boundingSphereNumber]=addCrystalToLayer3(p
article,handles.length,handles.width,handles.thickness,seeds,overlapCrystal
Array,boundingSphereNumber);

qtyCrystalsSeeded=0;
boundingSphereNumber=seeds+1;
```

```
newlayerData=zeros;
newlayerData2=zeros;
newlayerVertexData=zeros;
%this is the basic crystal to be seeded.
crystal=dlmread('seedCrystal.txt');
%crystal=dlmread('roundedHexD005H01f09f12.dat');


noFacets=crystal(1,1); %first numer in crystal file is number of facets
% each facet will have a number of vericies associated with it.  Number of
% verticies for nth facet given in row n+1
noVerticies=crystal(2:noFacets+1,1);
% Each vertex has x,y and z data in 3 colums after the list containing the
% number of verticies per facet
vertexData=crystal(noFacets+2:end,1:3);
%Centre of Gravity for the crystal
sp1=0;
sp2=0;
sp3=0;
n=0;
for r=(1:noFacets)
    for s=(1:noVerticies(r))
        n=n+1;
        sp1=vertexData(n,1)+sp1;
        sp2=vertexData(n,2)+sp2;
        sp3=vertexData(n,3)+sp3;
    end
end
sp1=sp1/n;
sp2=sp2/n;
sp3=sp3/n;

%get data on the seeding points for each crystal
%old method
    %layerfile=dlmread('mica.txt');
    %seedPointData=layerfile(1:end,1:3);
%new method
    seedPointData=particle;

%Determine total number of facets
%size(seedPointData,1) is the number of rows, i.e. the number of crystals
in
%the layer
% remove the; to generate the data to the command screen-note that only
% so much will be shown



%old method, now added during seeding

%NewNumberFacets=size(seedPointData,1)*noFacets;
% j=0;
% k=1;
% newlayerData2(1,1)=noFacets;
% for i=(1:NewNumberFacets)
%     k=k+1;
%     j=j+1;
%     if j>noFacets
%         j=1;
%         newlayerData2(k,1)=0;
%         k=k+1;
%          newlayerData2(k,1)=noFacets;
%          k=k+1;
%      end
%      newlayerData(i,1)=noVerticies(j);
%      %for layout2
%      newlayerData2(k,1)=noVerticies(j);
% end
% % remove the; to generate the data to the command screen-note that only
% % so much will be shown
% newlayerData2;
%



%take the x,y,z coordinates of seed point data and off-set each of the
%points for the crystal file by this amount and place the whole set in the
%new layer vertex data
for p=(1:size(seedPointData,1))
    vertexData=crystal(noFacets+2:end,1:3); %refresh vertex data each pass

    %it is at this point that we need to adjust the vertex data from its
    %original to a rotated crystal before updating the crystal.
    %Crystal Rotation
    alpha_euler=pi*rand*2;
    beta_euler=acos(1.0-2.0*rand);
    gamma_euler=pi*rand*2;

    s1=sin(alpha_euler);
    s2=sin(beta_euler);
    s3=sin(gamma_euler);
    c1=cos(alpha_euler);
    c2=cos(beta_euler);
    c3=cos(gamma_euler);

    r11=-c2*s1*s3+c1*c3;
    r12=-c2*s1*c3-c1*s3;
    r13=s2*s1;
    r21=c2*c1*s3+s1*c3;
    r22=c2*c1*c3-s1*s3;
    r23=-s2*c1;
    r31=s2*s3;
    r32=s2*c3;
    r33=c2;
    z=0;
    for r=(1:noFacets)
        for s=(1:noVerticies(r))
            z=z+1;
            tempVertexData(z,1)=vertexData(z,1)-sp1;
            tempVertexData(z,2)=vertexData(z,2)-sp2;
            tempVertexData(z,3)=vertexData(z,3)-sp3;
        end
    end
    z=0;
    for r=(1:noFacets)
        for s=(1:noVerticies(r))
            z=z+1;

vertexData(z,1)=tempVertexData(z,1)*r11+tempVertexData(z,2)*r12+tempVertexD
ata(z,3)*r13;
```

```
vertexData(z,2)=tempVertexData(z,1)*r21+tempVertexData(z,2)*r22+tempVertexD
ata(z,3)*r23;

vertexData(z,3)=tempVertexData(z,1)*r31+tempVertexData(z,2)*r32+tempVertexD
ata(z,3)*r33;
            %vertexData(z,4)=p;
        end
    end

end

%finish buggering about with the crystal file vertex data and apply
       %new crystal data to seedpoints for the crystal and store data
%newlayerVertexData=zeros;
%below needs to be updated so that particles which completely lie
       %outside the bounds of the box are completely removed. This means
       %writing to a temp array before then adding to the full array...
       %if all points are>max z or<min z reject
       %if all points are>max y or<min y reject
       %if all points are>max x or<min x reject

j=0;
k=1;
m=0;
n=0;
NewNumberFacets=0;
%newlayerData=zeros;
for p=(1:size(seedPointData,1))
    maxX=0;
    minX=width;
    maxY=0;
    minY=length;
    maxZ=0;
    minZ=thickness;

    for q=(1:size(vertexData,1));

        tempVertexData(q,1:3)=vertexData(q,1:3)+seedPointData(p,1:3);

        if maxX<tempVertexData(q,1)
            maxX=tempVertexData(q,1);
        end
        if minX>tempVertexData(q,1)
            minX=tempVertexData(q,1);
        end
        if maxY<tempVertexData(q,2)
            maxY=tempVertexData(q,2);
        end
        if minY>tempVertexData(q,2)
            minY=tempVertexData(q,2);
        end
        if maxZ<tempVertexData(q,3)
            maxZ=tempVertexData(q,3);
        end
        if minZ>tempVertexData(q,3)
            minZ=tempVertexData(q,3);
        end
    end
    %This bit removes any crystals that lie outside the layer
    if (maxX<=0) || ( maxY<=0) || (minX>=length) || (minY>=width) ||
(minZ>=thickness)%|| (maxZ<=0)
        %maxX           minX               maxY               minY
maxZ               minZ
        %maybe not remove this next line as in truth, the seeds
        %seeds=seeds-1
    else
        qtyCrystalsSeeded=qtyCrystalsSeeded+1;
        %adding facets
        NewNumberFacets=NewNumberFacets+noFacets;
        %adding vertex data
        %set below outside the loop
        %j=0;
        %k=1;
        if k==1
            newlayerData2(1,1)=noFacets;
        end
        for i=(1:noFacets)
            k=k+1;
            j=j+1;
            m=m+1;
            if j>noFacets
                j=1;
                newlayerData2(k,1)=0;
                k=k+1;
                newlayerData2(k,1)=noFacets;
                k=k+1;
            end
            newlayerData(m,1)=noVerticies(j);
            %for layout2
            newlayerData2(k,1)=noVerticies(j);
        end
        % remove the; to generate the data to the command screen-note that
only
        % so much will be shown
        newlayerData2;

        for q=(1:size(vertexData,1));

newlayerVertexData(((n)*(size(vertexData,1)))+q,1)=boundingSphereNumber;
            newlayerVertexData(((n)*(size(vertexData,1)))+q,2)=p;

newlayerVertexData(((n)*(size(vertexData,1)))+q,3:5)=vertexData(q,1:3)+seed
PointData(p,1:3);
        end


        n=n+1;
        for q=(1:size(vertexData,1));

crystalVertexData(q,1:3,n)=vertexData(q,1:3)+seedPointData(p,1:3);
        end
        % remove the; to generate the data to the command screen-note that
only
        % so much will be shown
        newlayerVertexData;

    end

end
```

```
%             for p=(1:size(seedPointData,1))
%                 for q=(1:size(vertexData,1));
%                     newlayerVertexData( ((p-
1)*(size(vertexData,1)))+q,1:3)=vertexData(q,1:3)+seedPointData(p,1:3);
%                 end
%             end




 %This section generates the crystal layer file. The first line is the
 %number of facets, then it is a loop for each facet and then the array of
 %vertex data. Currently being stored as crystalfile.txt

output=fopen('crystalfile.txt','w');
fprintf(output,'%d \r\n',NewNumberFacets);
for i=(1:size(newlayerData))
    fprintf(output,'%d\r\n ', newlayerData(i));
end

for i=(1:size(newlayerVertexData))
    fprintf(output,'%d\t %d \t %5.5f \t %5.5f \t %5.5f
\r\n',newlayerVertexData(i,1),newlayerVertexData(i,2),newlayerVertexData(i,
3:5));
end

fclose(output);




%generate data for the facet matrix-this is used to determine overlaps
%between crystals
% currentPositionInVertexData=0;
% for i=(1:NewNumberFacets)
%     facet(i,1)=boundingSphereNumber;
%     facet(i,2)=newlayerData(i);
%
%     for j=(1:newlayerData(i))
%         p=(j-1)*3;
%         for k=(1:3)
%
facet(i,2+p+k)=newlayerVertexData(currentPositionInVertexData+j,k);
%         end
%     end
%     currentPositionInVertexData=currentPositionInVertexData+j;
% end



%generate data for the line matrix-again, used for overlaps between
%crystals
% currentPositionInVertexData=0;
% q=1;
% for i=(1:NewNumberFacets)
%
%
%     for j=(1:newlayerData(i))
%         if j==newlayerData(i)
%             endpoint=currentPositionInVertexData+1;
%         else endpoint=currentPositionInVertexData+j+1;
%         end
%         for k=(1:3)
%             line(q,1)=boundingSphereNumber;
%
line(q,k+1)=newlayerVertexData(currentPositionInVertexData+j,k);
%             line(q,k+4)=newlayerVertexData(endpoint,k);
%         end
%         q=q+1;
%     end
%     currentPositionInVertexData=currentPositionInVertexData+j;
%end


%This is new stuff, essentially generating a matrix of the line functions
%and the facet planes rather than calculating these on the fly every time.

%generate data for the plane matrix-this is used to determine overlaps
%between crystals
currentPositionInVertexData=1;
planeEquation=sym([]);
size(overlapCrystalArray);
%if size(overlapCrystalArray)>0
    for i=(1:NewNumberFacets)
        planeEquation(i,1)=boundingSphereNumber;

planeEquation(i,2)=newlayerVertexData(currentPositionInVertexData,2);
        %get plane points for the facet from 3 points in the plane
        %P1 P2 P3
        P1=newlayerVertexData(currentPositionInVertexData,3:5);
        P2=newlayerVertexData(currentPositionInVertexData+1,3:5);
        P3=newlayerVertexData(currentPositionInVertexData+2,3:5);

        normal=cross(P1-P2, P1-P3);
        syms x y z;
        P=[x,y,z];
        planefunction=dot(normal, P-P1);
        planeEquation(i,3)=planefunction;
        %dot(P-P1, normal);
        %realdot=@(u, v) u*transpose(v);
        %realdot(P-P1,normal);


currentPositionInVertexData=currentPositionInVertexData+newlayerData(i);
    end
%end
planeEquation;
%generate data for the line Equation matrix-again, used for overlaps
between
%crystals. Maybe replace the above line matrix?

%Despite being only 8 facets to a hex column,this means that there are 36
%lines (6x4 + 8x2), even though some of the lines are effectively the same.

currentPositionInVertexData=0;
q=1;
lineEquation=sym([0,0,0,0]);
for i=(1:NewNumberFacets)

    for j=(1:newlayerData(i))
```

```
        if j==newlayerData(i)
            endpoint=currentPositionInVertexData+1;
        else endpoint=currentPositionInVertexData+j+1;
        end

        P4=newlayerVertexData(currentPositionInVertexData+j,3:5);
        P5=newlayerVertexData(endpoint,3:5);
        syms t;
        line2=P4+t*(P5-P4);
        line2(1,1);
        line2(1,2);
        line2(1,3);
        lineEquation(q,3:5)=line2(1,1:3);
        lineEquation(q,1)=boundingSphereNumber;

lineEquation(q,2)=newlayerVertexData(currentPositionInVertexData+j,2);
        q=q+1;
    end
    currentPositionInVertexData=currentPositionInVertexData+j;
end

%save crystal.mat
```