Numerical Methods for Caclulating the Electrostatic Capacitance of Ice Crystal Aggregates

Zachary Waller
Student Number: 80214670
School of Physics and Astronomy
University of Manchester
Oxford Road
Manchester
M13 9PL
Email: zachary.waller@student.manchester.ac.uk
This project was performed in collaboration with Thomas Hayes

***Abstract*—A Monte Carlo method for accurately calculating the capacitance of arbitrary polyhedra was applied to a number of realistic ice crystal geometries. These were found to deviate from values obtained by approximating crystals to geometries with spherical symmetries - a result valuable to evapouration and deposition models in weather predictions.**

Aggregates of hexagonal plates with aspect ratio of 0.25 were generated with various aggregation indices and special attention paid to long, chain-like aggregates which are present in high altitude tropical storm clouds [1]. The normalised capacitance C/D_{max} was found to decrease with number of plates in the chain, but we expect it to reach an asymptotic value of $\approx 0.10-0.13$. Chain-like aggregates with aggregation indices approaching 1 were found to have lower normalised capacitances than more clumped aggregates.

As a test of the method's accuracy, the capacitance of a unit cube was calculated 0.6612 ± 0.0055, a value in close agreement with the literature [4]. The capacitance of hexagonal prisms as a function of aspect ratio were found in the range $\mathcal{A} = 0.5$ to $\mathcal{A} = 63.75$. A fit of $C = 0.58(1 + 0.936A^{0.76})$ was applied with a goodness-of-fit reduced χ^2 of 1.153. This gives a slightly higher value of capacitance at higher values of $\mathcal{A}$ than given by Westbrook et al. [12].

A number of Matlab programs were written from scratch to generate and test the capacitances of the various ice crystal geometries used and could be further extended to find the capacitances of other ice crystal types found in the atmosphere.

Contents

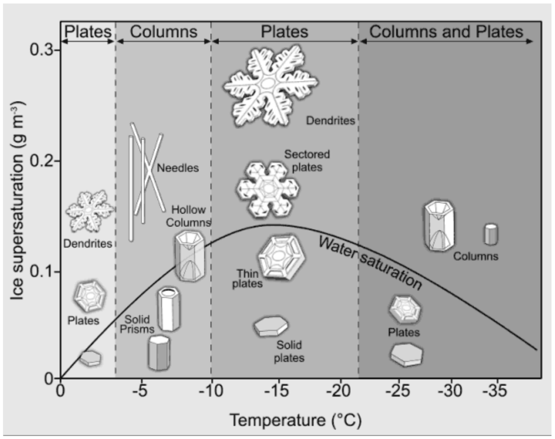

Figure 1. Diagram showing ice crystal morphology. The habits of ice crystals change as a function of ambient temperature and ice supersaturation [2].

I. Aims

1) Understand the governing equations for ice crystal growth in vapour fields.
2) Introduce an electrostatic analogy to aid in the solution of ice crystal growth equations.
3) Find a numerical method for calculating the capacitance of a general polyhedra.
4) Implement this method such that it can be used for realistic ice crystal geometries.
5) Expand this method to find the capacitances of complex aggregates of hexagonal prisms.

II. Introduction

The study of the growth and evapouration rates in clouds is an important and active area of research in numerical weather prediction models. For ice crystals, however, their often complex, polyhedral geometries mean any attempt at analytically calculating their growth rates in vapour feilds is futile. This has lead to - often crude - approximations of crystals and crystal aggregates to spheres or spheroids, which can overestimate the growth rates by a factor of 2 [12]. It has been shown that these approximations can give large errors in weather predictions with, for example, precipitation rates being significantly under estimated around mountain ranges due to these approximations [15]. Since ice crystals also affect the scattering of radiation, their existence in clouds alters the overall albedo of the Earth and consequently, atmospheric temperatures - so much so that it is comparable in magnitude to greenhouse gases [2]. The scattering properties ice clouds are dependent on the habits (geometries) of the constituent crystals which are, in turn, a product of the growth of crystals under different environmental conditions (see Fig.1). This shows that realistic models of ice crystal growths are necessary in the atmospheric sciences.

By making an analogy to electrostatics, the growth rate of an ice crystal can be simplified by introducing a capacitance fo each particular crystal geometry. This capacitance can be found directly by exploiting the electrostatic analogy fully and making conducting models of ice crystals and finding their

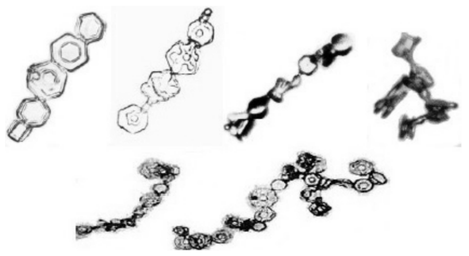

Figure 2. Photographs of chain aggregates formed in laboratory conditions [1]. Evidence for these chains forming naturally is found in high altitude, tropical thunderstorm clouds where strong electric fields ($\approx 10^5 Vm^{-1}$) induce dipole moments in ice crystals.

electrostatic capacitance [6]. However, a more accurate and adaptable method involving random walkers can be used - originating in problems of hydrodynamic friction [16] it was later applied to crystal growth by Westbrook et. al [12] and will be the method used for this project. The aim is to apply it to a number of crystal geometries with particular attention paid to chain-like aggregates of hexagonal plates.

Chain-like aggregates are found in tropical thunderclouds where strong electric fields induce dipole moments in small crystals. These dipole moments are sufficient enough to cause these crystals to coalesce preferentially along the direction of the electric field and form long chains (see Fig.**??**) [1]. These chains are typically made up of hexagon plates and have been little studied in terms of capacitances. These aggregates are also of particular interest for their scattering properties - they can give rise to highly reflective clouds [1]. Since the associated anvil storm clouds are most common in tropical and equatorial regions, where solar radiation is highest, the understanding of chain growth rates is an important research topic.

III. Theory

A. Growth Rates of Ice Crystals

The growth of an ice crystal in a vapour field is determined by the diffusion of vapour through the field and its subsequent deposition onto the surface of the crystal. The vapour current $\mathbf{j_m}$ is a measure of the mass flowing through a unit area per unit time and is given by Fick's Law of diffusion as

$$\mathbf{j_m} = -D\boldsymbol{\nabla}\rho_v \tag{1}$$

where D is the diffusivity of the vapour field and ρ_v is the vapour density. The vapour current satisfies the continuity equation

$$\frac{\partial \rho_v}{\partial t} + \boldsymbol{\nabla}\cdot\mathbf{j_m} = 0 \tag{2}$$

for a region of space with no sources or sinks. In the steady state, where $\frac{\partial \rho_v}{\partial t} = 0$, (1) and (2) can be combined to give

$$\boldsymbol{\nabla}\cdot\mathbf{j_m} = \boldsymbol{\nabla}\cdot(-D\boldsymbol{\nabla}\rho_v) = 0$$

or

$$\boldsymbol{\nabla}^2\rho_v = 0.$$

That is, ρ_v satisfies Laplace's equation.

The growth rate of the crystal can be found by integrating (1) over the surface of the crystal S to give

$$\frac{dm}{dt} = -D \int_S \nabla \rho_v \cdot d\mathbf{S}. \tag{3}$$

There are few geometries for which there exists an analytic solution for (3), and the few that do cannot be said to realistically represent an ice crystal (see Table 1). By use of an analogy with electrostaics, however, the solution to (3) can be simplified for real ice crystals.

1) Electrostatic Analogy: The electrostatic potential Φ of a perfect conductor with total charge Q in an electric field also satisfies Laplace's equation

$$\nabla^2 \Phi = 0.$$

Gauss' law for the setup gives

$$\frac{Q}{\epsilon_0} = -\int_S \nabla \Phi \cdot d\mathbf{S} \tag{4}$$

where ϵ_0 is the permittivity of free space. The right hand sides of (3) and (4) are equivalent in form, and since both ρ_v and Φ satisfy Laplace's equation the same solutions can be used for each. The usual solution to (4) involves rewriting Q in terms of the capacitance C_e and the potential difference $(\Phi_{surf} - \Phi_\infty)$

$$Q = C_e(\Phi_{surf} - \Phi_\infty)$$

such that the integral can be avoided:

$$-\int_S \nabla \Phi \cdot d\mathbf{S} = \frac{1}{\epsilon_0} C_e(\Phi_{surf} - \Phi_\infty). \tag{5}$$

This means that the problem of solving the integral on the right hand side of (4) equates to simply plugging in the value of the conductor's capacitance (a purely geometrical factor) and the boundary conditions of the potential.

Returning to the problem of growth rates, the same strategy can be employed to find

$$-\int_S \nabla \rho_v \cdot d\mathbf{S} = 4\pi C_v(\rho_{v,S} - \rho_{v,\infty}) \tag{6}$$

where C_v is the "vapour capacitance" of the crystal. The factor 4π arises in (6) but not in (5) due to the definition of ϵ_0 in SI units. This can then be used to solve 3 for the growth rate:

$$\frac{dm}{dt} = -D 4\pi C_v(\rho_{v,surf} - \rho_{v,\infty}). \tag{7}$$

Here, the electrostatic capacitance and vapour capacitance are related by

$$C_e = 4\pi C_v \epsilon_0,$$

that is, the electric capacitance and vapour capacitance are equivalent up to a numerical constant, so any results from electrostatics can be used here. In fact, MacDonald [6] put conducting models of ice crystals in electric fields to find their capacitance and solve the growth rate problem.

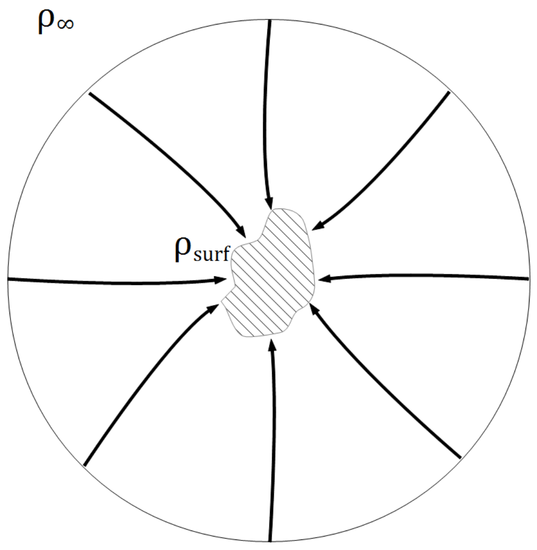

Figure 3. A crystal in a vapour field. The black arrows represent the flux of vapour towards the crystal. The field is isotropic on the surface of the circle (where there is no influence from the crystal), but becomes anisotropic nearer the crystal.

2) Boundary Conditions: The next important problem is to determine the boundary conditions on the vapour density; its value at the surface of the crystal and at "infinity". At an infinite distance form the crystal, the vapour density ρ_∞ will, by symmetry, be isotropic and constant.

At the surface of the crystal the vapour density ρ_{surf}, will be equal to the saturation density ρ_{sat} of the air. The saturation density at the surface is a function of temperature only. The mechanisms affecting the surface temperature are: the release of latent heat when vapour deposites onto the crystal, and conduction of heat from the crystal. The conduction of heat from the crystal is determined by Fourier's Law:

$$\mathbf{j_h} = -k \nabla T \tag{8}$$

where $\mathbf{j_h}$ is the heat current, k is the coefficient of heat conduction and T is the temperature of the surrounding vapour.

When one area of the crystal has more vapour deposited onto it than the surrounding area, its temperature will increase through the release of latent heat, but heat will also be conducted away from it at a similarly higher rate due to (8). These effects work together such that the surface temperature T_S is constant [6]. The constancy of T_S means that ρ_{sat} at the crystal surface and therefore ρ_{surf} are also constant and uniform [9]. This can also be linked back to the electrostatic case where, for a conductor, the potential will be uniform across the surface because any differences will cause a current to flow and equalise the potential. Hence, the boundary conditions are:

$$\rho_v = \rho_{surf} \text{ at the crystal surface and}$$

$$\rho_v = \rho_\infty \text{ far from the crystal.}$$

Table I
Exact capacitances for geometries often used as approximations in ice crystal problems [6].

Shape	Capacitance
Sphere, radius r	$C = r$
Thin disc, radius r	$C = 2r/\pi$
Prolate spheroid: major,	$C = A/\ln(a+A)/b$,
minor semiaxes a, b	where $A = \sqrt{a^2 - b^2}$
Oblate spheroid: major	$C = ae/\sin^1 e$,
minor semiaxes a, c	where $e = \sqrt{1 - c^2/a^2}$

B. Determining the Capacitance

Finding the growth rate of the ice crystal has now been vastly simplified, so long as the capacitance of the specific geometry is known - itself a non-trivial problem. Some analytic answers for simple shapes are given in Table 1 and are commonly used in approximations [13] but if a more accuracy is required the only way to find the capacitance is through numerical methods.

The method used in this project involves tracking the diffusion of individual vapour particles. The particles begin on a sphere where the vapour has a uniform distrbution and are then released to diffuse randomly until they hit the crystal surface S or escape back to the outer boundary. The flux of particles k_{surf} onto the ice crystal is

$$k_{surf} = k_R f \tag{9}$$

where k_R is the flux passing through a sphere of radius R that contains S for the first time and f is the fraction of particles passing through the sphere that go on to be absorbed by the crystal. R must be large enough such that the distribution of vapour on it is isotropic (see Fig.3).

The flux of vapour onto the bounding sphere *for the first time* is simply the flux onto a perfectly absorbing sphere. From (7):

$$k_R = D4\pi C_R \tag{10}$$

where C_R is the capacitance of the bounding sphere, which is R (see Table 1). An important point to note is that the *net* flux through the sphere is not given by (10) because vapour can pass through the sphere and still "escape" back out to the outer boundary. The *net* flux will be the same as the flux onto the crystal surface, as the crystal is the only vapour sink within the volume of the sphere [12].

By setting $\rho_{v,S} = 0$ and $\rho_{v,\infty} = 1$ the flux through the surface is then

$$k_{surf} = D4\pi C_S \tag{11}$$

where C_S is the crystal surface's capacitance (the quantity we wish to find).

Substituing (10) and (11) into (9) yields

$$C_S = Rf \tag{12}$$

that is, the capacitance or an arbitrary shape can be found from the radius of a bouding sphere, and the likelihood that a vapour particle passing through this sphere goes on to deposite itself on the crystal. A method must now be found for calculating the value of f.

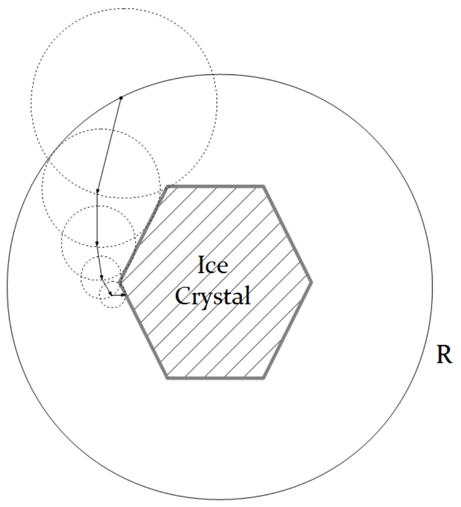

Figure 4. Schematic diagram showing an example of a random walker path inside the bounding sphere that eventually hits the ice crystal. The dashed circles show the step size at each point - equal to the shortest distance to the crystal. The grey border of the crystal is an exaggerated δ absorbing layer. Once the random walker has hit the crystal 1 is added to the number of $hits$.

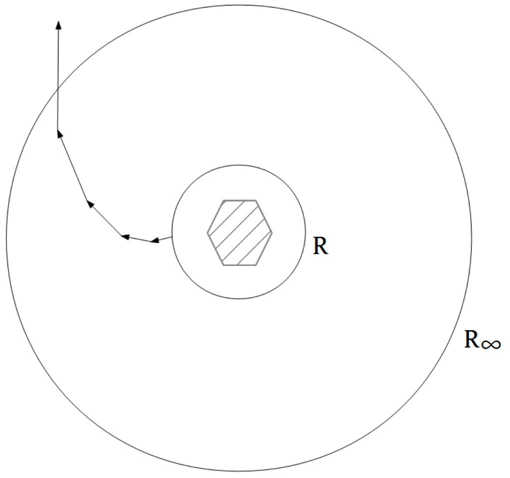

Figure 5. Schematic diagram showing an example of a random walker released from the bounding sphere and escaping to R_∞. The step size is equal to the minimum distance to the bounding sphere at R. Once the random walker escapes 1 is added to the number of $losses$.

IV. Experiment Method

A. Walk on Spheres Algorithm

The method used to find the probability of a vapour particle released from a bounding sphere hitting the crystal surface is the "Walk on Spheres" algorithm [8]. A vapour particle is released from a random position on the bounding sphere and released to move in a random direction by a certain step size.

The vapour particle (or "random walker") repeats this step of moving in a random direction until it either hits the crystal and is absorbed or escapes to "infinity" and is lost. For each case the result adds 1 to counter of the number of $hits$ or $losses$ and another walker is released from the bounding sphere (see Fig.4 and Fig.5). From the $hits$ and $losses$ f can be found:

$$f = \frac{hits}{hits + losses}. \tag{13}$$

This method will then accurately sample f to arbitrary precision, and the only intrinsic error is due to the nature of statistical sampling [4]. The error on the $hits$ counted is $\sqrt{N}$ where N is the total number of walkers ($hits + losses$). This gives the error in f as

$$\sigma_f = \frac{1}{\sqrt{N}}$$

and from (12) the error in capacitance σ_C is

$$\sigma_C = \frac{R}{\sqrt{N}}. \tag{14}$$

The algorithm can be optimised by using a variable step length as to save the number of steps the random walker spends in empty space. The first optimisation is to vary the step size whilst outside of the bounding sphere to be equal to the distance between the walker's position and the bounding sphere: if r is the walker's distance from the origin and R is the radius of the sphere, then the minimum distance between the two is simply $r - R$ (provided the sphere is centred on the origin, which will be ensured by construction).

The next optimisation involves changing the step size whilst inside the bounding sphere, such that it is equal to the minimum distance to the crystal. Determining the minimum distance between a polyhedra and an abitrary point is actuallly a rather involved calculation and will be explained in the next section.

Both optimisations give no bias to the statistics of the problem as they both exploit the isotropic nature of the probability distribution: because each optimisation avoids the possibility of the walker hitting the crystal at each step, there are no sources of sinks of vapour within reach of the walker - that is, the probability of each new position being chosen at each time step is as likely as it would be for a series of small steps to get to the same point.

These optimisations, however, bring a systematic error into the simulation; because the walkers can never actuallty hit the crystal (the step size being only equal to the distance to the crystal means the likelihood of this happening is effectively zero), an absorbing layer δ must be introduced wherein the walker can be said to have been absorbed by the crystal. This, unavoidably skews the result of the measurement - we are infact finding the capacitance of a subtly different geometry. By reducing the thickness of δ though, this error can be reduced to be negligibly small but at the expense of computing time. The δ error can be removed by the use of Green's functions [7] but it should be small enough to have little effect on our results.

A further optimisation available is the reduction in size of the bounding sphere. By reducing its size such that it just contains the crystal, the amount of empty space between the sphere and crystal is greatly diminished. This relies on the statistical fact that the distribution of walkers passing through R from R_∞ for the first time will be uniform - there is nothing to distort the spherical symmetry but the crystal, which is inside of R. As explained earlier, the *net* distribution of walkers at R will, of course, not be spherically symmetric though due to the influence of the ice crystal.

The other error arising from this method is the fact that walkers must be tracked out to R_∞ before they can be said to be truly lost. Of course, letting walkers actually escape to an infinite distance is impossible, so a distance must be chosen as a placeholder for R_∞. This has to be, at the very least, large enough that the probability distribution is uniform, meaning the effect on the symmetry by the cystal is negligible. The chosen value will be subject to considerations of both accuracy and computing time.

B. Hit Detection - GJK Algorithm

The most difficult part of this simulation is quickly and accurately determining the minimmum distance from the random walker to the polyhedral ice crystal. The method used for this is a modified version of the Gilbert-Johnson-Keerthi Algorithm adapted from Lindemann [5] but extended to $3D$ by myself. The way the algorithm works is by filling a "simplex" (just a collection of vertices from the polyhedra) with points and finding the shortest distance from the simplex to the random walker and then iteratively converging on closer simplices. When the algorithm returns the same simplex at two consecutive iterations it terminates as the current simplex must be the closest one.

The random walker is always just a single point, but the simplex can be either a point, line, triangle face or a tetrahedron. So at any iteration the algorithm must find the distance between: 1) a point and a point, 2) a point and a line, or 3) a point and a face. Each of these requires a separate method of caculating the distance.

1) Calculating Distances for each Geometry: 1) *Point to point distance*: For a point $\mathbf{P}$ and simplex of just one point $\mathbf{A}$ the minimum distance d_{point} is simply the magnitude of the vector between $\mathbf{A}$ and $\mathbf{P}$:

$$d_{point} = |\mathbf{P} - \mathbf{A}| = \sqrt{(\mathbf{P} - \mathbf{A}) \cdot (\mathbf{P} - \mathbf{A})} \tag{15}$$

see Fig.6.1.

2) *Point to line distance*: Imagine a point $\mathbf{P}$ and a line segment $\mathbf{AB} = \mathbf{B} - \mathbf{A}$, the shortest distance is now the perpendicular distance from $\mathbf{AB}$ to $\mathbf{P}$. First, construct a perpendicular line $\mathbf{perp}$ to $\mathbf{AB}$ in the direction of $\mathbf{P}$

$$\mathbf{perp} = (\mathbf{AB} \times \mathbf{AP}) \times \mathbf{AB} \tag{16}$$

where $\mathbf{AP} = \mathbf{P} - \mathbf{A}$. The minimum distance $d_{line}$is then the magnitude of $\mathbf{AP}$ in the direction of normalised $\mathbf{perp}$. That is

$$d_{line} = \frac{\mathbf{perp}}{|\mathbf{perp}|} \cdot \mathbf{AP}. \tag{17}$$

see Fig.6.2.

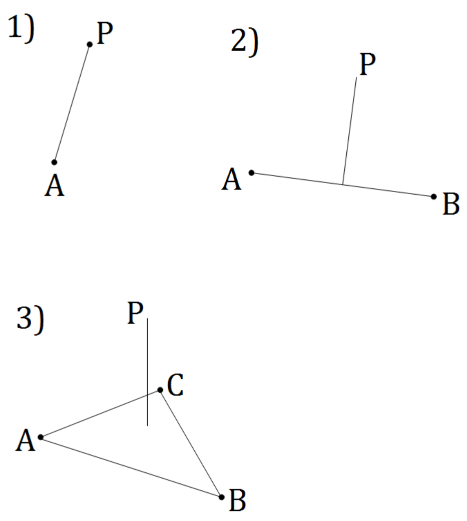

Figure 6. Diagram showing the minimum distance for each case: 1) a point and a point, 2) a point and a line and 3) a point and a plane.

3) *Point to face distance*: The minimum distance to a plane is similar to the line case; it is equal to the magnitude of a line perpendicular to the face and pointing to $\mathbf{P}$. Imagine a triangle $\mathbf{ABC}$ and a point $\mathbf{P}$, the normal to the triangle $\mathbf{norm}$ is

$$\mathbf{norm} = \mathbf{AB} \times \mathbf{AC} \tag{18}$$

where $\mathbf{AB} = \mathbf{B} - \mathbf{A}$ and $\mathbf{AC} = \mathbf{C} - \mathbf{A}$. The minimum distance d_{plane} is then the magnitude of the vector $\mathbf{AP}$ in the direction of the unit normal $\frac{\mathbf{norm}}{|\mathbf{norm}|}$.

$$d_{plane} = \frac{\mathbf{norm}}{|\mathbf{norm}|} \,.\, \mathbf{AP} \tag{19}$$

see Fig.6.3.

This method can give negative values of d_{plane} if the "winding" (whether going from $\mathbf{A}$ to $\mathbf{B}$ to $\mathbf{C}$ takes you clockwise or anticlockwise) of the points in the face aren't correct. This is simply remedied by taking the absolute value of the value of d_{plane}.

2) Converging on the closest simplex: At this point it may be tempting to plug all the faces, edges and vertices into these formulae, find the minimum of them all and then use this as the distance for the next step in the Walk on Spheres algorithm. However, this will lead to incorrect distances - it could return a distance of zero even though the point is not at all touching (or even near to) the crystal - leading to errors in hit detection. The reason these problems arise is due to the fact that though the problem really requires the distance to line *segments* and *bounded* planes, whereas the methods used above would find the distance to *infinite* lines and planes. This could lead to a situation where the point is not at all touching the crystal but is in the same plane as one of the faces; leading to a 'minimum distance' of 0. To combat this error, a method for finding the best simplex to find the minimum distance from is needed.

The machinery used for this is a *search function* which essentially "searches" for the farthest point in a given direction.

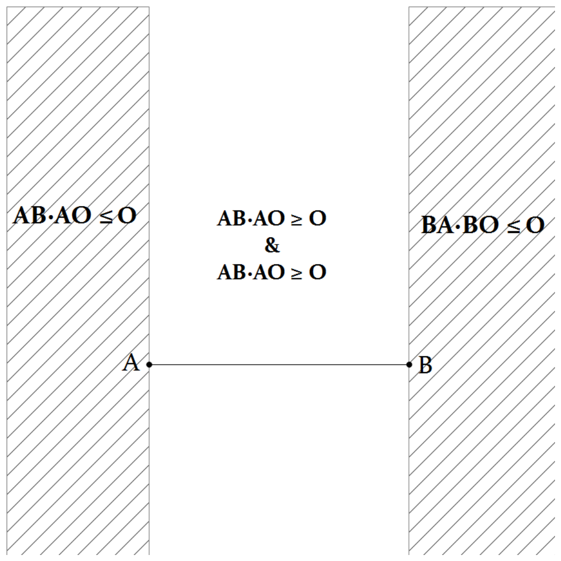

Figure 7. Diagram showing the regions "cut off" by the checks in (21) and (22).

The first step is to choose a search direction $\mathbf{S}$ (with $|\mathbf{S}| = \mathbf{1}$) and to find the vertex of the crystal furthest in that direction. This involves a dot product with every vertex of the crystal; for the ith vertex $\mathbf{v_i}$ its distance is

$$dist_i = \mathbf{v_i} \,.\, \mathbf{S}. \tag{20}$$

These distances are stored in an array and the vertex corresponding to the largest value of $dist$ will be the first member of the simplex $\mathbf{A}$. The distance from $\mathbf{A}$ to $\mathbf{P}$ is then found from (15). The search direction is now changed to $\mathbf{S} = \mathbf{P} - \mathbf{A}$ (and normalised). The $searchfunction$ is then called on again and another point $\mathbf{B}$ is added to the simplex. If $\mathbf{B} = \mathbf{A}$ then the algorithm terminates as $\mathbf{A}$ must be the nearest point on the whole crystal to the random walker at $\mathbf{P}$.

If $\mathbf{B} \neq \mathbf{A}$ then the simplex becomes a line. For the next distance check, one needs to make sure that $\mathbf{P}$ is between the points $\mathbf{A}$ and $\mathbf{B}$ such that (17) is valid to use.

3) Voronoi Regions: Voronoi Regions are geometrical constructions which provide an intuitive method for finding which part of a polyhedra a point is nearest to in $3D$ space.

For a line segment $\mathbf{AB}$, to make sure that the point is between $\mathbf{A}$ and $\mathbf{B}$ then two dot product tests can be made. The point $\mathbf{P}$ is within the line if and only if both

$$\mathbf{AB} \,.\, \mathbf{AP} \geq \mathbf{0} \tag{21}$$

and

$$\mathbf{BA} \,.\, \mathbf{BP} \geq \mathbf{0} \tag{22}$$

see Fig.7.

Both are either true, or one is true and the other isn't - they can't both be false. If (21) is false, then the point is closer to $\mathbf{A}$ than the line and similarly for (22) and $\mathbf{B}$. If both are true, then (17) can be used to find the distance and $\mathbf{S} = \mathbf{perp}$ from (16). This new $\mathbf{S}$ *always* point in the direction of the random walker.

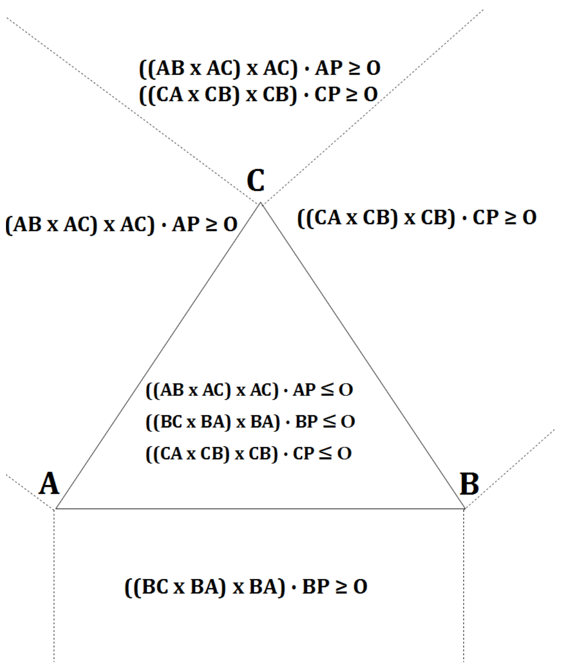
((AB x AC) x AC) · AP ≥ O
((CA x CB) x CB) · CP ≥ O
C
(AB x AC) x AC) · AP ≥ O
((CA x CB) x CB) · CP ≥ O
((AB x AC) x AC) · AP ≤ O
((BC x BA) x BA) · BP ≤ O
((CA x CB) x CB) · CP ≤ O
A
B
((BC x BA) x BA) · BP ≥ O

Figure 8. Diagram showing the regions of a triangle and the corresponding checks in each region. Similar checks to that shown near the $\mathbf{C}$ corner are used for $\mathbf{A}$ and $\mathbf{B}$ but have been omitted for clarity.

Similar tests must be taken for when the simplex is a triangle. All three vertices and edges need to be checked as well as the face. For the $\mathbf{AC}$ edge:

$$((\mathbf{AB} \times \mathbf{AC}) \times \mathbf{AC}) \,.\, \mathbf{AP} \leq \mathbf{0}, \tag{23}$$

for the $\mathbf{BA}$ edge:

$$((\mathbf{BC} \times \mathbf{BA}) \times \mathbf{BA}) \,.\, \mathbf{BP} \leq \mathbf{0}, \tag{24}$$

and for the $\mathbf{CB}$ edge:

$$((\mathbf{CA} \times \mathbf{CB}) \times \mathbf{CB}) \,.\, \mathbf{CP} \leq \mathbf{0}. \tag{25}$$

All three must be true for $\mathbf{P}$ to be within the $\mathbf{ABC}$ face (see Fig.8). Pairs of false equations means $\mathbf{P}$ is in a corner region. For edges of the triangle, tests like (21) and (22) need to be used along with the corresponding equation from (23), (24) and (25) - the point must be between the two ends of the edge and outside of the face. If the point is nearest the plane then () can be used for the distance and $\mathbf{S} = \mathbf{norm}$ form (18).

If the point is nearest a corner or edge, then the distance is calculated accordingly and $\mathbf{S}$ is changed. The simplex must also be reduced in these cases: if, for example, the point were nearest to the $\mathbf{AC}$ edge, then $\mathbf{B}$ must be gotten rid of and $\mathbf{C}$ relabelled as the new $\mathbf{B}$.

For a tetrahedron, the checks become more numerous still as the extra point creates three more edges and three more faces. The checks for the faces remain the same. The edges, however need a check to make sure the point is outside of the two faces that share the edge. The points also require extra checks - they need to be checked from the point of view of each of the three edges that share it.

The GJK algorithm sorts through these Voronoi Regions in such a way that it is not necessary to check every single point, edge and face of the crystal, as it reduces the points to a simplex and even then it is only necessary to do a few checks at each iteration as the simplex builds up.

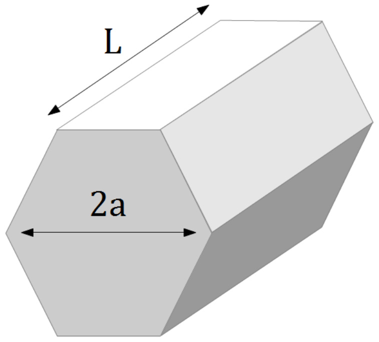

Figure 9. A hexagonal prism showing the definitions of a and L.

After the first iteration, it is actually no longer necessary to ever check the $\mathbf{A}$ vertex ever again, and after the second the $\mathbf{AB}$ edge need never be checked again. Once the simplex becomes a tetrahedron, the $\mathbf{ABC}$ triangle need never be checked, nor do its corners or edges. This leaves the $\mathbf{D}$ vertex; $\mathbf{AD}$, $\mathbf{BD}$ and $\mathbf{CD}$ edges; and the $\mathbf{ABD}$, $\mathbf{BCD}$ and $\mathbf{CAD}$ faces - all of which can be checked by the methods previously established.

When the algorithm starts looping through a number of the same simplices over and over, then the algorithm terminates and the current value of the shortest distance is used. This shortest distance is the step size for the next step in the Walk on Spheres simulation. When the minimum distance drops below δ, a hit is counted and the next random walker is released from the bounding sphere.

C. Making the Chains

The purpose of this project is not only to find the capacitance of simple, convex shapes (like a hexagonal plate), but to find it for aggregates of simple crystal shapes. To this end we need to find a useful way of 1) making chains of simple shapes and 2) parameterising these chains so useful results can be found.

1) Geometry of Hexagons: First of all, it is useful to parameterise the properties of a single hexagon and later more complex aggregates can be made.

The geometry of interest for this project is that of a plate-like hexagonal prism - the most common shape of ice crystal [3]. The geometry of a regular hexagonal plate is usually characterised by two parameters; the length L and width $2a$. The length is the distance from one hexagonal face to the other, whereas the width is the largest distance across the hexagonal faces (see Fig.9). The aspect ratio, $\mathcal{A}$ is defined as the length over the width, $\mathcal{A} \equiv \frac{L}{2a}$.

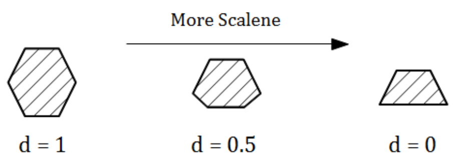

Figure 10. Examples of hexagons with various values of d. A regular hexagon has $d = 1$ and as d is changed it becomes more scalene. When $d = 0$ the hexagon becomes a trapezium.

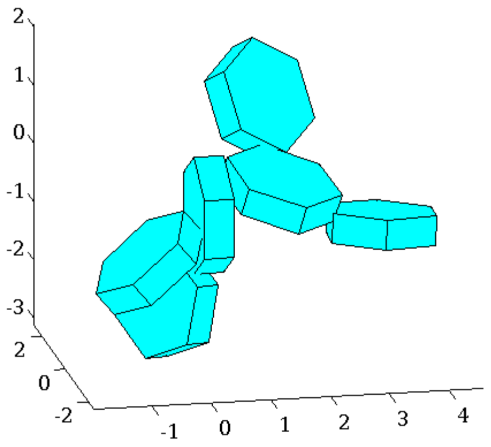

Figure 11. Example of a chain with aggregation index of 0.5388 generated by our MATLAB code.

Another parameter can also be introduced to make the hexagons more scalene which has been observed in real ice crystals. This 'scalene-parameter' d is factor which multiplies onto the points on the bottom half of the hexagonal faces; it gives a regular hexagon with $d = 1$ and a trapezium with $d = 0$ (see Fig.10).

2) Aggregation Index: The aggregation index AI supplies a method to parameterise aggregations of chains [11]. It is essentially a measure of how "clumped up" the aggregation is. It is defined as:

$$AI = \frac{\sum_{i=1}^{N}\sum_{j=1}^{N} d_{ij}}{\text{Max}(\sum_{i=1}^{N}\sum_{j=1}^{N} d_{ij})} \tag{26}$$

where d_{ij} is the distance between the centres of the ith and jth plates in the chain. The denominator is the value of the sum when the plates are arranged to be as extended as possible while still touching. The agggregation index, therefore, will always give a value between 0 (the centres are exactly on top of one another) and 1 (the chain is extended as possible). If a random aggregation of plates with a set number of plates is made, the capacitance for each chain can be found and plotted against AI. A library of chains can also be made and the capacitance plotted against the number of plates.

D. GJK Algorithm for Two Polyhedra

Chains can be made up of multiple simple shapes by randomly attaching a new hexagon to the chain from the previous

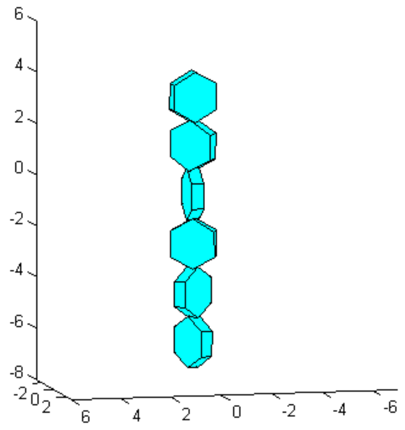

Figure 12. Example of a chain with aggregation index of 0.96 generated by our MATLAB code.

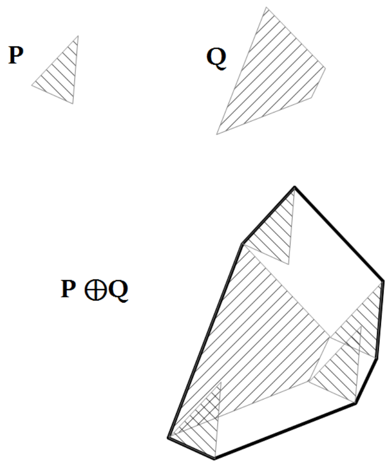

Figure 13. Schematic of a Minkowski Sum of two polygons P and Q. A simple way to visualise the Minkowski Sum is to'sweep' one shape around the perimeter of the other while keeping one corner fixed to perimeter. The Minkowski Difference would require the same method, but one shape would be mirrored about the $y = -x$ line.

iteration. This raises the problem of realistically attaching the hexagons together; real ice crystals won't intersect with one another but will be joined at the vertices, edges or faces in some way. To realistically emulate this requires an algorithm to detect the intersection between two convex polyhedra. This is similar, but subtley different to the algorithm in (IV-B) but the distance does not need to be found at any point - the algorithm only returns *true* or *false* (the polyhedra are either intersecting or they aren't).

This algorithm exploits the properties of the Minkowski Sum [14].

1) The Minkowski Sum: The Minksowski Sum of two polyhedra P and Q involves the vector addition of every point in P with every point in Q and is denoted as

$$P \oplus Q = \{\mathbf{p} + \mathbf{q} \mid \mathbf{p} \in P, \mathbf{q} \in Q\}.$$

A Minkowski difference can also be constructed, which involves the negative vectors of all the points in P or Q:

$$P \ominus Q = \{\mathbf{p} - \mathbf{q} \mid \mathbf{p} \in P, \mathbf{q} \in Q\}.$$

The Minkowski difference is essentially the same as the Minkowski sum, but one of the shapes will be mirrored about the $x = -y$ line.

The Minksowski difference is the most useful for finding the intersection between shapes, because if the two polyhedra intersect, then the Minkowski Difference will contain the origin - there will be at least one point in P that is the same as a point in Q such that $\mathbf{p} - \mathbf{q} = \mathbf{0}$. Therefore, if a method can be found that first of all finds the Minkowski difference of two polyhedra, and then searches for the origin within this Minkoswki difference, we can find out whether or not two polyhedra intersect.

To find this out, it is not necessary to calculate the full Minkowski difference (that is, it isn't necessary to find the vector difference between every point in one polyhedra and every point in the other) - you can get away with only finding the Minkowski difference of each of the vertices because all the other points in the full Minkowski difference will be within these points. This relies on both polyhedra being convex, though concave polyhedra can be split into multiple convex polyhedra which can be checked individually.

The machinery for calculating the Minkowski differences needed is contained within a segment of code known as the *support function*. This searches in a direction $\mathbf{S}$ for the furthest vertex $\mathbf{v_1}$ (in the same way that the *search function* from (IV-B) searched fro the furthest vertex in a given direction) of one the polyhedra along $\mathbf{S}$ and then searches along $-\mathbf{S}$ for the furthest vertex $\mathbf{v_2}$ in the other polyhedra. This way, the largest possible bounding area for the origin is found, which will reduce the convergence time for the algorithm. The *support function* then finds the vector difference of these two points (this is a single point in the complete Minkowski difference) to find the first point in the simplex $\mathbf{A}$: $\mathbf{v_1} - \mathbf{v_2} = \mathbf{A}$.

The search direction is then switched to $-\mathbf{S}$ and this step is repeated to find $\mathbf{B}$. From here a check can be made to see whether the origin is somewhere between $\mathbf{A}$ and $\mathbf{B}$ - if it isn't then the origin cannot be contained within the full Minkowski difference of the polyhedra, that is, they cannot be intersecting. This *outside check* is as follows

$$\mathbf{B} \boldsymbol{.} \mathbf{S} > \mathbf{0} \tag{27}$$

if this is true, then the origin may be contained within the full Minkowski difference, but if it is false, then the origin cannot be contained within the full Minkowski difference. What this essentially does is check whether the most recent point in the simplex is actually along the search direction - if it isn't, then is is on the other side of the origin and therefore the origin cannot be contained by the Minkowski difference; the algorithm will return *false*, because the polyhedra aren't intersecting.

If (27) is true, then the next step of the algorithm can be taken; (27) doesn't tell us if the origin is contained within the Minkowski difference, only whether it is not. The next step is an *origin check*. This finds where the origin is with respect to the current simplex and chooses a new search direction pointing towards the origin. There are 3 cases:

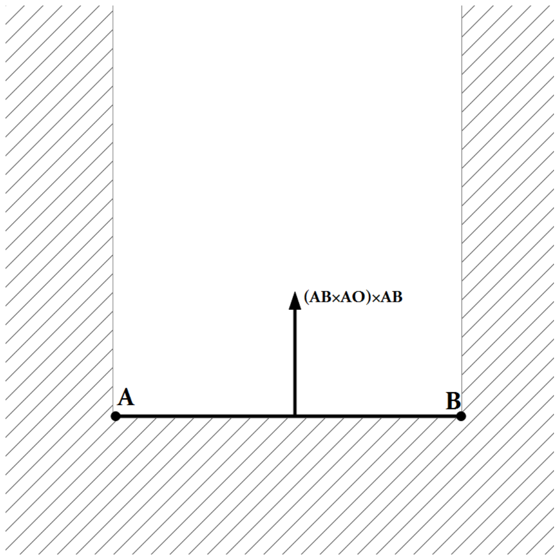

Figure 14. Schematic of a simplex with two points in it. The hatched area represents where the origin canot lie, and the white area is where is could lie. The areas to the left and right of $\mathbf{A}$ and $\mathbf{B}$ respectively were eliminated by 'outside check' from earlier. The new search direction $\mathbf{S} = (\mathbf{AB} \times \mathbf{AO}) \times \mathbf{AB}$ will always point in the direction of the origin. If $\mathbf{S} \boldsymbol{.} \mathbf{AO} > \mathbf{0}$ then the possibility of the origin being 'below' the line $\mathbf{AB}$ is eliminated.

1) The simplex is a line, $\mathbf{AB}$

The search direction is taken to be perpendicular to the line and in the direction of the origin (see Fig.**??**).

$$\mathbf{S} = (\mathbf{AB} \times \mathbf{AO}) \times \mathbf{AB}. \tag{28}$$

The algorithm now returns to the start and goes through the *support function* and then the *outside check*. If the *outside check* returns true then the simplex will be a triangle $\mathbf{ABC}$.

2) The simplex is a triangle $\mathbf{ABC}$

The origin can be in one of three places: outside of the triangle, but near side $\mathbf{AC}$; outside the triangle, but near side $\mathbf{BC}$ or inside the triangle (or above or below it, but bounded by the three vertices). This requires two checks:

$$((\mathbf{CB} \times \mathbf{CA}) \times \mathbf{CA}) \boldsymbol{.} \mathbf{CO} > \mathbf{0} \tag{29}$$

and

$$((\mathbf{CA} \times \mathbf{CB}) \times \mathbf{CB}) \boldsymbol{.} \mathbf{CO} > \mathbf{0} \tag{30}$$

each of these constructs a vector in the plane of the triangle, perpendicular to the sides and pointing out of the triangle and dots products it with a vector pointing from a vertex of that side to the origin (see Fig.**??**). Either only one is true, or both are false. If (29) is true, then the search direction becomes $((\mathbf{AC} \times \mathbf{AO}) \times \mathbf{AC})$ and the simplex gets rid of point $\mathbf{B}$, but if (30) is true, the search direction is $((\mathbf{BC} \times \mathbf{BO}) \times \mathbf{BC})$ and the simplex gets rid of point $\mathbf{A}$. After this the algorithm returns to the *support function* and *outside check*.

If both (29) and (30) are false, then the origin lies somewhere within an infinite triangular prism defined by the $\mathbf{ABC}$ triangle (see Fig.**??**). To find the new search direction, it is

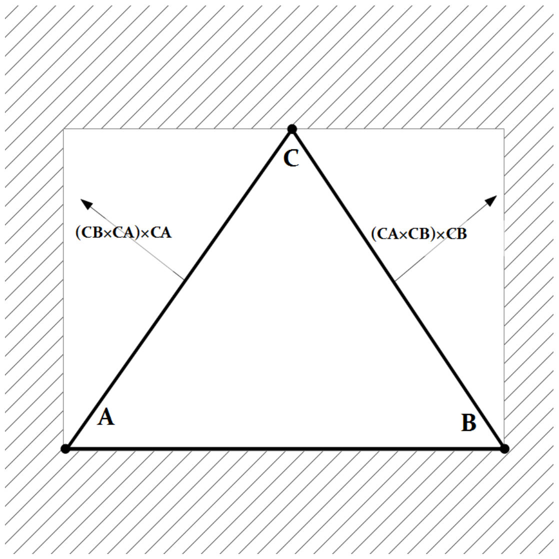

Figure 15. Schematic of a simplex with two points in it. The hatched area represents where the origin canot lie, and the white area is where is could lie. The areas to the left and right of $\mathbf{A}$ and $\mathbf{B}$ respectively were eliminated by 'outside check' from earlier. The new search direction $\mathbf{S} = (\mathbf{AB} \times \mathbf{AO}) \times \mathbf{AB}$ will always point in the direction of the origin. If $\mathbf{S} \,.\, \mathbf{AO} > \mathbf{0}$ then the possibility of the origin being 'below' the line $\mathbf{AB}$ is eliminated.

necessary to find out whether the origin is above or below the triangle. This requires the check

$$(\mathbf{CA} \times \mathbf{CB}) \,.\, \mathbf{CO} > \mathbf{0}. \tag{31}$$

If this returns true then the search direction is now $(\mathbf{CA} \times \mathbf{CB})$ and the algorithm returns to the support function and outside check with the simplex $\mathbf{ABC}$. If this check returns false, however, then the search direction becomes $-(\mathbf{CA} \times \mathbf{CB})$ and the order of the simplex must be changed. The labels of two points muct be swapped (this is very important for the next part of the algorithm with four points); these can be any two points, so long as it is only two points.

3) The simplex is a tetrahedron $\mathbf{ABCD}$

For a tetrahedron, three more checks must be made For the $\mathbf{DAB}$ face

$$(\mathbf{DA} \times \mathbf{DB}) \,.\, \mathbf{DO} \geq \mathbf{0}, \tag{32}$$

for the $\mathbf{DBC}$ face

$$(\mathbf{DB} \times \mathbf{DC}) \,.\, \mathbf{DO} \geq \mathbf{0}, \tag{33}$$

and for the $\mathbf{DCA}$ face

$$(\mathbf{DC} \times \mathbf{DA}) \,.\, \mathbf{DO} \geq \mathbf{0}. \tag{34}$$

As with the triangle case, with these checks either one is true and the others are false, or all three are false. When only one is true, then the face that it checks becomes the new simplex and where $\mathbf{D}$ is relabelled as the missing letter (if (32) was true then $\mathbf{D}$ would be relabelled as $\mathbf{C}$). This is where the switching of labels for the triangle case becomes useful; if no switch was made, then the vectors dotted with $\mathbf{DO}$ could point into the tetrahedron, instead of out of it, which would produce errors (see Fig.**??**). For each outcome here, the algorithm returns to the *support function* and *outside check*.

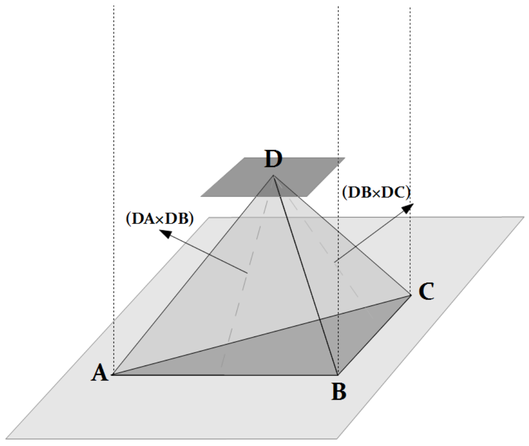

Figure 16. Schematic of a simplex with four points in it, forming a tetrahedron. The origin must lie above light plane on which $\mathbf{ABC}$ sits (this was established by the three point simplex checks) and below the dark plane at $\mathbf{D}$ (due to the 'outside check' after $\mathbf{D}$ was added to the simplex). The dashed lines define the prism that the origin cannot lie outside of - also established by the three point simplex step. The vectors $\mathbf{DA} \times \mathbf{DB}$ and $\mathbf{DB} \times \mathbf{DC}$ (and $\mathbf{DC} \times \mathbf{DA}$ which isn't shown for clarity's sake) will be used to determine where the origin is. This shows the importance of the counter-clockwise 'winding' of $\mathbf{ABC}$; had they been 'wound' clockwise, these vectors would point inwards.

In the case where all three checks return false, the origin must be contained within the tetrahedron. The algorithm will halt and return $true$ - the polyhedra are intersecting! This is the only point where the algorithm returns *true* however it can return *false* if and of the *outside checks* return *false*.

When making chains of hexagonal plates, the implementation of this algorithm is as follows: a hexagonal plate is placed at the origin and oriented randomly. Another hexagon is made at the origin and moved in a random direction, the GJK algorithm then checks whether the two plates are intersecting or not. If they are, then hexagonal plate is moved further along in the translation direction, if they are not, it is moved toward the origin along the translation vector. This is repeated until the hexagons are touching, but moving it one step further would cause the hexagons to not be touching. Basically, the hexagons can touch, but they must intersect as little as possible to give the most realistic ice crystals.

2) Controlling the Aggregation Index: The method just outlined gives very little control over the aggregation indices of the chains formed; because the orientation is random and the translation is in a random direction, the chains tend to end up with very low aggregation indices, i.e. they are very clumped up.

To remedy this, Thomas introduced the use of a bias for the direction of the translation vector. By using spherical polar coordinates and a probability distribution, the value of the θ component of the translation vector can be controlled. For chains, the required directions for the plates to be moved mainly along are the $\theta = 0$ and $\theta = 2\pi$ directions. This requires a probability distribution which varies between 0 and π with maxima at each end and a minima at exactly π. MATLAB contains a statistical toolbbox with a number of probability distributions that that make this possible. We settled on using a Beta distribtuion:

$$P(x) = \frac{x^{\alpha-1}(1-x)^{\beta-1})}{B(\alpha,\beta)} \tag{35}$$

where $P(x)$ is the probability as a function of x and α and β are constants and $B(\alpha,\beta)$ is the normalising factor. For $\alpha = \beta$ the distribution becomes symmetric about $x = 0.5$ and for $\alpha, \beta < 1$ the distribution dips in the middle; for $\alpha = \beta = 1$ the distribution is uniform. These are the conditions we want to set on the distribution. Values of x will be picked from the distribution with x related to θ by $\theta = \pi x$. From this, the directions of the translation vectors can be biased towards the positive and negative z directions with the degree of bias controlled by the values of α and β (see Fig.**??**).

Further, one can introduce rotations about the x, y and z axes for each hexagonal plate, so that we can also control how much the plates line up with one another. This gives further control of the aggregation index because if all the hexagons line up tip to tip then the highest aggregation index of 1 can be achieved. These rotations can also be controlled by the Beta distribution so that certain directions can be biased more than others.

The process of generating chains goes as follows: first the 'basic hexagon' is constructed about the origin and oriented according to a Beta distribution. Another hexagon is then added and oriented with the same beta biases. It is translated by a vector with completely random ϕ but a Beta distributed θ. The *GJK polyhedra* algorithm is used, as before, to give "just touching" hexagonal plates. This continues for all the hexagons in the chain until a certain number of plates n is reached. Once one chain is finished, the next chain is started with varied values of α and β - this so a wide range of AI values can be attained.

This way a number of different aggregation indices can be generated for each length of chain (length in the sense of number of constituent plates n). These chains can be stored in a $4D$ array - the first dimension stores the coordinates of a point, the second gives multiple points forming a plate, the third can hold other plates in the chain and the fourth can hold other chains. This way, one can loop through the array and find the capacitances of each of the chains.

When finding the capacitance of a chain, the *GJK distance algorithm* must be done for each plate in the chain, which increases the simulation time. The reason for this is that a chain of hexagonal plates won't be a convex polyhedra, but each hexagonal plate will be. This means the *GJK distance algorithm* won't work for the chain as a whole, but will for each constituent plate; this will give the same answer. The distances from each plate has to be stored in its own array and the minimum of them taken as the next step size.

V. Results and Discussion

A. Unit Cube

The unit cube is perhaps the polyhedra with the most accurately determined capacitance and therefore provides a useful test or method. A value of 0.6612 ± 0.0055 was

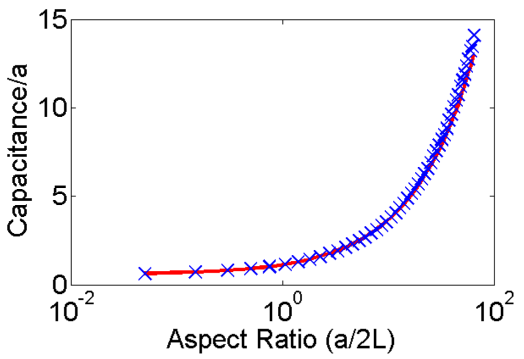

Figure 17. Graph showing the capacitance of hexagonal plates with $\mathcal{A} = \frac{1}{2}$ as their scalene parameter is varied between 0 and 1.

found after $100,000$ walkers. This compares well with the work of Hwang and Mascagni [4], who calculated it to be $0.66067813 \pm 1.01 \times 10^{-7}$. This gives reason to that believe our method is valid and our code is accurate.

B. The effect of R_∞ and δ

The effect of changing R_∞ between $500R$ to $1000R$ gives a change in capacitance of only $\approx 0.1\%$ but the increase in computation time was significant enough that $R_\infty = 500R$ was chosen to be optimum.

The absorbing layer δ was chosen to be less than 0.1% of the length of the shortest edge of each crystal. The small error that arises from this layer can be removed, but this is beyond the scope of this project [4].

C. Hexagonal Prisms with Varied Aspect Ratios

The aspect ratio of a regular hexagonal prism was varied between $\mathcal{A} = 0.5$ and $\mathcal{A} = 60$ and the capacitance found for each $\mathcal{A}$. A fit for the capacitance as a function of aspect ratio was found to be

$$C = 0.58\Big(1 + 0.936\mathcal{A}^{0.77}\Big). \tag{36}$$

with a reduced χ^2 of 1.153. This fit was found by the use of a $\log - \log$ graph. For comparison, the fit given by Westbrook et al. [12] is

$$C = 0.58\Big(1 + 0.95\mathcal{A}^{0.75}\Big)a. \tag{37}$$

Hexagonal columns are often approximated to cylinders and Smythe [10] gives the fit

$$C = 0.637\Big(1 + 0.868\mathcal{A}^{0.76}\Big)a_{cyl} \tag{38}$$

where a_{cyl} is the radius of the cylinder. Smythe's fit, however is only valid up to an aspect ratio of around 10. Notice that all three fits are of the same form.

Our fit and that of Westbrook et al. noticeably differ for large values of $\mathcal{A}$, however, we have far more data points at higher aspect ratios - Westbrook et al. use only three greater than $\mathcal{A} = 10$ whereas our results have over thirty with a very clear, unwavering trend which is followed exceptionally well by (36). It could be argued that because Westbrook used

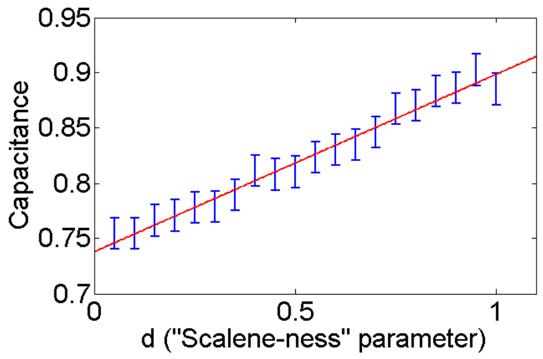

Figure 18. Graph showing the capacitance of hexagonal plates with $\mathcal{A} = \frac{1}{2}$ as their scalene parameter was varied between 0 and 1.The fit of $C = 0.74 + 0.16d$ has been plotted also.

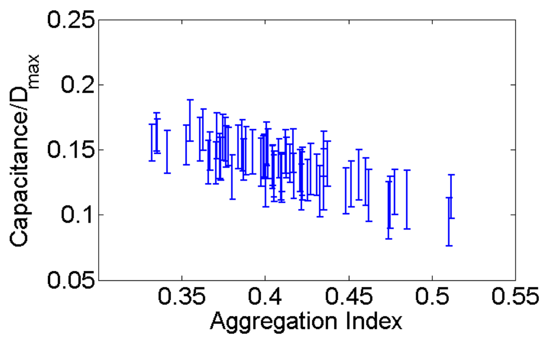

Figure 20. The normalised capacitance of chain-like aggregates again aggregation index. The plot shows a clear downward trend as aggregation index increases.

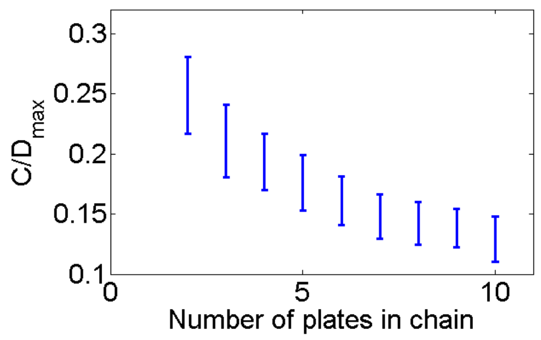

Figure 19. The normalised capacitance of chain-like aggregates with different aggregation indices against number of plates in the chain.

$250,000$ walkers to his $100,000$ we could expect his fit if we just used more walkers, but this is unlikely because if the fit was due to the capacitance not converging one would expect a noisier line; the final value of the capacitance can be approached from either above or below (it is, by nature, random). This means, if our values were yet to converge to the correct value they wouldn't all be clearly above the values of Westbrook. Similarly, the values for lower $\mathcal{A}$ would also be far noisier, yet they are also exceptionally smooth.

D. Varied scalene hexagons

The data for the scalene varied hexagonal plates is shown in Fig.**??**. An aspect ratio of $\mathcal{A} = \frac{1}{2}$ was used with $a = 1$. The capacitance here seems to vary linearly with d so the linear fit

$$C = 0.74 + 0.16d \tag{39}$$

was applied, with a reduced χ^2 of 0.27. This is quite a low reduced χ^2, but we believe this is due to the fact that too few walkers were used - only $10,000$. If more walkers were used, smaller errors and a more definite trend could be achieved.

E. Aggregates of hexagonal plates

The capacitance of roughly one thousand aggregates made up of between 1 and 10 hexagonal plates were found. These had all been generated using the method given above to give a wide spread in the aggregation index. All the generated plates had aspect ratios of 0.25, with $a = 1$ to best emulate the geometry of crystal chains.

The number of walkers used for each aggregate was $10,000$ which is far fewer than was used for the regular hexagonal plates, the reason for this is purely because of computational time; as the aggregate become more and more complex, the computation time increases dramatically. However, since multiple aggregates are used with various aggregation indices for each chain length, we expect the statistics to be roughly correct.

For all lengths of chain a scatter of capacitances against aggregation index was found. However, plotting the normalised capacitance (C/D_{max}) (where D_{max} is the maximum dimension of the aggregate) against aggregation index proved to be more illuminating. This showed a general trend of lower normalised capacitance for more extended chains (higher AI) - see Fig20.

When the chain length is plotted against normalised capacitance (see Fig.**??**) an unusual result is found. The normalised capacitance tends to decrease with the length of chain - a result contradictory to the findings of Westbrook et al. [12] and Field and Heymsfield [**?**]. Their findings indicate that the normalised capacitance tends to a constant value for larger aggregates.

It is possible, however, that the work of Westbrook et al. is not readily applicable to chain aggregates - it is difficult to get an idea of whether they used any chains at all or whether their investigations were only concerned with more “clumpy”, snowflake-like aggregates. Clumpier aggregates much more efficiently fill space than chains - much like the difference between a sphere and a cylinder.

As a tentative justification for our results, we can compare our results to that of a cylinder. A cylinder of similar dimensions to our chains can be constructed with $a_{cyl} = 1$ and $D_{max} = \sqrt{L^2 + (2a)^2}$. The justification for $a_{cyl} = 1$ is that in their most extended arrangement, a cylinder bounding a chain would have a diameter roughly equal to the maximum span of a the hexagonal face - 2 in this case, as $a = 1$.

Using the capacitance fit from (38) gives the normalised capacitance with length as

$$C/D_{max} = \frac{0.637}{\sqrt{L^2+4}}(1 + 0.868\frac{L}{2}^{0.76}) \tag{40}$$

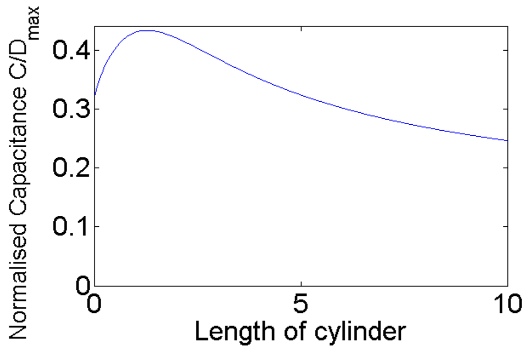

Figure 21. The normalised capacitance of a cylinder. Notice the similarity in trend to Fig.19.

which is shown in Fig.21. Since the length of a chain aggregate is proportional to the number of plates in its length a comparison to Fig.**??** seems valid. Westbrook et al. show that the capacitance of a cylinder must be larger than the shape it encloses [12], so a lower value for an actual chain is expected. Further investigation into the capacitance of chains with more plates would be necessary to show the validity of this approximation and whether C/D_{max} becomes asymptotic with higher more plates - as this approximation and Westbrook et al. suggest. From (40) we estimate that the normalised capacitance would asymptotically reach a value of $\approx 0.10 - 0.13$.

Regardless, this shows that a direct comparison of our results to other work on aggregates in the field may not be justified.

The approximation of a crystal aggregate to a sphere (where $C/D_{max} = 0.5$) has previously been shown to be flawed [12], and our results show that this approximation is yet less accurate for chains.

VI. Conclusion

A numerical method for calculating the capacitance of an abitrary polyhedra has been presented and we have written a number of MATLAB programs to apply this method. A program for generating chains has also been written such that the aggregation index can be controlled by a proabability distribution. These programs can, quite easily, be extended to other ice crystal geometries such as hollow crystals or columns, which are little explored in terms of capacitances.

New results for chain-like aggregates have been found and could be applicable to cloud physics in tropical storm clouds, but require fruther investigation. Our results for scalene crystals parameterised by a scalene-parameter is also a novel result of our project.

Most of the project time was spent writing and trouble-shooting the GJK algorithms used. This method is not at all mentioned in any of the literature on the Walk on Spheres method. Any information on the $3D$ version of the GJK distance algorithm is usually vague and sparse, meaning the execution of it in our code is entirely our own.

The programs written can be used, with little modification, for simulations where similar algorithms are used such as the hydrodynamic friction of arbitrary polyhedra [8].

The workload was split between the two of us such that the GJK algorithms were mainly worked on by myself and the Walk on Spheres algorithm and aggregation generators were written by my lab partner Thomas Hayes.

References

[1] P. Connolly and M. G. et al., "Aircraft observations of the influence of electric fields on the aggregation of ice crystals," *Q.J.R. Meteorol. Soc.*

[2] D. C. Emersic, "Cloud physics research - light scattering by ice crystals," http://www.cas.manchester.ac.uk/resactivities/cloudphysics/topics/lightscattering/, accessed: 2010-09-30.

[3] P. Hobbs, *Ice Physics*. Clarendon Press, 1974.

[4] C. Hwang, M. Mascagni, and T. Won, "Monte carlo methods for computing the capacitance of the unit cube," *Mathematics and Computer in Simulation.*

[5] P. Lindermann, "The gilbert-johnson-keerthi distance algorithm."

[6] J. Macdonald, "Use of the electrostatic analogy in studies of ice crystal growth," *Z. Angew. Math. Phys.*

[7] M. Mascagni and C.-O. Hwang, "Epsilon-shell error analysis for 'walk on spheres' algorithms," *Mathematics and Computers in Simulation*, vol. 63.

[8] S. Northrup, S. Allison, and J. McCammon, "Brownian dynamics simulation of diffusion influenced biomolecular reactions," *The Journal of Chemical Physics.*

[9] H. Pruppacher and J. Klett, *Microphysics of Clouds and Precipitation.* Kluwer Academic Publishers, 1997.

[10] W. Smythe, "Charged right circular cylinder," *J. Appl. Phys.*, vol. 33.

[11] J. Um and G. McFarquhar, "Single-scattering properties of aggregates of plates," *Q. J. R. Meteorol. Soc.*, vol. 135.

[12] C. Westbrook, R. Hogan, and A. Illingworth, "The capacitance of pristine ice crystals and aggregate snowflakes," *JAS.*

[13] D. Wilson and S. Ballard, "A microphysically based precipitation scheme for the uk meteorological office unified model," *Q. J. R. Meteorol. Soc.*, vol. 125.

[14] ——, "A microphysically based precipitation scheme for the uk meteorological office unified model," *Q. J. R. Meteorol. Soc.*, vol. 125.

[15] C. Woods, M. Stoelinga, and J. Locatelli, "The improve-1 storm of 1-2 february 2001. part iii: Sensitivity of a mesoscale model simulation to the representation of snow particle types and testing of a bulk microphyscal scheme with snow habit prediction," *Journal of the Atmospheric Sciences*, vol. 64.

[16] H.-X. Zhou, A. Szabo, J. Douglas, and J. Hubbard, "A brownian dynamics algorithm for calculating the hydrodynamic friction and and electrostatic capacitance of an arbitrarily shaped object," *J. Chem. Phys*, vol. 100.