TALLINN UNIVERSITY OF TECHNOLOGY
SCHOOL OF SCIENCES
DEPARTMENT OF CYBERNETICS

Generating Non-Intersecting Fiber Distributions for Finite Element Modeling of Short Fiber Reinforced Composites

Master's Thesis

student: Artur Raag
studentcode: 231938LAFM
Supervisor: Dr. rer. nat. habil. Heiko Jens Herrmann
Senior Researcher
Department of Cybernetics
Curriculum: Applied Physics and Data Science

Tallinn 2026

TALLINNA TEHNIKAÜLIKOOL
LOODUSTEADUSKOND
KÜBERNEETIKA INSTITUUT

Mittelõikuvate kiudude jaotuste genereerimine lühikeste kiududega tugevdatud komposiitide lõplike elementide modelleerimiseks

Magistritöö

Üliõpilane: Artur Raag
Üliõpilaskood: 231938LAFM
Juhendaja: Dr. rer. nat. habil. Heiko Jens Herrmann
Senior Researcher
Küberneetika Instituut
Õppekava: Rakendusfüüsika ja andmeteadus

Tallinn 2026

Author's declaration of originality

I hereby declare that I have written this thesis independently and the thesis has not previously been submitted for defence. All works and major viewpoints of the other authors, data from sources of literature and elsewhere used for writing this paper have been properly cited.

Author: Artur Raag

Supervisors resolution

The thesis complies with the requirements for master's theses.

Supervisor: Heiko Jens Herrmann

18.05.2026

Abstract

The objective of this thesis is to develop a computational framework for the explicit geometric modelling and finite element simulation of steel fiber reinforced concrete and other short fiber composites. The developed framework generates three-dimensional fiber geometries, places them inside a concrete specimen while preventing geometric intersections, and prepares the resulting models for finite element analysis.

Fibers are represented as piecewise-linear geometries that are transformed into cylindrical volumes during geometric modelling. Fiber orientations are generated using an orientation distribution function together with rejection sampling, allowing preferential alignment along prescribed directions. A two-stage collision detection algorithm based on axis-aligned bounding boxes and segment-segment distance computations is used to prevent intersections between fibers.

The generated geometries are exported to Gmsh, where constructive solid geometry operations are used to construct separate volumetric regions for the concrete matrix and the embedded fibers. The resulting meshes are converted for finite element simulations in ElmerFEM. The simulations are performed under uniaxial tensile loading, and the resulting stress and deformation fields are analysed in ParaView.

Three specimen configurations are investigated: a homogeneous cube filled only with the matrix material, a specimen reinforced with hooked U-shaped fibers, and a specimen reinforced with straight fibers. The simulations demonstrate that the developed modelling pipeline successfully generates geometrically valid fiber reinforced composite models suitable for finite element analysis. The computed elastic properties remain close to those of the composite matrix due to the relatively low fiber volume fraction, although slight directional stiffness differences are observed in the preferred fiber orientation direction.

The developed framework demonstrates a complete workflow for fiber geometry generation, orientation sampling, collision detection, mesh generation, finite element simulation, and visualization. The work also provides a foundation for future studies involving more realistic fiber geometries, boundary effects, higher fiber volume fractions, and non-linear fracture simulations.

The thesis is in English and contains 36 pages of text, 6 chapters, 9 figures, 4 tables.

Annotatsioon

Mittelõikuvate kiudude jaotuste genereerimine lühikeste kiududega tugevdatud komposiitide lõplike elementide modelleerimiseks

Käesoleva lõputöö eesmärk on arendada arvutuslik raamistik teraskiududega tugevdatud betooni ning teiste lühikeste kiududega komposiitmaterjalide geomeetriliseks modelleerimiseks ja lõplike elementide simulatsiooniks. Arendatud raamistik genereerib kolmemõõtmelisi kiugeomeetriaid, paigutab need homogeensesse maatriksisse, vältides geomeetrilisi lõikumisi ning valmistab saadud mudelid ette lõplike elementide analüüsiks.

Kiud genereeritakse murdjoonelistest segmentidest, mis teisendatakse geomeetrilise modelleerimise käigus silindrilisteks ruumaladeks. Kiudude orientatsioonid genereeritakse orientatsioonijaotusfunktsiooni ja valikumeetodi (inglise keeles *rejection sampling*) abil, võimaldades kiududel joonduda eelistatavalt etteantud suunas. Kiudude omavaheliste lõikumiste vältimiseks kasutatakse kaheastmelist kokkupõrgete tuvastamise algoritmi, mis põhineb telgedega joondatud piirdekastidel (inglise keeles *axis-aligned bounding boxes*) ning kahe sirglõigu vahelise minimaalse kauguse arvutamisel.

Genereeritud geomeetriad eksporditakse programmi Gmsh, kus moodustatakse eraldi ruumalad maatriksi ja kiudude jaoks ning teisendatakse komposiit lõplike elementide võreks. Saadud võre teisendatakse ElmerFEM-i jaoks sobivasse vormingusse ning simulatsioonid viiakse läbi ühesuunalise tõmbekoormuse korral. Tulemuseks saadud pinge- ja deformatsioonivälju analüüsitakse programmis ParaView.

Töös uuritakse kolme katsekeha konfiguratsiooni: homogeenne kuupmaatriks, U-kujuliste konksotstega kiududega tugevdatud katsekeha ning sirgete kiududega tugevdatud katsekeha. Simulatsioonid näitavad, et arendatud modelleerimisahel võimaldab genereerida korrektseid komposiitmudeleid, mis sobivad lõplike elementide analüüsiks. Madala kiudude ruumalaosa tõttu jäävad elastsed omadused lähedaseks maatriksi omadustele, kuid eelistatud orientatsiooni suunas esineb väikeseid jäikuse erinevusi.

Arendatud raamistik hõlmab kiugeomeetria ja selle orientatsioonide genereerimist, lõikumiste tuvastamist, võrgu genereerimist, simulatsiooni ja visualiseerimist. Töö loob aluse edasisteks uuringuteks, mis käsitlevad realistlikumaid kiugeomeetriaid, suuremat kiudude arvu ning mittelineaarseid purunemissimulatsioone.

Lõputöö on kirjutatud inglise keeles ning sisaldab teksti 36 leheküljel, 6 peatükki, 9 joonist, 4 tabelit.

List of abbreviations and terms

AABB	Axis-Aligned Bounding Box
CSG	Constructive Solid Geometry
FEA	Finite Element Analysis
FEM	Finite Element Method
FRC	Fibre Reinforced Concrete
ODF	Orientation Distribution Function
RNG	Random Number Generator

Table of contents

List of figures

List of tables

1 Introduction

Fiber reinforced concrete (FRC) is a commonly used composite material in which fibers are distributed within a cementitious matrix to improve its load bearing capacity under horizontal and vertical stresses [1]. The addition of fibers improves mechanical properties such as tensile strength, ductility, crack resistance and post-cracking behavior, which makes FRC an attractive and practical composite to explore [1]. Different types of fibers yield different results for specific mechanical properties, the most common of which are steel fibers with hooked ends due to their superior mechanical anchorage and enhanced pull-out resistance compared to straight fibers [2].

The mechanical properties of FRC depend not only on the material properties of the fibers and the matrix, but also the geometric and topological characteristics of the fibers. This includes fiber shape, spatial distribution and orientation within the concrete volume [3]. For example, hooked fibers introduce additional geometric complexity due to their non-linear shape, which impacts their interaction with the surrounding matrix [2]. Thus, realistic modelling of such fibers and their distributions within the matrix is essential for accurate numerical simulations of FRC.

In this thesis, a computational framework is developed for the generation and meshing of randomly oriented hooked fibers within a three-dimensional concrete specimen. The approach is based on representing fibers as polylines, which are subsequently transformed into volumetric geometries suitable for finite element analysis. Fiber orientations are sampled using a probabilistic approach based on an orientation distribution function, allowing control over the degree of alignment within the specimen. A two-stage collision detection algorithm is used to ensure that fibers do not intersect during placement. The resulting fiber geometries are then converted into solid representations and embedded into the concrete domain using constructive solid geometry operations. Afterwards, using specialized software called GMSH [4], a three-dimensional finite element mesh is generated, in which fibers and the homogeneous matrix are represented as distinct material regions. Finally, a finite element analysis (FEA) will be performed with the ElmerFEM software [5] on a number of different meshes. The resulting deformation and stress fields are subsequently visualized and analysed using ParaView [6], allowing qualitative inspection of fiber behaviour as well as extraction of deformation quantities required for estimating effective material properties.

The aim of this work is to develop a robust and flexible methodology for explicitly modelling hooked fibers in concrete and other short fiber composites, enabling the generation of geometrically accurate and mesh-compatible representations suitable for finite element analysis.

2 Theory

In this chapter, the theoretical background relevant to the modelling of fiber reinforced concrete is presented. The focus is placed on the geometric representation of fibers, their spatial orientation, and the computational methods required for their generation and placement within a three-dimensional domain. Furthermore, key concepts related to probabilistic sampling, collision detection, and geometric modelling for finite element analysis are introduced. These concepts will be presented generally and then further elaborated on in the methodology section, due to several nuances in implementation that might not be universal.

2.1 Fiber reinforced concrete and fiber geometry

Many structures are built using concrete, however, concrete on its own lacks sufficient mechanical performance under certain conditions, particularly in tension, shear, and bending [7]. As a result, it is prone to cracking and brittle failure. This limitation can be mitigated by incorporating reinforcement e.g. fibrous materials into the concrete matrix [1], forming what is known as fiber reinforced concrete (FRC).

The use of fiber reinforcement has a long historical background. Early applications included natural materials such as horsehair and straw, while more modern developments utilize materials such as steel, glass, and synthetic fibers [8]. The addition of fibers significantly improves mechanical properties such as tensile strength, crack resistance, toughness, and post-cracking behaviour [1, 8].

The effectiveness of fiber reinforcement depends on several factors, including the properties of the concrete matrix, such as viscosity and workability during casting, which influence fiber orientation, porosity, and fiber–matrix bonding, the material type of the fibers, and their geometric characteristics. In particular, fiber shape, length, and aspect ratio influence how effectively stresses are transferred across cracks [9]. For example, straight fibers primarily contribute through frictional resistance, whereas hooked or deformed fibers provide enhanced mechanical anchorage, resulting in improved pull-out resistance and energy absorption [9]. Additionally, the spatial distribution and orientation of fibers within the concrete volume play a crucial role in determining the overall mechanical response [3]. Randomly oriented fibers may provide isotropic reinforcement, while aligned fibers can introduce directional strength and anisotropic behaviour.

It is also important to note that fiber reinforcement is not the only method used to improve the mechanical performance of concrete. Traditional reinforcement using steel bars (rebar) is widely employed in structural applications, where it provides high tensile strength

and is particularly effective in load-bearing elements [7]. However, rebar reinforcement requires precise placement, is labour-intensive, and may be susceptible to corrosion over time, especially in aggressive environments [7]. In contrast, fiber reinforcement can be distributed throughout the concrete volume, allowing for more uniform crack control and greater flexibility in design [8]. For this reason, fiber reinforced concrete is often used as a partial or complete alternative to conventional reinforcement in applications such as tunnel linings, precast elements, and industrial flooring [8]. While the utilization of rebar for concrete reinforcement is an interesting field to explore, we will only be dealing with fiber reinforced concrete in our work.

2.2 Representation of fiber orientation in three-dimensional space

The spatial orientation of fibers within a three-dimensional domain plays a crucial role in determining the mechanical behaviour of fiber reinforced concrete [3]. In order to generate fibers with arbitrary orientations, it is necessary to define a mathematical representation of direction in three-dimensional space and a method for sampling such directions.

A fiber orientation can be represented by a unit vector $\mathbf{n} \in \mathbb{R}^3$, such that

$$\|\mathbf{n}\| = 1. \tag{2.1}$$

This unit vector defines the direction of the fiber axis. Geometrically, all possible orientations correspond to points on the surface of the unit sphere S^2 [10].

A common parametrization of the unit sphere is given in spherical coordinates [10]:

$$\mathbf{n} = \begin{pmatrix} \cos\phi \sin\theta \\ \sin\phi \sin\theta \\ \cos\theta \end{pmatrix}, \tag{2.2}$$

where $\phi \in [0, 2\pi)$ is the azimuthal angle and $\theta \in [0, \pi]$ is the polar angle.

A naive approach would be to sample the spherical angles θ and ϕ uniformly:

$$\phi \sim \mathcal{U}(0, 2\pi), \quad \theta \sim \mathcal{U}(0, \pi). \tag{2.3}$$

However, this leads to clustering of points near the poles, due to the non-uniform area element of the sphere [10]. This effect is illustrated in Figure 1.

An intuitive way to understand this effect is to consider a number line with uniformly distributed points. If the ends of this line are compressed while the middle remains unchanged, the points near the ends become more densely packed. A similar effect occurs when sampling θ uniformly. Regions near the poles correspond to smaller surface areas, causing points to accumulate there.

This issue arises because the surface area element on the sphere is not uniform in θ [10], but instead given by

$$d\Omega = \sin\theta \, d\theta \, d\phi. \tag{2.4}$$

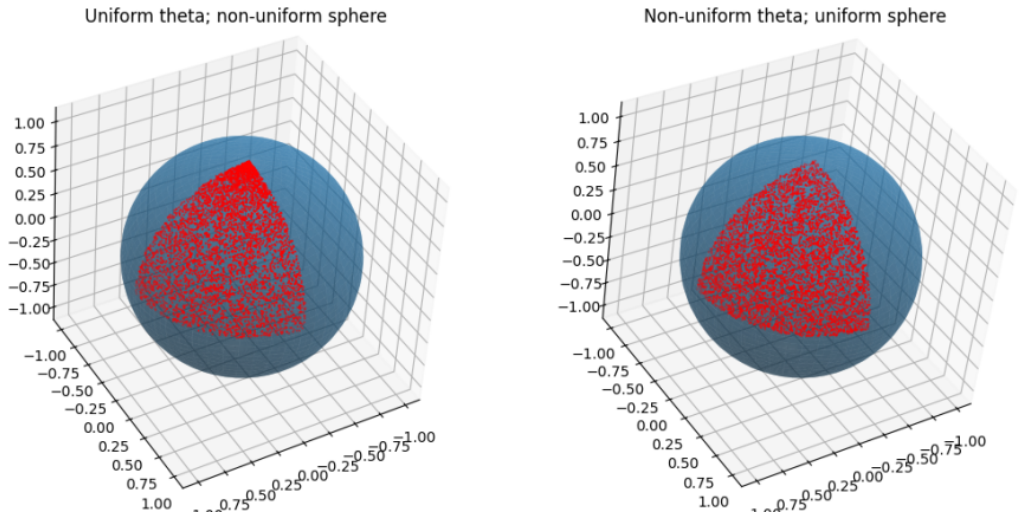

Figure 1. Uniform sampling of the polar angle θ results in clustering of points near the poles (left). A correct uniform distribution on the sphere is obtained when $\cos\theta$ is sampled uniformly instead (right). Plot generated using the script provided from [11].

To achieve a uniform distribution on the unit sphere, the sampling must account for this area element. This can be accomplished by sampling ϕ uniformly in $[0, 2\pi)$ and sampling $\cos\theta$ uniformly in $[-1, 1]$ [12]. If we set $u_1, u_2 \sim \mathcal{U}(0, 1)$ to be our independent random variables, then the correct transformation is

$$\phi = 2\pi u_1, \quad \cos\theta = 2u_2 - 1. \tag{2.5}$$

The corresponding Cartesian coordinates are then computed as

$$\mathbf{n} = \begin{pmatrix} \cos\phi\sqrt{1-\cos^2\theta} \\ \sin\phi\sqrt{1-\cos^2\theta} \\ \cos\theta \end{pmatrix}. \tag{2.6}$$

With this we ensure that directions are uniformly distributed over the surface of the sphere. This allows us to generate candidate fiber orientations before applying further probabilistic filtering using an orientation distribution function, which is described in the following section.

In addition to the direction vector, a rotation around the fiber axis is also required to fully define the fiber geometry [3]. This is introduced as an additional angle $\psi \in [0, 2\pi)$, which represents a rotation about the direction vector $\mathbf{n}$. This degree of freedom is particularly important for non-axisymmetric fiber shapes, such as hooked fibers, where the orientation around the axis affects the final geometry in space.

Together, the unit direction vector $\mathbf{n}$ and the rotation angle ψ will provide a sufficient description of fiber orientation in three-dimensional space.

2.3 Orientation distribution functions

In order to model the orientation of fibers within a three-dimensional domain, it is necessary to introduce a mathematical description of how directions are distributed in space. Since each fiber can be associated with a unit direction vector $\mathbf{n} = (n_1, n_2, n_3)$, all possible orientations lie on the surface of the unit sphere. Consequently, the orientation of fibers can be described using a probability density function defined on this sphere [13].

This function is referred to as the orientation distribution function (ODF), and is denoted by $f(\mathbf{n})$. The ODF characterizes the likelihood of observing a fiber aligned with direction $\mathbf{n}$. As a probability density, it must satisfy the normalization condition

$$\int_{S^2} f(\mathbf{n})\, d\Omega = 1, \tag{2.7}$$

where S^2 denotes the unit sphere and $d\Omega$ is the surface element [13]. In the case of a completely random (isotropic) distribution, the ODF is constant over the sphere, i.e. $f(\mathbf{n}) = \frac{1}{4\pi}$ [13].

A convenient way to represent directional distributions on the sphere is through the use of spherical harmonics. These functions form an orthogonal basis for functions defined on the unit sphere and can be viewed as the spherical analogue of Fourier series [13]. Using this basis, the ODF can be expressed as

$$f(\mathbf{n}) = \sum_{l=0}^{\infty} \sum_{m=-l}^{l} \alpha_{lm} Y_l^m(\mathbf{n}), \tag{2.8}$$

where Y_l^m are spherical harmonic functions and α_{lm} are coefficients that determine the contribution of each mode [13]. Lower-order terms describe simple, smooth variations in orientation, while higher-order terms capture increasingly complex directional patterns.

In practical applications, it is often unnecessary to keep the full infinite expansion. Instead, the series can be truncated at a low order, yielding a compact approximation of the distribution. In our work, a second-order approximation ($l = 2$) is used, which is sufficient to describe the dominant anisotropic features of fiber orientation.

Rather than working directly with spherical harmonics, it is advantageous to use an equivalent tensor representation. In this formulation, the ODF is expressed as a series in terms of the components of the direction vector:

$$f(\mathbf{n}) = \frac{1}{4\pi} \left(1 + \sum_{l=1}^{\infty} \frac{(2l+1)!!}{l!} a_{\mu_1 \ldots \mu_l} n_{\mu_1} \cdots n_{\mu_l} \right), \tag{2.9}$$

where $(2l+1)!!$ denotes the double factorial of an odd integer, defined as the product of every second integer [14]. More precisely the double factorial is defined as

$$n!! = \begin{cases} n(n-2)\cdots 5\cdot 3\cdot 1, & \text{for odd } n > 0, \\ n(n-2)\cdots 6\cdot 4\cdot 2, & \text{for even } n > 0, \\ 1, & \text{for } n = -1, 0. \end{cases} \tag{2.10}$$

This provides a tensorial expansion equivalent to the spherical harmonic representation [13].

Truncating this expansion at second order leads to

$$f(\mathbf{n}) = \frac{1}{4\pi}\left(1 + \frac{15}{2}a_{ij}n_i n_j\right), \tag{2.11}$$

where a_{ij} is a second-order tensor known as the alignment tensor [13].

The alignment tensor is defined as

$$a_{ij} = \langle n_i n_j \rangle - \frac{1}{3}\delta_{ij}, \tag{2.12}$$

where $\langle\cdot\rangle$ denotes averaging over all orientations and δ_{ij} is the Kronecker delta [13]. The subtraction of the isotropic component ensures that the tensor is traceless, meaning that it represents only deviations from a uniform distribution.

Since a_{ij} has two indices, it can be represented as a 3×3 matrix. This provides a compact and intuitive description of fiber alignment: the eigenvectors of the tensor indicate the principal directions of orientation, while the corresponding eigenvalues describe the degree of alignment along those directions.

This tensor-based formulation is particularly convenient for computational purposes, as it allows the orientation distribution to be controlled using a small number of parameters. We will later see that this representation is used to define a target orientation distribution in our simulation, from which fiber directions are subsequently sampled using a probabilistic method.

2.4 Sampling from non-uniform distributions

In order to generate fiber orientations according to a prescribed orientation distribution function, it is necessary to sample random directions from a non-uniform probability distribution defined on the unit sphere. While uniform sampling of directions is relatively straightforward, sampling from an arbitrary distribution such as the ODF introduced in the previous section is a little bit trickier.

A widely used method for this purpose is rejection sampling [15, 16]. The fundamental idea of rejection sampling is to generate candidate samples from an "easy to sample" distribution and accept or reject them based on a criterion derived from the target probability density function.

If we let $f(\mathbf{n})$ denote the target probability density function defined on the unit sphere, and have M be a constant such that

$$f(\mathbf{n}) \leq M \quad \text{for all } \mathbf{n} \in S^2, \tag{2.13}$$

then the rejection sampling algorithm can be described as follows [15]:

1. Generate a candidate direction $\mathbf{n}$ uniformly on the unit sphere.
2. Generate a random number u from the uniform distribution on the interval $[0, M]$.
3. If $u \leq f(\mathbf{n})$, accept the sample, otherwise, reject it.

This procedure is repeated until enough acceptable samples are obtained. It can be shown that the accepted samples follow the desired distribution $f(\mathbf{n})$ [16].

We generate the candidate directions using a method described in Section 2.2 that ensures uniform sampling on the unit sphere. This is achieved by sampling the azimuthal angle ϕ uniformly from $[0, 2\pi)$ and the cosine of the polar angle $\cos\theta$ uniformly from $[-1, 1]$. This approach avoids clustering of points near the poles, which would occur if the polar angle θ itself were sampled uniformly.

The acceptance criterion is based on the orientation distribution function defined in the previous section 2.3. For each candidate direction $\mathbf{n}$, the value of the ODF $f(\mathbf{n})$ is evaluated using the tensor-based formulation. A random number u is then generated in the interval $[0, M]$, where M is chosen as an upper bound for the ODF. If $u \leq f(\mathbf{n})$, the direction is accepted as a valid fiber orientation.

The efficiency of rejection sampling depends on the choice of the constant M. If M is significantly larger than the maximum value of $f(\mathbf{n})$, many candidate samples will be rejected, resulting in increased computational cost [15]. In this work, M is chosen based on the maximum possible value of the ODF corresponding to the selected alignment tensor.

A useful way to visualize this process is to consider a two-dimensional plot of a probability density function. A random point is generated within a rectangular region that fully encloses the curve. The horizontal coordinate corresponds to a candidate sample, while the vertical coordinate represents a randomly chosen value between 0 and the maximum of the function. The sample is accepted only if the point lies below the curve of the probability density function, otherwise, it is rejected.

Once the fiber orientation has been determined, the fiber is then translated from the global origin point into the box where the concrete mesh will be. This translation is generated randomly via three uniformly distributed values.

2.5 Collision detection methods

When generating fiber geometries, it is necessary to check that individual fibers do not overlap in space. If two fibers intersect, the resulting geometry no longer represents a physically possible configuration, and thus the mesh generation may fail or produce invalid samples. For this reason, collision detection is an important step in generating a fiber filled mesh.

Checking collisions between all fibers at full geometric resolution would be computationally expensive, especially when the number of fibers is large. We adress this problem by using a two-stage collision detection strategy. The first stage performs a fast rough estimate check via axis aligned bounding boxes (AABB) to get rid of the obvious non-intersecting fibers, while the second stage performs a more detailed geometric test, by checking the minimum distance between two line segments, when necessary [17].

2.5.1 Broad-phase collision detection using axis-aligned bounding boxes (AABB)

The first stage of the collision detection algorithm is based on axis-aligned bounding boxes (AABBs). An AABB is the smallest rectangular box, aligned with the global coordinate axes, that fully encloses a given object [17]. In the present case, each fiber is represented by a polyline and assigned an artificial radius corresponding to its physical thickness. The bounding box is therefore constructed from the minimum and maximum coordinates of all polyline points and then expanded by the fiber radius in all three coordinate directions. An example of a fiber together with its axis-aligned bounding box is shown in Figure 2.

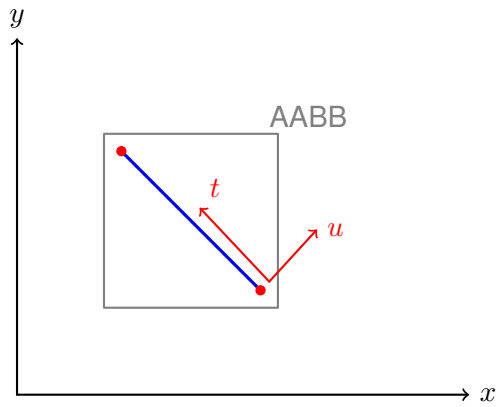

Figure 2. Fiber and its global-axis-based bounding box. Although the fiber is oriented at an angle, the bounding box is aligned with the global coordinate system.

For a fiber represented by points $\mathbf{p}_k = (x_k, y_k, z_k)$, the bounding box is defined by

$$x_{\min} = \min_k x_k, \qquad x_{\max} = \max_k x_k, \tag{2.14}$$

$$y_{\min} = \min_k y_k, \qquad y_{\max} = \max_k y_k, \tag{2.15}$$

$$z_{\min} = \min_k z_k, \qquad z_{\max} = \max_k z_k, \tag{2.16}$$

after which an additional buffer of fiber radius r is added on each side of the box.

Two AABBs overlap if and only if they overlap along all three coordinate directions [17]. Thus, for bounding boxes A and B, overlap occurs when

$$A_{\min,x} \leq B_{\max,x} \text{ and } A_{\max,x} \geq B_{\min,x}, \tag{2.17}$$

$$A_{\min,y} \leq B_{\max,y} \text{ and } A_{\max,y} \geq B_{\min,y}, \tag{2.18}$$

$$A_{\min,z} \leq B_{\max,z} \text{ and } A_{\max,z} \geq B_{\min,z}. \tag{2.19}$$

If these conditions are not satisfied, the two fibers cannot intersect and no further checking is required. This makes the AABB test a very efficient filter, as it quickly rejects any non-colliding fiber pairs [17]. A visual example of overlapping axis-aligned bounding boxes is shown in Figure 3.

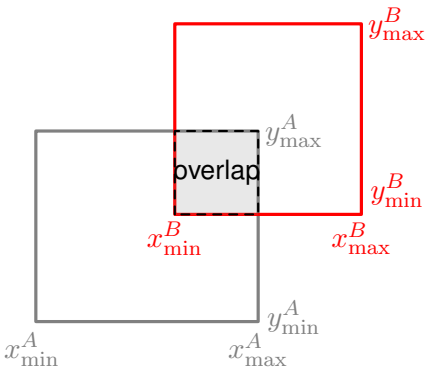

Figure 3. Visual demonstration of two overlapping axis-aligned bounding boxes.

2.5.2 Narrow-phase collision detection using segment–segment distance

The overlap of axis-aligned bounding boxes only indicates a potential collision between fibers and does not guarantee an actual geometric intersection. Therefore, we need a more precise geometric test to determine whether two fibers intersect.

Since each fiber is represented as a polyline, the problem reduces to computing the minimum distance between pairs of line segments in three-dimensional space [17].

Let two line segments be defined by their endpoints $\mathbf{p}_1, \mathbf{q}_1$ and $\mathbf{p}_2, \mathbf{q}_2$. These segments can be expressed in parametric form as

$$\mathbf{L}_1(s) = \mathbf{p}_1 + s(\mathbf{q}_1 - \mathbf{p}_1), \quad s \in [0, 1], \tag{2.20}$$

$$\mathbf{L}_2(t) = \mathbf{p}_2 + t(\mathbf{q}_2 - \mathbf{p}_2), \quad t \in [0, 1]. \tag{2.21}$$

The squared distance between two arbitrary points on the segments is then given by

$$d^2(s,t) = \|\mathbf{L}_1(s) - \mathbf{L}_2(t)\|^2, \tag{2.22}$$

and the goal is to determine the values of s and t that minimise this distance.

To simplify the problem, one first considers the corresponding infinite lines obtained by extending the segments beyond their endpoints. In this case, the parameters s and t are unconstrained, and the minimisation problem can be solved analytically [17].

Introducing the direction vectors

$$\mathbf{d}_1 = \mathbf{q}_1 - \mathbf{p}_1, \quad \mathbf{d}_2 = \mathbf{q}_2 - \mathbf{p}_2, \tag{2.23}$$

and the relative vector

$$\mathbf{r} = \mathbf{p}_1 - \mathbf{p}_2, \tag{2.24}$$

the squared distance can be written as

$$d^2(s,t) = \|\mathbf{r} + s\mathbf{d}_1 - t\mathbf{d}_2\|^2. \tag{2.25}$$

Expanding this expression and minimizing it with respect to the parameters s and t leads to a system of linear equations for the closest points on the two lines [17].

$$as - bt = -c, \tag{2.26}$$

$$-bs + et = f, \tag{2.27}$$

where

$$a = \mathbf{d}_1 \cdot \mathbf{d}_1, \quad b = \mathbf{d}_1 \cdot \mathbf{d}_2, \quad c = \mathbf{d}_1 \cdot \mathbf{r}, \tag{2.28}$$

$$e = \mathbf{d}_2 \cdot \mathbf{d}_2, \quad f = \mathbf{d}_2 \cdot \mathbf{r}. \tag{2.29}$$

This linear system can be solved analytically [17]. Rearranging the first equation gives

$$s = \frac{bt - c}{a}, \tag{2.30}$$

which can be substituted into the second equation. Solving for s yields

$$s = \frac{bf - ce}{ae - b^2}. \tag{2.31}$$

Once s is known, the corresponding value of t is obtained from

$$t = \frac{bs + f}{e}. \tag{2.32}$$

These expressions correspond to the closest points on the supporting infinite lines and are used as candidate values before enforcing the constraints required for finite segments [17].

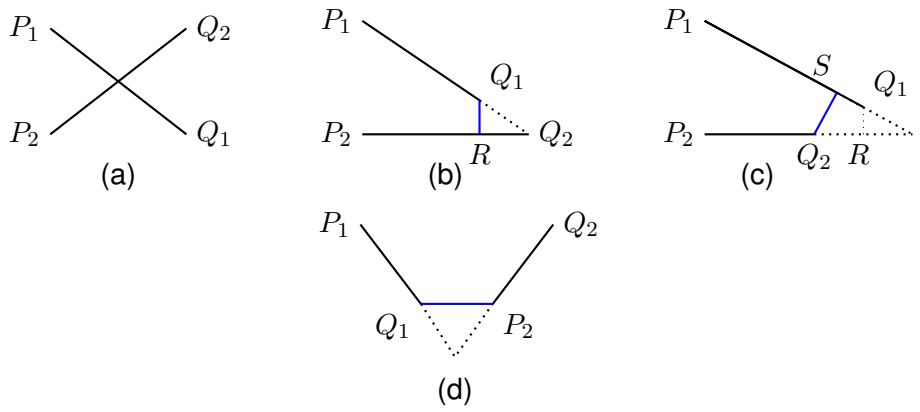

Figure 4. Different closest-point configurations arising in segment–segment distance computations: (a) closest points located within the interiors of both segments, (b)–(c) closest point located at an endpoint of one segment and within the interior of the other segment, and (d) closest points located at endpoints of both segments. Figure inspired by [17].

However, these values are not guaranteed to lie within the interval $[0, 1]$, and therefore may not correspond to valid points on the finite segments. The parameters s and t have a direct geometric interpretation: values $s = 0$ and $s = 1$ correspond to the endpoints of the first segment, while values $0 < s < 1$ represent points along the interior of the segment. The same applies to t for the second segment. Consequently, values outside the interval $[0, 1]$ correspond to points on the supporting infinite lines rather than on the actual segments.

If the solution obtained from the infinite-line minimisation satisfies $s \in [0, 1]$ and $t \in [0, 1]$, then the closest points lie within the interiors of both segments, and the solution is valid. Otherwise, one or both parameters fall outside this interval, indicating that the closest point on at least one line lies beyond the corresponding segment endpoint.

In such cases, the invalid parameter is projected back onto the interval $[0, 1]$, effectively selecting the nearest endpoint of the segment. Once this endpoint is fixed, the problem reduces to determining the closest point on the other segment relative to that endpoint, resulting in a point-to-segment distance computation [17]. Different possible closest-point configurations between two segments are illustrated in Figure 4.

Once the closest points on the two segments have been determined, the minimum distance between them can be computed. A collision is detected if this distance is smaller than the combined radii of the fibers. If each fiber has radius r, the collision condition can be written as

$$d(\overline{\mathbf{p}_1\mathbf{q}_1}, \overline{\mathbf{p}_2\mathbf{q}_2}) < 2r, \tag{2.33}$$

or equivalently,

$$d^2(\overline{\mathbf{p}_1\mathbf{q}_1}, \overline{\mathbf{p}_2\mathbf{q}_2}) < (2r)^2. \tag{2.34}$$

Using the squared distance avoids the need for computing square roots and is therefore more efficient in numerical implementations [17].

2.6 Geometric modelling for finite element analysis

In order to perform finite element analysis of fiber reinforced concrete, it is necessary to construct a geometric model that explicitly represents both the concrete matrix and the embedded fibers. This enables the mechanical behaviour of each phase to be modelled separately and allows interactions between the matrix and the reinforcement to be captured.

In explicit geometric modelling approaches, the domain is represented as a combination of simple geometric primitives, such as boxes, cylinders, or other parametric shapes. More complex geometries are constructed using operations from constructive solid geometry (CSG), including union, intersection, and difference [4]. These operations allow multiple primitives to be combined into a single object or used to remove one region from another.

In the context of fiber reinforced materials, fibers are typically represented as cylindrical inclusions embedded within a surrounding matrix. To obtain a physically meaningful model, the individual fiber segments must first be merged into continuous volumes. These volumes are then subtracted from the matrix domain using Boolean difference operations. The result is a partitioned domain consisting of non-overlapping subregions corresponding to the different material phases.

For finite element simulations, it is essential to assign material properties and boundary conditions to specific regions of the geometry. This is achieved through the use of physical groups, which associate geometric entities such as volumes or surfaces with predefined labels [4]. Physical volumes are used to distinguish material regions, while physical surfaces are used to define boundary conditions.

In addition, appropriate mesh resolution must be chosen to accurately capture geometric details. Regions containing small features, such as fibers, require a finer mesh than the surrounding matrix. This leads to a non-uniform discretisation in which the element size varies according to the local geometric complexity.

We will be performing the geometric modelling and mesh generation using Gmsh [4]. Then the resulting meshes are used for finite element simulations in ElmerFEM [5]. Finally, the resulting stress and deformation fields are visualized and analysed in ParaView [6]. The specific implementation details of these steps will be described in the following chapter.

2.7 Evaluation of effective elastic properties

After generating the fiber reinforced geometries and corresponding finite element meshes, mechanical simulations are performed to estimate the effective elastic properties of the composite material. These properties are evaluated from uniaxial loading simulations performed along the principal coordinate directions.

For small deformations and linear elastic behaviour, the relationship between stress and strain is described by Hooke's law [18]. When a specimen is subjected to a uniaxial load,

the normal stress is defined as

$$\sigma = \frac{F}{A}, \tag{2.35}$$

where F is the applied force and A is the cross sectional area perpendicular to the load direction.

The corresponding normal strain is defined as the relative deformation of the specimen,

$$\varepsilon = \frac{\Delta L}{L_0}, \tag{2.36}$$

where L_0 denotes the original specimen length and ΔL is the change in length after the deformation.

Under linear elastic conditions, Young's modulus is obtained from the ratio between stress and strain,

$$E = \frac{\sigma}{\varepsilon}. \tag{2.37}$$

In anisotropic materials such as fiber reinforced composites, the effective Young's modulus depends on the loading direction. Thus, separate simulations are performed along the x-, y-, and z-directions to estimate

$$E_x = \frac{\sigma_x}{\varepsilon_x}, \qquad E_y = \frac{\sigma_y}{\varepsilon_y}, \qquad E_z = \frac{\sigma_z}{\varepsilon_z}. \tag{2.38}$$

In addition to axial deformation, loading in one direction also produces deformation in the transverse directions. This behaviour is quantified through Poisson's ratio [18], defined as the negative ratio between transverse strain and axial strain,

$$\nu_{ij} = -\frac{\varepsilon_j}{\varepsilon_i}, \tag{2.39}$$

where ε_i is the strain in the loading direction and ε_j is the strain measured in the transverse direction.

For example, when the specimen is loaded along the x-direction, the Poisson ratios are computed as

$$\nu_{xy} = -\frac{\varepsilon_y}{\varepsilon_x}, \qquad \nu_{xz} = -\frac{\varepsilon_z}{\varepsilon_x}. \tag{2.40}$$

Similarly, loading in the remaining coordinate directions yields the corresponding transverse Poisson ratios.

The required deformation quantities are extracted from the finite element simulations using ParaView [6]. The undeformed and deformed geometries are compared by analysing the coordinate bounds of the specimen before and after deformation. From these values, the axial and transverse deformations are determined and subsequently used to compute strains, Young's moduli, and Poisson ratios.

3 Implementation

This chapter describes the practical implementation of the fiber generation and simulation pipeline developed in this work. The objective of the implementation is to generate geometrically valid fiber reinforced concrete or fiber composite specimens that can subsequently be used for finite element analysis.

The chapter begins with the construction of individual fiber geometries and the generation of fiber orientations using probabilistic sampling methods. The implemented collision detection algorithm used to prevent fiber intersections is then presented, followed by the export of the generated geometries into Gmsh for mesh generation and finite element preparation.

Finally, the finite element simulation setup used in ElmerFEM is described, together with the parameter choices adopted for the numerical experiments performed in this thesis.

3.1 Fiber geometry generation

In this section we will explain, and if necessary also elaborate, on our implementation of the whole pipeline for generating fiber reinforced concrete meshes. This includes the assignment of fiber geometry, the fiber orientation and transformations in space, collision detection among the placed fibers, exporting the defined geometry as a valid file for use in mesh generating and finite element analysis software, and finally we'll also cover the setup for our experiments conducted on our defined meshes.

The first step in the modelling pipeline is the generation of individual fiber geometries. In our work, fibers are represented as piecewise-linear curves or polylines, which are later interpreted as cylindrical segments during mesh generation. This allows complex shapes to be constructed while keeping the implementation relatively simple.

The chosen programming language for this task is C++, and we take a very functional programming design approach, where the generation process is structured such that the geometry, orientation, and spatial placement of fibers are handled as separate steps. This separation is intentional, as it allows the same geometric template to be reused while independently controlling orientation and position.

3.1.1 Random number generator

The generation of fiber orientations, positions, and rotational angles relies on pseudo-random sampling. Thus, before we begin generating any fibers, we need to establish

what kind of random number generator we will be using. In our implementation, random numbers are generated using the `std::mt19937_64` engine from the C++ standard library, which is based on the Mersenne Twister algorithm [19].

The Mersenne Twister is a widely used pseudo-random number generator due to its computational efficiency, which makes it a suitable candidate for Monte Carlo type sampling procedures such as the rejection sampling method used for orientation generation [19].

To ensure reproducibility of the results, the generator is initialized with a fixed seed. This guarantees that identical input parameters produce identical fiber configurations, which is necessary for verification and comparison of simulation results.

3.1.2 U-shaped fiber template

Each fiber is initially defined in a local coordinate system as a U-shaped polyline. The geometry consists of four sequentially connected points, forming two straight segments (legs) connected by a central segment (stem).

The shape of the fiber is controlled by two parameters:

- the half-width of the hook, which determines the horizontal length of the fiber,
- the leg height, which determines the length of the vertical segments.

These parameters directly correspond to the geometric construction used in the implementation, where the coordinates of the polyline points are defined relative to the origin.

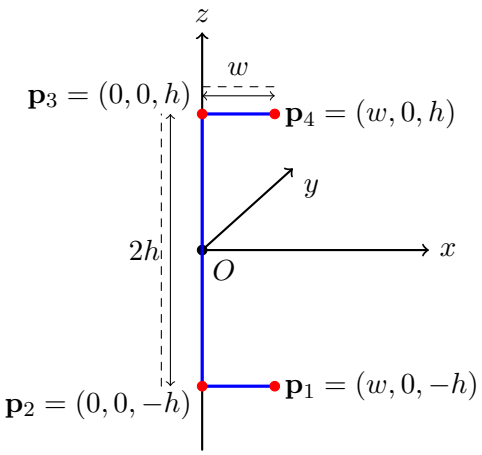

Figure 5. U-shaped fiber template in the local coordinate system. The stem of the fiber is aligned with the local z-axis, while the hook segments extend along the local x-direction.

At this stage, the fiber is centred around the origin and aligned with the local coordinate axes (see figure 5). The coordinates of each point

$$\mathbf{p}_{\text{local}} = (x, y, z) \tag{3.1}$$

should therefore be interpreted purely as defining the shape of the fiber, without any

information about its final orientation or position in space.

This local representation is simply a design choice. By defining the geometry independently of its placement, the same template can later be reused for all fibers, while transformations are applied separately.

We will also emphasize for any potential users of this pipeline, that the fiber geometry should be defined in a manner, where the stem of the fiber is placed symetrically on the Z-axis as demonstrated in figure 5. This is important because future orientations and transformations in the pipeline will use the local Z-axis as the main axis for those operations.

3.1.3 Local coordinate frame construction

To orient the fiber in three-dimensional space, a local orthonormal coordinate frame is constructed. This frame consists of three unit vectors

$$\{\mathbf{u}, \mathbf{v}, \mathbf{t}\}, \tag{3.2}$$

which define the directions of the local coordinate axes after transformation.

The vector $\mathbf{t}$ represents the primary orientation of the fiber and is obtained from the orientation distribution function described in the theory section (chapter 2.4). This vector can be interpreted as the direction along which the stem of the fiber is aligned.

However, specifying $\mathbf{t}$ alone is not enough to fully define the orientation of the fiber. Specifically, it does not determine how the fiber is rotated around its own axis. To resolve this, two additional perpendicular vectors must be constructed.

An additional reference vector $\mathbf{r}$ is selected such that it is not parallel (nor nearly parallel) to $\mathbf{t}$. This vector has no physical meaning and is used simply to construct a perpendicular direction. Using the cross product, the vector

$$\mathbf{u} = \frac{\mathbf{r} \times \mathbf{t}}{\|\mathbf{r} \times \mathbf{t}\|} \tag{3.3}$$

is obtained, which is perpendicular to $\mathbf{t}$. A third vector is then computed as

$$\mathbf{v} = \mathbf{t} \times \mathbf{u}. \tag{3.4}$$

The resulting vectors $\mathbf{u}$, $\mathbf{v}$, and $\mathbf{t}$ form an orthonormal basis, which defines a rotated coordinate system aligned with the fibers stem.

3.1.4 Orientation and twist

The resulting vectors $\mathbf{u}$, $\mathbf{v}$, and $\mathbf{t}$ form an orthonormal basis defining a local coordinate system aligned with the fiber axis. While the direction vector $\mathbf{t}$ determines the main fiber

orientation, the local frame still possesses one remaining rotational degree of freedom corresponding to rotation about the axis $\mathbf{t}$ itself.

To account for this, a random twist angle ψ is introduced. This angle is used to rotate the vectors $\mathbf{u}$ and $\mathbf{v}$ within the plane perpendicular to $\mathbf{t}$:

$$\mathbf{u}' = \cos\psi\,\mathbf{u} + \sin\psi\,\mathbf{v}, \quad \mathbf{v}' = -\sin\psi\,\mathbf{u} + \cos\psi\,\mathbf{v}. \tag{3.5}$$

This ensures that fibers with identical direction vectors $\mathbf{t}$ can still differ in their spatial configuration. Without this step, all fibers aligned in the same direction would also share the same rotational orientation, which would introduce an artificial regular patern into the model.

3.1.5 Transformation to global coordinates

Once the local coordinate frame has been constructed, the fiber geometry is transformed into global coordinates.

A point

$$\mathbf{c} = (c_x, c_y, c_z) \tag{3.6}$$

is sampled uniformly within the domain of the concrete. This point represents the centre of the fiber and determines its spatial location.

Each point of the fiber, originally defined in local coordinates as

$$\mathbf{p}_{\text{local}} = (x, y, z), \tag{3.7}$$

is then mapped to global coordinates using the transformation

$$\mathbf{p}_{\text{global}} = \mathbf{c} + x\mathbf{u} + y\mathbf{v} + z\mathbf{t}. \tag{3.8}$$

This equation expresses the local coordinates in terms of the rotated coordinate system defined by $(\mathbf{u}, \mathbf{v}, \mathbf{t})$. In other words, the directions of the local axes are replaced by the corresponding vectors of the constructed frame. The resulting vector is then translated by $\mathbf{c}$, placing the fiber at its final position within the concrete domain.

It is important to emphasize that this transformation does not modify the shape of the fiber. Instead, it only affects its orientation and position. Thus, all fibers share the same geometric template while differing in their spatial arrangement.

3.1.6 Summary of the generation process

The complete fiber generation procedure can be summarised as follows:

1. Define a U-shaped fiber in a local coordinate system,

2. Sample an orientation vector $\mathbf{t}$ from the ODF,
3. Construct an orthonormal frame $(\mathbf{u}, \mathbf{v}, \mathbf{t})$,
4. Apply a random twist angle ψ,
5. Transform all template points into global coordinates and assign a position $\mathbf{c}$.

This structured approach allows geometry, orientation, and spatial placement to be controlled independently, resulting in a flexible and extensible implementation.

3.2 Orientation sampling

In the previous section, we showed how a fiber can be constructed and oriented in space once a direction vector $\mathbf{t}$ is known. But how is this direction vector achieved?

As discussed in Sections 2.2–2.4, fiber orientations are described using an orientation distribution function (ODF) defined on the unit sphere, and sampled using rejection sampling.

3.2.1 Evaluation of the orientation distribution function

The orientation distribution function is evaluated using the second-order approximation introduced in Section 2.3. For a unit direction vector

$$\mathbf{n} = (n_1, n_2, n_3), \tag{3.9}$$

the ODF is computed as

$$f(\mathbf{n}) = \frac{1}{4\pi}\left(1 + \frac{15}{2} a_{ij} n_i n_j\right). \tag{3.10}$$

In our code, the quadratic form $a_{ij}n_in_j$ is evaluated explicitly using the components of the direction vector and the alignment tensor:

$$a_{ij}n_in_j = a_{11}n_1^2 + a_{22}n_2^2 + a_{33}n_3^2 + 2a_{12}n_1n_2 + 2a_{13}n_1n_3 + 2a_{23}n_2n_3. \tag{3.11}$$

Due to some numerical effects or certain tensor configurations, the computed value of $f(\mathbf{n})$ may become slightly negative. Since the ODF represents a probability density, negative values are not physically meaningful and are therefore clipped to zero in the implementation.

3.2.2 Generation of candidate directions

Candidate direction vectors are generated using the uniform sampling procedure on the unit sphere described in Section 2.2. This ensures that all directions are initially sampled without bias before applying the orientation distribution function.

3.2.3 Rejection sampling procedure

To obtain directions that follow the prescribed ODF, rejection sampling is employed as described in Section 2.4. The implementation follows these steps:

1. Generate a candidate direction $\mathbf{n}$ uniformly on the unit sphere,
2. Evaluate the orientation distribution function $f(\mathbf{n})$,
3. Generate a random number $u \in [0, M]$,
4. Accept the direction if $u \leq f(\mathbf{n})$, otherwise reject it.

This process is repeated until a direction is accepted. The accepted direction is then used as the fiber orientation vector $\mathbf{t}$.

3.2.4 Choice of the upper bound

The efficiency of rejection sampling depends on the choice of the upper bound M, which must satisfy

$$f(\mathbf{n}) \leq M \quad \text{for all } \mathbf{n} \in S^2. \tag{3.12}$$

In the present implementation, the alignment tensor is prescribed in diagonal form. The upper bound M is therefore computed from the largest diagonal component of the tensor, which corresponds to the maximum eigenvalue in this special case.

Thus, the upper bound is taken as

$$M = \frac{1}{4\pi}\left(1 + \frac{15}{2}\lambda_{\max}\right), \tag{3.13}$$

where $\lambda_{\max}$ is obtained as the largest diagonal entry of the prescribed alignment tensor.

This choice ensures that the rejection sampling procedure remains valid while avoiding excessive rejection of candidate samples.

3.2.5 Summary of the sampling procedure

The orientation sampling procedure can be summarised as follows:

1. Generate a candidate direction uniformly on the unit sphere,
2. Evaluate the ODF using the alignment tensor,
3. Accept or reject the direction using rejection sampling,
4. Use the accepted direction as the fiber orientation vector $\mathbf{t}$.

This implementation provides a flexible mechanism for generating fiber orientations with controlled anisotropy, while remaining efficient and relatively simple to integrate into the overall fiber generation pipeline.

3.3 Collision detection algorithm

During the fiber placement process, it is necessary to ensure that fibers do not intersect. In the present implementation, each fiber is represented as a sequence of cylindrical segments with radius r. A collision is therefore defined as occurring when the distance between any pair of segments is smaller than $2r$.

A direct comparison between all fibers would be computationally expensive. To address this, a two-stage collision detection algorithm is employed. A fast broad-phase check is first used to eliminate clearly non-interacting fibers, followed by a more detailed narrow-phase check applied only to potential collision candidates.

3.3.1 Broad-phase: axis-aligned bounding boxes

For each fiber, an axis-aligned bounding box (AABB) is constructed based on the minimum and maximum coordinates of its polyline points. The bounding box is expanded by the fiber radius to ensure that it fully encloses the cylindrical geometry.

Two fibers are considered potential collision candidates if their bounding boxes overlap. If no overlap is detected, the fibers cannot intersect and no further computation is required.

This stage provides a computationally inexpensive filtering step that significantly reduces the number of segment comparisons required in the narrow-phase.

3.3.2 Narrow-phase: segment-to-segment distance

If two bounding boxes overlap, a more precise collision check takes place. Since fibers are represented as polylines, we reduce the problem to evaluating the distance between all pairs of line segments belonging to the two fibers.

For each pair of segments, the minimum distance is computed using an analytical solution for the closest points between two line segments in three-dimensional space. In the implementation, this is performed by the function `segSegDist2`, which evaluates the squared distance between segments.

The method first computes the closest points on the corresponding infinite lines. The lines are expressed as parametric equations using parameters s and t. However, since the actual geometry consists of finite segments, these parameters have to be restricted to the interval $[0, 1]$.

This restriction is introduced via the clamping operation:

$$s = \mathrm{clamp}(s, 0, 1), \quad t = \mathrm{clamp}(t, 0, 1), \tag{3.14}$$

which ensures that the resulting points lie on the actual segments rather than on the extended lines.

If either parameter falls outside the valid range, the corresponding point is projected

onto the nearest segment endpoint, and the remaining parameter is recomputed accordingly. This effectively reduces the problem to point-to-segment or point-to-point distance calculations in such cases.

Some special cases, such as very small lengths or nearly parallel segments, are handled separately in the implementation to ensure numerical robustness.

Once the parameters have been determined, the closest points on the segments are computed and the squared distance between them is evaluated.

A collision is detected if

$$d^2 < (2r)^2, \tag{3.15}$$

where r is the fiber radius. The use of squared distance avoids unnecessary square root operations and improves computational efficiency.

This procedure is repeated for all segment pairs between the two fibers. If any of the pairs satisfy the collision condition, the fibers are considered to intersect.

3.3.3 Placement limit

For the fiber volume fractions considered in this work, the probability of successfully placing a new fiber without intersection is relatively high. However, as the density of fibers increases, the placement problem becomes progressively more constrained. In such cases, the algorithm may require a large number of attempts to find a valid, non-colliding configuration. In extreme situations, it may become geometrically impossible to place additional fibers.

To prevent the algorithm from entering an infinite loop during candidate generation, a maximum number of placement attempts is introduced for each fiber. If a valid configuration is not found within this limit, the algorithm exits the placement process and reports the number of successfully placed fibers.

The choice of limit is arbitrary, yet has trade-offs. A value that is too small may cause the algorithm to give up too early, leading to an underestimation of the achievable fiber density. Similarly, a value that is too large may result in unnecessary computational cost in situations where no further valid placements are possible.

In our current implementation, the maximum number of placement attempts is set to

$$N_{\text{max}} = 10000. \tag{3.16}$$

3.4 Geometry export and finite element model preparation

Once the fiber geometries have been generated and positioned within the domain, the next step is to construct a corresponding geometric model suitable for finite element analysis. In the present work, this is achieved by exporting the generated fiber data to a `.geo` file,

which serves as input for the mesh generator Gmsh.

The overall process consists of three main stages:

1. Construction of fiber geometries using cylindrical primitives,
2. Assembly of the concrete matrix and fiber volumes using Boolean operations,
3. Assignment of physical groups for material properties and boundary conditions.

Most of these steps are generated programmatically in C++, while some simulation-specific definitions, such as the identification of boundary surfaces, are currently assigned manually in Gmsh.

3.4.1 Representation of fibers using cylindrical segments

Each fiber is represented internally as a polyline consisting of a sequence of points in three-dimensional space. For the purpose of mesh generation, these polylines are converted into cylindrical segments, where each segment connects two consecutive points.

More specifically, for each pair of points $\mathbf{p}_i$ and $\mathbf{p}_{i+1}$, a cylinder is defined with its base at $\mathbf{p}_i$ and its axis aligned with the vector $\mathbf{p}_{i+1} - \mathbf{p}_i$. The radius of the cylinder corresponds to the physical radius of the fiber.

This representation allows complex fiber shapes, such as hooked fibers, to be approximated as a sequence of connected cylindrical elements, while remaining compatible with the geometric primitives supported by Gmsh. An example of the corresponding Gmsh code is shown below.

```
SetFactory("OpenCASCADE");
Box(1) = {-2,-2,-2,13,13,13};
```

Here, the first three values define the lower corner of the box, while the remaining three values define its dimensions along the coordinate axes.

```
Cylinder(2) = {3.80654,1.30022,5.79953,
   -0.184018,-0.446965,0.127906,
0.1,2*Pi };
```

Here, the first three values define the starting point of the cylinder, the next three values define the segment direction vector, and the final value before `2*Pi` defines the fiber radius.

3.4.2 Construction of fiber and matrix volumes

Since each fiber consists of multiple cylindrical segments, these segments must first be combined into a single continuous volume. This is achieved using Boolean union operations, which merge all cylinders belonging to a given fiber into one solid object.

```
BooleanUnion{Volume{2}; Delete;}{Volume{3,4}; Delete;}
```

After constructing the individual fiber volumes, the concrete matrix is defined as a rectangular box representing the specimen domain. The fiber volumes are then subtracted from this box using a Boolean difference operation. This ensures that the resulting geometry consists of two non-overlapping regions: the concrete matrix and the embedded fibers.

```
BooleanDifference{ Volume{1}; Delete;}{
  Volume{32,33,34,35,36,37,38,39,40,41};
}
```

The outcome of this procedure is a geometrically consistent model in which the fibers occupy distinct volumetric regions within the concrete, enabling separate material properties to be assigned during finite element analysis.

3.4.3 Assignment of physical groups

In order to use the generated geometry in finite element simulations, it is necessary to assign physical groups to the relevant geometric entities.

Physical volumes are defined to distinguish between the different material phases. In our current work, one physical volume is assigned to the concrete matrix, while a second physical volume represents all fiber inclusions. These groups are later used in the finite element solver to assign appropriate material models.

```
Physical Volume("concrete") = {1};
Physical Volume("allFibers") = {32,33,34,35,36,37,38,39,40,41};
```

The physical volumes define separate material regions for the concrete matrix and the embedded fibers.

In addition to material regions, boundary conditions must be applied to the outer surfaces of the specimen. For this purpose, the six external faces of the concrete domain (top, bottom, left, right, front, and back) are identified and assigned to physical surface groups.

```
Physical Surface("top") = {189};
Physical Surface("bottom") = {191};

Physical Surface("left") = {187};
Physical Surface("right") = {192};

Physical Surface("front") = {190};
Physical Surface("back") = {188};
```

In the current implementation, the identification of these boundary surfaces is performed manually within Gmsh. The corresponding surface identifiers are then used to define physical surface groups with consistent labels. This allows the same boundary condition definitions to be reused across multiple simulations, provided that the surface labeling remains unchanged.

3.4.4 Mesh resolution and export

To accurately capture the geometry of the fibers, a non-uniform mesh is employed. A coarser mesh is used for the concrete matrix, while a finer mesh is applied in regions corresponding to the fibers. This is achieved by assigning different characteristic lengths to the points belonging to each volume.

```
Characteristic Length{
  PointsOf{ Volume{32,33,34,35,36,37,38,39,40,41}; }
} = 0.1;

Characteristic Length{
  PointsOf{ Surface{189,191,187,192,190,188}; }
} = 1.0;
```

Here, a smaller characteristic length is assigned to the fiber regions in order to better resolve their geometry, while a larger characteristic length is used for the outer concrete surfaces to reduce the overall computational cost. An example of the resulting finite element mesh is shown in Figure 6.

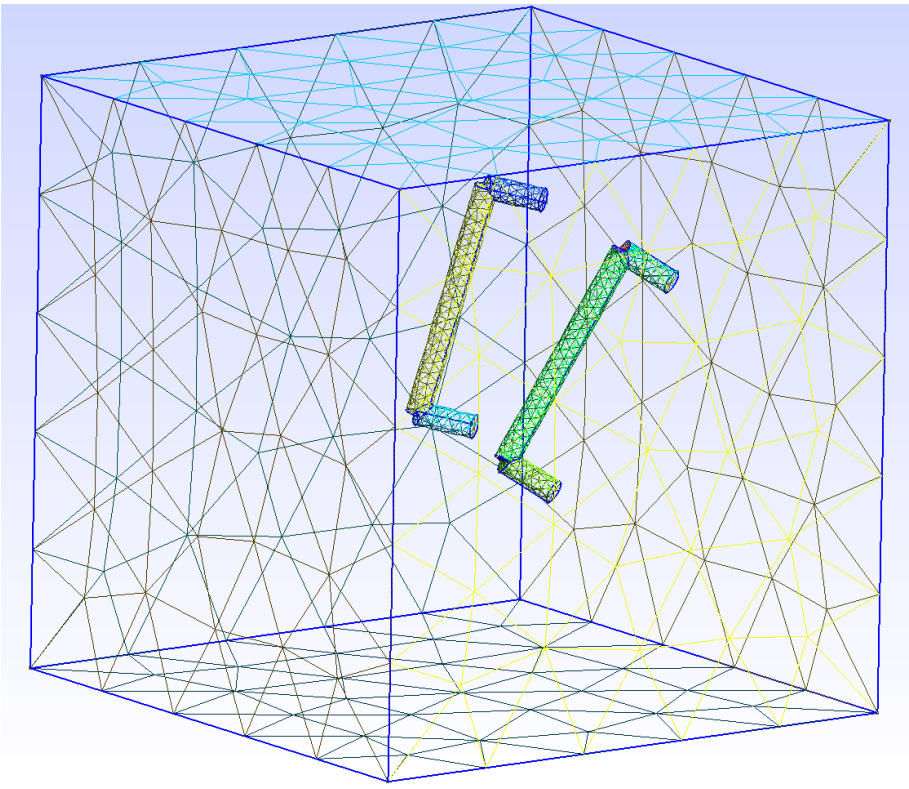

Figure 6. Example of the generated finite element mesh with locally refined mesh resolution near the hooked fibers.

After defining the geometry and mesh size parameters, a three-dimensional mesh is generated using Gmsh. The resulting mesh is exported in `.msh` format, which is subsequently used as input for finite element simulations.

In practice, the generated mesh is converted into a format compatible with ElmerFEM using

the `ElmerGrid` utility. This conversion step preserves the physical groups defined in Gmsh, ensuring that material properties and boundary conditions can be applied consistently in the simulation environment.

3.4.5 Integration with finite element simulations

The generated mesh, including its physical volume and surface groups, forms the basis for the finite element simulations performed in this work. The physical volumes are used to assign material properties to the concrete matrix and fiber inclusions, while the physical surfaces are used to impose boundary conditions such as prescribed displacements or loads.

An important practical aspect of this setup is that multiple simulations can reuse the same solver configuration, as long as the naming and ordering of physical groups remain consistent. This allows different fiber configurations to be analysed efficiently by simply replacing the mesh file while keeping the simulation setup unchanged.

3.5 Simulation setup

With the fiber generation pipeline established, we proceed to define the simulation cases used to evaluate its performance and to investigate the influence of fiber geometry.

Three specimen configurations are considered. The first is a homogeneous specimen without fibers, which is used as a reference case. The second uses hooked U-shaped fibers, while the third uses straight fibers. In order to ensure a fair comparison between the hooked and straight fiber cases, a fixed random seed is used for the orientation sampling. This guarantees that both reinforced specimens are based on identical sets of orientation vectors, allowing any observed differences in the results to be attributed primarily to the fiber geometry.

The geometry of the hooked fibers is defined through the parameters `halfWidth` and `legHeight`, which control the horizontal length of the hooks and the length of the vertical segments, respectively. Straight fibers are obtained by modifying the template such that only two points remain. In this case, the `legHeight` parameter does not influence the final geometry.

Fiber orientations are generated using a prescribed alignment tensor. In the present simulations, a tensor corresponding to preferential alignment along the global z-axis is used.

The placement domain is defined as a cubic region of size $10 \times 10 \times 10$. For the purpose of geometry export and mesh generation, a slightly larger bounding box is used to ensure that all fibers are fully contained within the computational domain.

Mesh resolution is controlled through two characteristic length scales. A coarser mesh size of 1.0 is assigned to the concrete matrix, while a finer mesh size of 0.1 is used in

regions containing fibers. This allows for improved resolution of the fiber geometry without excessively increasing the overall computational cost.

A summary of the input parameters used in the simulations is provided in Table 1.

Table 1. Input parameters used in the fiber generation pipeline.

Parameter	**Value**
Random seed	1000
Number of fibers	10
Fiber radius	0.1
Hook half-width	1.0
Hook leg height	0.5
Placement domain	$[0,0,0] \to [10,10,10]$
Exported concrete domain	$[-2,-2,-2] \to [13,13,13]$
Alignment tensor	$\begin{pmatrix} -1/3 & 0 & 0 \\ 0 & -1/3 & 0 \\ 0 & 0 & 2/3 \end{pmatrix}$
Maximum placement attempts	10000
Concrete mesh size	1.0
Fiber mesh size	0.1

4 Results

This chapter presents the finite element simulations performed on the generated fiber reinforced “concrete” specimens. The simulations were carried out using ElmerFEM, while ParaView was used for post-processing and visualization of the resulting deformation and stress fields.

The purpose of these simulations is to demonstrate the complete modelling pipeline and to obtain estimates of the effective elastic response of the generated specimens under uniaxial tensile loading.

4.1 Finite element setup

The generated Gmsh meshes were converted into a format compatible with ElmerFEM using the ElmerGrid utility. The simulations were performed using the linear elasticity solver.

Three different specimen configurations were considered:

1. a homogeneous “concrete” cube without fibers,
2. a homogeneous cube reinforced with ten hooked U-shaped fibers,
3. a homogeneous cube reinforced with ten straight fibers.

All simulations used the same “concrete” specimen dimensions of

$$0.15 \times 0.15 \times 0.15 \text{ m}.$$

Real concrete typically has a Young’s modulus in the range of 30–50 GPa. To better observe the influence of a relatively low number of fibers, a softer homogeneous matrix material was selected for the simulations. The chosen stiffness corresponds more closely to structural plastics than to conventional concrete.

The matrix was modelled as a linear elastic isotropic material with

$$E_c = 2.7 \times 10^9 \text{ Pa}, \tag{4.1}$$

and

$$\nu_c = 0.3. \tag{4.2}$$

The steel fibers were assigned

$$E_f = 200 \times 10^9 \text{ Pa}, \tag{4.3}$$

and

$$\nu_f = 0.4. \tag{4.4}$$

For the fiber reinforced simulations, the fibers were generated using the same orientation distribution and identical random seed in order to allow comparison between hooked and straight fiber geometries. The orientation tensor was chosen such that the fibers exhibited preferential alignment along the global z-axis.

The resulting displacement and stress fields were visualized in ParaView. The effective elastic properties were estimated from axial and transverse displacements measured using the `Hover Points` tool.

4.2 Boundary conditions

Three separate uniaxial tensile loading cases were considered, corresponding to loading along the global x-, y-, and z-directions.

For the x-direction loading case, displacement in the x-direction was constrained on the right boundary surface,

$$u_x = 0, \tag{4.5}$$

while a traction load of

$$14500 \text{ Pa}$$

was applied to the opposite left surface.

For the y-direction loading case, displacement in the y-direction was constrained on the back boundary surface,

$$u_y = 0, \tag{4.6}$$

while the same traction load was applied to the front surface.

For the z-direction loading case, displacement in the z-direction was constrained on the top surface,

$$u_z = 0, \tag{4.7}$$

while the traction load was applied to the bottom surface.

These boundary conditions produce uniaxial tensile deformation along the selected coordinate direction while constraining rigid body motion of the specimen.

4.3 Verification using an empty “concrete” cube

As an initial verification step, a simulation was performed using a homogeneous “concrete” cube without fibers. Since the material model is isotropic, loading along a single coordinate direction is sufficient for validation.

The extracted elastic properties are summarized in Table 2.

Table 2. Computed elastic properties for the empty “concrete” cube.

Property	Value
E_Y	2.699999 GPa
ν_{YX}	0.300125
ν_{YZ}	0.300108

The computed Young’s modulus and Poisson ratios agree closely with the material parameters prescribed in the ElmerFEM model. This confirms that the simulation setup, traction boundary conditions, displacement extraction procedure, and post-processing computations are functioning correctly.

4.4 Hooked fiber reinforced “concrete”

The second set of simulations considered a “concrete” cube containing ten hooked U-shaped fibers. Tensile loading was applied independently along all three coordinate directions.

Figure 7 shows an example of the resulting von Mises stress field visualization.

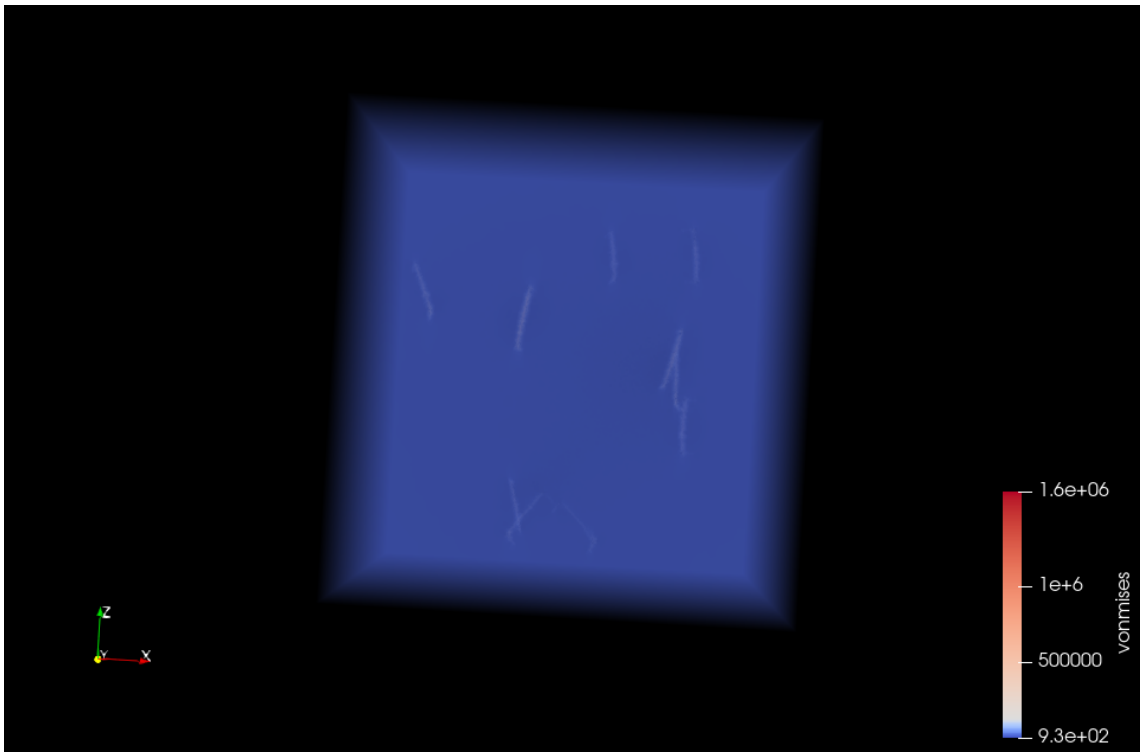

Figure 7. Von Mises stress visualization for the specimen reinforced with hooked fibers.

The extracted elastic properties are summarized in Table 3.

The computed elastic properties remain close to those of the “concrete” cube. Since the fiber volume fraction is relatively small compared to the surrounding “concrete” matrix, only minor changes in the overall elastic response are observed.

Table 3. Computed elastic properties for the specimen reinforced with hooked fibers.

Loading direction	Young's modulus [GPa]	Poisson ratio 1	Poisson ratio 2
X	2.704665	$\nu_{XY} = 0.300276$	$\nu_{XZ} = 0.298760$
Y	2.702693	$\nu_{YX} = 0.297445$	$\nu_{YZ} = 0.299351$
Z	2.708198	$\nu_{ZX} = 0.300409$	$\nu_{ZY} = 0.298399$

Small differences between the loading directions can nevertheless be observed. In particular, the specimen exhibits slightly larger stiffness in the z-direction, which is consistent with the preferential fiber alignment along the global z-axis introduced by the orientation distribution.

4.5 Straight fiber reinforced "concrete"

The final set of simulations considered straight fibers generated using the same orientation distribution and random seed as the hooked fiber case.

Figure 8 shows the corresponding von Mises stress visualization.

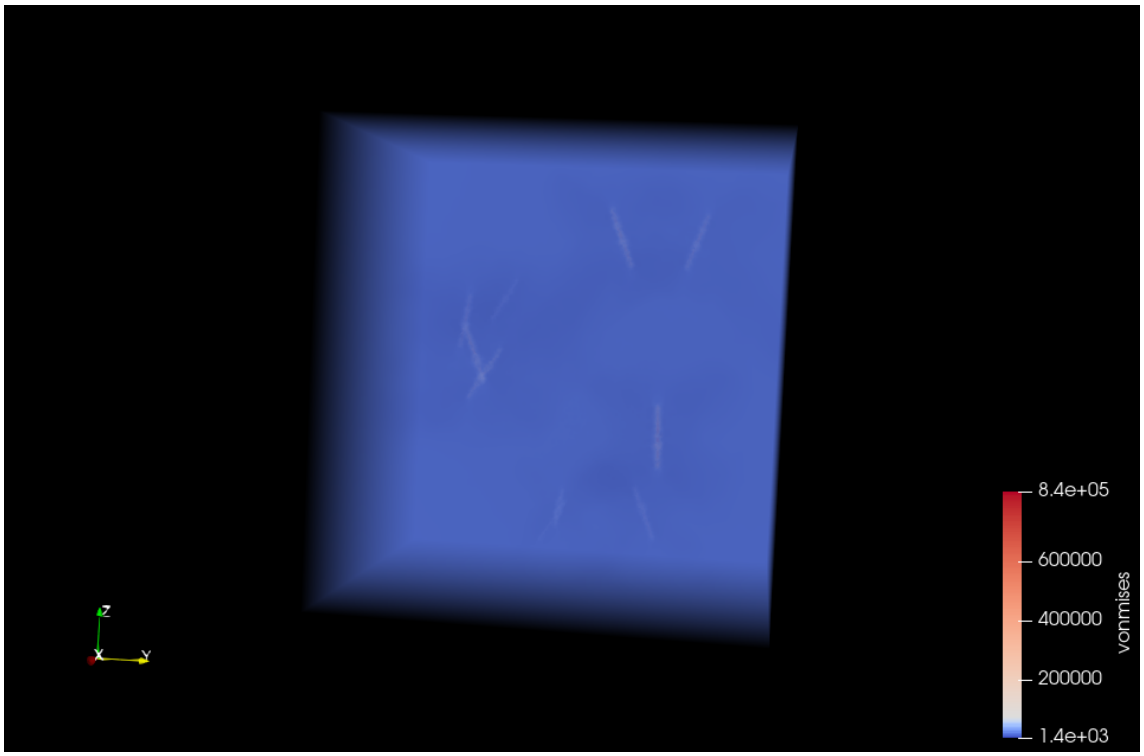

Figure 8. Von Mises stress visualization for the specimen reinforced with straight fibers.

The extracted elastic properties are summarized in Table 4.

Similarly to the hooked fiber simulations, the resulting elastic properties remain close to the properties of the plain "concrete" specimen. The differences between the loading directions are again relatively small, indicating that the low fiber volume fraction only slightly modifies the overall stiffness of the composite.

Table 4. Computed elastic properties for the specimen reinforced with straight fibers.

Loading direction	Young's modulus [GPa]	Poisson ratio 1	Poisson ratio 2
X	2.702723	$\nu_{XY} = 0.300630$	$\nu_{XZ} = 0.299852$
Y	2.701622	$\nu_{YX} = 0.302788$	$\nu_{YZ} = 0.304060$
Z	2.706463	$\nu_{ZX} = 0.299452$	$\nu_{ZY} = 0.300303$

4.6 Comparison between hooked and straight fibers

A comparison between the hooked and straight fiber simulations indicates that both geometries produce very similar effective elastic properties under the considered loading conditions. It is noteworthy to mention, that the maximum von Mises stress shown in Figure 7 is larger than in Figure 8.

The computed Young's modulus values differ only slightly between the two cases, with both fiber configurations exhibiting marginally increased stiffness in the preferred fiber orientation direction. The Poisson ratios also remain close to the reference "concrete" value of 0.3.

These results suggest that, for the relatively small fiber volume fraction considered in this work, the global linear elastic response is dominated primarily by the "concrete" matrix. Thus, the influence of the specific fiber geometry remains limited in small-strain elastic simulations.

Nevertheless, the developed modelling framework successfully demonstrates the capability to explicitly generate, place, mesh, and simulate different fiber geometries within a finite element environment. This provides a foundation for future studies involving larger fiber contents, more complex fiber geometries, or non-linear material behaviour such as crack propagation and fiber pull-out.

5 Summary

In this thesis, a computational framework for the explicit geometric modelling and finite element simulation of steel fiber reinforced concrete was developed. The primary objective was to generate realistic three-dimensional fiber geometries, place them within a concrete specimen without geometric intersections, and prepare the resulting models for finite element analysis.

The developed methodology represents fibers as polylines consisting of connected line segments. A local coordinate representation was introduced for defining hooked U-shaped fibers, after which random orientations and translations were applied to position the fibers within the specimen domain. Fiber orientations were generated using an orientation distribution function together with rejection sampling, allowing preferential alignment along prescribed directions.

To ensure geometric consistency, a two-stage collision detection algorithm was implemented. First, axis-aligned bounding boxes were used as a broad-phase collision test to efficiently eliminate non-intersecting candidates. Potential overlaps were then verified using a narrow-phase segment-segment distance computation based on closest-point minimization between finite line segments.

The resulting fiber geometries were converted into volumetric representations using cylindrical primitives in Gmsh. Boolean union and Boolean difference operations were subsequently used to construct separate material regions for the fibers and the surrounding concrete matrix. The generated geometries were then meshed and exported for finite element simulations in ElmerFEM.

Finite element simulations were performed under uniaxial tensile loading along the principal coordinate directions. The resulting deformation and stress fields were visualized in ParaView using warped displacement visualizations and von Mises stress rendering. The extracted elastic properties for the empty concrete specimen closely matched the prescribed material parameters, confirming the correctness of the simulation setup, loading conditions, and post-processing procedure.

Additional simulations were performed for concrete specimens reinforced with both hooked and straight fibers. The computed effective elastic properties remained close to those of the plain concrete matrix, which is consistent with the relatively low fiber volume fraction considered in this work. Slight increases in stiffness were observed in the preferred fiber orientation direction, demonstrating that the generated fiber distributions influence the mechanical response of the composite.

The simulations demonstrated successful integration of:

1. fiber geometry generation,
2. probabilistic fiber orientation sampling,
3. collision-free fiber placement,
4. constructive solid geometry modelling,
5. mesh generation,
6. finite element simulation,
7. and visualization of stress and deformation fields.

Overall, the present thesis demonstrates a complete computational workflow for explicit geometric modelling and finite element preparation of fiber reinforced concrete specimens. The developed framework provides a foundation for future investigations involving more realistic fiber geometries, advanced material models, and large-scale simulations of fiber reinforced concrete behaviour.

5.1 Future work and outlook

Several possible extensions and improvements of the present framework can be identified.

The current fiber representation intentionally uses simplified hooked geometries consisting of only three connected cylindrical segments. While this approximation is sufficient for demonstrating the modelling pipeline and finite element workflow, industrial steel fibers often possess significantly more complex geometries. Future work could therefore incorporate more realistic fiber models using larger numbers of cylindrical and spherical primitives. For example, a more realistic hooked fiber could be represented using five cylindrical segments together with four spherical transition regions, as illustrated in Figure 9. Such geometries would allow improved representation of curved hooks and local shape transitions.

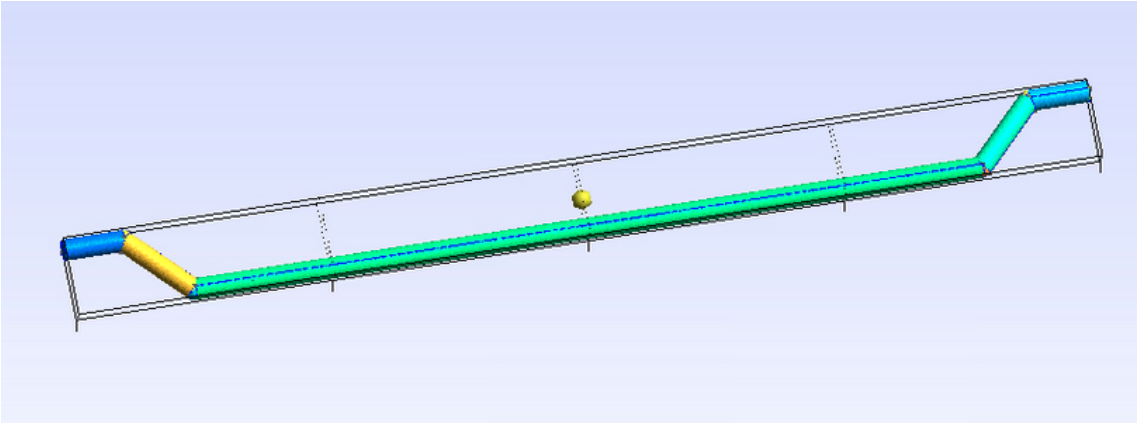

Figure 9. Example of a more geometrically detailed hooked fiber representation using multiple cylindrical segments and spherical transition regions.

Another possible extension concerns collision handling near specimen boundaries. In the current implementation, fibers are prevented from approaching the outer surfaces too

closely by using conservative placement constraints. This simplification was intentionally chosen in order to keep the collision detection procedure manageable. However, in real fiber reinforced concrete, fibers located near boundaries often exhibit orientation-dependent placement restrictions, commonly referred to as boundary effects. Future work could therefore extend the collision detection algorithm to allow fibers to be placed closer to specimen surfaces when their orientations remain physically admissible, for example when fibers align approximately parallel to the boundary.

The present work also considered only small-deformation linear elastic simulations. Consequently, important non-linear mechanisms such as crack initiation, fracture propagation, debonding, and fiber pull-out were not modelled. Future work could therefore investigate alternative finite element tools or constitutive formulations capable of simulating fracture processes in fiber reinforced concrete. Such approaches would enable investigation of post-cracking behaviour and failure mechanisms, which are among the most important mechanical characteristics of steel fiber reinforced concrete.

In addition, the developed framework could be extended to larger specimens containing substantially greater fiber counts and more advanced orientation distributions. This would allow investigation of anisotropic reinforcement behaviour, fiber clustering, and statistical variability of the effective material response.

6 Acknowledgements

I would first like to express my sincere gratitude to my supervisor, Heiko Jens Herrmann, who, despite his many responsibilities, always found time to guide me and help explain concepts that were initially difficult to grasp.

I would also like to thank the lecturers of the Applied Physics and Data Science curriculum at TalTech, who have been excellent role models throughout my studies.

In addition, I would like to acknowledge the use of artificial intelligence tools, specifically ChatGPT, during the writing process of this thesis. These tools were primarily used for grammar checking, language refinement, and improving the clarity of phrasing.

References

[1] Arnon Bentur and Sidney Mindess. *Fibre-Reinforced Cementitious Composites*. CRC Press, 1990.

[2] Hu Feng et al. „Pullout Behaviour of Different Types of Steel Fibres Embedded in Magnesium Phosphate Cementitious Matrix“. In: *International Journal of Concrete Structures and Materials* (Jan. 2019). URL: https://ro.uow.edu.au/articles/journal_contribution/Pullout_Behaviour_of_Different_Types_of_Steel_Fibres_Embedded_in_Magnesium_Phosphate_Cementitious_Matrix/27763614.

[3] Suresh G. Advani and Charles L. Tucker III. „The Use of Tensors to Describe and Predict Fiber Orientation in Short Fiber Composites“. In: *Journal of Rheology* 31.8 (1987), pp. 751–784. DOI: 10.1122/1.549945.

[4] Christophe Geuzaine and Jean-François Remacle. „Gmsh: A 3-D finite element mesh generator with built-in pre- and post-processing facilities“. In: *International Journal for Numerical Methods in Engineering* 79.11 (2009), pp. 1309–1331. DOI: https://doi.org/10.1002/nme.2579. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/nme.2579. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/nme.2579.

[5] CSC – IT Center for Science. *Elmer Models Manual*. https://www.nic.funet.fi/index/elmer/doc/ElmerModelsManual.pdf. Accessed: 2026-05-11. 2026.

[6] James Ahrens, Berk Geveci, and Charles Law. „ParaView: An End-User Tool for Large Data Visualization“. In: *The Visualization Handbook* (2005), pp. 717–731.

[7] A. M. Neville. *Properties of Concrete*. Pearson Education, 2011.

[8] ACI Committee 544. *ACI 544: State-of-the-Art Report on Fiber Reinforced Concrete*. 2009.

[9] Antoine E. Naaman. „Engineered Steel Fibers with Optimal Properties for Reinforcement of Cement Composites“. In: *Journal of Advanced Concrete Technology* 1 (2003), pp. 241–252. URL: https://api.semanticscholar.org/CorpusID:56288346.

[10] George B. Arfken and Hans J. Weber. *Mathematical Methods for Physicists*. Academic Press, 2013.

[11] Nick Loomis. *Generating Random Directions*. https://loomsci.wordpress.com/2015/03/14/generating-random-directions/. Accessed: 2026-05-11. 2015.

[12] Eric W. Weisstein. *Sphere Point Picking*. https://mathworld.wolfram.com/SpherePointPicking.html. Accessed: 2026-05-17.

[13] Heiko Herrmann and Miriam Beddig. „Tensor series expansion of a spherical function for the use in constitutive theory of materials containing orientable particles“. In: *Proceedings of the Estonian Academy of Sciences* 67.1 (2018), pp. 73–92.

[14] Eric W. Weisstein. *Double Factorial*. https://mathworld.wolfram.com/DoubleFactorial.html. Wolfram MathWorld, Accessed: 2026-05-11. 2002.

[15] Luc Devroye. *Non-Uniform Random Variate Generation*. Springer, 1986.

[16] Christian P. Robert and George Casella. *Monte Carlo Statistical Methods*. Springer, 2004.

[17] Christer Ericson. *Real-Time Collision Detection*. San Francisco: Morgan Kaufmann, 2005. ISBN: 9781558607323.

[18] Ansel C. Ugural and Saul K. Fenster. *Advanced Strength and Applied Elasticity*. 5th ed. Prentice Hall, 2011.

[19] Makoto Matsumoto and Takuji Nishimura. „Mersenne Twister: A 623-Dimensionally Equidistributed Uniform Pseudo-Random Number Generator“. In: *ACM Transactions on Modeling and Computer Simulation* 8.1 (1998), pp. 3–30. DOI: 10.1145/272991.272995.

Appendix 1 – Non-exclusive licence for reproduction and publication of a graduation thesis[1]

I Artur Raag

1. Grant Tallinn University of Technology free licence (non-exclusive licence) for my thesis “Generating Non-Intersecting Fiber Distributions for Finite Element Modeling of Short Fiber Reinforced Composites”, supervised by Dr. rer. nat. habil. Heiko Jens Herrmann
    1.1. to be reproduced for the purposes of preservation and electronic publication of the graduation thesis, incl. to be entered in the digital collection of the library of Tallinn University of Technology until expiry of the term of copyright;
    1.2. to be published via the web of Tallinn University of Technology, incl. to be entered in the digital collection of the library of Tallinn University of Technology until expiry of the term of copyright
2. I am aware that the author also retains the rights specified in clause 1 of the nonexclusive licence.
3. I confirm that granting the non-exclusive licence does not infringe other persons' intellectual property rights, the rights arising from the Personal Data Protection Act or rights arising from other legislation.

18.05.2026

[1] The non-exclusive licence is not valid during the validity of access restriction indicated in the student's application for restriction on access to the graduation thesis that has been signed by the school's dean, except in case of the university's right to reproduce the thesis for preservation purposes only. If a graduation thesis is based on the joint creative activity of two or more persons and the co-author(s) has/have not granted, by the set deadline, the student defending his/her graduation thesis consent to reproduce and publish the graduation thesis in compliance with clauses 1.1 and 1.2 of the non-exclusive licence, the non-exclusive licence shall not be valid for the period.

Appendix 2 – Gmsh geometry files

This appendix contains example `.geo` files generated by the developed fiber generation pipeline. The files correspond to specimens containing ten fibers and are included to demonstrate the exported geometric definitions used for mesh generation in Gmsh.

Straight fiber specimen

The following listing shows the generated `.geo` file for a specimen reinforced with ten straight fibers.

```
SetFactory("OpenCASCADE");
Box(1) = {-2,-2,-2,15,15,15};
Cylinder(2) = {3.62252,0.853255,5.92744,0.331738,0.41518,1.9281,0.1,2*
    Pi};
Cylinder(3) = {9.1064,2.35763,4.49306,0.0183187,1.14684,1.63842,0.1,2*
    Pi};
Cylinder(4) =
    {8.72504,2.87382,4.99697,0.473615,-0.678038,1.82098,0.1,2*Pi};
Cylinder(5) =
    {3.78445,8.63829,-0.477821,-0.397739,-0.868596,1.75708,0.1,2*Pi};
Cylinder(6) =
    {3.41499,4.82754,-0.272112,1.28518,0.395002,1.48064,0.1,2*Pi};
Cylinder(7) =
    {9.85276,8.96419,4.6666,0.0202755,0.107838,-1.99699,0.1,2*Pi};
Cylinder(8) = {9.75999,7.0843,10.6558,0.271473,0.829278,-1.79961,0.1,2*
    Pi};
Cylinder(9) = {0.66252,1.4426,6.91891,-0.801124,0.920374,1.58465,0.1,2*
    Pi};
Cylinder(10) =
    {6.90211,9.51059,8.95869,-0.253557,0.726007,1.84625,0.1,2*Pi};
Cylinder(11) =
    {5.33468,5.4891,0.90727,1.38989,-0.792117,-1.20031,0.1,2*Pi};
//
//
BooleanDifference{ Volume{1}; Delete;}{Volume{2,3,4,5,6,7,8,9,10,11};}
//
Physical Volume("concrete") = {1};
Physical Volume("allFibers") = {2,3,4,5,6,7,8,9,10,11};

Physical Surface("top") = {39};
Physical Surface("bottom") = {41};
Physical Surface("left") = {37};
Physical Surface("right") = {42};
Physical Surface("front") = {40};
```

```
Physical Surface("back") = {38};

//
Characteristic Length{ PointsOf{ Volume{2,3,4,5,6,7,8,9,10,11}; } } =
    0.1;
Characteristic Length{ PointsOf{ Surface{39, 41, 37, 42, 40, 38}; } } =
     1;
//
```

Hooked fibre specimen

The following listing shows the generated `.geo` file for a specimen reinforced with ten hooked fibres.

```
SetFactory("OpenCASCADE");
Box(1) = {-2,-2,-2,15,15,15};
Cylinder(2) =
    {3.80654,1.30022,5.79953,-0.184018,-0.446965,0.127906,0.1,2*Pi};
Cylinder(3) = {3.62252,0.853255,5.92744,0.331738,0.41518,1.9281,0.1,2*
    Pi};
Cylinder(4) =
    {3.95426,1.26843,7.85554,0.184018,0.446965,-0.127906,0.1,2*Pi};
Cylinder(5) =
    {9.25878,2.74696,4.21883,-0.152382,-0.389334,0.274224,0.1,2*Pi};
Cylinder(6) = {9.1064,2.35763,4.49306,0.0183187,1.14684,1.63842,0.1,2*
    Pi};
Cylinder(7) =
    {9.12472,3.50447,6.13148,0.152382,0.389334,-0.274224,0.1,2*Pi};
Cylinder(8) =
    {8.50174,2.43869,4.89303,0.223306,0.435122,0.103938,0.1,2*Pi};
Cylinder(9) =
    {8.72504,2.87382,4.99697,0.473615,-0.678038,1.82098,0.1,2*Pi};
Cylinder(10) =
    {9.19866,2.19578,6.81795,-0.223306,-0.435122,-0.103938,0.1,2*Pi};
Cylinder(11) =
    {3.63138,9.07785,-0.295178,0.153065,-0.43956,-0.182644,0.1,2*Pi};
Cylinder(12) =
    {3.78445,8.63829,-0.477821,-0.397739,-0.868596,1.75708,0.1,2*Pi};
Cylinder(13) =
    {3.38671,7.76969,1.27926,-0.153065,0.43956,0.182644,0.1,2*Pi};
Cylinder(14) =
    {3.77805,4.90327,-0.607449,-0.363061,-0.0757356,0.335337,0.1,2*Pi};
Cylinder(15) =
    {3.41499,4.82754,-0.272112,1.28518,0.395002,1.48064,0.1,2*Pi};
Cylinder(16) =
    {4.70017,5.22254,1.20853,0.363061,0.0757356,-0.335337,0.1,2*Pi};
Cylinder(17) =
    {10.3462,8.88364,4.66726,-0.493468,0.0805556,-0.000660186,0.1,2*Pi
    };
Cylinder(18) =
    {9.85276,8.96419,4.6666,0.0202755,0.107838,-1.99699,0.1,2*Pi};
```

```
Cylinder(19) =
   {9.87304,9.07203,2.66962,0.493468,-0.0805556,0.000660186,0.1,2*Pi};
Cylinder(20) =
   {9.26764,7.16263,10.6176,0.492349,-0.0783268,0.0381775,0.1,2*Pi};
Cylinder(21) =
   {9.75999,7.0843,10.6558,0.271473,0.829278,-1.79961,0.1,2*Pi};
Cylinder(22) =
   {10.0315,7.91358,8.85619,-0.492349,0.0783268,-0.0381775,0.1,2*Pi};
Cylinder(23) =
   {0.557585,1.84043,6.6348,0.104936,-0.39783,0.284113,0.1,2*Pi};
Cylinder(24) =
   {0.66252,1.4426,6.91891,-0.801124,0.920374,1.58465,0.1,2*Pi};
Cylinder(25) =
   {-0.138604,2.36297,8.50356,-0.104936,0.39783,-0.284113,0.1,2*Pi};
Cylinder(26) =
   {6.65424,9.90204,8.77072,0.247868,-0.391444,0.187971,0.1,2*Pi};
Cylinder(27) =
   {6.90211,9.51059,8.95869,-0.253557,0.726007,1.84625,0.1,2*Pi};
Cylinder(28) =
   {6.64855,10.2366,10.8049,-0.247868,0.391444,-0.187971,0.1,2*Pi};
Cylinder(29) =
   {5.03275,5.55496,0.5142,0.301925,-0.0658556,0.39307,0.1,2*Pi};
Cylinder(30) =
   {5.33468,5.4891,0.90727,1.38989,-0.792117,-1.20031,0.1,2*Pi};
Cylinder(31) =
   {6.72457,4.69698,-0.293044,-0.301925,0.0658556,-0.39307,0.1,2*Pi};
//
BooleanUnion{Volume{2}; Delete;}{Volume{3,4}; Delete;}
BooleanUnion{Volume{5}; Delete;}{Volume{6,7}; Delete;}
BooleanUnion{Volume{8}; Delete;}{Volume{9,10}; Delete;}
BooleanUnion{Volume{11}; Delete;}{Volume{12,13}; Delete;}
BooleanUnion{Volume{14}; Delete;}{Volume{15,16}; Delete;}
BooleanUnion{Volume{17}; Delete;}{Volume{18,19}; Delete;}
BooleanUnion{Volume{20}; Delete;}{Volume{21,22}; Delete;}
BooleanUnion{Volume{23}; Delete;}{Volume{24,25}; Delete;}
BooleanUnion{Volume{26}; Delete;}{Volume{27,28}; Delete;}
BooleanUnion{Volume{29}; Delete;}{Volume{30,31}; Delete;}
//
BooleanDifference{ Volume{1}; Delete;}{Volume
   {32,33,34,35,36,37,38,39,40,41};}
//
Physical Volume("concrete") = {1};
Physical Volume("allFibers") = {32,33,34,35,36,37,38,39,40,41};

Physical Surface("top") = {189};
Physical Surface("bottom") = {191};
Physical Surface("left") = {187};
Physical Surface("right") = {192};
Physical Surface("front") = {190};
Physical Surface("back") = {188};
```

```
//
Characteristic Length{ PointsOf{ Volume{32,33,34,35,36,37,38,39,40,41};
    } } = 0.1;
Characteristic Length{ PointsOf{ Surface{189, 191, 187, 192, 190, 188};
    } } = 1;
//
```

Appendix 3 – ElmerFEM solver input file

The following listing shows the ElmerFEM simulation input file used for the linear elastic finite element simulations presented in this thesis.

```
Header
  CHECK KEYWORDS Warn
  Mesh DB "." "."
  Include Path ""
  Results Directory ""
End

Simulation
  Max Output Level = 5
  Coordinate System = Cartesian
  Coordinate Mapping(3) = 1 2 3
  Simulation Type = Steady state
  Steady State Max Iterations = 1
  Output Intervals = 1
  Timestepping Method = BDF
  BDF Order = 1
  Solver Input File = case.sif
  Post File = case.vtu
End

Constants
  Gravity(4) = 0 -1 0 9.82
  Stefan Boltzmann = 5.67e-08
  Permittivity of Vacuum = 8.8542e-12
  Boltzmann Constant = 1.3807e-23
  Unit Charge = 1.602e-19
End

Body 1
  Target Bodies(1) = 1
  Name = "Body 1"
  Equation = 1
  Material = 1
End

Body 2
  Target Bodies(1) = 2
  Name = "Body 2"
  Equation = 1
  Material = 2
```

```
End

Solver 1
  Equation = Linear elasticity
  Variable = -dofs 3 Displacement
  Procedure = "StressSolve" "StressSolver"
  Exec Solver = Always
  Stabilize = True
  Bubbles = False
  Lumped Mass Matrix = False
  Optimize Bandwidth = True
  Steady State Convergence Tolerance = 1.0e-5
  Nonlinear System Convergence Tolerance = 1.0e-7
  Nonlinear System Max Iterations = 20
  Nonlinear System Newton After Iterations = 3
  Nonlinear System Newton After Tolerance = 1.0e-3
  Nonlinear System Relaxation Factor = 1
  Linear System Solver = Iterative
  Linear System Iterative Method = BiCGStab
  Linear System Max Iterations = 500
  Linear System Convergence Tolerance = 1.0e-10
  BiCGstabl polynomial degree = 2
  Linear System Preconditioning = ILU0
  Linear System ILUT Tolerance = 1.0e-3
  Linear System Abort Not Converged = False
  Linear System Residual Output = 10
  Linear System Precondition Recompute = 1
End

Equation 1
  Name = "Equation 1"
  Calculate Stresses = True
  Active Solvers(1) = 1
End

Material 1
  Name = "Material 1"
  Poisson ratio = 0.3
  Youngs modulus = 2.7e9
End

Material 2
  Name = "Material 2"
  Poisson ratio = 0.4
  Youngs modulus = 200e9
End

Boundary Condition 1
  Target Boundaries(1) = 5
  Name = "BoundaryCondition 1"
  Displacement 1 = 0
```

```
End

Boundary Condition 2
  Target Boundaries(1) = 6
  Name = "BoundaryCondition 2"
  Force 1 = 14500
End
```

Appendix 4 – C++ source code for fiber geometry generation

This appendix provides a link to the GitHub repository containing the source code for the developed fiber generation framework.

https://github.com/ArturRaag/FRC-fiber-geometry-generator/blob/main/main.cpp