Determining the conductance of networks created by randomly dispersed cylinders

S. Appella [1], A. Atayev [2], O. G. Bond [3], B. Collins [4], M. Dablander [3], N. Fadeev [5], A. A. Lacey [6], P. Morawiecki [1], H. Ockendon [3], D. Polvara [7], E. Powell [7]†, L. Quintavalle [8], E. Wilson [4] and Y. Zhou [1]

[1] *University of Bath*
[2] *University of Warwick*
[3] *University of Oxford*
[4] *University of Bristol*
[5] *Research Institute for Symbolic Computation*
[6] *Heriot–Watt University*
[7] *Durham University*
[8] *Deutsches Elektronen-Synchrotron*

(*Communicated to* MIIR *on 14 August 2021*)

Study Group: ESGI165, Durham University, 12-16 April 2021

Communicated by: Bernard Piette

Industrial Partner: Peratech Holdco Limited

Presenter: Tim Wiles

Team Members: S. Appella, University of Bath; A. Atayev, University of Warwick; O. G. Bond, University of Oxford; B. Collins, University of Bristol; M. Dablander, University of Oxford; N. Fadeev, Research Institute for Symbolic Computation; A. A. Lacey, Heriot–Watt; M. Lichota, University of Edinburgh; P. Morawiecki, University of Bath; H. Ockendon, University of Oxford; J. Ockendon, University of Oxford; D. Polvara, Durham University; E. Powell, Durham University; L. Quintavalle, Deutsches Elektronen-Synchrotron; M. Saragnese, Deutsches Elektronen-Synchrotron; C. Shi, Humboldt Univeristy Berlin; E. Wilson, University of Bristol; Y. Zhou, University of Bath.

Industrial Sector: Electronics

Tools: Finite volume simulation, Geogebra, MATLAB (Robotics Toolbox), Octree, OpenFOAM, Python, Solving Laplace Equation, Spatial trees

Key Words: Electric Networks, Resistivity, Cylinder Packing, Quantum Tunnelling, Homogenisation

MSC2020 Codes: 78A25; 78M12; 81U26; 78M40; 05C90

† Corresponding Author: `ellen.g.powell@durham.ac.uk`

Summary

This report investigates various aspects of modelling a composite material using randomly packed cylinders in 3d space. In particular it considers: simulating configurations of non-intersecting cylinders; generating weighted networks from such simulations, including calculation of minimal cylinder separation distances; estimating the conductance or resistivity of such a network. This is based on work carried out during the ESGI165 Study Group at Durham University, April 2021.

1 Introduction

1.1 Background and problem

An aspect of Peratech's sensing technology relies upon understanding connectivity properties of collections of cylindrical objects (i.e. particles) placed in a box. To this end, the company have posed the following problem.

1.1.1 *The initial question*

A configuration of a finite number of identical, uniform density cylinders are placed in a box $B \subset \mathbf{R}^3$ of arbitrary length, width, and depth. For this initial question, let us suppose that the configuration is *given*; we will come back to the problem of generating the configuration and its properties later on. The cylinders cannot overlap, and have identical diameter D and length L. The centre of mass (X, Y, Z) and rotation (θ, ϕ) of each cylinder is known. Rotations around the cylinder axes are irrelevant to this problem, due to symmetry, leading to only the five degrees of freedom above.

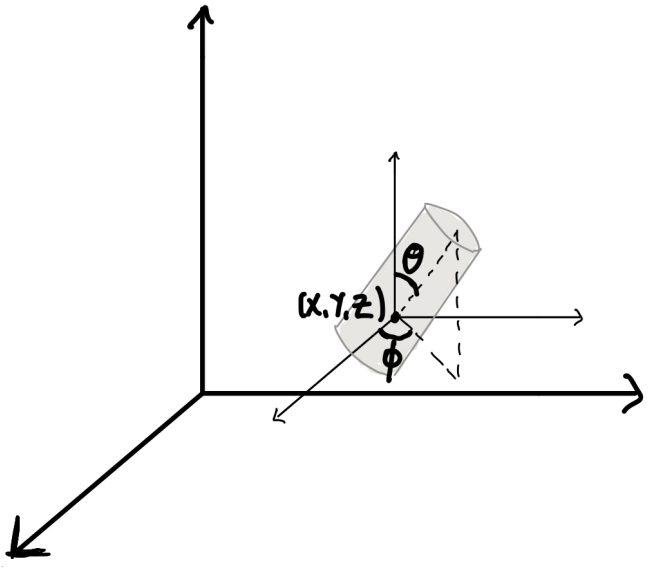

Figure 1. An illustration of a single cylinder described by (X, Y, Z, θ, ϕ).

Two cylinders are said to have a *connection* if the minimum distance between them is smaller than a cut-off length, l_c. If a connection exists, we write $d_{\min}$ for the minimum distance between the objects. This also defines two *nodes*, one on each object, which are the points achieving this minimum distance.[1]. Note that this gives rise to a finite collection of nodes, resulting from all connected pairs of cylinders in the box. Nodes corresponding to "adjacent cylinders" also come with a natural associated distance. We

[1] Suppose that the configuration is such that these nodes are uniquely defined

call this collection of nodes (and associated distances) the *network* generated by the configuration.

Question
Given a list of (X, Y, Z, θ, ϕ) for each cylinder in the configuration, what is the most efficient way to produce the network above. That is, to give a complete description of the associated nodes, edges and distances. Peratech advised that a Python script demonstrating this process would be an ideal way to convey a successful solution of this problem.

1.1.2 *Additional questions*

- Using the network described above, or otherwise, how can global connectivity properties of the cylinder configurations be (efficiently) described?
- Peratech are also interested in simulating the cylinder configurations themselves. The final question is thus to consider methods for generating such configurations, given some information on desirable properties.

1.2 Group structure and outline

This problem attracted a large team, and the working group naturally divided into four distinct subsets. The results of these are described separately in Sections 2, 3, 4 and 5 below.

- Section 2 considers, broadly speaking, the initial question described above. In fact, the main challenge here turns out to be the calculation of precise minimal distances between given pairs of cylinders.
- Section 3 then addresses the question of calculating certain global connectivity properties of the configuration of cylinders. Numerical and analytical approaches are discussed, as well as two separate models corresponding to different assumptions on the material being represented by the cylinders.
- Section 4 is based on the problem of generating cylinder configurations. It also describes a method for generating the network given the cylinders, using an existing MATLAB toolbox.
- Finally, Section 5 describes a somewhat different perspective on the overall problem, focusing on average quantities based on a homogenisation assumption.

2 Networks from cylinder configurations

This first section focuses on converting a configuration of cylinders in a three-dimensional box of arbitrary size into a network configuration. This translation is done by:

- considering those cylinders whose minimal separation distance d_{min} is lower than a certain threshold l_c to be connected;
- adding two connected nodes to the network for each such pair, corresponding to the two points at minimal distance on these cylinder surfaces;
- associating this pair of connected nodes with the corresponding minimal distance; and
- further connecting any two nodes that belong to the same cylinder.

The first natural step to address in this task is to compute the minimal distances between cylinders. This ended up taking most of the working time during the ESGI study group, and the final step of converting spatial configurations to network configurations was not completed. With more time at our disposal, it would have been possible to combine our distance-computing algorithm with a *spatial trees* approach to efficiently produce network configurations.

Before moving to the main problem studied in this section, which is the generation of a function to compute the minimal distance between cylinders, we briefly discuss how to optimise the network generation *once* such a function is given. Let us consider a uniform distribution of cylinders within a three-dimensional box; computing the distance between each cylinder in the box and all the others requires a considerable amount of time. However this is unnecessary since most of the cylinders are far from one another, with distance between them bigger than the cut-off length l_c. To optimise the problem it is reasonable to split the original box, in which the cylinders are located, into smaller sub-boxes of size L'. We say that a cylinder belongs to a certain sub-box, let call it sub-box (1), if the centre of mass of the cylinder is located inside (1). We can then properly tune the parameter L' in a way such that cylinders lying in boxes that do not touch each other are automatically at a distance greater than the cut-off length l_c. As a result, for the given sub-box (1), we only need to compute the distance of the cylinders in (1) from one another and the distance between the cylinders in (1) and those living in the 26 sub-boxes surrounding (1).

To properly tune the sub-box length L' we present the following argument. Suppose there are two cylinders of length L and radius R with centres of mass O_1 and O_2 in two non-neighbouring boxes (1) and (3) as depicted in Figure 2. In this configuration the shortest distance that can be reached between the two cylinders maintaining their centres of mass in the boxes (1) and (3) is given by

$$d_{min} = L' - 2\sqrt{\left(\frac{L}{2}\right)^2 + R^2}$$

which corresponds to the red line in Figure 2. We tune the box size L' by requiring that such minimal distance cannot be smaller than the cut-off length l_c. This is achieved by choosing the length L' for the sub-box to satisfy

$$L' \geq l_c + 2\sqrt{\left(\frac{L}{2}\right)^2 + R^2}.$$

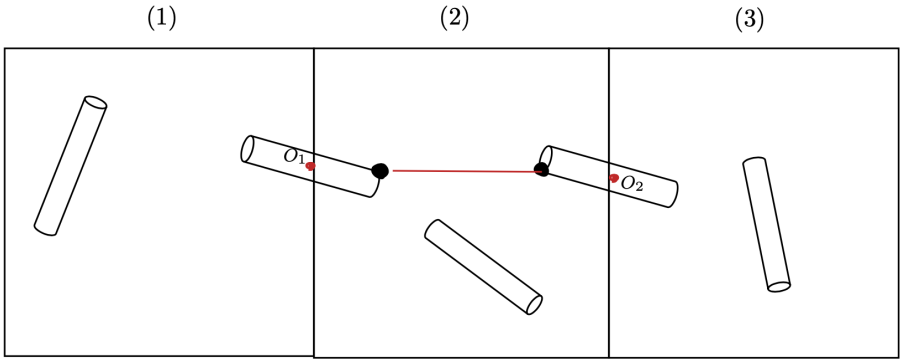

Figure 2. Minimum distance between two cylinders in boxes that do not touch each other.

In this manner, it is possible to considerably reduce the number of cylinder distance calculations required. As mentioned previously, we did not write any code for this during the study group, but believe it should be relatively straightforward compared to the distance calculation problem that we did address.

This leads us onto the core of the section, which is the implementation of a function to compute the minimal distance between cylinders. In the following subsections we will describe the algorithm we coded. The overall structure is partially based on the one suggested in [1].

2.1 Overview of the algorithm

A cylinder in a three-dimensional space is identified by its radius, R, and by the positions of the end points of its symmetry axis. It can therefore be represented by 3 elements $(R, \mathbf{a}, \mathbf{b})$; in this case R represents the radius of the cylinder while $\mathbf{a}$ and $\mathbf{b}$ are the coordinates in $\mathbb{R}^3$ of the symmetry axis end points. We will refer to the surface parallel to the cylinder axis as the cylinder *shell*, while the rest of the surface is made of two disks, whose boundaries are two circles. The present program receives as input data a pair of cylinders $(R_1, \mathbf{a_1}, \mathbf{b_1})$ and $(R_2, \mathbf{a_2}, \mathbf{b_2})$, defined as above, and returns as an output the minimum distance between them together with the pair of points on their surfaces achieving this minimal distance, that will correspond to the *nodes* of the network.

To produce such a minimum distance, and the corresponding nodes, we prepared four different sub-functions:

- Line-line distance,
- Line-circle distance,
- Circle-disk distance,
- Circle-circle distance,

which are called in the algorithm to find out the minimum value.

The logic we followed is summarized in the flow chart of Figure 3, which we will now describe in more detail. The first step is to compute the *line-line distance* between the

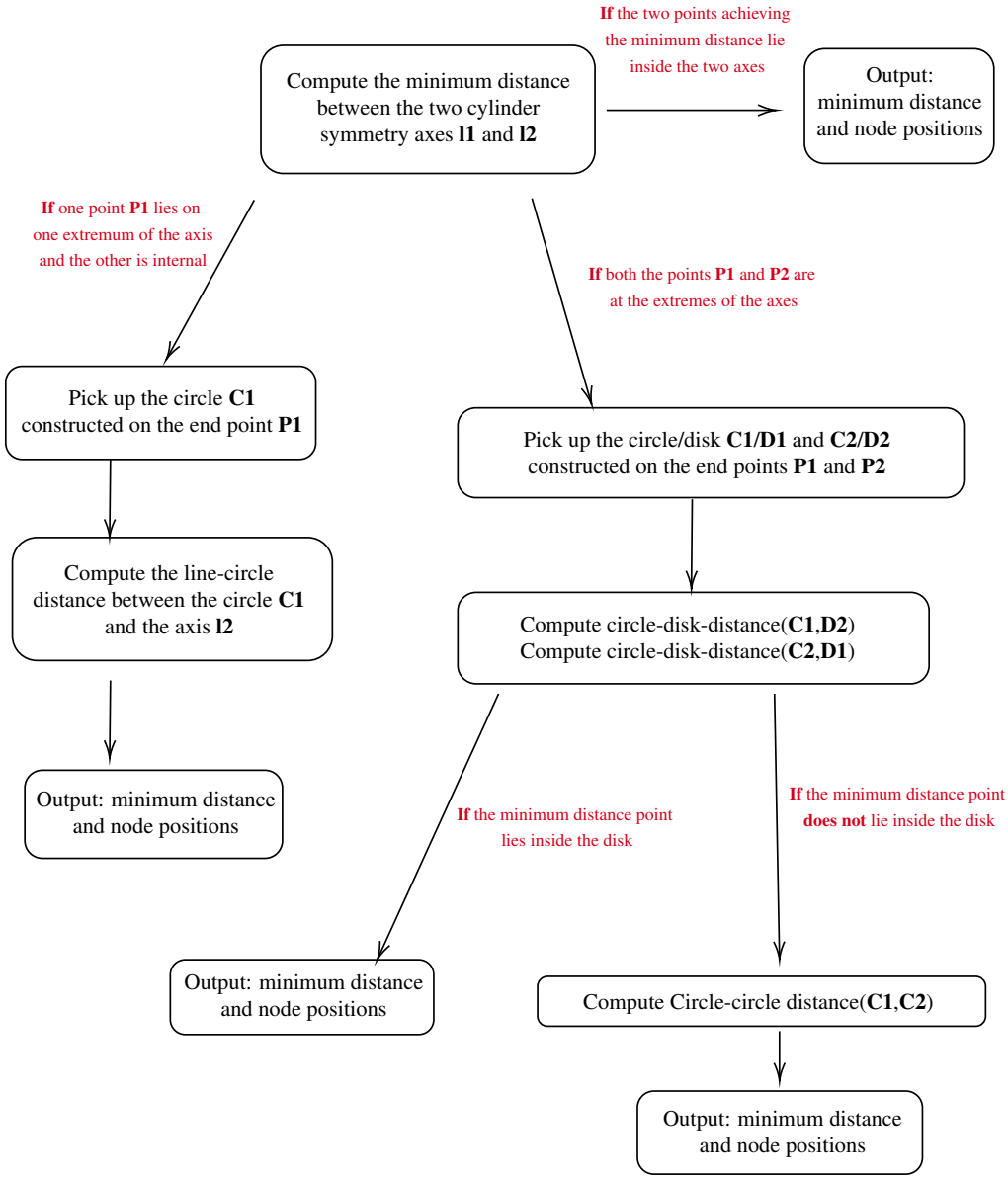

Figure 3. Flow chart of the algorithm for the computation of the minimal distance between a pair of cylinders and the associated nodes.

two axes of the cylinders, which leads to three possible distinct outcomes, depending on the location of the minimal distance points.

a) If the minimal distance points lie within the interior of both axes, the two cylinders are then in a configuration such that the minimal distance is reached between the two cylinder shells. This is easily computed by taking the already computed minimal line-line distance and subtracting the two radii from it. See the left-most path indicated on Figure 3, and also Section 2.2 below for more details.

b) If one of the minimal distance points coincides with an extreme of the corresponding cylinder axis, and the other lies within the interior of the other axis, then the minimal distance is reached between the circle centered on the point lying on the extreme, and the shell of the second cylinder. This case is then computed using the *line-circle distance* function and subtracting the radius of the cylinder from this result.

c) If both minimal distance points coincide with extremes of the cylinder axes, then the minimal distance is achieved either between one disk and one circle centered on these extremes, or between the two circles centered on these same extremes. To discriminate between these two cases, the *circle-disk distance* function is executed for both combinations of disk and circle. This function returns the point of minimum distance between the circle and the plane of the disk, together with its projection on the plane and the distance between these two. If the projection on the plane of the minimum distance point lies inside the disk, then the distance produced by the previous function also corresponds to the minimum distance between the two cylinders. If, instead, the minimum distance point lies outside of the disk for both circle-disk pairs, it means that the minimum distance between cylinders must be computed with the remaining *circle-circle distance* function.

In the following subsections we describe our implementation of the four distance functions, alongside some theoretical explanation of these.

2.2 Line-line distance

Consider two finite line segments in three-dimensional Euclidean space, $L_{(a,b)}, L_{(c,d)} \subset \mathbb{R}^3$, which are given by their respective endpoints, $(a,b),(c,d) \in \mathbb{R}^3 \times \mathbb{R}^3$. We assume that both line segments are nonempty and do not intersect, i.e. $a \neq b$, $c \neq d$ and $L_{(a,b)} \cap L_{(c,d)} = \emptyset$. In the context of our problem we think of both line segments as the axes of two respective cylinders with common radius $R > 0$. We are interested in computing the minimal distance between $L_{(a,b)}$ and $L_{(c,d)}$ as well as the pair of critical points $(p,q) \in L_{(a,b)} \times L_{(c,d)}$ at which this minimal distance is attained. Figure 4 depicts this situation.

As mentioned in the previous subsection, knowing the points p, q associated with the minimal distance between two cylinder axes will help us to find the overall minimal shell-to-shell distance between the cylinders. Indeed in case (a) above, when both critical points p and q fall into the interior parts of their respective line segments, the distance vector $p - q$ is perpendicular to the directions of both line segments $b - a$ and $d - c$. If this is the case, then we can easily find the minimal shell-to-shell distance between both cylinders by computing $\|p - q\| - 2R$. The associated critical points on both cylinder shells can further be found via

$$p^{\text{shell}} = p + R\frac{(q-p)}{\|q-p\|}, \qquad q^{\text{shell}} = q + R\frac{(p-q)}{\|p-q\|}.$$

On the other hand, if one or both of the points p, q coincide with an endpoint of a line segment, then we are in scenario (b) or (c) above, and a different method will be needed to calculate the shell-to-shell distance between the cylinders. These cases are treated in the following subsections, and corresponds to the right half of Figure 3. The critical points p, q which describe the minimal distance between the axes of two cylinders thus

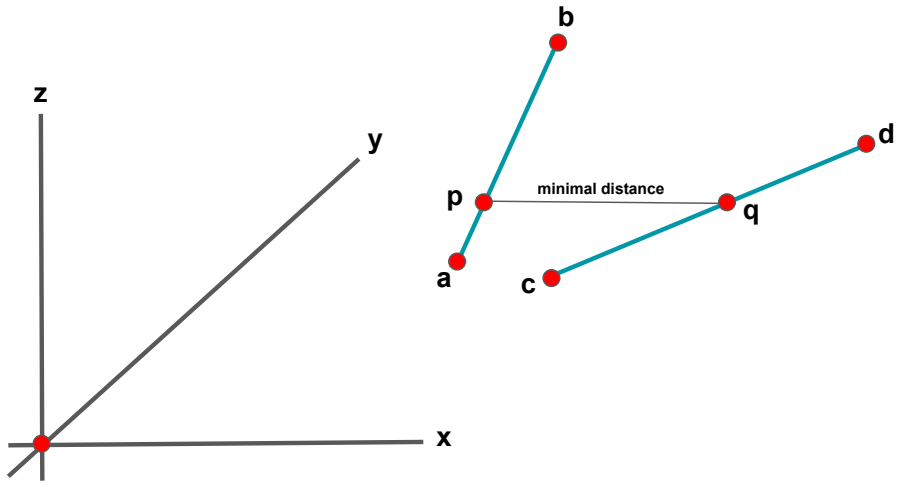

Figure 4. The points p, q which minimise the Euclidean distance between two finite line segments $L_{(a,b)}$ and $L_{(c,d)}$ in $\mathbb{R}^3$.

deliver a first high-level criterion which can be used to determine which computational path to follow in our algorithm for the computation of the minimal shell-to-shell distance between two cylinders.

To determine the shortest distance between $L_{(a,b)}$ and $L_{(c,d)}$, we start by parameterising both line segments via

$$L_{(a,b)} = \{a + \lambda(b-a) \mid \lambda \in [0,1]\}, \qquad L_{(c,d)} = \{c + \mu(d-c) \mid \mu \in [0,1]\}.$$

We can then define a function

$$f : [0,1]^2 \to \mathbb{R}_{>0}, \quad f(\lambda,\mu) = \|(a + \lambda(b-a)) - (c + \mu(d-c))\|^2,$$

which describes the squared Euclidean distance between two points on our line segments as a function of their associated parameter values. Finding the critical points (p, q) which minimise the distance between both line segments can thus be reduced to the problem of minimising f over its domain $[0,1]^2$. We first simplify f by using standard properties of the scalar product in $\mathbb{R}^3$:

$$\begin{gathered} f(\lambda,\mu) = \|(a + \lambda(b-a)) - (c + \mu(d-c))\|^2 = \\ \langle a + \lambda(b-a)) - (c + \mu(d-c)) \mid a + \lambda(b-a)) - (c + \mu(d-c))\rangle = \\ \langle \lambda(b-a) + \mu(c-d) + a - c \mid \lambda(b-a) + \mu(c-d) + a - c\rangle = \\ \|b-a\|^2\lambda^2 + \|d-c\|^2\mu^2 + 2\langle b-a \mid c-d\rangle\lambda\mu + \\ 2\langle b-a \mid a-c\rangle\lambda + 2\langle a-c \mid c-d\rangle\mu + \|a-c\|^2 =: \\ \theta_1\lambda^2 + \theta_2\mu^2 + \theta_3\lambda\mu + \theta_4\lambda + \theta_5\mu + \theta_6. \end{gathered}$$

This calculation reveals that f is a two-dimensional polynomial in λ and μ with constant coefficients $\theta_1, ..., \theta_6 \in \mathbb{R}$. Since f is a continuous function on the compact domain $[0,1]^2$,

there must exist at least one critical point $(\lambda_p, \mu_q) \in [0,1]^2$ which minimises f. It is feasible to find (λ_p, μ_q) using solely analytical methods. To do so,

- we collect the corner points of the domain $\{(0,0),(0,1),(1,0),(1,0)\}$ as candidate points for (λ_p, μ_q),
- we look for minimisers of f when restricted to the sides of its domain $\{(\lambda,0) \mid \lambda \in (0,1)\}, \{(\lambda,1) \mid \lambda \in (0,1)\}, \{(0,\mu) \mid \mu \in (0,1)\}, \{(1,\mu) \mid \mu \in (0,1)\}$ and collect these minimisers (if they exist) as candidate points for (λ_p, μ_q),
- we look for a minimiser of f when restricted to the interior of its domain $(0,1)^2$ and collect it (if it exists) as a candidate point for (λ_p, μ_q), and
- we compute the value of f at all previously collected candidate points for (λ_p, μ_q) and define the point with the lowest value to be a global minimiser (λ_p, μ_q) of f.

To find the potential minimiser of f when restricted to the side $\{(\lambda,0) \mid \lambda \in (0,1)\}$, we compute

$$\frac{d}{d\lambda} f(\lambda, 0) = 0 \iff 2\lambda\theta_1 + \theta_4 = 0 \iff \lambda = -\frac{\theta_4}{2\theta_1}.$$

If $-\frac{\theta_4}{2\theta_1} \in (0,1)$, then we collect $(-\frac{\theta_4}{2\theta_1}, 0)$ as a candidate point for (λ_p, μ_q), otherwise we ignore it due to it lying outside of the feasible set $[0,1]^2$. We further identify the potential candidate points $(-\frac{\theta_3+\theta_4}{2\theta_1}, 1), (0, -\frac{\theta_5}{2\theta_2}), (1, -\frac{\theta_3+\theta_5}{2\theta_2})$ by analogously investigating the other three sides $\{(\lambda,1) \mid \lambda \in (0,1)\}, \{(0,\mu) \mid \mu \in (0,1)\}, \{(1,\mu) \mid \mu \in (0,1)\}$ of the square $[0,1]^2$.

To identify the potential candidate point in the interior of the domain $(0,1)^2$, we need to solve

$$\nabla f(\lambda, \mu) = 0.$$

This results in the following linear system:

$$\begin{aligned} 2\theta_1\lambda + \theta_3\mu + \theta_4 &= 0, \\ \theta_3\lambda + 2\theta_2\mu + \theta_5 &= 0. \end{aligned}$$

By studying the determinant $4\theta_1\theta_2 - \theta_3^2$ of the matrix associated with this linear equation, one can show that the system is degenerate if and only if the line segments $L_{(a,b)}$ and $L_{(c,d)}$ are parallel. In this case, the (then non-unique) global minimiser (λ_p, μ_q) can always be chosen to lie on one of the sides or corners of the domain $[0,1]^2$ and one can thus ignore the interior points $(0,1)^2$. However, if the linear system above is not degenerate then its explicit solution can be found using Cramer's rule:

$$\lambda = \frac{\theta_3\theta_5 - 2\theta_2\theta_4}{4\theta_1\theta_2 - \theta_3^2}, \qquad \mu = \frac{\theta_3\theta_4 - 2\theta_1\theta_5}{4\theta_1\theta_2 - \theta_3^2}.$$

One needs to check whether this solution lies in $(0,1)^2$ and if this is indeed the case then one has to collect it as a candidate point for (λ_p, μ_q).

The above procedures result in a maximum number of nine candidate points for the global minimiser (λ_p, μ_q) of f on $[0,1]^2$. Computing the value of f at each candidate point and setting the point with the lowest value to be (λ_p, μ_q) finally delivers the critical points

$$p = a + \lambda_p(b-a) \in L_{(a,b)}, \qquad q = c + \mu_q(d-c) \in L_{(c,d)},$$

which minimise the distance between $L_{(a,b)}$ and $L_{(c,d)}$. Consequently, the minimal distance between both line segments can simply be obtained by calculating $\|p - q\|$.

A computationally efficient `Python`-implementation of this method for the computation of the points p, q associated with the shortest distance between two line segments in three dimensional space can be found in the GitHub repository associated with this project.

2.3 Line-circle distance

In principle, the shortest distance between a line and a circle can be computed in a very similar way to that between two line segments. This is because both objects in each pair are two-dimensional sets in a three-dimensional space, and therefore a point on an object in each pair can be stated in terms of one parameter (typically arc length). The Euclidean distance between a pair of points on each object can then be given as a function of two parameters, thus forming the basis of a multivariable optimisation problem.

Rather than starting with first principles, we outline the shortest circle-line distance result as provided in [2], which in turn considers the shortest distance between a point and a circle as given in [3].

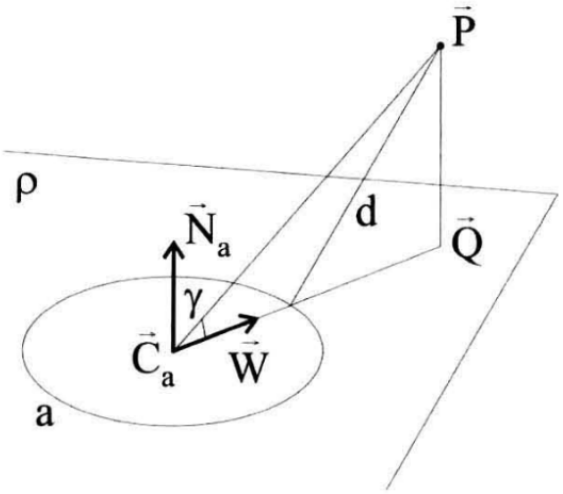

Figure 5. A diagram illustrating the geometry behind the point-circle calculation, taken from [2].

The shortest distance d between the point and the circle as shown in Figure 5 is given by

$$d^2 = R_a^2 + |\mathbf{C}_a - \mathbf{P}|^2 - 2R_a|\mathbf{P} - \mathbf{C}_a - (\mathbf{N}_a \cdot (\mathbf{P} - \mathbf{C}_a) \cdot \mathbf{N}_a)| \tag{2.1}$$

Parametrising the line segment as $\mathbf{P}(t) = \mathbf{r}_0 + t\mathbf{d}$, and substituting this into equation (2.1), one can recover the square of the distance between the points on the line and circle that minimise this distance. That is,

$$f(t) = a_6 t^2 + a_5 t + a_4 + a_3\sqrt{a_2 t^2 + a_1 t + a_0}, \tag{2.2}$$

where the coefficients are given by

$$a_6 = \mathbf{M}, \cdot\mathbf{M} \qquad a_5 = 2, \mathbf{D} \cdot \mathbf{M} \qquad a_4 = |\mathbf{D}|^2 + R_a^2, \tag{2.3}$$

$$a_3 = -2R_a, \qquad a_2 = |\mathbf{E}|^2, \qquad a_1 = 2\mathbf{E} \cdot \mathbf{F}, \qquad a_0 = |\mathbf{F}|^2, \tag{2.4}$$

and

$$\mathbf{D} = \mathbf{B} - \mathbf{C}_a, \qquad \mathbf{E} = \mathbf{M} - (\mathbf{N}_a \cdot \mathbf{M})\mathbf{N}_a, \qquad \mathbf{F} = \mathbf{D} - (\mathbf{N}_a \cdot \mathbf{D}) \cdot \mathbf{N}_a. \tag{2.5}$$

The minimum distance can then be found by minimising $f(t)$ in equation (2.2). This can be done numerically by applying, for example, the Newton-Raphson method to $f'(t)$. This is because we are looking for where $f'(t) = 0$, that is, where is $f(t)$ minimised.

As well as incorporating the above approach into Python, we developed an interactive diagnostic tool in a package called GeoGebra (www.geogebra.org). This is an educational tool designed for maths teachers in secondary schools, but it can also be used to develop applets for other purposes. In particular, as shown in Figure 12, we managed to use this platform in order to compile an app allowing the user to move around points in order to change the geometry of the lines and/or circles, and generate test cases with coordinates, radii and approximate shortest distances and critical points. The reason for the approximate nature of the shortest distances from this package is because it is not optimised to be able to perform calculations in more than one dimension; in particular, optimisation problems involving more than one variable. To rememdy this, we used GeoGebra's in-built spreadsheet functionality to take a finite number of sample points on each curve, and then compute the distance between them for all possible combinations of parameters on each curve. Although this was found to work well for numerous test cases, including when the circle and line intersect, this coarse numerical approach is likely to break down in cases where the circles have very large radius, as the sample points will become further apart. Despite these limitations, this diagnostic tool was highly useful in checking the Python code that was developed, and being able to easily generate cases where the shortest distance was known. This was positively received by our contacts at Peratech, and this tool in particular could be useful for the developers.

2.4 Circle-disk distance

In this part we discuss how to calculate the distance between a circle and a disk or two disks in three-dimensional Euclidean space. We first explain the definition of 3d circles and disks, and then present how to calculate the shortest distance between one disk and one circle analytically. At the end we present optimised methods using vectors to calculate the distance numerically. The pseudo-algorithm is presented at the end of this subsection.

A disk in three dimensions is defined as the following set,

$$\{x \in \mathbb{R}^3, \quad (\mathbf{x} - \mathbf{C}) \cdot \hat{\mathbf{n}} = \mathbf{0}, \quad |\mathbf{x} - \mathbf{C}| \leq r\}, \tag{2.6}$$

where $\mathbf{C} \in \mathbb{R}^3$ is the vector for the disk circle centre, $\hat{\mathbf{n}}$ is the normal vector of the disk surface, and $r > 0$ is the scalar of disk radius.

A circle in three dimensions is defined as the following set,

$$\{x \in \mathbb{R}^3, \quad (\mathbf{x} - \mathbf{C}) \cdot \hat{\mathbf{n}} = \mathbf{0}, \quad |\mathbf{x} - \mathbf{C}| = r\}, \tag{2.7}$$

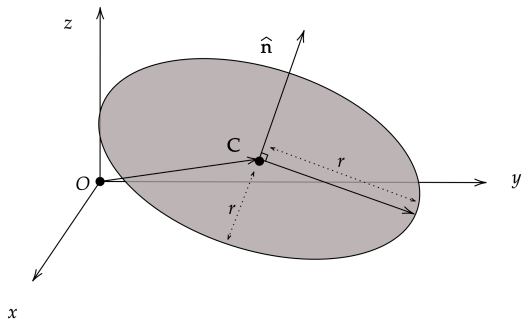

Figure 6. Graph of a 3d disk.

where $\mathbf{C}$ is the vector for the circle centre, $\hat{\mathbf{n}}$ is the normal vector of the circle surface, and r is the scalar of the circle radius.

All of those points that are defined by the set (2.6) form a 3d disk as shown in Fig (6). Additionally, all of those points that satisfy (2.7) form a 3d circle as shown in Fig (7).

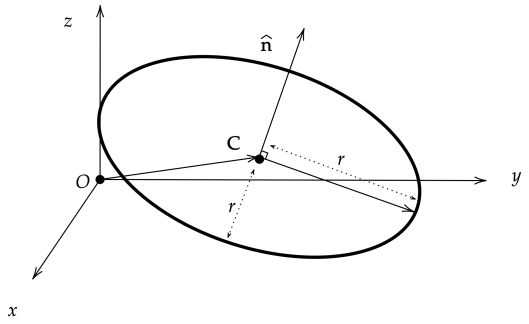

Figure 7. Graph of a 3d circle.

Now let us suppose that we have the two objects: a disk as in (2.6) with $\mathbf{C} =: \mathbf{C}_d, \hat{\mathbf{n}} =: \hat{\mathbf{n}}_\mathbf{d}, r =: r_d$ and a circle as in (2.7) with $\mathbf{C} =: \mathbf{C}_c, \hat{\mathbf{n}} =: \hat{\mathbf{n}}_\mathbf{c}, r =: r_c$. In order to find the minimal distance between the two, we will rotate and translate our system so that the disk is centred at the origin and lying in the x, y plane. This is illustrated in Figure (8).

Let us denote by $\mathbf{x_d}, \mathbf{x}_c$ the sets (2.6),(2.7) respectively, with $\mathbf{C} = \mathbf{0}$ and $\hat{\mathbf{n}} = (0, 0, 1)^T$ in both cases, and radii r_d, r_c respectively. Our disk and circle can then be described in the original coordinates by:

$$C_d = R_1 \mathbf{x_d} + \mathbf{C_d} \tag{2.8}$$

$$C_c = R_2 \mathbf{x_c} + \mathbf{C_c}, \tag{2.9}$$

where R_1 is the rotation matrix that takes $(0, 0, 1)^T$ to $\hat{\mathbf{n}}_d$ and R_2 takes $(0, 0, 1)^T$ to $\hat{\mathbf{n}}_c$.

Therefore, to rotate the system as in Figure (8), we need to do a translation by $-\mathbf{C}_d$ followed by the rotation R_1^T. In the new coordinate system, the disk and the circle are thus described as follows:

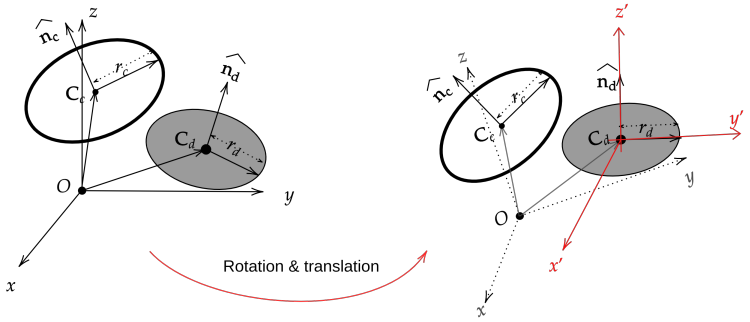

Figure 8. Rotate and translate the original coordinates so that the new origin is located at the disk centre and the new z-axis points in direction $\hat{\mathbf{n}}_\mathbf{d}$.

$$C'^{d} = \mathbf{x_d} \tag{2.10}$$

$$C'^{c} = R_1^T R_2 \mathbf{x_c} + R_1^T (C_c - C_d) \tag{2.11}$$

Having done this rotation, we now need to consider 3 main cases as shown in Figure (9). The projection of the lowest point of the circle to the plane of the disk determines the existence of the shortest distance. The ideal case is shown in a) of Figure (9), where the projection of the circle overlaps with the disk and the lowest point is projected within the disk. The shortest distance is then z_{low}.However, even when the projection of the circle overlaps with part of the the disk, we might not have the lowest point projected within the disk, as shown in b) of (9). When this happens, we can swap the circle and disk and use the analysis from a) to find the new shortest distance. When the projection of the circle does not overlap with the disk at all, we need to call the circle-circle algorithm to calculate the shortest distance. This is shown in c).

The analytical approach will be combining the calculation shown in equations (2.9), (2.8), (2.11) and (2.10). However, doing this numerically requires the construction of mesh points, and this is fairly costly in computing. We use the geometrical features of circles and disks to find the high point and the low point as follows in (2.13) and (2.14):

$$\mathbf{v} = r_c \frac{(\hat{\mathbf{n}}_\mathbf{c} \times \hat{\mathbf{n}}_\mathbf{d} \times \hat{\mathbf{n}}_\mathbf{c})}{||\hat{\mathbf{n}}_\mathbf{c} \times \hat{\mathbf{n}}_\mathbf{d} \times \hat{\mathbf{n}}_\mathbf{c}||} \tag{2.12}$$

$$\mathbf{x_{high}} = \mathbf{C_c} + \mathbf{v}, \tag{2.13}$$

$$\mathbf{x_{low}} = \mathbf{C_c} - \mathbf{v}, \tag{2.14}$$

where $\mathbf{v}$ gives the radius vector pointing from the circle centre to highest point $\mathbf{x_{high}}$, so the vector for $\mathbf{x_{high}}$ can be obtained by adding $\mathbf{v}$ to the centre $\mathbf{C_c}$. The coordinate of $\mathbf{x_{low}}$ can be obtained by subtracting $\mathbf{v}$ from $\mathbf{C_c}$.

The **pseudo code** for the *circle-disk distance algorithm* is as follows:

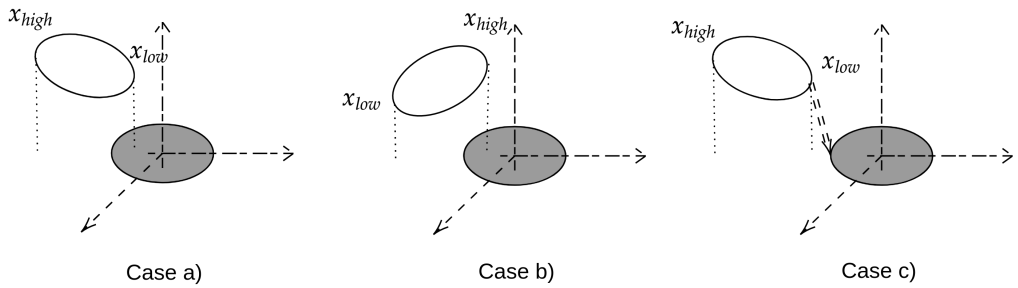

Figure 9. Three cases for finding the shortest distance between a circle (white colour) and a disk (grey colour). Case a): the circle has the lowest point projected within the disk radius, in this case z_{low} is the shortest distance. Case b): the highest point is projected within the disk, but the shortest point is outside. Then we need to swap circle and the disk to find the distance like in case a). Case c): all of the points from the circle after projection are outside of the disk. In this case we need to use the circle-circle algorithm to find the shortest distance.

(1) Check if the Circle and the Disk are parallel:

- if C_c is projected on C_d, $d_{min} = z_c$;
- if C_c is not projected on C_d but within the disk, $d_{min} = z_{low}$;
- if C_c is not projected within the disk, call circle-circle distance.

(2) else if $\mathbf{x_{low}}$ is projected within the disk: $d_{min} = z_{low}$,

(3) else if $\mathbf{x_{low}}$ is not projected within the disk but the $\mathbf{x_{high}}$ is, then call the function by swapping the circle and disk objects.

(4) else if neither $\mathbf{x_{low}}$ nor $\mathbf{x_{high}}$ are projected within the disk, call circle-circle distance.

Along with the shortest distance, the location of the two points on the circle and the disk are calculated and returned as outputs. All of the codes are implemented in the method `distanceCircleDisk(C2)`, which is in the class `circle` in the Python file objects3D.py. The input C2 is considered as the circle and the object is the disk. By calling `C1.distanceCircleDisk(C2)`, we will receive a flag (**True** or **False**) indicating whether the shortest distance was found or not.

When the flag returns **True**, the shortest distance is found and the two location points

are returned in the output. If flag returns **False**, we need to swap the disk and circle by calling `C2.distanceCircleDisk(C1)`. If still shortest distance is not found, the algorithm will call the next method `distanceCircleCircle`, which will be explained in the following subsection.

2.5 Circle-circle distance

In the general algorithm for the computation of the distance between two cylinders, this is the last item. More precisely, when we are in a configuration where the closest objects are the endpoints of the axis for both cylinders, we fall in the disk-disk distance computation. If we confirm in this case that there is no overlap between both disks[2], we fall in the case where we have to compute the distance between two circles in three-dimensional space. In order to implement this computation, we were mostly inspired by the results of [2, Section 2], that we summarise here.

The configuration, represented in the Figure 10, is the following: in the (O, x, y, z) frame, we have two circles $\mathscr{C}_a$ and $\mathscr{C}_b$ of same radius r, with respective centres defined by the vectors $\mathbf{C_a}$ and $\mathbf{C_b}$, and respective normal (and normalised) vectors $\hat{\mathbf{n}}_\mathbf{a}$ and $\hat{\mathbf{n}}_\mathbf{b}$. We denote by $\hat{\mathbf{u}}$ and $\hat{\mathbf{v}}$, two vectors perpendicular to $\hat{\mathbf{n}}_\mathbf{b}$ such that $(\hat{\mathbf{n}}_\mathbf{b}, \hat{\mathbf{u}}, \hat{\mathbf{v}})$ is an orthonormal basis of $\mathbb{R}^3$. Any point of the circle $\mathscr{C}_b$ can be parametrised using a variable $t \in [-\pi, \pi)$, by

$$\mathbf{X}(t) = \mathbf{C_a} + r\left[\cos(t)\hat{\mathbf{u}} + \sin(t)\hat{\mathbf{v}}\right]$$

The formula that gives the minimal distance d between a circle $\mathscr{C}_a$ and a point $\mathbf{P}$, computed in [3] and recalled by [2], is the following:

$$d = \sqrt{r^2 + |\mathbf{C_a} - \mathbf{P}|^2 - 2r\,|\mathbf{P} - \mathbf{C_a} - [\hat{\mathbf{n}}_\mathbf{a} \cdot (\mathbf{P} - \mathbf{C_a})] \cdot \hat{\mathbf{n}}_\mathbf{a}|} \tag{2.15}$$

In the case $\mathbf{P} = \mathbf{X}(t)$, plugging this into (2.15) we get the distance squared as a function of t. More precisely we have the following function $f(t) := d^2(t)$:

$$\begin{aligned} f(t) = {} & a_9 \cos t + a_8 \sin t + a_7 \\ & + a_6 \sqrt{a_5 \cos^2 t + a_4 \sin^2 t + a_3 \cos t + a_2 \sin t + a_1 \cos t \sin t + a_0} \end{aligned} \tag{2.16}$$

where the coefficients a_i, $i = 1, \ldots, 9$ are defined as follows:

$$\begin{aligned} a_9 &= 2r\mathbf{C_b} \cdot \hat{\mathbf{u}} - 2r\mathbf{C_a} \cdot \hat{\mathbf{u}} \\ a_8 &= 2r\mathbf{C_b} \cdot \hat{\mathbf{v}} - 2r\mathbf{C_a} \cdot \hat{\mathbf{v}}, \\ a_7 &= 2r^2 + \mathbf{C_a} \cdot \mathbf{C_a} + \mathbf{C_b} \cdot \mathbf{C_b} - 2\mathbf{C_a} \cdot \mathbf{C_b}, \\ a_6 &= -2r, \\ a_5 &= -\left(r\hat{\mathbf{n}}_\mathbf{a} \cdot \hat{\mathbf{u}}\right)^2 \\ a_4 &= -\left(r\hat{\mathbf{n}}_\mathbf{a} \cdot \hat{\mathbf{v}}\right)^2 \\ a_3 &= 2r\mathbf{C_b} \cdot \hat{\mathbf{u}} - 2r\mathbf{C_a} \cdot \hat{\mathbf{u}} - 2r\left(\hat{\mathbf{n}}_\mathbf{a} \cdot \mathbf{C_b} - \hat{\mathbf{n}}_\mathbf{a} \cdot \mathbf{C_a}\right)\hat{\mathbf{n}}_\mathbf{a} \cdot \hat{\mathbf{u}} \\ a_2 &= 2r\mathbf{C_b} \cdot \hat{\mathbf{v}} - 2r\mathbf{C_a} \cdot \hat{\mathbf{v}} - 2r\left(\hat{\mathbf{n}}_\mathbf{a} \cdot \mathbf{C_b} - \hat{\mathbf{n}}_\mathbf{a} \cdot \mathbf{C_a}\right)\hat{\mathbf{n}}_\mathbf{a} \cdot \hat{\mathbf{v}}, \\ a_1 &= -2r^2 \hat{\mathbf{n}}_\mathbf{a} \cdot \hat{\mathbf{u}} \hat{\mathbf{n}}_\mathbf{a} \cdot \hat{\mathbf{v}}, \\ a_0 &= r^2 + \mathbf{C_b} \cdot \mathbf{C_b} + \mathbf{C_a} \cdot \mathbf{C_a} - 2\mathbf{C_a} \cdot \mathbf{C_b} - \left(\hat{\mathbf{n}}_\mathbf{a} \cdot \mathbf{C_b} - \hat{\mathbf{n}}_\mathbf{a} \cdot \mathbf{C_a}\right)^2 \end{aligned} \tag{2.17}$$

[2] In the correctly translated and rotated frame, as described in the section before.

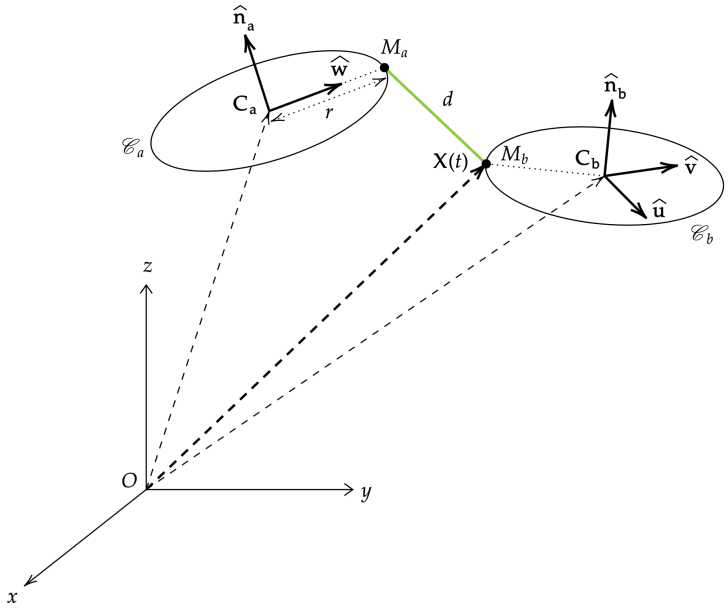

Figure 10. Detailed configuration of two circles $\mathscr{C}_a$ and $\mathscr{C}_b$ for which we want to compute the minimal distance d, obtained respectively between the points M_a and M_b of those circles.

The aim now is to find the global minimum of the function f that will correspond to the minimal distance between the circles. For that, the usual technique would be to compute the derivative $f'(t)$ and simply find its zeros. What happens nevertheless is that when we compute f' and do to the usual trigonometric-to-polynomial change of variable $Z = \cos(t)$, $\sin(t) = \sqrt{1 - Z^2}$, we end up getting a function that has a denominator of the following form:

$$\sqrt{a_2\sqrt{1 - Z^2} + a_0 + \cdots}$$

And at the same time terms proportional to Z^2 in the numerator. The usual trick to solve $f'(Z) = 0$ in this kind of situation is to square until no roots are left. In our situation, this means that we would have to square twice, which would lead to a polynomial of degree 8 in Z. Properly eliminating the square roots and solving numerically the $8^{\text{th}}-$degree polynomial being cumbersome[3] and leading to numerical losses, we decided to use the special numerical technique proposed by [2, Section 2.1].

The idea behind this technique consists, morally, of solving $f(t) = 0$ instead of $f(t) = f_{min}$ where f_{min} is the global minimum of f. This is simpler since we do not require the

[3] One quick attempt at eliminating the square roots in the general case through use of Gröbner basis techniques in Mathematica ended up taking too much time to be efficient.

explicit derivative of f; having only factors proportional to Z outside of its root[4], will lead to a polynomial of degree 4 instead of 8 when we get rid of the roots. More precisely, these are the steps that we have to undertake to find the minimum of f:

- Find one minimum t_m of f, it doesn't matter which one, on the interval $I = [-\pi, \pi)$. Indeed, by construction f will be well-defined on this interval and furthermore, $f = f(\cos t, \sin t)$ is $2\pi-$periodic, so we can restrain to the interval I. In Python, we use the method `minimize` of the function `optimize` from the package `scipy` to find such a minimum.
- Shift the function f by $D := f(t_m)$ by defining the function $g(t) := f(t) - D$.
- Since by construction g has a double zero at $t = t_m$, we can make a change of variables $g(t) \overset{Z=\cos(t)}{\longrightarrow} P_4(Z)$ where $P_4 := g \circ \arccos$ is polynomial of degree 4. The explicit form of this polynomial is given in [2] and is of the following form:

$$P_4(Z) = b_4 Z^4 + b_3 Z^3 + b_2 Z^2 + b_1 Z + b_0$$

Where the b_i, $i = 0, \ldots, 4$ are given by:

$$\begin{aligned} b_4 &= c_6^2 + c_5^2 \\ b_3 &= 2c_1c_5 + 2c_4c_6 \\ b_2 &= c_1^2 - c_5^2 + 2c_3c_6 + c_4^2 \\ b_1 &= -2c_1c_5 + 2c_3c_4 \\ b_0 &= c_3^2 - c_1^2 \end{aligned} \tag{2.18}$$

And the c_i, $i = 1, \ldots, 6$ are given by:

$$\begin{aligned} c_1 &= 2a_8\,(a_7 - D) - a_6^2 a_2 \\ c_2 &= a_6^2 a_4 \\ c_3 &= (a_7 - D)^2 - a_6^2 a_0 + a_8^2 - a_6^2 a_4 \\ c_4 &= 2a_9\,(a_7 - D) - a_6^2 a_3 \\ c_5 &= 2a_9 a_8 - a_6^2 a_1 \\ c_6 &= -a_8^2 + a_6^2 a_4 + a_9^2 - a_6^2 a_5 \end{aligned} \tag{2.19}$$

- Now, since P_4 admits a double zero at $Z_m = \cos(t_m)$, we actually only need to find both zeros of

$$P_2 = \frac{P_4}{(Z - Z_m)^2},$$

which can be done very easily numerically[5]. There are two possibilities at this point:

- If we find no real solutions to $P_2(Z) = 0$, it means that we already reached the global minimum, since that means that the second minimum is above the $t-$axis. Therefore, t_m is the minimum of f that we were searching for.
- If we find two distinct roots Z_a and Z_b[6], this means that the actual global minimum

[4] Using the same change of variables $Z = \cos t$ as before.

[5] In order to carry the polynomial division of P_4 by $(Z - Z_m)^2$ in Python, we use the `numpy` function `polydiv`, and then to find the roots in the form of a `numpy` array, we use the function `roots`.

[6] Given that we are in a configuration where we exclude parallel overlapping circles, a case that is treated in the circle-disk distance case, we can't have another double root, i.e. two different but equal minimal distances.

is located in the interval $[Z_a, Z_b]$, so we can actually search for it, using for instance once more the `minimize` method of the `optimize` function with a different starting point, say $t_a := \arccos(Z_a)$.

• In any case, we call the real global minimum $t_{\min}$, and plugging it into $f(t)$ gives us back $d = \sqrt{f(t_{\min})}$. The whole procedure is illustrated schematically for the case when the first minimum is local in the figure 11.

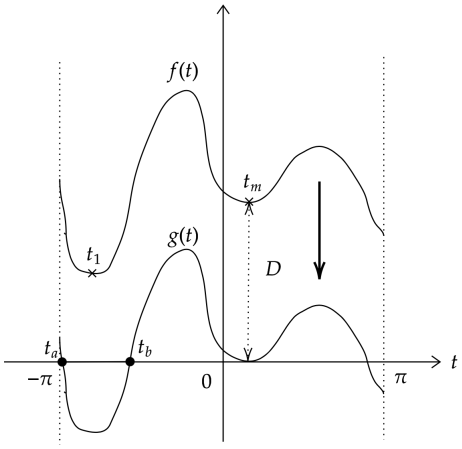

Figure 11. Schematic representation of the numerical computation of the minimum of f

In our case, we are nevertheless also interested in the points M_a and M_b of the circles $\mathscr{C}_a$ and $\mathscr{C}_b$ respectively that attain this minimum distance. For the circle $\mathscr{C}_b$, this is simply the point M_b with the same coordinates as the vector $\mathbf{X}(t_{\min})$. For the circle $\mathscr{C}_a$, in order to find this point, we have to compute a normal vector $\hat{\mathbf{w}}$ perpendicular to $\hat{\mathbf{n}}_\mathbf{a}$ that is in the same plane as $\hat{\mathbf{n}}_\mathbf{a}$ and $\mathbf{X}(t_{\min})$, since the point to minimise the distance on $\mathscr{C}_a$ will obviously be one that is the closest to M_b. Using cross products, we have that

$$\hat{\mathbf{w}} = \frac{\hat{\mathbf{n}}_\mathbf{a} \times \mathbf{X}(t_{\min}) \times \hat{\mathbf{n}}_\mathbf{a}}{\|\hat{\mathbf{n}}_\mathbf{a} \times \mathbf{X}(t_{\min}) \times \hat{\mathbf{n}}_\mathbf{a}\|}$$

Finally, the minimal point M_a on $\mathscr{C}_a$ is defined by the coordinates of the vector $\mathbf{C}_\mathbf{a} + r\hat{\mathbf{w}}$.

All this procedure is implemented in the method `distanceCircleCircle` of the class `circle` in the Python code. In order to compute the minimum distance between two circle objects `c1` and `c2`, it suffices to simply call `c1.distanceCircleCircle(c2)` in the Python console upon execution of the `objects3d.py` script.

2.6 Results and Conclusion

Results

In the overview section, we have explained how these four functions described above may be combined to give us the shortest distance between two cylinders in three-dimensional Euclidean space. Towards the end of the study group, we tested our algorithm with special cases. We calculated the distance between some random cylinders provided by Peratech and compared our result to the black box model used in the simulation group, which will be explained in Section (4).

We have also provided a case visualisation tool in GeoGebra which allows us to visualise special cases that we are interested in.One example is shown in Figure (12). This tool was developed and used in testing our algorithm.

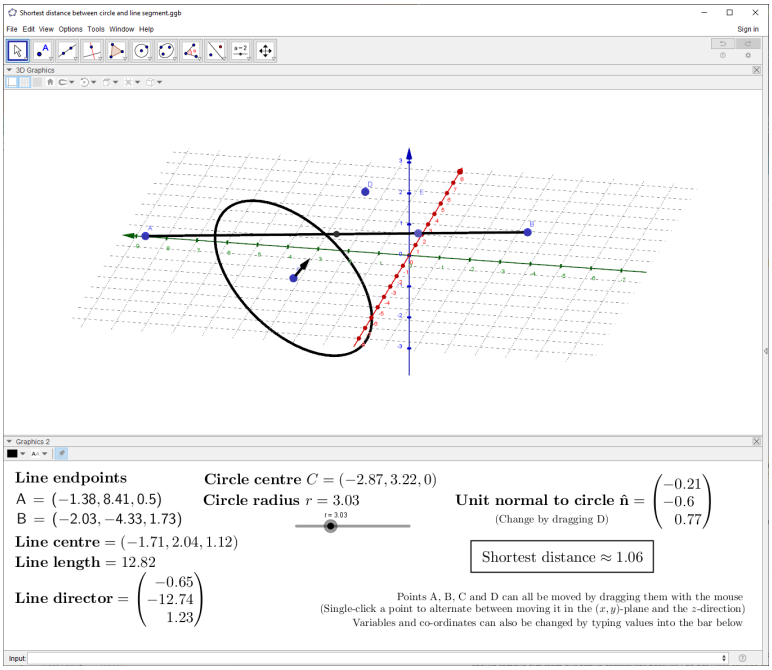

Figure 12. An illustrated interactive applet developed in GeoGebra (www.geogebra.org) which can be used to generate test cases for the circle-line distance.

Conclusion and Further work

- We have developed a fully working software to calculate distances, but this still needs to be combined with a spatial trees approach to generate the final desired network. We have no visualisation tools to present as yet.

- We also developed interactive tools for individual case analysis and case testing. The interactive tool could be extended to cylinders for a better result.

• We tested our algorithm on input data, in small amounts due to the time limit. As such, it remains to be tested on big input data, and it would also be desirable to develop a fast model to access the data.

• To use the software we have developed, we still need to couple a Visualisation tool to present the calculated results in 3D network.

• Currently the tools we have developed are available on a private Github repository because it was designed for Peratech. The status of this is to be discussed with Peratech.

3 Electric conductivity of the network

In the previous section we described how to measure distances between nearby cylinders in a non-overlapping configuration, to form a network. In fact, the configuration of cylinders is a model for a material whose resistance/conductance (when a potential difference is applied between two sides of the box) is a quantity of interest. Thus, the next problem is to actually use the obtained network to analyse electric flow through the material represented by the cylinders. This can be broken into two steps (see Figure 13):
(1) understanding the current-voltage characteristic between all pairs of cylinders (see Section 3.1 and 3.2),
(2) using it to find net current-voltage characteristic between cathode and anode for the entire network (see Section 3.3).

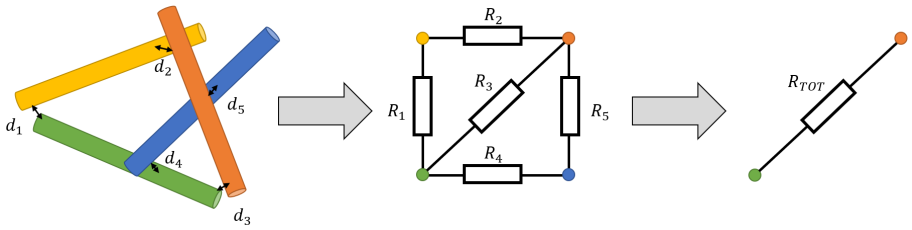

Figure 13. Illustration of two step procedure for calculating conductance.

Of course we must make some assumptions about the material that the cylinders are representing (to be discussed), but given this, the two steps above ultimately allow us to find the conductivity of the material for any configuration of cylinders. By applying it to multiple realisations of cylinder configurations, one can estimate the expected variance between produced samples and find properties of the material (such as cylinder aspect ratio), which increase the variance.

For most of the project we assumed that cylinders are located in a poorly conducting medium of a constant conductivity which is many orders of magnitude lower than the conductivity of the cylinders. Therefore, in the first approach we assumed that cylinders are perfect conductors and medium between them satisfies Ohm's law with very low, but constant conductivity. This approach is summarised in Section 3.1.

In the second part of this project, we investigated the resistance under the rather different assumption that the medium between the cylinders is insulating. Insulating materials typically have a nonlinear current-voltage characteristic and do not conduct electricity at all unless a certain (very high) voltage is applied to their ends. We discuss the consequences of this characteristic in Section 3.2.

3.1 Resistance between two cylinders in low conductive medium

3.1.1 *Mathematical model*

Let us consider two cylinders is an arbitrary configuration relative to one another. Let the voltage between the cylinders by V, i.e. electric potential ϕ on the first cylinder can be assumed to be $\phi\Big|_{\Gamma_1} = V$ and potential on the second cylinder $\phi\Big|_{\Gamma_2} = 0$. Now we assume

that the medium satisfies Ohm's law, i.e. $\mathbf{J} = \sigma\mathbf{E}$, where $\mathbf{J}$ is the current density, σ is a constant conductivity and $\mathbf{E} = \nabla\phi$ is an electric field. If we consider an electrostatic potential, then since $\nabla \cdot \mathbf{J} = 0$, the electric field ϕ has to satisfy the Laplace equation:

$$\Delta\phi = 0 \tag{3.1}$$

After solving above equation for ϕ we can find the current between the cylinders by integrating the current density flux $\mathbf{J}\cdot\mathbf{n}$ over any surface surrounding one of the cylinders. Here $\mathbf{n}$ is a normal vector to the surface. For the integration surface we can also pick an infinite plane between cylinders, since the current density disappears in infinity. The current is given by:

$$I = \oiint_S \mathbf{J} \cdot \mathbf{n}\mathrm{d}S = \oiint_S \sigma\nabla\phi \cdot \mathbf{n}\mathrm{d}S. \tag{3.2}$$

Knowing the potential difference V (set at the start) and resultant electric current I between the cylinders, one can find the resistance between them as $R = \frac{V}{I}$ (or conductance as $G = \frac{I}{V}$). Since the Laplace equation is linear, ϕ can be scaled with V and therefore electric current I will be proportional to V. As a result the current-voltage characteristic will be described using a proportional relationship with a constant resistance R.

Unfortunately, analytic solutions of the Laplace equation are known only in few simple cases, and the solution in the case of two cylinders in an arbitrary configuration almost certainly does not exist. Therefore the full solution was obtained using numerical methods. However, an analytic approximation was also developed for a region where cylinders are close to each other; this was in high agreement with the numerical results.

3.1.2 *Numerical approach*

The numerical solutions were obtained in OpenFoam (https://www.openfoam.com/), an open source CFD (Computational Fluid Dynamics) software based on the C programming language, designed for solving PDEs using the finite volume method. Its utilities allow the user to create meshes for irregular geometries, refine them in regions of interest, solve the Laplace equation and visualise the solution, and it also offers a range of post-processing tools, for example for integrating the solution over specified region. The mesh was generated using blockMesh and SnappyHexMesh OpenFoam utilities, the first of which allows one to generate a mesh of a desired size, and the second to cut out cylinders from the domain and refine the mesh around their edges and around the intersection, in order to obtain precise results in these regions. The electric potential is found using ElectrostaticFoam built-in OpenFoam solver with charge density ρ set to 0. The solver was additionally was modified to calculate $\nabla\phi$ automatically. Visualisations and post-processing was done in the Paraview software.

The numerical procedure is illustrated in Figure 14. Note that since we have to constrain our geometry to finite size we have to set an additional boundary condition on the geometry's outer boundary, which in this case is von Neumann boundary condition $\nabla\phi|_{\text{wall}} = 0$, which represents the fact that no significant current should be observed

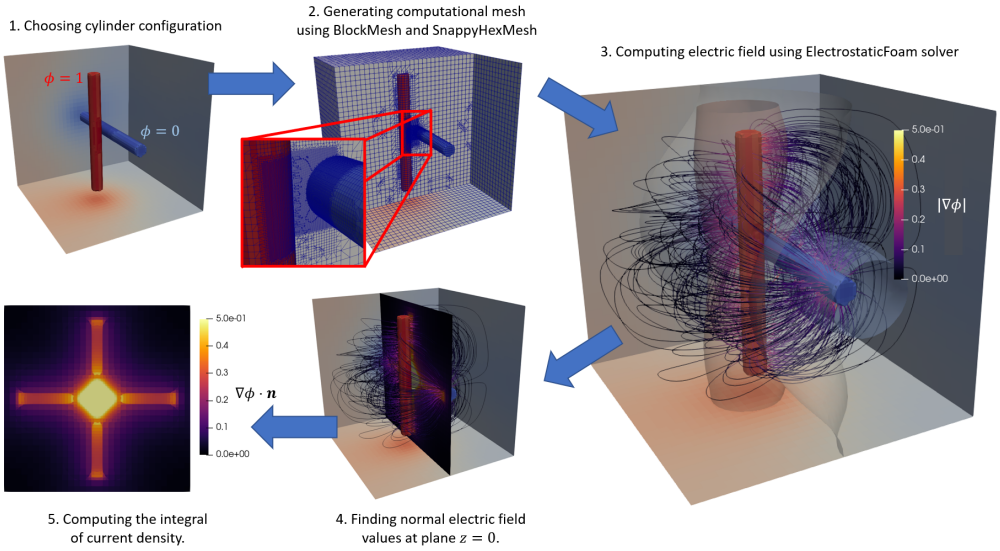

Figure 14. Numerical procedure used to calculate electric current between two cylinders in OpenFoam.

far away from the cylinders "intersection" (by intersection we mean the region where cylinders are closest to one another).

To understand the impact of cylinder configuration on the resistance between them we designed a series of numerical tests. In the first series we considered two cylinders perpendicular to each other in the x-y plane and varied the distance between them along the z-direction. The cylinders have diameter 1 and length 10, and the distance between them was changed from 0.02 to 1. A non-dimensional case was considered with potential difference $V = 1$ and medium conductance $\sigma = 1$. The results are presented in Figure 15.

As one might have expected the current between the cylinders decreases as distance between them increases. It, however, decreases slowly, linearly with a logarithm of distance at least for distances (< 1) much smaller than the length of the cylinders (10). The electric density field value near the cylinder "intersection" drops quickly with distance. However, intensity in other parts of the field (characteristic cross representing regions close to one of the cylinders) is not significantly impacted by the distance. Looking at the electric field lines (see 3$^{\text{rd}}$ image in Figure 14) one can see that this long-distance interaction comes from field lines spanning wide distances away from the cylinder intersection. If there was another cylinder in this region (which in volume fraction $5 - 20\%$ is almost certain) it would affect the field lines' structure and therefore change the current flowing between the cylinders. In such a case it is not possible to represent the cylinder structure using a network of independently considered resistances between pairs of cylinders, and other approaches should be used (such as homogenisation methods as discussed in Section ...).

In the second series we kept the distance between the cylinders constant (0.2) and rotated one relatively to another. All angles between cylinder axes were considered, from $0°$ to $90°$. The results are presented in Figure 16. As the angle between the cylinders

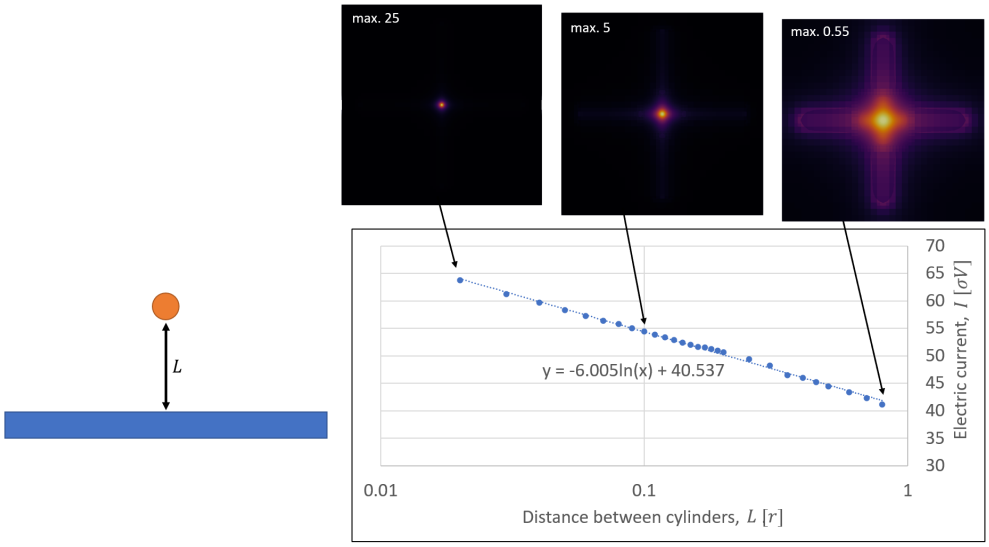

Figure 15. Dependence between the distance L between two cylinders and electric current flowing between them.

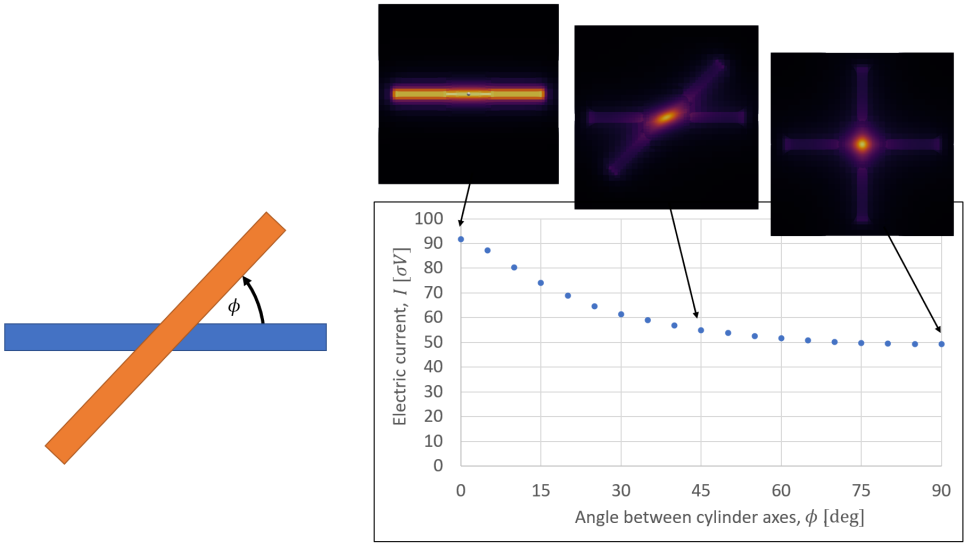

Figure 16. Dependence between the angle ϕ between two cylinders and electric current flowing between them.

decreases the surface area of the region close to both cylinders increases, and consequently the total current increases as well. It is highest when the cylinders are parallel to each other. Therefore, unlike in the initial problem formulation, not only distance between the cylinders matters, but also their relative orientation.

There is a third degree of freedom in cylinder orientation, namely the shift of one

cylinder along its axis direction. This however was not investigated in detail since we expect significant effect of this shift only when 1) cylinder are parallel to each other, or 2) the distance between the cylinders in constraint by their length (e.g. when they point at each other tip-to-tip). As long as the aspect ratio of both cylinders and cut-off distance are small both scenarios are unlikely to occur.

3.1.3 *Analytic approximation*

The main disadvantage of the numerical approach is the time required to generate mesh and compute the current between any two cylinders. Therefore during the study group we developed an analytic approximation of the field near the cylinder intersection region. We made two assumptions here (see Figure 17):

(1) Electric field lines in this region are parallel to each other and perpendicular to cylinder axes.

(2) The surface of the cylinders may be approximated using parabolas.

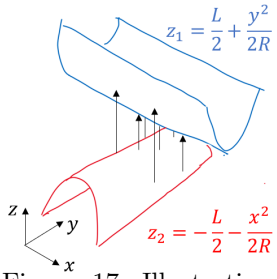

Figure 17: Illustration of analytic approach.

Assumption 1 is confirmed by numerical results. Assumption 2 is not necessary, but is accurate near closest contact point and allows us to find analytic expression for current between the cylinders. These assumptions may be taken to be valid within some distance κR around the intersection (where κ is a parameter of the order of magnitude of 1). Considering geometry and axes' orientation from the first series of numerical experiments one can write the equation for the surface of the first cylinder as $z_1(x,y) = -\frac{L}{2} - \frac{1}{2R}x^2$ and the surface of the second cylinder as $z_1(x,y) = \frac{L}{2} + \frac{1}{2R}y^2$. Therefore the distance between the cylinders if $\Delta z = L + \frac{1}{2R}\left(x^2 + y^2\right)$. The resulting current density is:

$$J(x,y) = \frac{V}{\Delta z} = \frac{1}{L + \frac{1}{2R}\left(x^2 + y^2\right)} \tag{3.3}$$

or

$$J(r,\theta) = \frac{1}{L + \frac{r^2}{2R}} \tag{3.4}$$

in polar coordinates. This result is consistent with the current around $r = 0$ obtained using full 3D numerical simulation (see Figure 18). It is not valid for $r \gg R$, where: 1) the decay rate of the current with distance should be faster, 2) we can observe additional current resulting from long-distance cylinder interactions along axial direction, which is not predicted by approximated model due to its assumptions.

Integration of this current density over the circle of region κR using polar coordinates gives:

$$I \approx \sigma \int_0^{\kappa R} \frac{2\pi r \mathrm{d}r}{L + \frac{r^2}{2R}} = 2\pi R\sigma \int_L^{L+\frac{1}{2}\kappa^2 R} \frac{\mathrm{d}\xi}{\xi} = 2\pi R\sigma \ln\left(1 + \frac{\kappa^2 R}{2L}\right) \tag{3.5}$$

Here we used the substitution $\xi = L + \frac{r^2}{2R}$, $\mathrm{d}\xi = \frac{r\mathrm{d}r}{R}$. This allows one to evaluate short-

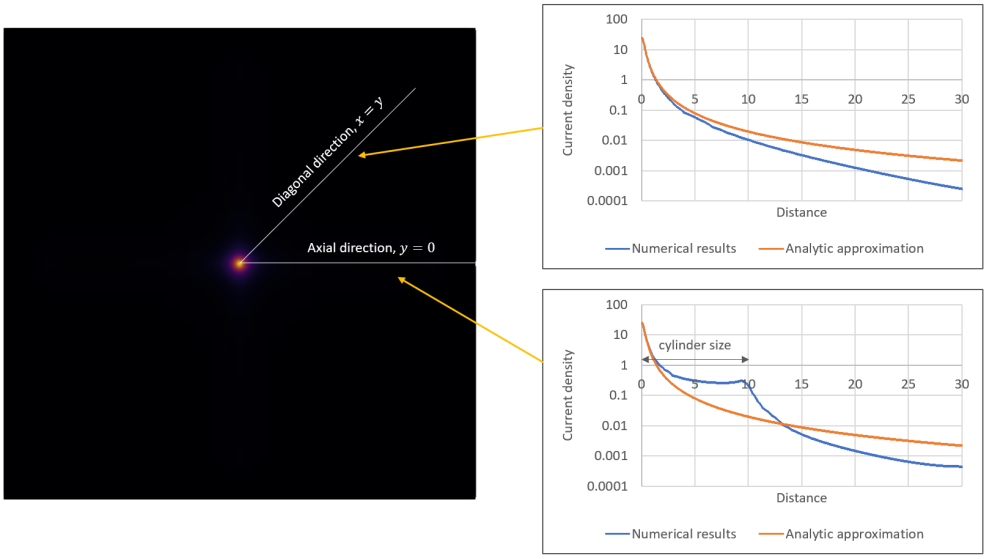

Figure 18. Comparison of current density predicted by 3D numerical simulation and analytic approximation valid near $r = 0$ for two close perpendicular cylinders. The value of current density was compared along diagonal direction and axial direction. The second includes characteristic high current density along one of the cylinder, which corresponds to long-distance interaction between cylinders.

distance interactions between the cylinders, but does not include long distance ones, the simple flow (3.3) not properly describing slow rate of current decay with distance. To evaluate this we recommend using functions fitted to numerical results instead. (The numerical calculations for larger distances need only be done for one "typical" case of the cylinders being close together.)

3.2 Resistance between two cylinders in an insulating medium

3.2.1 *Mechanisms for electron flow*

Insulating media such as resin, unlike the low-conductive medium analysed in the previous section, do not conduct electric current unless we apply a very high voltage (the so called breakdown voltage), and therefore do not satisfy the relation $\mathrm{J} = \sigma \mathrm{E}$ on which previous results were based. If the conductive cylinders are located within such a medium there are three possible mechanisms for electron flow between them [4]:

(1) If the potential gradient is higher than a certain value, the insulator atoms may become ionised and realised free electrons may cause current flow.

(2) If the distance between cylinders is very small (of order of magnitude of 1nm, possibly down to the order of magnitude of size of molecules) we may observe a tunnelling current, due to electrons tunnelling through the potential barrier.

(3) If there is a contact between the cylinders the electrons can pass between them directly, without a need to pass through the insulating medium.

The first mechanism might seem least likely considering the fact that non-zero conductance is observed even for small voltages applied between the cathode and anode, although it must be recognised that the gaps are very narrow so there might still be significant voltage gradients. There are models in the literature, which describe the electron flow through similar composite materials as a result of both 2nd and 3rd mechanisms [5]. (Assuming perfectly smooth circular cylinders, quick calculations indicate to get current flow between two touching ones, they would need at least contact along a line parallel to their axes. Contact at a point on their curved surfaces with the axes not parallel, for example, appears to give infinite resistance.)

However, note that for those mechanisms to take place cylinders have to be very close to one another (< 1nm). Such small distances between the cylinders are hardly ever observed in configurations generated numerically in Section 4 (which for a volume density of 5% are of the order of magnitude of 100nm). In such configurations we should not observe any flow through the bulk of material. The typical separation distance between nearest cylinders may greatly reduce for higher volume density. If the typical distances are still higher than expected, one should focus on better understanding the production process of the material and especially factors which may cause the cylinders to keep close to one another (such as imperfection of mixing process or potential adhesion forces between the cylinders located very closely to each other).

3.2.2 *Mathematical model for electron tunnelling*

If there is direct contact between the cylinders (mechanism 3rd) but it does not create a shortcut from cathode to anode it is safe to assume that the potential across the entire material is constant and potential drops will only occur at the locations where electrons have to tunnel through narrow gaps between cylinders. In this section we outline the physical model describing the tunnelling current between two model surfaces separated with an insulating layer based on paper by Peng Zhang from 2015 [6].

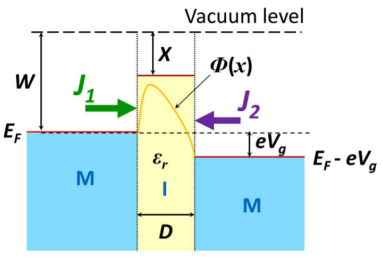

Figure 19. Metal-insulator-metal tunnelling junction with key parameters impacting current density J_1 and J_2 are marked on the figure. [6]

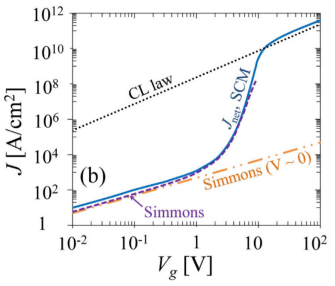

Figure 20. Current density as a function of applied gap voltage V_g, for two gold electrodes separated by a vacuum gap of width $D = 1nm$, at $T = 300K$. [6]

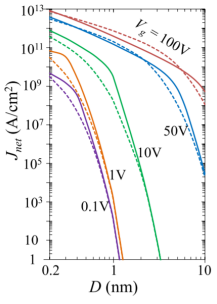

Figure 21. Current density J_{net} as a function of vacuum gap size D, for various applied voltages V_g. [6]

The metal-insulator-metal junction is presented in Figure 19. The metal electrodes have Fermi level E_F and work function W. The insulator thin film has electron affinity X, relative permittivity ϵ_r, and thickness D. The applied voltage bias is V_g, the effective potential between the electrodes is $\phi(x)$. We are interested in finding the current density J_1 emitted from the cathode. All of these quantities affect the size of the energy barrier $\Phi(x)$ that electrons need to cross. More details on this relationship can be found in the source paper. Let's consider an electron of longitudinal energy E_x (normal to the surface) trying to pass through this barrier. The probability for that is given by the WKBJ approximation:

$$D(E_x) = \exp\left[-\frac{2}{\hbar}\int_{x_1}^{x_2}\sqrt{2m\left[\Phi(x) - E(x)\right]}dx\right] \tag{3.6}$$

where x_1 and x_2 are the two roots of equation $E_x - \Phi(x) = 0$. The current density tunnelling through the barrier from electrode 1 to the right is calculated by:

$$J_1 = e\int_{-\infty}^{+\infty} N_1(E_x)D(E_x)dE_x \tag{3.7}$$

where $N_1(E_x)dE_x$ is the total number of electrons inside electrode 1 with longitudinal energy between E_x and $E + dE_x$ impinging on the surface of electrode 1 across a unit area per unit time. It is calculated using the free-electron theory of metal as:

$$N_1(E_x) = \frac{mk_BT}{2\pi^2\hbar^3}\ln\left[1 + \exp\left(-\frac{E_x - E_F}{k_BT}\right)\right] \tag{3.8}$$

where, with kB and T being the Boltzmann constant and the absolute temperature, respectively.

Potential $\Phi(x)$ was evaluated by authors of [6] over a range of junction parameters. Figure 20 presents an example current density-voltage characteristics. The obtained non-linear function in low and high voltage limits can be fitted to two earlier models, namely the Simmons model and Child-Langmuir (CL) law, respectively. The CL law states that current density is proportional to $V_g^{3/2}$, while the Simmons model, derived using an approach similar to that presented above, gives a proportional relation between J and V_g in a small voltage limit. This sounds very promising from the point of view of the material investigated in this study, since the conductivity is being measured using relatively small voltage, which is additionally distributed among many gaps between cylinders. In such case we may assume that a linear Ohm-like relationship still holds.

However the electron tunnelling current density is much more distance sensitive than in the low-conductivity model presented in Section 3.1. The impact of insulating gap width on current density is presented in Figure 21. The decrease of current is relatively slow for extremely small gaps ($\ll$ 1nm), however it drops significantly around a gap size of 1nm. This means that we should not expect to observe any current until the cylinders are almost touching each other. In this case the electrons are practically passing only through the small gap between the cylinders. This allows us to use a similar approximation to the one presented in Section 3.1.3, namely expressing $J(D)$ as a function of distance between the cylinder, and we estimate the total current by:

$$I \approx 2\pi \int_0^{\kappa R} J(D = L + \frac{r^2}{2R}) r \mathrm{dr}. \tag{3.9}$$

For example we may consider approximating the dependence between electron tunnelling density and gap width using exponential function $J(D) = J_0 \exp(-\alpha D)$. This case corresponds to a rectangular energy barrier $\Phi(x) = \text{const.}$ and approximately reflects the behaviour of $J(D)$ shown in Figure 21. In this case we get:

$$I \approx 2\pi J_0 \int_0^{\infty} \exp\left(-\alpha\left(L - \frac{r^2}{2R}\right)\right) r dr = \frac{2\pi R J_0}{\alpha} \int_{\alpha L}^{\infty} \exp(-\xi)\, d\xi = \frac{2\pi R J_0}{\alpha} \exp(-\alpha L) \tag{3.10}$$

Here we used substitution $\xi = \alpha L + \alpha \frac{r^2}{2R}$, $\mathrm{d}\xi = \frac{\alpha r \mathrm{d}r}{R}$. Note that we used ∞ as upper limit unlike in (3.5), since impact of $J(D)$ is negligible everywhere apart from the nearest neighbourhood of cylinder intersection point, and therefore the limit of integration not have significant impact on estimated current I.

One can try using the exponential model presented above to run preliminary conductivity simulations, however it is important to stress that to obtain a more accurate $I(L)$ function one should use Simmons formula presented in [6], formulated in a paper by John Simmons from 1963 [7].

We also note here that the approach in Section 3.1.3, taking current flow and field lines to be all approximately parallel, and normal to the conductors' surfaces near where they are closest, can be applied for general non-linear relations between current density and electric field. (It might be interesting to know if any experiments could be carried out, varying the applied voltage somewhat, to test for (non-)linearity of the current/voltage relation.)

3.3 Resistance of cylinder network

Suppose the previous analysis yields a network which will be represented by a directed graph of $i = 1, 2, \ldots, m$ nodes and $j = 1, 2, \ldots, n$ edges. Each edge j is specified as pointing from a start node s_j to a terminal node t_j. The weights analysis also provides a resistance r_j for each edge (which we will assume is ohmic in the first instance). Note that we will assume that each node-pair is joined by either 0 or 1 edges. The point of this construction is that we can represent the current on each edge by the scalar x_j, where a change in sign denotes a change in the direction of the current. Note that we can assume without loss of generality that the adjacency matrix is upper triangular — that is we assume the positive current direction for each edge points from the lower numbered node to the higher numbered node ($s_j < t_j$ for all j).

It is best to summarise the network structure by the matrix $\mathbf{B}$, which is an $m \times n$ node-edge incidence matrix. We set $b_{ij} = +1$ if edge j points into node i (that is, if $t_j = i$), $b_{ij} = -1$ if edge j points out of node i (that is, if $s_j = i$), and 0 otherwise. Current conservation at any given interior node i can then be expressed in the form $\sum_j b_{ij} x_j = 0$. Generally, we will write $\mathbf{Bx} = \mathbf{s}$, where x is the vector of all edge currents and s is a source vector — which has non-zero elements at boundary nodes of the network

where current is either created or destroyed — these are specified as part of the boundary conditions of the problem, described shortly.

There are then two (equivalent) formulations of Kirchoff's laws to consider. 1. In the first, the key variables are the node potentials V_i. These are prescribed at some of the boundary points but are unknown (and to be solved for) in the interior of the network. They are coupled to the currents by the edges, so that for edge j, we have $V_{t_j} - V_{s_j} = x_j r_j$ (that is, '$V = IR$'), and the edge currents must also satisfy the conservation constraints at each node. However, it is well-known that the best approach computationally is rather 2. to work in a current-oriented formulation. The Kirchoff currents turn out to minimise energy dissipation, so we should seek to minimise $f := \mathbf{x}^{\mathrm{T}}\mathbf{R}\mathbf{x}$, where $\mathbf{R} := \mathrm{diag}(\mathbf{r})$ is the Hessian matrix with the resistances r_j arranged along its diagonal. This minimisation is subject to $\mathbf{B}\mathbf{x} = \mathbf{s}$, as before.

But how do we prescribe source and sink currents at the boundaries of the network? The best way is to establish a group of source nodes that constitute the anode (and a corresponding group of sink nodes that constitute the cathode). We then specify a total source current by I by introducing a dummy node of source strength I, which is connected by a set of dummy edges of zero resistance to each of the real source nodes. The real source nodes satisfy conservation constraints (flow in equals flow out) and the solution of the minimisation problem will naturally distribute current across the real source nodes in the correct way. The same device is used for the sink nodes (and the dummy sink must have the same strength I).

To complete the picture, we assign an arbitrary potential to the source nodes (0 say). Each edge will have a corresponding voltage drop $r_j x_j$ which gives a kind of edge length. As if by magic, we then find that the distance (in voltage terms) between any two nodes in the network is path-independent, so that each node has a well-defined voltage. Further, all sink nodes will have the same voltage, thus the potential difference across the network is obtained. Since the current was prescribed, the effective impedance is determined.

Quadratic programs (QPs) with linear constraints, like that we have constructed here, have many applications and consequently have been intensively explored by the numerical analysis community. There are many efficient general purpose solvers available, and in this project we used the quadprog function in Matlab's optimisation toolbox. Since the matrices $\mathbf{B}$ and $\mathrm{diag}(\mathbf{r})$ are mostly made up of zeroes, very large problems can be tackled quickly by using a sparse linear algebra implementation. Note that for non-ohmic resistances, the objective functional is no longer quadratic: the Matlab set-up would be similar but would require the more general fmincon function.

As a simple example of what can be achieved, Figure 22 presents a 2D synthetic network built with a Delaunay triangulation on randomly distributed nodes. The anode is made up of the 6 nodes on the left and the cathode of the six nodes on the right. Edge resistances here have been modelled as proportional to the length of the edge squared. The thickness of each edge indicates the (solved) current through it — note the relatively small number of high current paths that dominate, similar to force chains in granular media contact problems. The shade of each node denotes its potential: which gradually drops from left to right and of course, so that all nodes in the anode (resp. all nodes in the cathode) have the same value. The computation shown here takes only a fraction

of a second, and the extension to 3D networks with perhaps tens of thousands of nodes (cylinders) is entirely tractable.

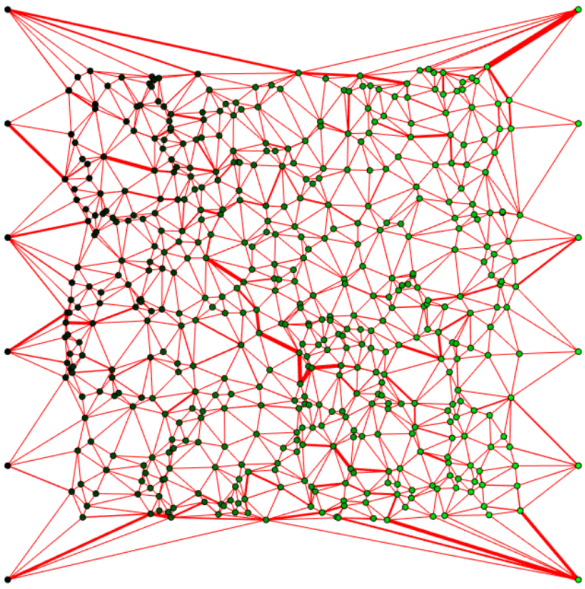

Figure 22. Current flow through a random 2D graph from the anode (nodes on left) to the cathode (nodes on right). The shading of each node denotes its potential.

4 Cylinder simulations

This subsection describes the outcomes of a working subgroup aiming to provide code for the simulation of a series of *long*, non-intersecting, cylinders or radius R and length L located in some domain $\Omega \subset \mathbb{R}^3$. Given the radius, length, domain size and volume fraction, the first goal is to generate positions and orientations of the cylinders such that there are no intersections. From this point, the second goal is to produce a network where the resistance distance between two nodes of a graph is akin to treating the graph as a grid of resistors with a resistance equal to the provided weight. Our weights will be given by minimal distances between cylinders, which we compute using a (non-open source) MATLAB function. However, these could alternatively be calculated using the code discussed in Section 2.

For the first goal, it is of particular importance to create *realistic* configurations of cylinders *efficiently*. With large numbers of cylinders, say N, detecting intersections is an $\mathcal{O}(N^2)$ operation. Currently, Peratech employs a physics simulator, giving cylinders a random initial velocity and allowing the system to evolve. The most pressing constraint is the volume fraction, the fraction of space occupied by cylinders.

The naive $\mathcal{O}(N^2)$ approach is to simply place cylinders in the domain randomly and then check to see if that cylinder intersects the boundary of the domain or any other cylinder.

This approach can be improved by considering a spatial tree type approach to improve the search for intersections. (This method is also considered in Section 2, although for a slightly different purpose). We partition the domain into cubes with side length $2\sqrt{L^2 + R^2}$. It is then sufficient to only check for collisions with cylinders in cubes adjacent to the cube containing the cylinder in question.

4.1 Simulating Packed Configurations

4.1.1 *Naive spatial tree type approach*

There are various algorithms that can pack cylinders into a domain, ranging from the iterative approach described here to a molecular dynamics approach, where cylinders are simulated with repulsive physics and allowed to move freely about one another, see [8, 9, 10].

To produce realistic, useful configurations of cylinders we turned to MATLAB's Robotics Toolbox package. This provided a number of black-box functions that enabled fast prototyping of code. The two most relevant are `collisionCylinder` which generates cylinders with particular sizes, orientations and positions and `checkCollision` which checks if two cylinders intersect, and also provides information on distances between cylinders (as well as the closest points between them) if there is no collision.

We realise that these are not open source functions, but hope that the general idea can be of use. Furthermore, much of the work in Section 2 is focused on computing the minimum distance between two cylinders. This process could be used to check for collisions between cylinders and replace `checkCollision`.

The naive algorithm to produce packed configurations is outlined in the flowchart in Figure 23. We first specify the size of the domain, the radius R and length L of the

cylinder as well as the desired volume fraction V_{frac}. The volume fraction specifies the number of cylinders through

$$V_{\text{frac}} = \frac{V_{\text{cyl}}}{D_1 D_2 D_3} = \frac{N L \pi R^2}{D_1 D_2 D_3}$$

or equivalently

$$N = \left\lceil \frac{V_{\text{frac}} D_1 D_2 D_3}{\pi L R^2} \right\rceil . \tag{4.1}$$

where D_1, D_2, D_3 are the dimensions of the domain. Note that (4.1) is not valid if $L, 2R > \max D_1, D_2, D_3$ as the cylinders will not fit. Furthermore, one should note that for the cylinders considered here (where the ends of the cylinders are flat circles), the maximum volume fraction that the cylinders can achieve is 90.69%, see https://en.wikipedia.org/wiki/Circle_packing.

The domain is partitioned into cubes with side length $2\sqrt{L^2 + R^2}$. Therefore two cylinders cannot touch if they reside in boxes with 1 or more empty boxes between them. This is illustrated in Figure 24.

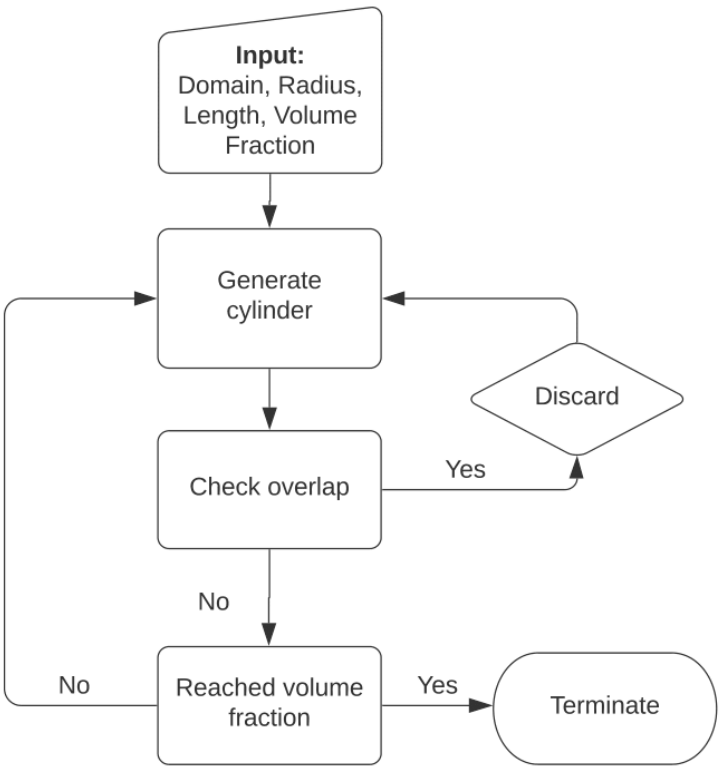

Figure 23. Flow diagram of the iterative packing approach.

We then randomly generate a cylinder with center (x, y, z) inside the domain and

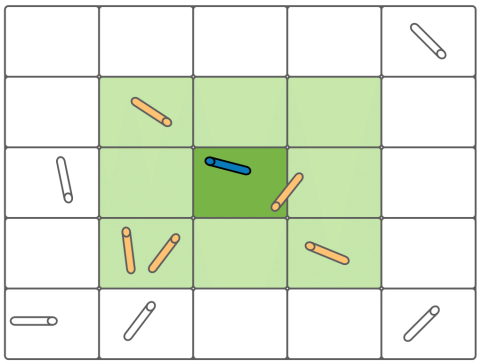

Figure 24. Spatial tree type approach to the search algorithm

random orientation (θ, ϕ). The angles θ and ϕ are the standard polar and azimuthal angles in spherical polar coordinates. We also record the box coordinates of the centre of the cylinder.

A cylinder may not intersect with any other cylinders. The brute force approach is to check with each other cylinder in the domain. However, this is an $\mathcal{O}(N^2)$ operation. Instead we check all 27 surrounding boxes with Chebyshev distance ≤ 1. While this does not reduce the complexity, it greatly reduces the pre-factor. The cylinder must also be checked for intersections with the edge of the domain. If there are no collisions then the cylinder is added to the configuration. If there are collisions then the cylinder is discarded and a new one generated.

After each cylinder is added to the configuration we compute the volume fraction. If this has been reached, the algorithm terminates and outputs an $N \times 5$ list of cylinder configurations (x, y, z, θ, ϕ). A plot of the output is shown in Figure 30.

4.1.2 *Refined Spatial Tree Type Approach*

The refined spatial tree approach adapts Figure 24. For very slender cylinders ($L \gg R$) of a fixed volume, the partitioning boxes may get large (since they must be of length $2\sqrt{L^2 + R^2}$ or greater). This requires checking many other cylinders. Instead we partition the domain into smaller boxes of size not smaller than (cutoff distance) + 2R.

In this case, for each cylinder we obtain a list of boxes which are intersected by the centre line of the cylinder (dark green in Figure 25). We need only compare with cylinders with centre lines that intersect the neighbouring boxes (light green). This method improves the algorithm for slender cylinders, at the cost of memory as it is required to store all the indices of the boxes intersected by each cylinder.

This was demonstrated during the study group by dividing the domain into a different number of boxes and checking how it impacts: 1) the number of selected cylinders for which the distance need to be measured, 2) computational time required to run the refined spatial tree algorithm and 3) total allocated memory by the program. The results are presented in Figure 26. Note that the number of spatial cells neighbouring each cylinder increases linearly with n, but the average number of cylinders per cell decreases as n^{-3}.

Therefore the average number of neighbouring cylinders decreases as $n \cdot n^{-3} = n^{-2}$ and therefore so does the algorithm computational complexity, therefore it is beneficial to use the refined spatial tree approach over the naive one.

4.1.3 *Overcrowding*

The algorithm outlined in 4.1.1 and Figure 23 can suffer from 'overcrowding'. In fact, for large volume fractions there is no room for the cylinder to be placed. The algorithm will not halt in this case and an upper limit on the number of trial cylinders is required. Typically, we saw the algorithm begin to struggle around a volume fraction 5%. This is an issue that is suffered by all random, fixed placement algorithms. There is simply not enough room to place the cylinders.

4.1.4 *Hierarchical Spatial Tree Approach*

In the hierarchical tree approach, we subdivide the domain using a *Octree* data structure, in which each internal node has exactly eight children. Their are generally used to partition a three-dimensional space by recursively subdividing it into eight octants. Each internal node represents a sub-box, while its eight children are uniform sub-boxes obtained after a further refinement. The refinement of a specific branch of a tree stops when only one or zero cylinders are found within the refined sub-box, that becomes a leaf of the overall tree. To summarize, we compose the Octree by doing the following steps:

(1) Divide the initial box into eight boxes
(2) If any box has more than one cylinder then divide it further into eight boxes. Do not divide the box if zero or one cylinder is contained in it
(3) Repeat the process until all the boxes contain one or zero cylinders

A visual representation of the data structure is given in Figure 27. One would then perform collision checks with neighbouring boxes.

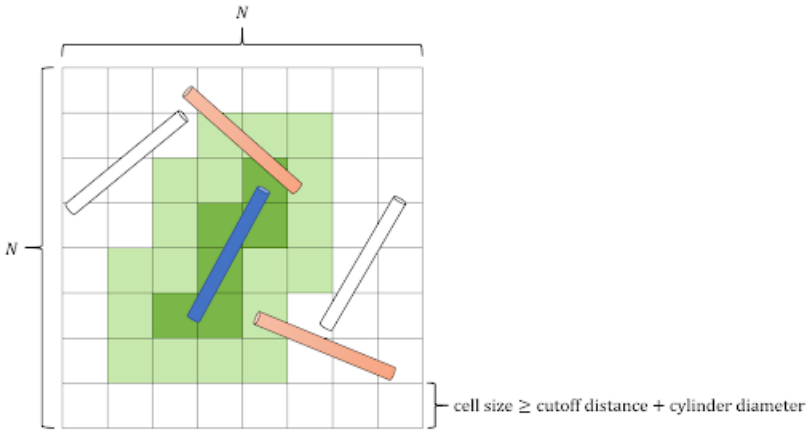

Figure 25. Refined spatial tree type approach to the search algorithm, where N denotes the number of subdivisions.

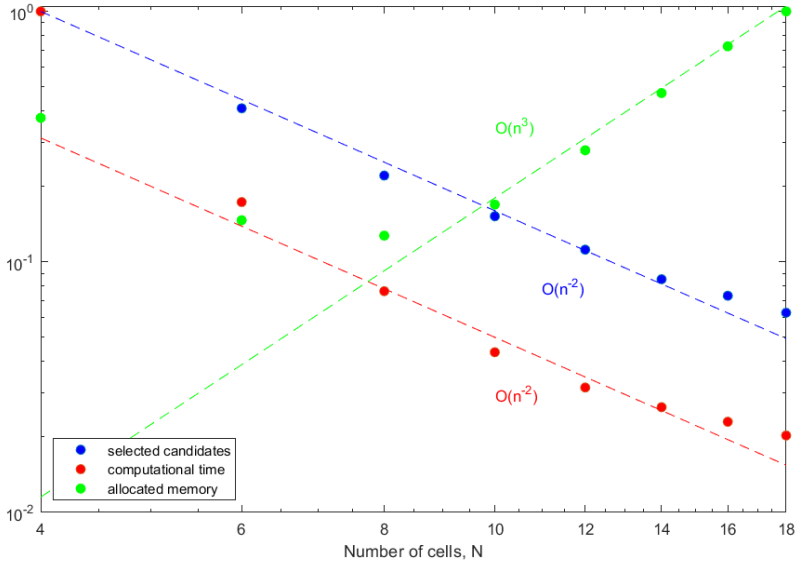

Figure 26. Dependence between the number of selected candidates, computational and memory of refined spatial tree algorithm depending on the number of cells into which the domain is divided. All the quantities are normalised to 1.

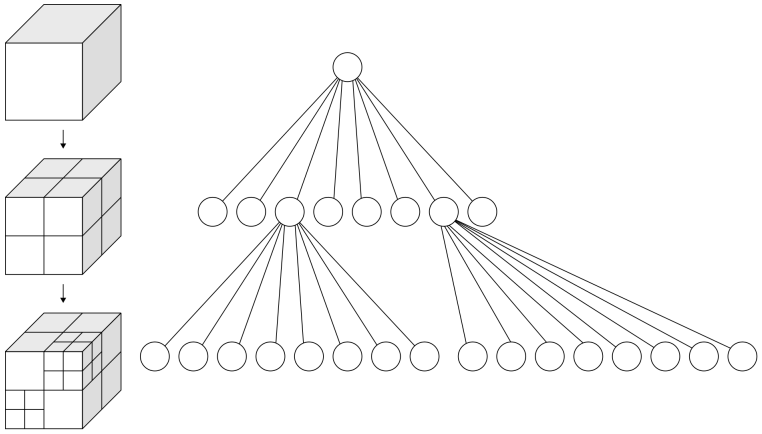

Figure 27. Left: Recursive subdivision of a cube into octants. Right: The corresponding octree.

Advantages

By traversing the nodes using depth-first search, only the cylinders that share a common parent node can be processed for collision detection. Furthermore, in case we are interested in a physical simulation, all the cylinders lying on other inner nodes can be viewed as a single element located at barycentre of that box. Other applications of Octree involve 3D computer graphics games and nearest neighbor search [11, 12].

4.2 Results and Comparison

The time taken for the algorithm in 4.1.1 to run was very dependent on the domain size, the cylinder size and the desired volume fraction. As mentioned in 4.1.3, for high volume fractions the naive approach may take a long time or even not finish. Comparison between time taken on our machines is unlikely to be of much use.

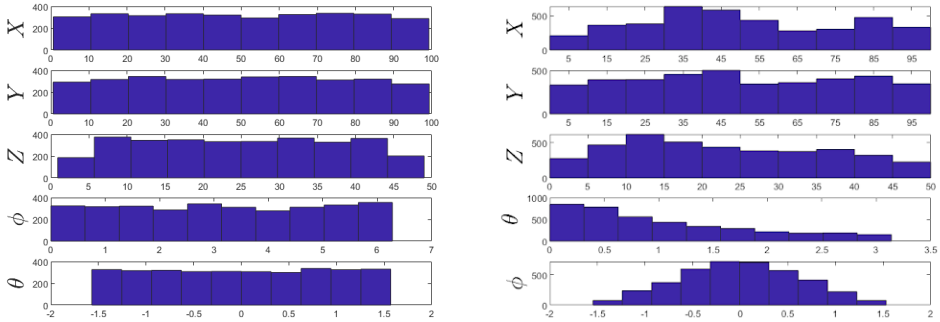

Figure 28. Left: Example 1 data from Peratech. Right: Data produced by the algorithm in 4.1.1.

Data was provided by Peratech in the form of N cylinders with position and orientation vectors (x, y, z, v_1, v_2, v_3). The orientation is assumed to be specified by a unit vector point in the direction of the symmetry axis of the cylinder. It was also assumed the cylinder dimensions are $L = 10$ and $R = 0.5$.

The distributions of the (X, Y, Z, θ, ϕ) are shown in 28. For the example data given by Peratech, we notice that the ϕ and θ are not distributed uniformly. Compare this with the data from 4.1.1 where all the variables are uniformly distributed. The preferential choice of angles could be due to a number of factors depending on the data generation. If the data was generated by randomly picking the unit vector (v_1, v_2, v_3), then this may result in a non-uniform distribution of ϕ and θ. The distribution could also be a result of the physics simulation. By packing more cylinders into the domain and forcing others out of the way some order is likely to arise.

Without knowing more about the physical system, it is hard to say what the true distribution should be. For large volume fractions a non-uniform distribution of angles is to be expected.

4.3 Network Creation

Given the final configurations of cylinders with no intersections, we are able to create a network with nodes and edges representing the cylinders and its neighbours, respectively. The network is generated from an *EdgeTable* with the following information:
A *NodeTable* must be also provided, which contains the center of coordinates of each cylinder. Both tables are first stored in a .csv file and then imported in Python. We use open source software *NetworkX* for the subsequent analysis [13]. We convert into a

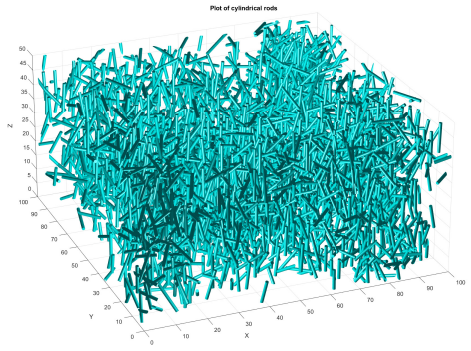

Figure 29. Plot of cylinders generated using `Example1.csv` data file, assuming a radius of 0.5, and length of cylinders of 10.

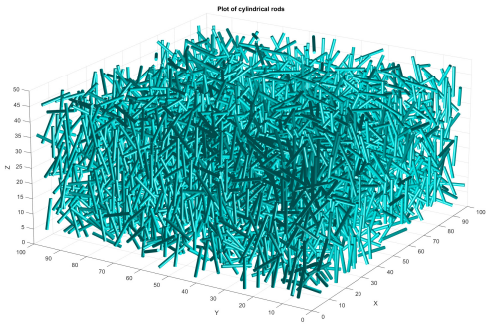

Figure 30. Plot of cylinders generated during study group using own generation methods, assuming a radius of 0.5, and length of cylinders of 10. The volume fraction is 0.05.

EndNodes_1	EndNodes_2	Weight	X1	Y1	Z1	X2	Y2	Z2
1	74	0.423	5.973	6.584	9.950	6.239	6.462	9.644
1	74	0.423	6.239	6.463	9.644	5.973	6.585	9.950
3	25	0.432	3.613	7.661	3.509	3.677	7.872	3.880
3	25	0.432	3.677	7.872	3.880	3.613	7.661	3.509
4	16	0.298	4.187	1.057	3.855	4.075	0.808	3.735
4	16	0.298	4.075	0.808	3.735	4.187	1.057	3.855

Figure 31. EdgeTable for cylinder configuration. The first two columns denote the label of the adjacent cylinders. The Weight is given by the distance between the first cylinder, at the point (X1,Y1,Z1), and the second cylinder, at the point (X2,Y2,Z2). Note that a 2 cylinders might have multiple entries due to the distance of different points on both the surface of the 2 cylinders whose distance is below the threshold.

network a cylinder configuration having the following properties: radius $R = 0.5$, length $L = 10.00$, volume fraction $v_f = 0.05$.

For sake of visualization, we decide to represent each cylinder as a single node and we

consider only edges with other cylinders with minimum weight. The nodes are labelled according to the NodeTable, while each edge has the attributes given by the EdgeTable.

The *NetworkX* package provides built-in algorithms for the computation of the shortest path distance and the resistance distance between two nodes of a graph (i.e. treating the graph as a grid of resistors with a resistance equal to the provided weight). The two combined algorithm will be used to determine the overall resistance of the network.

The computation of the network resistance is performed according to the following steps:

(1) Create a 'dummy' anode and cathode to represent the electrical circuit at the center of two opposite faces of the box (in figure 32 anode and cathode are located at $x = 0$ and $x = 100$).

(2) Connect both anode and cathode to the nearest nodes according to a threshold distance. The weight assigned to the edges must be infinitesimal (e.g. 1×10^{-12}).

(3) Find the shortest path between anode and cathode, in order to obtain the minimum network resistance.

A visual representation of the synthetic network is given in Figure 32.

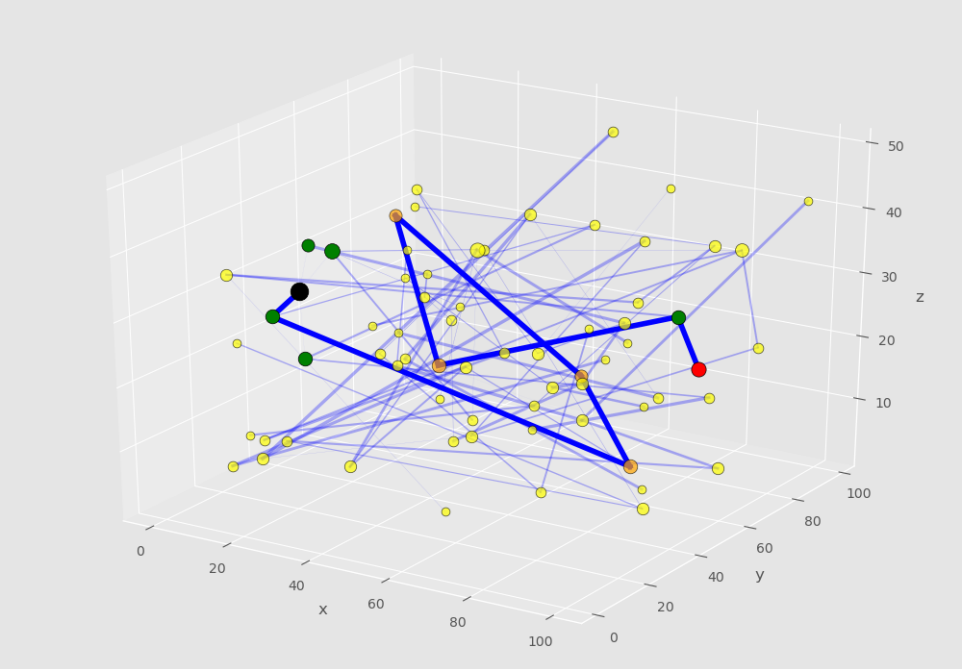

Figure 32. Network representing the configuration of generated cylinders. The red and black nodes represent the anode and the cathode of the electrical circuit. The green nodes are the closest to them and linked by an infinitesimal weight. The thick blue edges connect anode and cathode via the shortest path, so that the overall resistance is minimal. We also remark that the thickness of the other edges is proportional to their weights.

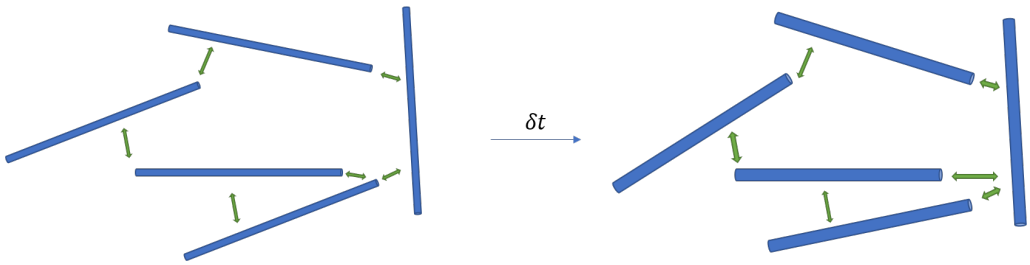

Figure 33. Depiction of the *Molecular Dynamic Packing* algorithm, i.e. the inflation of cylinders with time coupled with repulsive dynamics to avoid collisions.

4.4 Further Work

First and foremost, we should emphasise the importance of simulating a realistic cylinder distribution for any practical application. Indeed, the network topology is vital, as similar networks (with same sized cylinder/cylinder resistances) can give rise to significantly different overall conductances.

This is a clear direction for future work, but proceeding effectively would require much more specific information about the formation of the particle mixture and the interactions between particles.

In terms of efficiency, we note that the algorithms discussed simulate tightly packed cylinders in some domain via iteratively generating and checking collisions of cylinders. It is likely that this is *not* the optimal way to generate tightly packed cylinder configurations, and indeed alternative approaches to similar packing algorithms have been used. Such alternative approaches are the so-called *Molecular Dynamic Packing* and *Monte-Carlo Packing* methods, the former being inspired by molecular dynamics. For further details, see [10, 8, 9] and references therein.

4.4.1 *Molecular Dynamic Packing*

A brief overview of the molecular dynamic packing algorithm will be provided here.

Consider a domain $\Omega = [0, D_1] \times [0, D_2] \times [0, D_3]$ for some $D_1, D_2, D_3 > 0$ that we want to fill with cylinders of length L and radius R such that the volume occupied by the cylinders, V_{cyl} is some given portion of $D_1 D_2 D_3$. To satisfy this volume occupation, also often called, *volume fraction* V_{frac}, we require N cylinders, given by Equation 4.1.

Thus, assume we are provided with domain and cylinder dimensions, and are asked to generate a cylinder packing that achieves a given volume fraction V_{frac}, i.e. requires the use of N cylinders. Consider now some given time frame $[0, T]$ partitioned into M steps, i.e. $[0, T] := \bigcup_{i=1}^{M} [t_{i-1}, t_i]$, and begin (at time $t = 0$) by generating a sequence of N cylinders of radius R_0 and length L in random positions within Ω, where $R_0 \ll 1$ is very small (best to choose R_0 as small as possible, i.e. equal to machine precision, to avoid collisions between cylinders in the initial generation). If R_0 is chosen small enough, this should yield, quite efficiently, a simulation of non-intersecting cylinders occupied within the given domain, and if not, one can repeat the generation until a non-intersecting configuration is obtained. The main idea of this method of cylinder generation is to provide each cylinder with *repulsion physics*, by which we mean that the cylinders are

now made to repel one another under some given potential ϕ, which is primarily infinitely repulsive at zero, and weakly repulsive at a distance. One could for example use the Lennard-Jones potential to achieve this, although the choice of potential is up to the stakeholder.

Once repulsive physics between the cylinders have been implemented, at their given radius R_0, allow the cylinders to repel one another and re-position themselves for t_1 amount of time. Then at t_1, inflate slightly the cylinders to a new radius R_1, chosen small enough such that intersections between cylinders do not occur. The cylinders should again now be allowed to repel one another and re-position themselves until time t_2 and then inflated again to a radius R_2. Repeating this process M times, where at each time t_i, the cylinders are inflated to a radius $R_i := R_0 + \frac{(R-R_0)i}{M}$, where finally at time $t_M = T$, a cylinder configuration satisfying a given volume fraction should be obtained. See Figure 33 for a depiction of the physics. See also Figure 34 for a flowchart of the process.

Advantages

The main pitfall of the previous iterative packing algorithm implemented is that as the volume fraction is increased, the domain of feasible positions for new cylinders, with each new placed cylinder, dramatically decreases and thus makes it difficult and computationally expensive to find a viable position. Through experimentation we have deduced that the iterative packer struggles to achieve volume fractions greater that 5% within a feasible time frame. Indeed, this is where the advantages of the *molecular dynamic approach* lie. Since the required number of cylinders to achieve a given volume fraction are all already placed within the domain, no new positions have to be found, and thus as long as cylinder-repulsion physics have been well implemented, any feasible required volume fraction can be achieved.

Difficulties and Possible Problems

The main difficulties with this approach is that it is much more complex to implement than the iterative solver. The difficulties in the implementation lie in providing cylinders with repulsive physics. It is relatively straight froward to implement repulsive physics with spherical balls due to the axisymmetry of the repulsive potential, although the elongation of such spheres into cylinders makes this step difficult, since we not only have to have repulsion from the barycentres of the cylinders, but also along its whole length. One way to approximate this behaviour could be to fix to the ends and centre of each cylinder a repulsive sphere, each connected by some infinitely strong and inflexible bond, where one could increase the number of spheres within the cylinder to obtain better approximations on the repulsive physics.

Furthermore, by requiring that the cylinders repel one another, some underlying physics and possible attraction between the real world cylinders that we're trying to simulate may be lost. This could be accounted for by choosing a suitable repulsion potential ϕ.

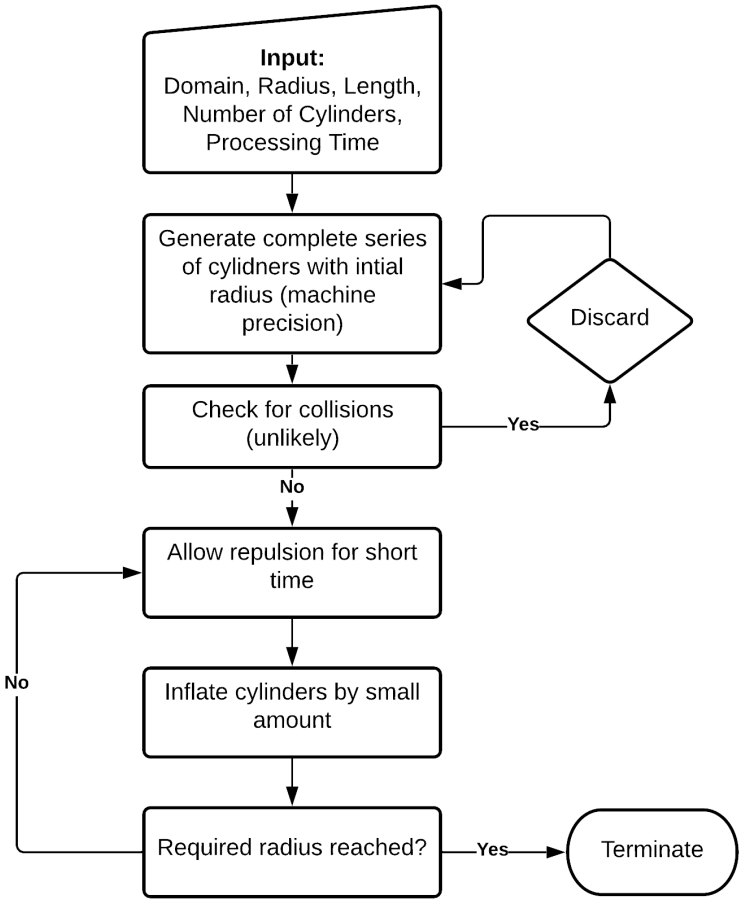

Figure 34. Flowchart of the *Molecular Dynamic Packing* approach.

4.4.2 *Monte Carlo Packing*

The *Monte-Carlo Packing* method follows a similar approach to the *Molecular Dynamical Packing* method, although instead of slowly inflating cylinders and allowing them to repel one another, one would simply perturb the rotation and position of cylinders that are intersecting until no intersection is detected. This, although simple to implement, is suspected to only work effectively for small volume fractions, and will have the same bottleneck as the iterative packer implemented above, i.e. in collision checking.

5 A Homogenised Model

In this section we discuss briefly an alternative approach in which the material is homogenised and general properties are derived in terms of averaged quantities. This will not provide a precise value for the conductance of a particular configuration of the cylinders within the box but, given that the exact arrangement of the cylinders is not easy to establish, this approach may still offer some useful insights.

5.1 The Conductance of the Homogenised Model

Some useful work has been done on conductive networks of carbon nanotubes [14] and we will follow these ideas here. A 'random stick' network model has been used to estimate the electrical conductivity of a film containing carbon nanotubes by [15] which shows how conductance depends on the number of inter-particle contacts. When the conductance across a junction between two particles (or cylinders) is much higher than the conductance along a cylinder, they show that a good approximation for the conductance across a box of volume V is $G_j n_j$ where n_j is the number of junctions in the box and G_j is the conductance at a junction.

To determine n_j, we can appeal to the paper [16] which provides an estimate for n_j when there are a large number N of cylinders of length L and diameter D in V. The arrangement of the cylinders in the material is characterised by a probability density function $P(X, Y, Z, \theta, \phi)$ which represents the probability of a cylinder with centre at $(X, Y.Z)$ having an orientation (θ, ϕ) in spherical polar coordinates. Then

$$n_j \doteq \frac{2DN^2L^2}{V} I(X, Y, Z) \tag{5.1}$$

where I is the orientation factor which can be calculated if P is known. If we assume that the cylinders are randomly oriented in space then $P = \frac{1}{2\pi}$ and $I = \frac{\pi}{4}$. It may also be more convenient to work with the volume fraction f of cylinders in the material in which case

$$n_j \doteq \frac{16f^2}{\pi^2 D^3} VI. \tag{5.2}$$

The factor G_j could be estimated from the calculations contained in Section 3.1 or it may be more practical to determine it by experiment.

Finally the conductance σ across the sample of length h in the Z-direction can be calculated by integrating so that

$$\sigma = \int_0^h G_j \left(\frac{16f^2}{\pi^2 D^3} I \right) dZ. \tag{5.3}$$

In this formula f and I could depend on X, Y, Z and h may depend on X, Y so that the value of σ may depend on X, Y. We will discuss the variability of the material briefly in the next section.

5.2 Non-uniformities in the Material

The material is manufactured via a multistage procedure (mixing, deposition, evaporation), and each of the individual processes will affect the position of the cylinders within

the sample. In particular, the deposition process will affect the orientation of the cylinders within the sample and the evaporation process will tend to change the orientation and the volume fraction of the cylinders particularly near the surface of evaporation.

The paper [17] describes a mathematical model of paint drying and shows that the free surface can become non-planar and hence h will depend on X, Y. The most likely cause for this is the dependence of the surface tension on the concentration of the solvent which creates an instability. This theory is developed further in [18] and [19]. The effect of the cylinders will also have an effect on the surface during the evaporation process and [20], which considers the evaporation of a multi-component fluid and how colloidal particles respond to these instabilities, may be helpful here.

We mention finally the possibility that the raw material may include bundles of parallel cylinders and that the mixing process may not be strong enough to tear all such bundles apart. When this happens the number of cylinders used to estimate the the number of junctions in formula (5.1) should be the number of separate cylinders plus the number of clumped cylinders and this may be significantly smaller than the total number of cylinders N.

6 Conclusion

In summary, progress was made during the study group on all aspects of the problem mentioned in the introduction.

With respect to the initial question of network/distance generation, a fully working programme to calculate exact minimal distances between two cylinders in $\mathbb{R}^3$ has been written, which turned out to be a non-trivial mathematical problem. Moreover, a detailed recommendation for converting these distances into a complete network given the cylinder configuration is given in Section 2. In Section 4 a pre-existing MATLAB function for such distance calculations and network generation is also identified.

Various methods for generating configurations of non-intersecting cylinders are discussed in Section 4, alongside analysis and recommendation as to the benefits and drawbacks of each. Having said this, the importance of generating a model with *realistic* features should be emphasised, and this was not possible during the working group due to the limited information available.

Perhaps the most theoretical part of this report is contained in Section 3, which provides a detailed methodology for calculating overall material conductance, based on various different underlying assumptions of the material represented by the cylinder configurations. Combining this analysis with experimental data should enable Peratech to enhance their understanding of the mechanisms that are contributing most to conductance within their material.

Finally, a somewhat different, "top down", approach to estimating conductance based on *homogenisation* is described in Section 5. Several recommendations for future investigation and relevant literature are also indicated in this section.

References

[1] D. Biermann, R. Joliet, and T. Michelitsch. Fast distance computation between cylinders for the design of mold temperature control systems. In *Advances in Computational Intelligence-Theory and Practice*, Series CI 258/08. Technical University of Dortmund, 2008.

[2] D. Vranek. Fast and accurate circle-circle and circle-line 3d distance computation. *Journal of Graphics Tools*, 7(1):23–31, 2002.

[3] D.H. Eberly. *3D Game Engine Design: A Practical Approach to Real-Time Computer Graphics*. Interactive Technologies Series. Taylor & Francis, 2000.

[4] I. Balberg, D. Azulay, D. Toker, and O. Millo. Percolation and tunneling in composite materials. *International Journal of Modern Physics B*, 18(15):2091–2121, 2004.

[5] G. Ambrosetti, C. Grimaldi, I. Balberg, T. Maeder, A. Danani, and P. Ryser. Solution of the tunneling-percolation problem in the nanocomposite regime. *Physical Review B*, 81(15), 2010.

[6] P. Zhang. Scaling for quantum tunneling current in nano-and subnano-scale plasmonic junctions. *Scientific Reports*, 5(1):1–11, 2015.

[7] J.G. Simmons. Generalized formula for the electric tunnel effect between similar electrodes separated by a thin insulating film. *Journal of Applied Physics*, 34(6):1793–1803, 1963.

[8] J.O. Freeman, S. Peterson, C. Cao, Y. Wang, S.V. Franklin, and E.R. Weeks. Random packing of rods in small containers. *Granular Matter*, 21(4), 2019.

[9] M. Skoge, A. Donev, F.H. Stillinger, and S. Torquato. Packing hyperspheres in high-dimensional Euclidean spaces. *Physical Review E*, 74(4), 2006.

[10] W. Zhang. *Experimental and Computational Analysis of Random Cylinder Packings with Applications*. PhD thesis, 2006.

[11] D. David Luebke, M. Reddy, J.D. Cohen, A. Varshney, B. Watson, and R. Huebner. *Level of detail for 3D graphics : application and theory*. Morgan Kaufmann, 2003.

[12] J. Elseberg, S. Magnenat, R. Siegwart, and A. Nuchter. Comparison on nearest-neigbour-search strategies and implementations for efficient shape registration. *Journal of Software Engineering for Robotics (JOSER)*, 3:2–12, 01 2012.

[13] A.A. Hagberg, A.D. Schult, and P.J. Swart. Exploring network structure, dynamics, and function using NetworkX. In *Proceedings of the 7th Python in Science Conference*, pages 11 – 15, Pasadena, CA USA, 2008.

[14] W. Shim, Y. Kwon, S. Jeon, and W. Yu. Optimally conductive networks in randomly dispersed CNT: graphene hybrids. *Scientific Reports*, 5:Article Number 16568, 2015.

[15] I. Zezelj, T. Stankovic. From percolating to dense random stick networks conductivity model investigation. *Physical Review B*, 86:134–202, 2012.

[16] T. Komori and K. Makishima. Numbers of fiber-to-fiber contacts in general fiber assemblies. *Textile Research Joural*, 47:13–17, 1977.

[17] S.D. Howison, J.A J.A. Moriarty, J.R. Ockendon, E.L. Terrill, and S.K. Wilson. A mathematical model for the drying of paint layers. *Journal of Engineering Mathematics*, 32:317–394, 1997.

[18] P.L. Evans, L.W. Schwartz, and R.V. Roy. A mathematical model for crater defect formation in a drying paint layer. *Journal of Colloid and Interface Science*, 227:191–205, 2000.

[19] P. Truong, X. Cheng, and S. Kumar. Drying of multicomponent thin films on substrates with topography. *Journal of Polymer Science Part B - Polymer Physics*, 55:1681–1691, 2017.

[20] M.H. Eres, D.E. Weidner, and L.W. Schwartz. Three-dimensional direct numerical simulation of surface-tension-gradient effects on the leveling of an evaporating multicomponent fluid. *Langmuir*, 15:1871–1859, 1999.