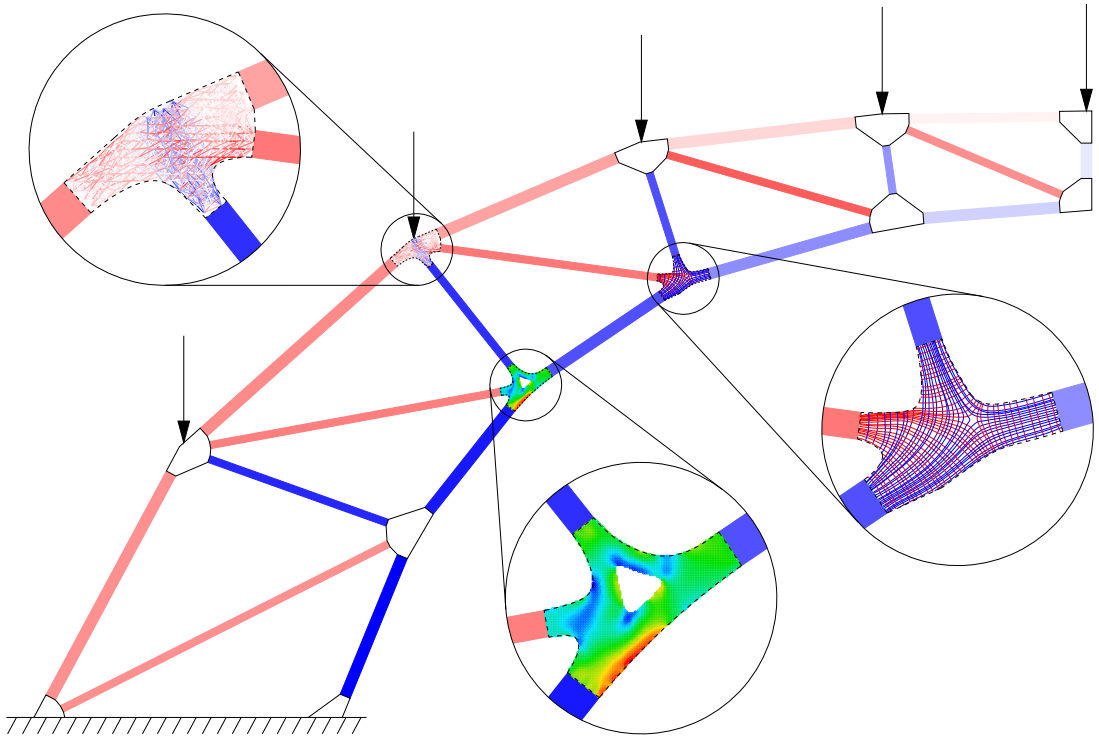
CHALMERS
UNIVERSITY OF TECHNOLOGY



Interactive Design Using Peridynamics

Development of a Design Tool for the Exploration of Structurally Efficient Geometries

Master's Thesis in Structural Engineering and Building Technology

GABRIEL EDEFORS
DANIEL JONSSON

Department of Architecture and Civil Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
Master's thesis ACEX30
Gothenburg, Sweden 2021

Master's thesis ACEX30

Interactive Design Using Peridynamics

Development of a Design Tool for the Exploration of Structurally Efficient Geometries

GABRIEL EDEFORS
DANIEL JONSSON

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Architecture and Civil Engineering
Research group for Architecture and Engineering
Chalmers University of Technology
Gothenburg, Sweden 2021

Interactive Design Using Peridynamics
Development of a Design Tool for the Exploration of
Structurally Efficient Geometries

GABRIEL EDEFORS
DANIEL JONSSON

Supervisor: Lic Eng Jens Olsson, Department of Architecture and
Civil Engineering
Examiner: Senior Lecturer Mats Ander, Department of Architecture and
Civil Engineering, Department of Industrial and Materials Science

Department of Architecture and Civil Engineering
Research group for Architecture and Engineering
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Cover: Curved truss in which three connections have been shaped using the design tool, see Figure 5.17 for more details.

Department of Architecture and Civil Engineering
Gothenburg, Sweden 2021

Interactive Design Using Peridynamics
Development of a Design Tool for the Exploration of Structurally Efficient Geometries
EDEFORS, JONSSON
Department of Architecture and Civil Engineering
Chalmers University of Technology

Abstract

With an ever-increasing demand on material efficiency, the use of optimization tools in the design process is becoming a more common practice. Rather than optimizing structures based on certain criteria, the authors propose an alternative approach where the designer interactively can change the geometry and get instant visual feedback on how this affects the stress field. The particle method peridynamics offers a flexible mathematical formulation that allows for shape alterations during runtime, making it a suitable theory to use for explorative design.

The purpose of this thesis is to develop an interactive computational design tool that can support the designer in finding structurally efficient and aesthetically appealing geometries in the early design stages. The tool is to be flexible to allow for a wide variety of solid mechanics problems, with the limitations of isotropic materials in two dimensions. Furthermore, the designer is to have an active role and interact in a design feedback loop.

The implementation is written in C# and exposed as a plugin to Grasshopper®, which runs within the CAD software Rhinoceros®. The theory used is a modified version of peridynamics that allows for having varying particle sizes. This makes it possible to have more densely packed particles at stress concentrations and boundaries. Furthermore, the underlying theory is tailored to quickly find the static solution after changes have been made to the geometry. For this purpose, a stable time stepping scheme has been derived along with useful stress measures.

The plugin has yielded results that agree well with benchmark solutions and the derived time step is conservative, yet close to the critical value. By using small particles on the boundaries and larger particles in the interior, maintained accuracy can be achieved for a significantly reduced computational cost. Furthermore, the plugin has been integrated in a parametric truss model to shape the structural connections. In this setting the tool showed high interactivity as it responded instantaneously to changes of shape and topology.

In conclusion, the plugin effectively augments the skills of the designer and opens up for an alternative, more intuitive, design approach.

Keywords: Particle Methods, Peridynamics, Smoothed Particle Hydrodynamics, Finite Element Method, Computational Design, Interactive Design, Grasshopper3d

Acknowledgements

We would like to express our deep gratitude to Lic. Jens Olsson for his guidance and enthusiastic support. Also, a special thanks for including us in your research and for sharing your findings. We also wish to acknowledge the excellent support from senior lecturer Mats Ander in his role as examiner. The help he gave us when writing the report and developing the theory is far more than we would have hoped for. Our thanks are also extended to Professor Chris Williams for helping us when we got stuck in the theory.

Finally, we would like to thank MSc. Erik Forsberg for all time and effort invested in this work, and for being an outstanding opponent.

Daniel Jonsson and Gabriel Edefors, Gothenburg, June 2021

Contents

List of Figures

List of Tables

Nomenclature

Symbols

$\boldsymbol{\omega}$	Angular velocity vector $[s^{-1}]$
T_{ij}	Arm tension between particle i and j $[N]$
$\boldsymbol{f}_i^{int}$	Internal force associated to particle i $[N]$
$\boldsymbol{b}_i$	Body force density of particle i $[N/m^2$ in 2D, N/m^3 in 3D$]$
$\boldsymbol{x}_i$	Position vector of particle i in deformed configuration $[m]$
δ_{ij}	Kronecker delta
$\mathcal{H}_i$	Family of particle i, i.e. the circular region around $\boldsymbol{X}_i$, defined by the horizon δ [-]
$\boldsymbol{t}_{ij}$	Force density vector acting on particle i, caused by particle j $[N/m^4$ in 2D, N/m^6 in 3D$]$
Γ	Boundary of the domain Ω
Δ	Grid size of the discretization $[m]$
δ	Horizon $[m]$
$\boldsymbol{X}_i$	Position vector of particle i in undeformed configuration $[m]$
T	Kinetic energy $[J]$
c	Centre of mass
$\otimes$	Dyadic product
U	Potential energy $[J]$
$\boldsymbol{\eta}_{ij}$	Relative displacement of particle i and j $[m]$
$\boldsymbol{\xi}_{ij}$	Relative position of particle i and j in undeformed configuration $[m]$
$\boldsymbol{q}_{ij}$	Relative position vector between particle i and j $[m]$
ρ_i	Density of particle i $[kg/m^2$ in 2D, kg/m^3 in 3D$]$
r_{ij}	Relative distance between particle i and j $[m]$
V_i	Volume of particle i $[m^3$ in 3D and m^2 in 2D$]$

Operators

$\bullet\vert_x$	Evaluated at x
∇_i	Gradient operator with respect to the undeformed position of particle i $[m]$
$\vert\bullet\vert$	Euclidean norm

Subscripts and Superscripts

$\bar{\bullet}$	Arithmetic mean
$\bullet'$	Dummy variable of integration
$\bullet_{ij}$	Relative vector between particle i and particle j
$\bullet^T$	Transpose
$\dot{\bullet}$	Time derivative
$\tilde{\bullet}$	Voigt representation

1
Introduction

This chapter presents the structural design process and identifies the need for an interactive design tool. It also introduces particle methods in general and outlines how peridynamics in particular can be used in a design tool. Finally, the aim of the thesis is specified together with the research questions and limitations.

1.1 Background

Structural and architectural design are often seen as separate processes with conflicting interests. This tends to result in design compromises and a segregation between the different disciplines. This does not need to be the case however. Geometry plays a central role in all design and could act as the link between aesthetics and structural performance. By focusing on the connection between form and force in an early design stage, force flows can inform the geometry and vice versa, and thereby efficient *and* aesthetically appealing designs can be achieved.

1.1.1 Context

At the time of writing, the building industry stands for a significant part of the total environmental impact. In 2010, roughly 20% of the total green house gases emissions could be attributed to the sector [12]. Materials such as steel and concrete require a lot of energy to manufacture, they are nevertheless strong building materials that are easy to produce. If used in a proper and economical way - the authors believe that they still can be a part of the solution in building efficient, long-standing, structures. For any material, it is however crucial to make the best use of its specific structural properties in the applications. Moreover, the need to decrease overall material usage is obvious. Therefore, the strive for structural efficiency should be an inherent part of any building design process, from the moment pen is put to paper.

Figure 1.1: The Olympic Stadium in Munich by Frei Otto (1972) (Photo by Jorgeroyan , licensed under CC BY-SA 3.0)

One way of saving material is to design lightweight, tensile structures. The structural behaviour is then often closely connected to the shape, and any added self-weight needs to be accounted for. One typical example of this is the design of structural nodes in cable structures, which often need to be lightweight in order not to lower the overall performance of the structure. In addition to being structurally efficient, lightweight structures often need to be aesthetically pleasing since they often are located in public areas. Figure 1.1 shows a close-up of some of the cable connections of the Munich Olympic Stadium. Designed by Frei Otto more than half a century ago, this extraordinary structure broke new ground for the field of tensile structures.

Figure 1.2: Palazzetto dello Sport in Rome by Pier Luigi Nervi (1958) [Photo]

Also for heavier structures, taking account to the force flows in the design process can yield great material savings - and result in beautiful patterns. Figure 1.2 shows the Palazzetto dello Sport in Rome, with its thin concrete dome designed by Pier Luigi Nervi in the 50s. Nervi often used the force patterns to guide his designs, resulting in both appealing and efficient structures.

Figure 1.3: Extended Waal Bridge in Nijmegen by Zwarts & Jansma Architecten and Witteveen+Bos as structural engineers (2015) [Photo by Jeroen van Lieshout , licensed under CC BY-NC-ND 2.0]

Bridges are another type of structure where the geometry is essential for achieving structural efficiency. Take for example the Waal extension bridge in Nijmegen (Figure 1.3), where the vault shape efficiently guides the compressive forces to the supports. The shape of the bridge is however not only based on the forces involved. The architects wanted the bridge to be perceived as being shaped by the force of streaming water [27]. By actively working with the geometry the architects and engineers managed to elegantly combine these two requirements.

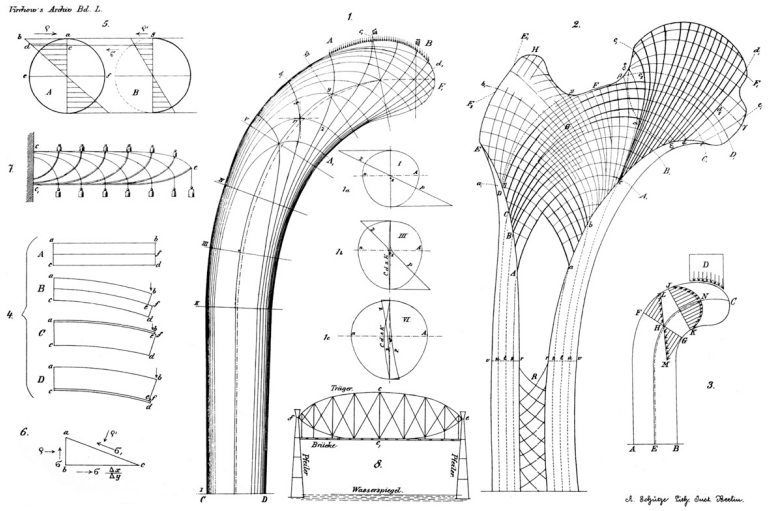


Figure 1.4: Composite drawing of Culmann's crane, a thighbone and more (Reprinted by permission from [Springer Nature Customer Service Centre GmbH]: [Nature Springer] [Archiv für pathologische Anatomie und Physiologie und für klinische Medicin] [Ueber die innere Architectur der Knochen und ihre Bedeutung für die Frage vom Knochenwachsthum, Dr. Julius Wolff], [Copyright Clearance Center])

The interplay between shape and force is not exclusive to built structures, but rather a law of nature. The eroded shape of a mountain, the tilt of a windswept tree and the

shape of a human skeleton are all results of external and internal forces constantly acting on the bodies. Through evolution, the skeleton has been shaped to withstand the loads caused by the self weight of the moving body. Wolff's law states that even during the short time span of a lifetime, your bones will adapt their shapes to better resist the loads, and resulting stresses, which they are exposed to. In Figure 1.4 a composition of drawings show how the form is tightly coupled to the force flow, note in particular how the principal stress elegantly follows the curvature of the bone.

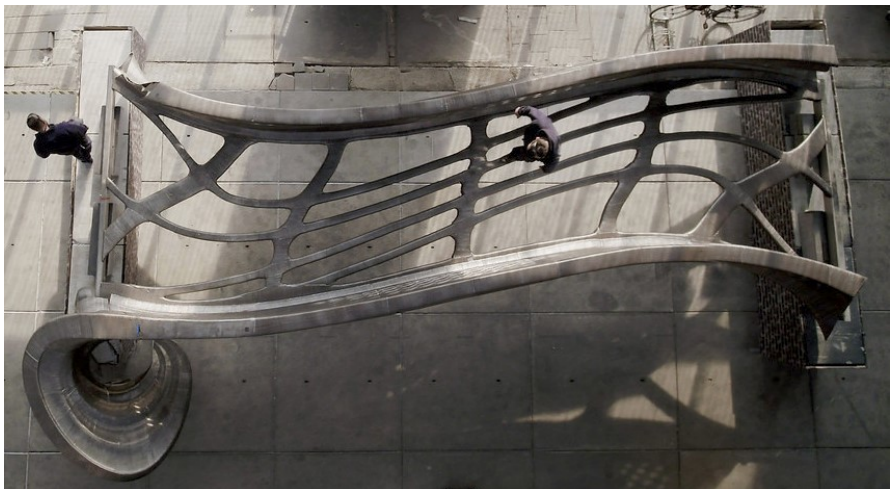


Figure 1.5: 3D printed stainless steel bridge in Amsterdam by MX3D and Arup (2018) [Photo by Arts Electronica , licensed under CC BY-NC-ND 2.0]

The emerging field of 3D printing of steel and other materials opens up the possibilities of mass customization, since complex components can be manufactured using a layer-by-layer technique[11]. Structural components can be tailored for specific requirements, which often requires a higher level of geometric complexity. An example of such a structure is the MX3D bridge in Amsterdam. The manufacturing of the complex organic shape of the bridge is made possible by using 3D printing. To fully capture the possibilities of 3D printing, computational design tools are often required. Optimization algorithms are powerful, but are often based on a limited set of criteria, and soft values such as aesthetics are difficult to integrate in the algorithms since they are hard to quantify. Therefore, the authors believe there is a need for more flexible and holistic tools that can give real-time feedback on how the shape and forces are associated. Thereby the designer can make informed decisions based on structural performance, material efficiency and aesthetical considerations.

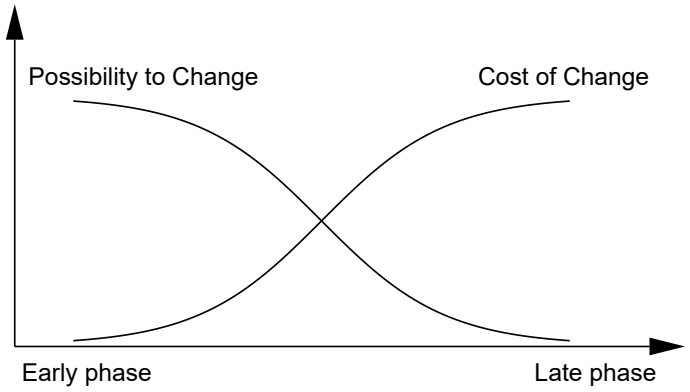


Figure 1.6: Principal graph showing how the possibility to make changes decrease and the cost increase as the design work progresses.

1.1.2 Particle Methods

In the implementation of numerical methods for the analysis of solids, there is a need to discretize the body. One way of doing so is to model the body as a *mesh*, where each vertex is connected to its neighbouring vertices by edges. The Finite Element Method (FEM) is a commonly used mesh-based method. Another way to discretize the domain is to use a finite number of *particles*. 'Particle methods' is an umbrella term for such methods, which can also be referred to as meshfree. Smoothed Particle Hydrodynamics (SPH) and Peridynamics (PD) are two examples of such methods. The two discretization methods are conceptually shown in Figure 1.7.

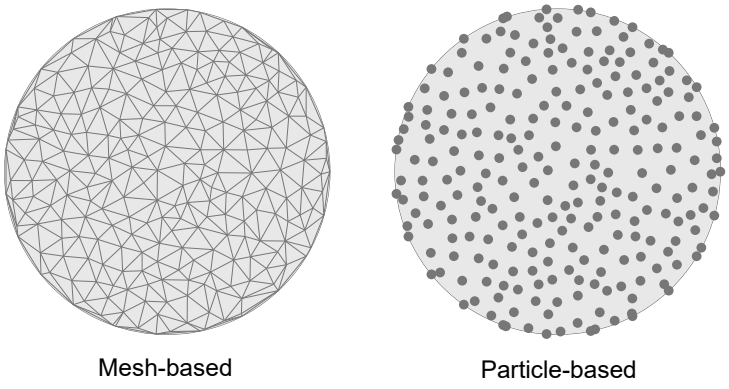


Figure 1.7: Two discretization methods applied to a circular domain

The classical continuum mechanics theory is based on an assumption of locality, as the stress state at a point is governed solely by the points in its immediate vicinity [23]. Methods based on this principle are referred to as *local* methods - continuum mechanics is an example. For *non-local* methods, on the other hand, the stress state at a point is governed by all its neighbouring points within a certain *horizon* δ. Particle methods, such as SPH and PD, belong to this category.

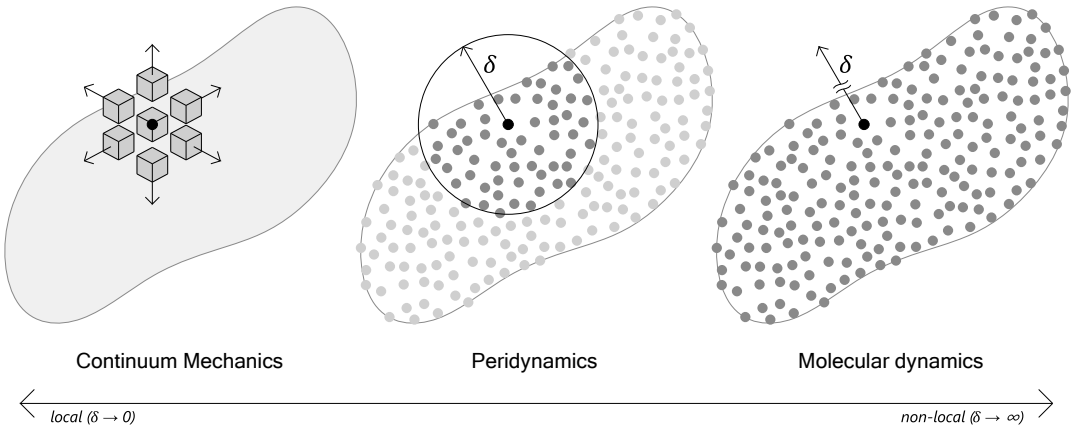


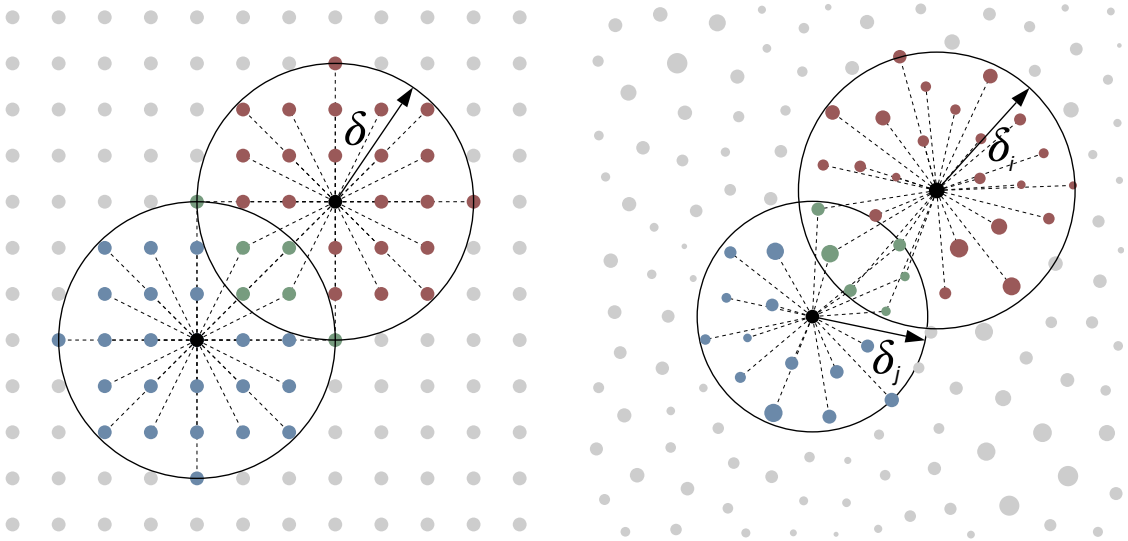
Figure 1.8: Local and non-local methods (Figure inspired by graphic in [17])

In addition, continuum mechanics is based on the assumption that the fields modelled, such as displacement and stress, vary continuously in the whole domain. In contrast, particle methods in general need no such assumption, which has made them suitable for modelling discontinuities such as cracks [17]. Sprung from earlier particle methods such as molecular dynamics simulation (MDS), the theory of Peridynamics was introduced by Stewart Silling in the beginning of the 21st century. By coarse-graining the molecular dynamics model into macro particles, non-local effects can be modelled, with less computational resources than MDS [17]. Instead of using a continuous formulation including partial derivatives, the peridynamics theory is utilizing spatial *integral* equations, which remain valid also at discontinuities.

1.1.3 Peridynamics in the Design Process

A research project *Smart Built Environment: Digitalization and industrialization for a sustainable built environment*, aiming at developing methods for exploration and design of nodal connections, has been conducted at Chalmers in cooperation with industrial partners. In [20], Olsson et al. investigate the application of peridynamics in different design stages. They conclude that the absence of continuity requirements in the formulation makes the theory flexible for domain changes related to the early design phase. They also emphasise the inherent property of peridynamics to capture stress concentrations and large deformations. This makes it possible to more easily explore how the geometry affects more complicated phenomena already in the early design phase. Another advantage of using a particle method instead of a mesh-based method is that it become easier to integrate with CAD environments. In addition it is easier to refine the discretization by simply adding particles, compared to having to refine a mesh [15].

1.1.4 Modification of Peridynamics Using Virtual Fibres



(a) Uniform particle distribution **(b)** Non-uniform particle distribution

Figure 1.9: Modified peridynamics allows for having varying particle sizes.

In [21] a new approach for establishing the arm tensions in-between particles in peridynamics is proposed. By introducing a virtual fibre mat, the influence on the arm tension from the orientation, relative the stress state, can be separated from the influence of the particle size and arm length. This decoupling enables the use of different particle sizes as opposed to in classical peridynamics, which is derived under the assumption of regular particle spacing. The influence of the arm length and particle size is governed by a Kernel function that reduces the arm tension for very short and very long arms. By including this formulation of arm tension in the peridynamic equation of motion the modelling of boundary regions possibly can be made more accurate.

1.2 Techniques for Finding Efficient Geometries

There are several approaches to finding efficient shapes and topologies. A common one being manual iteration in combination with an intuitive understanding of force flows. Other common approaches include applying mathematical optimization techniques and parameter studies.

1.2.1 Structural Optimization

There are three main types of structural optimization [22]. The most straight forward is optimizing the size of the component such that the material is maximally utilized. Another approach is to keep the outer boundary of the design domain constant and remove interior parts of the body thereby changing the *topology*[1]. The task is typically then to remove parts of the body such that the stiffness is maximized [22]. The last type of optimization technique is to change the *shape* of the body, this includes removing regions of the body from the outside or changing already introduced voids.

There are different techniques for achieving the optimal solution. There is the mathematical approach where the objective function is minimized under a set of constraints. This could be realized by adopting for instance the gradient descent method. There are also techniques that are called *evolution strategies* that to some extent mimics natural selection in the search for an optimum solution. This thesis will explore another technique where the designer is controlling the optimization by manually making changes to the body. The designer can by introducing voids in the body alter the topology. Or the designer can alter the shape by changing the outer boundary of the body. More importantly the optimization can be done more freely without having to bother about which method and constraints to use.

1.2.2 Generative Design

In contrast to the optimization algorithms described above, a generative design algorithm produces multiple outputs for a given set of inputs and constraints. Rather than generating one optimized output the designer is provided with multiple design alternatives. In an iterative process, the user can then fine tune the parameters in a feedback loop until the results are satisfactory, and then filter out the optimal (not necessarily optimized) outcome.

By using cloud computing, numerous design options can be generated in no time, and to further increase the power of generative design, artificial intelligence and machine learning can be utilized in the fine-tuning and selection processes (see [2]). For our design purposes, however, we want the designer, rather than a neural network, to be active in the fine tuning process and to interact with the software.

[1]Topology in this context refers to geometric topology, where two spaces are topologically equivalent if one space can be transformed into the other by applying a continuous deformation [3]

A potential set of parameters and boundary conditions to evaluate could be:

- Geometric constraints
- Load conditions
- Production method
- Aesthetical preferences
- Material usage

1.2.3 Existing Interactive Optimization Tools

There exist numerous software packages for both shape and topology optimization of structures, such as OptiStruct® [1] and Tosca® [4]. Also, most commercial FE-software have advanced capabilities for structural optimization. Most such tools require that the user identifies numerical parameters that can be fed to an algorithm. In an effort to make the optimization process more intuitive and more suitable for early design exploration, an app called TopOpt 3D® [6] was developed, see [18]. Here the user can freely alter the design parameters and get instant feedback on how that alters the optimized geometry. There also exist optimization tools in Grasshopper3d ® [24] where parameters such as loading and support conditions easily can be altered, e.g. the plugin AMEBA® [26]. However, all these tools lack the possibility for the user to sculpture the domain during runtime and get instant feedback on how this affects the stress field.

1.2.4 Proposed Design Approach

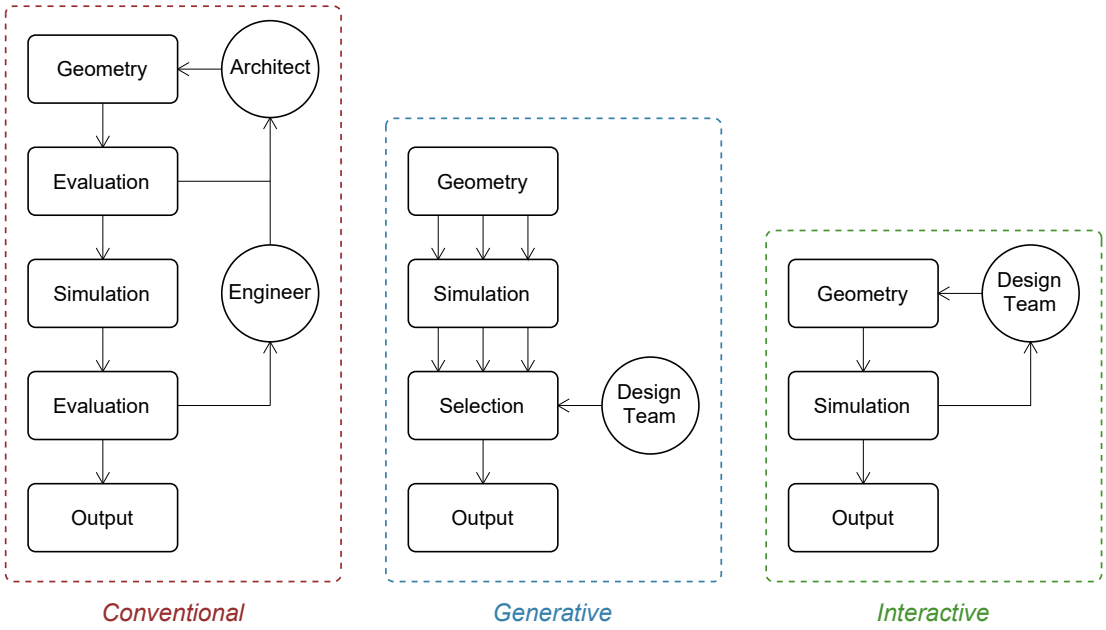


Figure 1.10: Flowcharts illustrating different design processes and how and when the key actors are involved in the design

Our design approach can be seen as a combination of classical shape and topology optimization and generative design. In contrast to most generative design procedures letting a machine learning algorithm handle the feedback-loop, here the designer will have a more active role. The user interacts with the program during the simulation, and is thereby able to customize and steer the design during run-time, rather than letting the algorithm generate multiple alternatives and then filter out one. The interactivity is the key feature in play here.

1.3 Aim

The objective of this thesis is to create an interactive design tool based on the modified theory of peridynamics. By exposing the mechanical simulation to the user, the design can be guided towards structural efficiency. The tool should be interactive and allow the designer to customize and alter the shape and topology in a feedback loop. The tool should be fast enough to give near to real-time feedback on the structural performance, thus enabling a user defined shape and topology optimization for designing material efficient, as well as aesthetically pleasing, structural components. Rather than generating a fully optimised geometry, the tool should guide the designer to combine efficiency and aesthetics. The tool is intended for early design stages, but should be flexible enough to easily be extended to simulate crack propagation and yielding. Since the trend in architectural and structural design is to integrate the whole design procedure in just a few, dynamically linked, programs, the potential use of the tool relies heavily on how it can be integrated in the design flow of larger structures. Instead of aiming on producing a stand-alone program the tool is to be integrated in the Rhino-Grasshopper environment, thus enabling automatic integration within an established framework for design.

1.4 Research Questions

Since the full potential of peridynamics in a design context is not known beforehand, but rather acquired by studying the underlying theory and by investigating design requirements, the following questions will help to guide the development of the tool.

- How should the dynamic equations of motion of peridynamics be implemented to quickly reach a steady state solution?
- How should the model parameters be tuned to minimize the computational time. In particular, how can the accuracy be increased by the means of the modified formulation of particle interaction presented in [21]?
- How do the user interact with the simulation, i.e. how is the result displayed to the designer and how can the domain be altered based on that?
- How to include shape and topology changes in the peridynamic model?
- How to integrate the tool with other design and analysis methods?

1.5 Limitations

The development of peridynamics sprung from the issue of handling discontinuities, especially concerning crack propagation [17]. Our focus is however shifted towards the early design phase, where the performance under static loading is of the greatest concern, hence no fracture modelling is considered. Also, a simple isotropic linear elastic material model is deemed sufficient for the same reason.

Since peridynamics in general is computationally more expensive than FEM, the tool is ideally integrated with structural elements modelled with FEM. Due to time limitations this is left out of scope of this project and is left as a natural continuation of the project. The tool can nevertheless be connected to models where the stiffness of the modelled part is not affecting the rest of the structure, for instance in a truss model. Even though the manual shaping of geometries is emphasized in this thesis, the tool could potentially benefit from also implementing convectional optimization techniques. This is however left for further work since the core functionality is prioritized. Finally, the tool is restricted to two dimensional geometries. Most of the work presented in this report can nevertheless be extended to 3D.

A summary of the limitations used in this thesis:

- No fracture analysis - Since the tool is aimed at the early design phase the static response is prioritized. Fracture modelling and progressive failure could relatively easily be implemented as described in [21]
- Linear elasticity - The tool is aimed at evaluating the performance under service conditions where plasticity is not occurring on a global scale.
- Isotropic materials - This is the most straight forward modelling approach and applies to common building materials such as steel.
- No implementation of conventional topology optimization in the tool - Rather the user should interactively explore efficient topologies.
- Tool restricted to 2D geometries - Even though the underlying theory and its implementation can be used for 3D, solving the user interaction becomes significantly harder.

2
Theory

This chapter starts by describing how the equations of motion for a system of particles can be derived from Hamilton's principle. From this the equations of motion used in peridynamics is presented together with some techniques for solving the equations. Finally a modified formulation of peridynamics is described. This will later be used to model particle systems with varying particle sizes.

2.1 Modelling the Evolution of a Particle System

Below follows a short introduction to how the equations of motion for a particle system can be derived. By first considering the motion of the particles and then applying Hamilton's principle of stationary action, a model for predicting the evolution of the system can be derived.

2.1.1 Lagrangian Description of Motion

Consider a reference configuration occupying the domain Ω, containing an arbitrary particle with position $\boldsymbol{X}_i$. Due to external stimuli the point will experience the displacement $\boldsymbol{u}_i$, changing the position to $\boldsymbol{x}_i$, the displacement vector is thus $\boldsymbol{u}_i(\boldsymbol{X}, t) = \boldsymbol{x}_i(t) - \boldsymbol{X}_i$. This approach of describing the motion of particles relative a fixed frame of reference conforms to a Lagrangian description and is commonly used in SPH and peridynamics [8]. In peridynamics the motion of a particle $\boldsymbol{X}_i$ is governed by the interaction with all other particles within a radius δ_i, referred to as the *horizon*. The zone within δ_i of $\boldsymbol{X}_i$ is called the *family* of $\boldsymbol{X}_i$, and is denoted $\mathcal{H}_{x_i}$ [17]. Furthermore, particle $\boldsymbol{X}_i$ has the volume V_i and the mass density ρ_i.

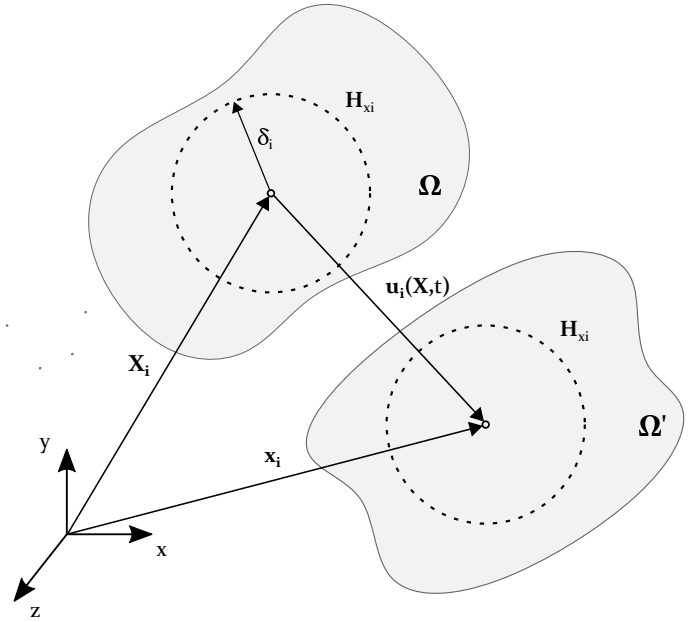


Figure 2.1: Deformation of a particle and associated quantities.

2.1.2 Strain Energy Density

The total strain energy density W_i of particle $\boldsymbol{X}_i$ arise due to interaction with all other particles $\boldsymbol{X}_j$ within its horizon. This can be written as a summation of micropotentials w_{ij} and w_{ji} according to

$$W_i = \frac{1}{2}\sum_{j=1}^{\infty}\frac{1}{2}\Big(w_{ij}\Big(\boldsymbol{x}_{(1^i)} - \boldsymbol{x}_{(i)}, \boldsymbol{x}_{(2^i)} - \boldsymbol{x}_{(i)} \dots \boldsymbol{x}_{(\infty^i)} - \boldsymbol{x}_{(i)}\Big) + w_{ji}\Big(\boldsymbol{x}_{(1^j)} - \boldsymbol{x}_{(j)}, \boldsymbol{x}_{(2^j)} - \boldsymbol{x}_{(j)} \dots \boldsymbol{x}_{(\infty^j)} - \boldsymbol{x}_{(j)}\Big)\Big)V_j, \tag{2.1}$$

following [17]. The micropotential scalars are family and material dependent. Thus, $w_{ij} \neq w_{ji}$, as w_{ij} depends on the material points belonging to $\mathcal{H}_{x_i}$, and w_{ji} to $\mathcal{H}_{x_j}$.

2.1.3 Hamilton's Principle - A Variational Description of Motion

Hamilton's principle states that

$$\delta\int_{t_1}^{t_2}(T-U)dt = 0. \tag{2.2}$$

where T and U are the kinetic and potential energy of the system, respectively. Note that δ in this equation does not refer to a particle horizon, but denotes a small change of the functional. The trajectory that satisfies Equation 2.2 is the one that makes the integral of the Lagrangian function $L = T - U$ stationary - hence the method is known as *the stationary action principle* [14]. The solution to this stationary condition can be found for particle i by solving the Euler-Lagrange equation

$$\frac{d}{dt}\Big(\frac{\partial L}{\partial \dot{\mathbf{u}}_i}\Big) - \frac{\partial L}{\partial \mathbf{u}_i} = 0. \tag{2.3}$$

The potential and kinetic energy of the system can be written as a summation over all particles according to

$$T = \sum_{k=1}^{\infty} \frac{1}{2} \rho_k \dot{\boldsymbol{u}}_k \cdot \dot{\boldsymbol{u}}_k V_k \tag{2.4}$$

and

$$U = \sum_{k=1}^{\infty} W_k V_k \; - \; \sum_{k=1}^{\infty} \boldsymbol{b}_k \cdot \boldsymbol{u}_k V_k \tag{2.5}$$

respectively. Here $\boldsymbol{b}_k$ is the body force density, V_k is the volume, W_k is the strain energy density and $\dot{\boldsymbol{u}}_k$ denotes the velocity of particle k (the first time derivative of the displacement) [17].

2.2 Classical Peridynamics

Inserting the Expression 2.4 and 2.5 into the Euler-Lagrange Equation 2.3 gives the discrete peridynamics equation of motion

$$\rho_i \ddot{\boldsymbol{u}}_i = \sum_{j=1}^{\infty} \Big(\boldsymbol{t}_{ij}\Big(\boldsymbol{u}_j - \boldsymbol{u}_i, \boldsymbol{X}_j - \boldsymbol{X}_i, t\Big) - \boldsymbol{t}_{ji}\Big(\boldsymbol{u}_i - \boldsymbol{u}_j, \boldsymbol{X}_j - \boldsymbol{X}_i, t\Big)\Big) V_j + \boldsymbol{b}_i, \tag{2.6}$$

where $\ddot{\boldsymbol{u}}_i$ is the acceleration (the second time derivative of the displacement) and $\boldsymbol{t}$ the force density, derived from the micro-potential w_{ik} by differentiation with respect to the relative position in the deformed state according to

$$\boldsymbol{t}_{ij} = \frac{1}{2V_j}\Big(\sum_{k=1}^{\infty} \frac{\partial w_{ik}}{\partial(\boldsymbol{x}_j - \boldsymbol{x}_i)} V_k \Big). \tag{2.7}$$

2.2.1 Continuous Peridynamic Equation of Motion

Since the volume V_j is infinitesimal, the summation can be replaced with integration over the family of each particle

$$\rho(\boldsymbol{X})\ddot{\mathbf{u}}(\boldsymbol{X}, t) = \int_{\mathcal{H}_x} \boldsymbol{t}\Big(\boldsymbol{u}' - \boldsymbol{u}, \boldsymbol{X}' - \boldsymbol{X}, t\Big) - \boldsymbol{t}'\Big(\boldsymbol{u} - \boldsymbol{u}', \boldsymbol{X} - \boldsymbol{X}', t\Big) dV_{\boldsymbol{x}'} + \boldsymbol{b}(\boldsymbol{X}, t), \tag{2.8}$$

where $\boldsymbol{t}$ is the force density (force per area squared in 2D) exerted on $\boldsymbol{x}$ by $\boldsymbol{x}'$ ($\boldsymbol{x}'$ is a dummy variable of integration) [25]. Equation 2.8 is the fundamental equation used in peridynamics and describes the time evolution of the system. A more concise formulation is given by

$$\rho \ddot{\boldsymbol{u}} = \int_{\mathcal{H}_x} \boldsymbol{t} - \boldsymbol{t}' \; dV_{\boldsymbol{x}'} + \boldsymbol{b}. \tag{2.9}$$

2.2.2 Different Formulations of the Bond Force

It is convenient to distinguish between three different types of peridynamics, which have different constraints on how the force density vectors between the particles are allowed to vary.

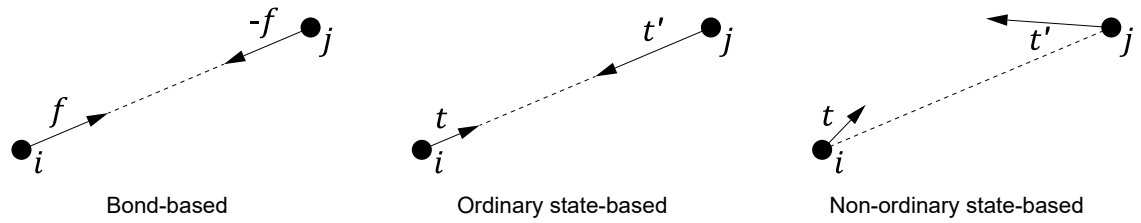


Figure 2.2: Three different formulations of the force density.

Bond-based peridynamics can be seen as a special case where the force density vectors of two interacting particles have the same magnitude and are opposite (parallel) to each other. This formulation is limiting the choice of modelling material to those with a constant Poisson's ratio $\nu = 1 / 3$ in two dimensions and $\nu = 1 / 4$ in three dimensions [13]. *Ordinary state-based* peridynamics is a more general approach, where the two magnitudes are allowed to differ, while the vectors still remain opposite. The most general formulation is called *Non-ordinary state-based* peridynamics, for which both the direction and magnitude of the vectors may differ [17].

2.2.3 Initial Conditions

Initial conditions need to be prescribed to solve Equation 2.8. For a quasi static analysis all particles in the domain Ω are assigned zero displacement and velocity at $t = 0$, i.e.

$$\left.\begin{aligned}\mathbf{u}(\boldsymbol{X}, t=0) &= \mathbf{0}\\ \dot{\mathbf{u}}(\boldsymbol{X}, t=0) &= \mathbf{0}\end{aligned}\right\} \quad for \;\; \mathbf{u} \;\in\; \Omega. \tag{2.10}$$

2.2.4 Boundary Conditions

In peridynamics, there is no straightforward way of enforcing boundary conditions. Compared to continuum mechanics, where boundary conditions arise naturally from problem statement, the peridynamic equations contains no such terms. There is however an artificial technique which introduces fictitious material layers in which conditions can be enforced [17]. Dirichlet (prescribed displacement) conditions are applied by setting all particles in the fictitious boundary layer Ω_D to $\mathbf{U}_p$. The fictitious layer must be thick enough to overcome a boundary effect called *edge softening*, which occurs because particles close to the edge have fever connections than internal particles, and as a consequence have a *softer* response[10]. In [25] Silling suggests, based on numerical experiments, using a thickness equal to the horizon δ.

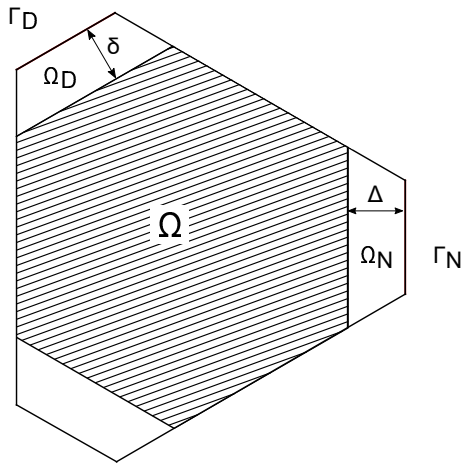


Figure 2.3: Boundary layers used for applying Dirichlet and Neumann boundary conditions.

For Neumann conditions (prescribed traction), a similar approach can be used. For the boundary layer Ω_N the body load $\mathbf{b}_\Gamma(\boldsymbol{X}, t)$ can be prescribed. To avoid too abrupt application of the load a linear fading between $t = t_0$ and $t = t_1$ can be applied [17]. For a total force of magnitude P, applied at boundary Γ_N with unit normal $\mathbf{n}_N$, $\mathbf{b}$ can be written as

$$\mathbf{b}_\Gamma(\boldsymbol{X}, t) = \frac{P}{\Gamma_N \Delta} \frac{t - t_0}{t_1 - t_0} \mathbf{n}_N, \tag{2.11}$$

where Δ is the grid size of the discretisation [17]. The same fading technique is preferably also used for prescribed displacements.

2.2.5 Volume Correction

For a given point, the entire volume of the family member points within its horizon are used to approximate the numerical integration. This may results in inaccurate results since some points may be only partially enclosed (see shaded areas in Figure 2.4). Thus, the contribution from points not fully contained inside the horizon, i.e. when $\delta - r \leq |\boldsymbol{X}_j - \boldsymbol{X}_i| \leq \delta$ should be scaled accordingly. The choice of kernel will however influence the need for volume correction. For a bell-shaped kernel, particles that are partially outside the horizon will have little influence on the internal force of particle k, which means that volume correction will be less important. Volume correction can nevertheless be used to marginally improve the accuracy. For a uniform grid, and with $r = \Delta/2$, [17] proposes a correction factor

$$v_j^c = (\delta + r - |\boldsymbol{X}_j - \boldsymbol{X}_i|)/2r. \tag{2.12}$$

This factor basically applies a linearly varying weight between 1 and 0.5 thus decreasing the influence of particles that are partially outside the horizon. Figure 2.4 illustrates a rectangular material surface discretized into a uniform grid, with two selected particles and their corresponding horizons, bonds and families. The edge of the rectangle represents the surface edge.

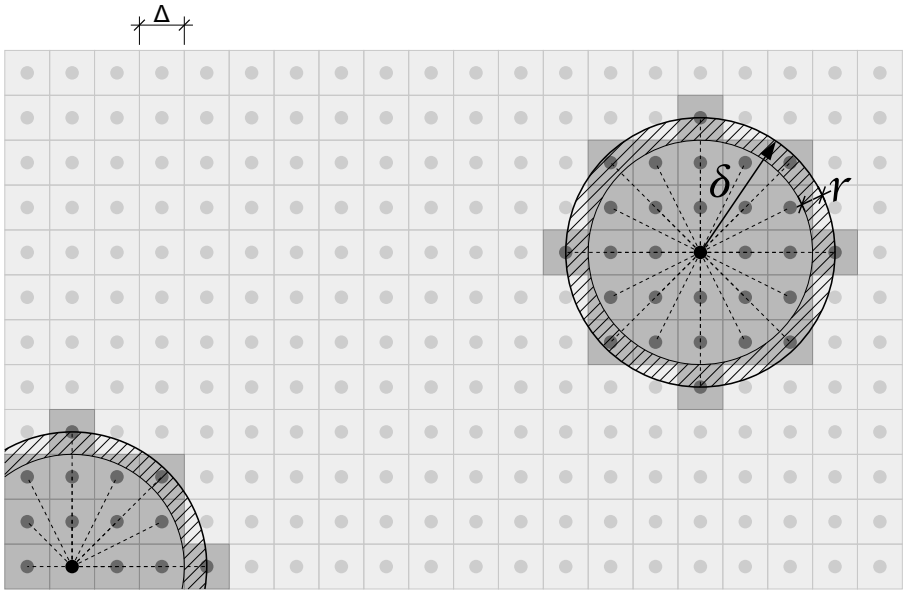


Figure 2.4: Uniform particle distribution with $\delta = 3\Delta$; families and volume correction regions for two chosen particles.

2.3 Modified Peridynamics Using Variable Particle Size

Using the model described below (Section 2.3.1) - which can be seen as type of ordinary state-based peridynamics - the particle size and density within a media are allowed to vary, see Figure 2.5. This makes it possible to have a non-uniform particle distribution (see Figure 3.5). Applications of this feature are discussed further in Section 5.2.3.

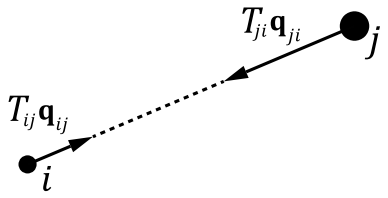


Figure 2.5: Ordinary state-based allowing for varying particle sizes.

2.3.1 Arm Tension Based on Virtual Fibres

By first considering the modelled body as a set of virtual fibres a model for how the stress changes with orientation can be derived without considering variations in position [21]. The complete description of stress state in the body is then achieved by introducing discrete particles in this fibre mat. From Equation 58 in [21] the total internal force on particle i, $\mathbf{f}_i^{int}$, can be written as a summation over all arm tensions T_{ij} due to interaction with all other particles j in the family $\mathcal{H}_{x_i}$

$$\mathbf{f}_i^{int} = \sum_j T_{ij}\mathbf{q}_{ij} = m_i \sum_j m_j \Big(\frac{S_i(\mathbf{q}_{ij})}{\rho_i^2} \nabla_i W_{ij} + \frac{S_j(\mathbf{q}_{ji})}{\rho_j^2} \nabla_i W_{ji} \Big), \tag{2.13}$$

where $\mathbf{q}_{ij} = (\boldsymbol{x}_j - \boldsymbol{x}_i)/r_{ij}$ is the unit vector between the current position of particle i and j, and $S(\mathbf{q_{ij}})$ is the *force flux density* at particle i in the direction $\mathbf{q}_{ij}$. The force flux density can be interpreted as the magnitude of the force that "flows" through a plane defined by $\mathbf{q}_{ij}$ (line in 2D) in a specific point $\boldsymbol{x}_i$ in the domain. Furthermore, $\nabla_i = 1/\partial \boldsymbol{X}_i$ is the gradient operator relative the undeformed position of particle i and W_{ij} is a Kernel function depending on the particle size h_i and relative distance $r_{ij} = |\boldsymbol{x}_j - \boldsymbol{x}_i|$. The kernel function is expressed using a dimensionless function $f(u)$ according to

$$W\Big(\frac{r_{ij}}{h_i}, h_i\Big) = \frac{1}{h_i^N} f\Big(\frac{r_{ij}}{h_i}\Big) = \frac{1}{h_i^N} f(u). \tag{2.14}$$

For the case of constant particle size and density, i.e. $m_i = m_j = m$ and $\rho_i = \rho_j = \rho$ and $\nabla_i W_{ij} = \nabla_i W_{ji}$, the internal force in Equation 2.13 simplifies to

$$\mathbf{f}_i^{int} = V_i \sum_j V_j \Big(S_i(\mathbf{q}_{ij}) \nabla_i W_{ij} + S_j(\mathbf{q}_{ji}) \nabla_i W_{ji} \Big). \tag{2.15}$$

Furthermore, the body force can be added and the identity $\nabla_i W_{ij} = -\nabla_j W_{ij}$ can be used in accordance with equation 55 in [21]. Note that $\nabla_i W_{ij} = \nabla_i W_{ji}$. It only remains to divide by V_i and replace the summation over j with an integration over $\mathcal{H}_x$ to arrive at the peridynamics equation of motion

$$\rho \ddot{\boldsymbol{u}} = \int_{\mathcal{H}_x} \Big(S(\boldsymbol{q}(\boldsymbol{x})) - S(\boldsymbol{q}(\boldsymbol{x}')) \Big) \nabla_{\boldsymbol{x}'} W(\boldsymbol{x}')\, dV_{\boldsymbol{x}'} + \boldsymbol{b}. \tag{2.16}$$

In 2D the unit of S is force per unit length and for ∇W the unit is one over length cubed. This is consistent with the unit of the force per unit area squared for the force density. However, to make use of the possibility to use variable particle sizes, Equation 2.13 is used in the implementation, and solved numerically for a certain time step by applying the body force of particle i and dividing by the mass in each direction m_i^x and m_i^y as in Equation 3.3.

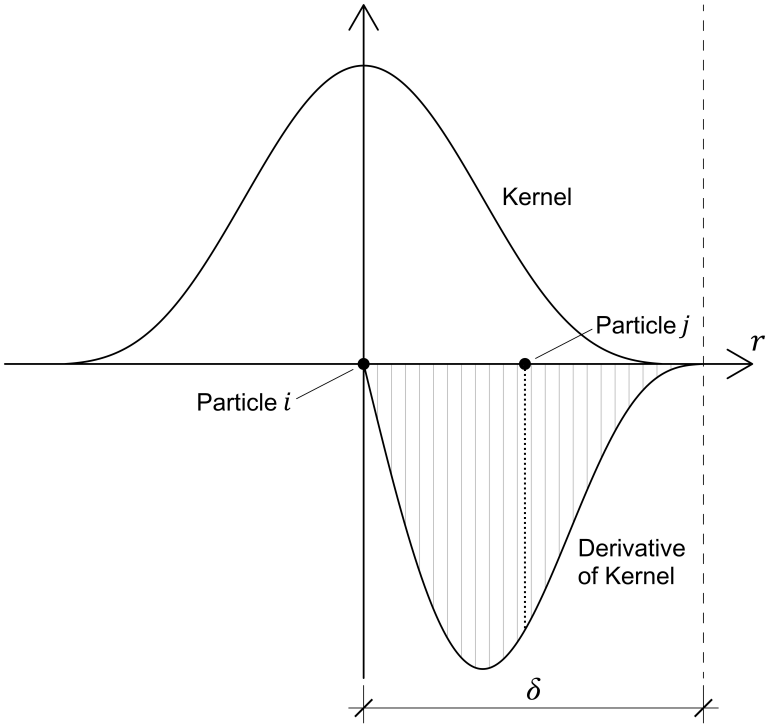


Figure 2.6: Typical kernel function.

The Kernel function chosen in [21] is used in the numerical implementation, and is given by

$$\begin{aligned} f(u) &= \frac{(1-\frac{u^2}{\alpha^2})^2}{2C(N-1)\pi\alpha^N(\frac{1}{N}-\frac{2}{N+2}+\frac{1}{N+4})} \quad for \quad u \leq \alpha \\ &= 0 \quad \text{for} \quad u > \alpha. \end{aligned} \tag{2.17}$$

Here $u = r/h$, with r and h are the radius and size respectively of a given particle, and α is a non-dimensional constant. In two dimensions, the Equation 2.17 is simplified to

$$\begin{aligned} f_{2D}(u) &= \frac{3}{C\pi\alpha^2}\Big(1-\frac{u^2}{\alpha^2}\Big)^2 \quad \text{for} \quad u \leq \alpha \\ &= 0 \quad for \quad u > \alpha, \end{aligned} \tag{2.18}$$

with the derivative

$$\begin{aligned} \frac{df_{2D}(u)}{du} &= \frac{12u}{C\pi\alpha^6}\Big(u^2-\alpha^2\Big) \quad \text{for} \quad u \leq \alpha \\ &= 0 \quad for \quad u > \alpha. \end{aligned} \tag{2.19}$$

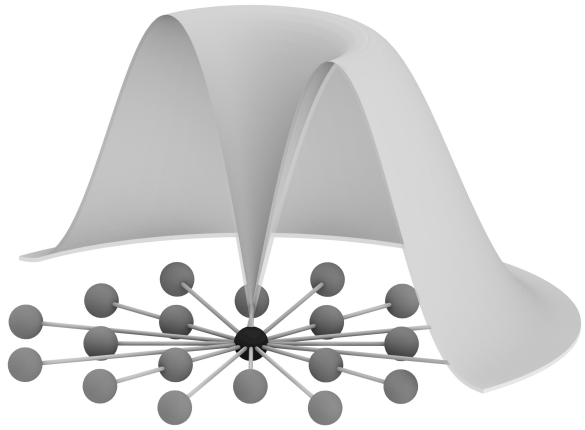


Figure 2.7: Illustration of the derivative of the kernel that is used for weighting the influence of each particle in the family.

The gradient of the kernel $\nabla_i W_{ij}$ as used in Equation 2.13 is in 2D evaluated as

$$\nabla_i W_{ij} = -\boldsymbol{q}_{ij} \frac{\partial W_{ij}}{\partial r_{ij}} = -\frac{\boldsymbol{q}_{ij}}{h_i^3} f'_{2D}(\frac{r_{ij}}{h_i}). \tag{2.20}$$

2.3.2 Constitutive Model

Following the virtual fibre approach in [21], a linear elastic model in N dimensions is formulated as

$$S(\mathbf{q}) = G(N+2)\mathbf{q} \cdot \boldsymbol{\epsilon} \cdot \mathbf{q} + (NK - G(N+2))\bar{\epsilon}, \tag{2.21}$$

where G and K are the shear and bulk modulus respectively, $\boldsymbol{\epsilon}$ is the strain tensor and $\bar{\epsilon}$ is the average strain. By considering the force flux of particle i in direction $\mathbf{q}_{ij}$, the strain can be reduced to the strain of the arm between particle i and j, ϵ_{ij}, and the force flux density can be given by

$$S_i(\mathbf{q}_{ij}) = G(N+2)\epsilon_{ij} + (NK - G(N+2))\bar{\epsilon}_i. \tag{2.22}$$

In two dimensions, this expression simplifies to

$$S_i(\mathbf{q}_{ij}) = 4G\epsilon_{ij} + (2K - 4G)\bar{\epsilon}_i. \tag{2.23}$$

The average strain $\bar{\epsilon}_i$ for particle i is computed by averaging the contribution from all neighbouring particles j using the kernel and particle volumes as weights

$$\bar{\epsilon}_i = \frac{\sum_j V_j \epsilon_{ij} W_{ji}}{\sum_j V_j W_{ji}}. \tag{2.24}$$

3
Extension of the Theory

The modified version of peridynamics presented in Section 2.3 is implemented in the design tool to simulate the mechanical behaviour of structural components. To fulfil the desired tool requirements outlined in Section 4.1, some modifications and adoptions need to be made to the theory. In particular, practical implementation issues need to be addressed, such as how to solve the equations numerically and how to compute useful measures that can inform geometry changes. This chapter covers the additional theory and derivations that are central to the implementation.

3.1 Finding the Static Solution by Using Dynamic Relaxation

The peridynamics equations describe the dynamic response of the particle system. By introducing artificial damping and mass, and tuning those parameters, the system can be forced to steady state in relatively short time.

3.1.1 Time Stepping Scheme

The time stepping is performed by applying a *central difference scheme*, which is an *explicit* numerical method (i.e. the evaluation is based only on known quantities from the previous time step). A suitable central difference scheme is adopted from chapter 8.1 in [16]. Knowing the displacement, velocity and acceleration of particle i at time t, the velocity a half step $\Delta t/2$ ahead can be approximated by extrapolation

$$\dot{\boldsymbol{u}}_i(t + \frac{\Delta t}{2}) = \dot{\boldsymbol{u}}_i(t) + \ddot{\boldsymbol{u}}_i(t)\frac{\Delta t}{2}. \tag{3.1}$$

Using this approximation of the midpoint velocity, the displacement at $t + \Delta t$ can be evaluated as

$$\boldsymbol{u}_i(t + \Delta t) = \boldsymbol{u}_i(t) + \dot{\boldsymbol{u}}_i(t + \frac{\Delta t}{2})\Delta t. \tag{3.2}$$

The internal force $\boldsymbol{f}_i^i$, caused by the displacement $\boldsymbol{u}_i(t+\Delta t)$ is then calculated using Equation 2.13, see Algorithm 1 in Section 4.5.1. If particle i is inside a Dirichlet boundary region the position of the particle is prescribed using the fading technique described in Section 2.2.4. The next step is to use the components of the fictitious mass m_i^x and m_i^y to evaluate the acceleration caused by the net force $\boldsymbol{f}_i = \boldsymbol{f}_i^i + \boldsymbol{f}_i^e$ acting on the particle i as

$$\ddot{\boldsymbol{u}}_i^x(t+\Delta t) = \frac{\boldsymbol{f}_i}{m_i^x},$$
$$\ddot{\boldsymbol{u}}_i^y(t+\Delta t) = \frac{\boldsymbol{f}_i}{m_i^y}. \tag{3.3}$$

Finally, the velocity at $t + \Delta t$ can be evaluated as

$$\dot{\boldsymbol{u}}(t+\Delta t) = \dot{\boldsymbol{u}}_i(t+\frac{\Delta t}{2}) + \frac{\Delta t}{2}\ddot{\boldsymbol{u}}_i(t+\Delta t). \tag{3.4}$$

Figure 3.1 summarizes the procedure described above, with the graphs merely acting as a support for the understanding.

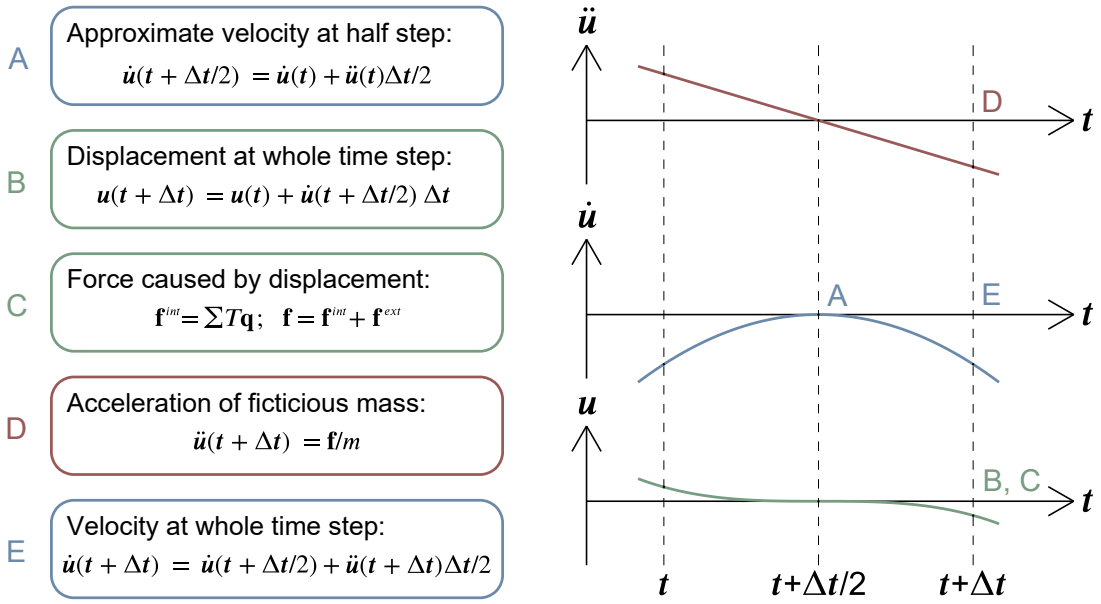


Figure 3.1: The time stepping scheme with time derivatives of the displacement

3.1.2 Numerical Stability from Courant–Friedrichs–Lewy Condition

When applying the time stepping scheme described in Section 3.1.1 it is necessary to use a small enough time step such that numerical instability does not occur. On the other hand one would like to use a large time step to reduce computation time. There are several conditions established for choosing a stable combination of time step and mass. One of those is the Courant–Friedrichs–Lewy (CFL) condition, which states that the time step must be small enough so that the wave does not propagate more than the smallest distance between two measurement points in the discretization. With c denoting the wave speed and r_{ij} being the arm length between particles i and j [16], in general

$$\Delta t_{crit} = \frac{r_{ij}}{c}. \tag{3.5}$$

Critical time step based on wave speed estimation
As a crude approximation of the wave propagation speed in the particle model the analytical wave speed based on the material is used. Two different sound waves in solids are distinguished; pressure waves c_p (generating volumetric deformations) and shear waves c_s (generating shear deformations) [7]. The larger of the two is inserted in Equation 3.5, as it will yield the smallest critical time step.

$$c = \max\Big(\sqrt{\frac{K + \frac{3}{4}G}{\rho}}, \quad \sqrt{\frac{G}{\rho}} \Big), \tag{3.6}$$

with K being the bulk modulus, G the shear modulus and ρ the density.

Derivation of a stable mass using the highest eigenvalue
Based on the Courant–Friedrichs–Lewy stability condition, a stable time step based on the maximum eigenvalue of the system can be derived

$$\Delta t_{crit} = \frac{2}{\sqrt{\lambda_{max}}}, \tag{3.7}$$

as presented in [19]. Evaluating this expression requires an approximation of the highest eigenvalue of the system, defined by the stiffness matrix $\boldsymbol{K}$ and mass matrix $\boldsymbol{M}$. One such approximation is obtained from the Gershgorin circle theorem, which provides an upper bound of λ_{max}

$$\lambda_{max} \leq \max_k \sum_l \frac{|K_{kl}|}{M_{kl}}. \tag{3.8}$$

By combining Equation 3.7 and 3.8 a stability condition in terms of the stiffness and mass matrices can be formulated [19]. Since the fictitious mass and the time step are coupled, the value of either one can be arbitrarily chosen after which the other value is determined. A convenient approach is to set $\Delta t = 1$ unit of time and solve for the maximum mass of each degree of freedom

$$m_{kk} \geq \frac{1}{4}\Delta t^2 \sum_l |K_{kl}|. \tag{3.9}$$

Consider particle i with degrees of freedom $k = 2i - 1$ in the x-direction and $k = 2i$ in the y-direction. By decomposing the internal force and the relative displacement $\boldsymbol{\eta}_{ij} = \boldsymbol{u}_j - \boldsymbol{u}_i$ into its components in the x and y direction, the masses for particle i can be written as

$$\begin{aligned} m_i^x &\geq \frac{1}{4}\Delta t^2 \sum_l |K_{2i-1,l}| = \frac{1}{4}\Delta t^2 \sum_j 2\left|\frac{\partial f_i^x}{\partial \eta_{ij}^x}\right| + 2\left|\frac{\partial f_i^x}{\partial \eta_{ij}^y}\right|, \\ m_i^y &\geq \frac{1}{4}\Delta t^2 \sum_l |K_{2i,l}| = \frac{1}{4}\Delta t^2 \sum_j 2\left|\frac{\partial f_i^y}{\partial \eta_{ij}^x}\right| + 2\left|\frac{\partial f_i^y}{\partial \eta_{ij}^y}\right|. \end{aligned} \tag{3.10}$$

The stiffness components in Equation 3.10 can be found by evaluating the change in internal force due to a perturbation of each relative displacement in the family.

For small displacements the tangent stiffness matrix in Equation 3.9 becomes the linear stiffness matrix. This means that it is sufficient to evaluate the stiffness once, preferable for zero displacement, i.e. around $\boldsymbol{\eta}_{ij} = \mathbf{0}$

$$\left.\frac{\partial \boldsymbol{f}_i}{\partial \boldsymbol{\eta}_{ij}}\right|_{\boldsymbol{\eta}_{ij}=\mathbf{0}} = m_i m_j \frac{\partial}{\boldsymbol{\eta}_{ij}}\Big(S_i(\boldsymbol{q}_{ij})\frac{\nabla_i W_{ij}}{\rho_i^2} + S_j(\boldsymbol{q}_{ji})\frac{\nabla_i W_{ji}}{\rho_j^2}\Big)\Bigg|_{\boldsymbol{\eta}_{ij}=\mathbf{0}}. \tag{3.11}$$

Since both the force flux density and the gradient of the kernel is dependent on $\boldsymbol{\eta}_{ij}$ the product rule must be applied

$$\begin{aligned}\left.\frac{\partial \boldsymbol{f}_i}{\partial \boldsymbol{\eta}_{ij}}\right|_{\boldsymbol{\eta}_{ij}=\mathbf{0}} = m_i m_j \Bigg[&\frac{\partial S_i(\boldsymbol{q}_{ij})}{\partial \boldsymbol{\eta}_{ij}} \otimes \frac{\nabla_i W_{ij}}{\rho_i^2} + \frac{S_i(\boldsymbol{q}_{ij})}{\rho_i^2}\frac{\partial \nabla_i W_{ij}}{\partial \boldsymbol{\eta}_{ij}} + \\ &\frac{\partial S_j(\boldsymbol{q}_{ji})}{\partial \boldsymbol{\eta}_{ij}} \otimes \frac{\nabla_i W_{ji}}{\rho_j^2} + \frac{S_j(\boldsymbol{q}_{ji})}{\rho_j^2}\frac{\partial \nabla_i W_{ij}}{\partial \boldsymbol{\eta}_{ij}}\Bigg]\Bigg|_{\boldsymbol{\eta}_{ij}=\mathbf{0}}.\end{aligned} \tag{3.12}$$

Both terms involving the force flux density (S_i and S_j) vanish however, when evaluated at $\boldsymbol{\eta}_{ij} = \mathbf{0}$. The derivative of the force flux density means differentiating the constitutive relation in Equation 2.21;

$$\frac{\partial S_i(\boldsymbol{q}_{ij})}{\partial \boldsymbol{\eta}_{ij}} = \frac{\partial}{\partial \boldsymbol{\eta}_{ij}} G(N+2)\epsilon_{ij} + \frac{\partial}{\partial \boldsymbol{\eta}_{ij}}(NK - G(N+2))\bar{\epsilon}_i. \tag{3.13}$$

Since the arm strain can be written as $\epsilon_{ij} = \frac{r_{ij} - |\boldsymbol{\xi}_{ij}|}{|\boldsymbol{\xi}_{ij}|}$, with $\boldsymbol{\xi}_{ij} = \boldsymbol{X}_j - \boldsymbol{X}_i$, we can write the the first term in Equation 3.13 as

$$\frac{\partial}{\partial \boldsymbol{\eta}_{ij}} G(N+2)\epsilon_{ij} = \frac{G(N+2)}{|\boldsymbol{\xi}_{ij}|}\frac{\partial r_{ij}}{\partial \boldsymbol{\eta}_{ij}} = G(N+2)\frac{\boldsymbol{q}_{ij}}{|\boldsymbol{\xi}_{ij}|}, \tag{3.14}$$

where we used that

$$\frac{\partial r_{ij}}{\partial \boldsymbol{\eta}_{ij}} = \frac{\partial |\boldsymbol{\eta}_{ij} + \boldsymbol{\xi}_{ij}|}{\partial \boldsymbol{\eta}_{ij}} = \frac{\boldsymbol{\eta}_{ij} + \boldsymbol{\xi}_{ij}}{|\boldsymbol{\eta}_{ij} + \boldsymbol{\xi}_{ij}|} = \boldsymbol{q}_{ij}. \tag{3.15}$$

For the second term in Equation 3.13, the derivative of the average strain defined in Equation 2.24 is needed. Since the kernel function is present in both numerator and denominator, the quotient rule is needed for an exact evaluation. After some manipulations the derivative of the numerator around $\epsilon_{ij} = 0$ is reduced to

$$\frac{\partial}{\partial \boldsymbol{\eta}_{ij}} \sum_k V_k \boldsymbol{\epsilon}_{ik} W_{ki} = V_j W_{ji}\frac{\partial \epsilon_{ij}}{\partial \boldsymbol{\eta}_{ij}} = V_j W_{ji}\frac{\boldsymbol{q}_{ij}}{|\boldsymbol{\xi}_{ij}|}, \tag{3.16}$$

which is combined with the denominator in the quotient rule to yield the expression for the derivative of the average strain

$$\frac{\partial \bar{\epsilon}_i}{\partial \boldsymbol{\eta}_{ij}} = \frac{(V_j W_{ji})^2}{\sum_k (V_k W_{ki})^2}\frac{\boldsymbol{q}_{ij}}{|\boldsymbol{\xi}_{ij}|}. \tag{3.17}$$

The derivative of the opposite force flux density S_j is calculated in the same fashion. The difference is however that the average strain now is based on particle j, which includes a summation over all its neighbouring particles l

$$\frac{\partial \bar{\epsilon}_j}{\partial \boldsymbol{\eta}_{ij}} = \frac{\sum_l (V_l W_{lj})^2 \frac{\partial \epsilon_{jl}}{\partial \boldsymbol{\eta}_{ij}}}{\sum_l (V_l W_{lj})^2} = \frac{-\sum_l (V_l W_{lj})^2 \delta_{li} \frac{\boldsymbol{q}_{jl}}{|\boldsymbol{\xi}_{jl}|}}{\sum_l (V_l W_{lj})^2} = \frac{(V_i W_{ij})^2}{\sum_l (V_l W_{lj})^2} \frac{\boldsymbol{q}_{ij}}{|\boldsymbol{\xi}_{ij}|}. \tag{3.18}$$

The components of Equation 3.12 are evaluated with respect to the basis vectors $\hat{\boldsymbol{e}}_x$ and $\hat{\boldsymbol{e}}_y$ as

$$\frac{\partial \boldsymbol{f}_i}{\partial \boldsymbol{\eta}_{ij}} = \begin{bmatrix} \frac{\partial f_i^x}{\partial \eta_{ij}^x} & \frac{\partial f_i^x}{\partial \eta_{ij}^y} \\ \frac{\partial f_i^y}{\partial \eta_{ij}^x} & \frac{\partial f_i^y}{\partial \eta_{ij}^y} \end{bmatrix}, \tag{3.19}$$

which can be inserted in Equation 3.10 after which the stable mass can be calculated.

3.1.3 Damping

Damping is introduced in the equation of motion 2.8 in order to achieve the steady state. Numerically this is achieved by introducing a *carry over factor* which is multiplied with velocity in each time step. This has the effect of reducing the kinetic energy that is transferred to the next time step. Although the optimal value can be calculated, the calibration is delegated to the user.

3.1.4 Convergence Criteria

It is not necessary for the simulation to have reached static equilibrium for the user to alter the domain. It could however be beneficial to terminate the simulation to save computational resources and make the program more responsive. A suitable scalar quantity to apply a convergence criterion to is the sum of the residual force magnitude of all particles.

$$R_{tot} = \sum_{k=1}^{n} |\ddot{\boldsymbol{u}}_k| m_k. \tag{3.20}$$

An alternative to this could be to add the force residual of all particles and then take the magnitude. One drawback of this approach is that local equilibrating forces are not captured. Another measure is the total kinetic energy of all particles, note that n is the number of particles

$$T_{tot} = \sum_{k=1}^{n} \frac{1}{2} m_k |\dot{\boldsymbol{u}}_k|^2. \tag{3.21}$$

3.2 Stress and Strain Measures

Since the displacement vectors only are known in discrete set of points the deformation tensor is preferably obtained by studying the deformation of nearby particles. By approximating the position of neighbouring points using a first order Taylor expansion an approximation of the deformation tensor can be obtained. A consequence of this linearization is that the deformation field is poorly described in regions with large deformation gradient. The strain measure used is the Green-Lagrange strain tensor, which converge to the engineering strain when the deformations are approaching zero.

3.2.1 Derivation of Green-Lagrange Strain Components From a Discrete Displacement Field

The Green-Lagrange strain tensor $\boldsymbol{E}$ can be expressed in terms of the deformation tensor $\boldsymbol{F}$ according to

$$\boldsymbol{E} = \frac{1}{2}(\boldsymbol{F}^T \cdot \boldsymbol{F} - \boldsymbol{I}), \tag{3.22}$$

where $\boldsymbol{I}$ denotes the identity tensor. $\boldsymbol{F}$ describes how an infinitesimal line is transformed from the reference to the deformed configuration. Following the notation in Section 2.1.1 this can in each point be described as $d\boldsymbol{x} = \boldsymbol{F}d\boldsymbol{X}$. Also consider the deformed position of a point a finite distance away, $\boldsymbol{x} + \Delta\boldsymbol{x} = \boldsymbol{X} + \Delta\boldsymbol{X}$, which can be written as a linear function $f(\boldsymbol{X} + \Delta\boldsymbol{X})$ of the initial position. Following [9], a second order Taylor expansion of the deformed position of $\boldsymbol{x} + \Delta\boldsymbol{x}$ gives

$$f(\boldsymbol{X} + \Delta\boldsymbol{X}) \approx \frac{\partial f(\boldsymbol{X})}{\partial \boldsymbol{X}}\Delta\boldsymbol{X} + \frac{\partial^2 f(\boldsymbol{X})}{\partial \boldsymbol{X}^2}\Delta\boldsymbol{X}\Delta\boldsymbol{X}. \tag{3.23}$$

From here an approximation of the deformation gradient can be formulated based on both terms of the expansion, as in [9]. For the current implementation however, a simpler approach is used where the deformation is approximated from the first term only. By using that $f(\Delta\boldsymbol{X}) = \Delta\boldsymbol{x}$, subtracting $f(\boldsymbol{X})$ from both sides an approximation of $\boldsymbol{F}$ using finite relative positions is established

$$\Delta\boldsymbol{x} \approx \frac{\partial \boldsymbol{x}}{\partial \boldsymbol{X}}\Delta\boldsymbol{X} = \boldsymbol{F}\Delta\boldsymbol{X}. \tag{3.24}$$

It should be noted that this linearization approaches the true deformation gradient for small $\Delta\boldsymbol{X}$. For implementation in peridynamics $\Delta\boldsymbol{X}$ is chosen as the undeformed distance between two particles $\boldsymbol{X}_j - \boldsymbol{X}_i$ and $\Delta\boldsymbol{x}$ as the deformed distance between the same two particles $\boldsymbol{x}_j - \boldsymbol{x}_i$. The evaluation of the deformation tensor and thus the strain will therefore be dependent on the particle spacing.

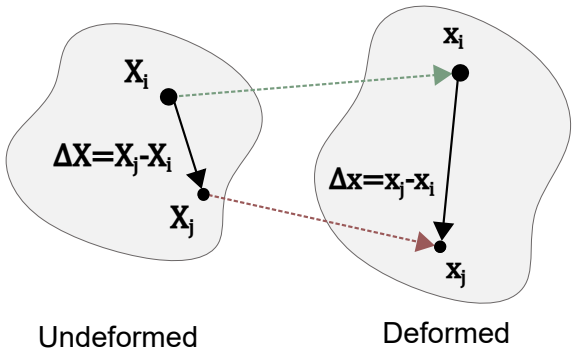


Figure 3.2: Illustration of the relative position of two particles i and j in undeformed and deformed configuration.

In 2D, $\boldsymbol{F}$ has four independent components and consequently four displacement components are required in order to solve for the unknowns in $\boldsymbol{F}$. Since each particle has many more bonds, a least square solution is used. Each bond contributes with two equations (one in each direction). In a particular coordinate system (with unit vectors $\hat{\boldsymbol{e}}_x$ and $\hat{\boldsymbol{e}}_y$) the bond between particle i and j will contribute with the equations

$$\begin{bmatrix} \boldsymbol{x}_i^x - \boldsymbol{x}_j^x \\ \boldsymbol{x}_i^y - \boldsymbol{x}_j^y \end{bmatrix} = \begin{bmatrix} F_i^{xx} & F_i^{xy} \\ F_i^{yx} & F_i^{yy} \end{bmatrix} \begin{bmatrix} \boldsymbol{X}_i^x - \boldsymbol{X}_j^x \\ \boldsymbol{X}_i^y - \boldsymbol{X}_j^y \end{bmatrix}. \tag{3.25}$$

For a particle i with family particles $j = 1, 2, ..., n$, a matrix $\boldsymbol{A}$ and a solution vector $\boldsymbol{b}$ can be established. Using this format all equations for particle i can be combined in the system $\boldsymbol{A}_i \tilde{\boldsymbol{F}}_i = \boldsymbol{b}_i$, or explicitly

$$\begin{bmatrix} \boldsymbol{X}_i^x - \boldsymbol{X}_1^x & \boldsymbol{X}_i^y - \boldsymbol{X}_1^y & 0 & 0 \\ 0 & 0 & \boldsymbol{X}_i^x - \boldsymbol{X}_1^x & \boldsymbol{X}_i^y - \boldsymbol{X}_1^y \\ \boldsymbol{X}_i^x - \boldsymbol{X}_2^x & \boldsymbol{X}_i^y - \boldsymbol{X}_2^y & 0 & 0 \\ 0 & 0 & \boldsymbol{X}_i^x - \boldsymbol{X}_2^x & \boldsymbol{X}_i^y - \boldsymbol{X}_2^y \\ \vdots & \vdots & \vdots & \vdots \\ \boldsymbol{X}_i^x - \boldsymbol{X}_n^x & \boldsymbol{X}_i^y - \boldsymbol{X}_n^y & 0 & 0 \\ 0 & 0 & \boldsymbol{X}_i^x - \boldsymbol{X}_n^x & \boldsymbol{X}_i^y - \boldsymbol{X}_n^y \end{bmatrix} \begin{bmatrix} F_i^{xx} \\ F_i^{xy} \\ F_i^{yx} \\ F_i^{yy} \end{bmatrix} = \begin{bmatrix} \boldsymbol{x}_i^x - \boldsymbol{x}_1^x \\ \boldsymbol{x}_i^y - \boldsymbol{x}_1^y \\ \boldsymbol{x}_i^x - \boldsymbol{x}_2^x \\ \boldsymbol{x}_i^y - \boldsymbol{x}_2^y \\ \vdots \\ \boldsymbol{x}_i^x - \boldsymbol{x}_n^x \\ \boldsymbol{x}_i^y - \boldsymbol{x}_n^y \end{bmatrix}. \tag{3.26}$$

This system is overdetermined since there are more than four equations. A suitable approach to find values for $\tilde{\boldsymbol{F}}$ that reflects the true deformations is to solve the least square problem

$$\tilde{\boldsymbol{F}}_i = (\boldsymbol{A}_i^T \boldsymbol{A}_i)^{-1} \boldsymbol{A}_i^T \boldsymbol{b}_i. \tag{3.27}$$

Since $\Delta \boldsymbol{X}$ is greater for particles further away in the family their contribution to the least square approximation is preferably limited. One obvious way to decrease the influence is to use the kernel function as weights in the least square approximation. In accordance with [5], a weighted least square solution of the system of equations

in Equation 3.26 can be found by introducing the weighting matrix $\boldsymbol{W}$, whereby the deformation tensor is approximated as

$$\tilde{\boldsymbol{F}}_i = (\boldsymbol{A}_i^T \boldsymbol{W}_i \boldsymbol{A}_i)^{-1} \boldsymbol{A}_i^T \boldsymbol{W}_i \boldsymbol{b}_i. \tag{3.28}$$

The weighting matrix proposed by the authors contains the Kernel weights associated to each particle j according to

$$\boldsymbol{W}_i = \begin{bmatrix} W_{1i} & 0 & 0 & 0 & \dots & 0 & 0 \\ 0 & W_{1i} & 0 & 0 & \dots & 0 & 0 \\ 0 & 0 & W_{2i} & 0 & \dots & 0 & 0 \\ 0 & 0 & 0 & W_{2i} & \dots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \dots & W_{ni} & 0 \\ 0 & 0 & 0 & 0 & 0 & \dots & W_{ni} \end{bmatrix}. \tag{3.29}$$

Once the components of the deformation tensor are established the strain components can be calculated using Equation 3.22

$$\epsilon_{xx} = \frac{1}{2}(F_{yx}^2 + F_{xx}^2 - 1), \tag{3.30}$$

$$\epsilon_{xy} = \epsilon_{yx} = \frac{1}{2}(F_{xx}F_{xy} + F_{yx}F_{yy}), \tag{3.31}$$

$$\epsilon_{yy} = \frac{1}{2}(F_{xy}^2 + F_{yy}^2 - 1). \tag{3.32}$$

3.2.2 Stress Components

In 2D the stress components are either

$$\begin{bmatrix} \sigma_{xx} \\ \sigma_{yy} \\ \sigma_{xy} \end{bmatrix} = \frac{E}{(1+\nu^2)} \begin{bmatrix} 1 & \nu & 0 \\ \nu & 1 & 0 \\ 0 & 0 & 1-\nu \end{bmatrix} \begin{bmatrix} \epsilon_{xx} \\ \epsilon_{yy} \\ \epsilon_{xy} \end{bmatrix}, \tag{3.33}$$

using a plane stress assumption, or

$$\begin{bmatrix} \sigma_{xx} \\ \sigma_{yy} \\ \sigma_{xy} \end{bmatrix} = \frac{E}{(1+\nu)(1-2\nu)} \begin{bmatrix} 1-\nu & \nu & 0 \\ \nu & 1-\nu & 0 \\ 0 & 0 & 1-2\nu \end{bmatrix} \begin{bmatrix} \epsilon_{xx} \\ \epsilon_{yy} \\ \epsilon_{xy} \end{bmatrix}, \tag{3.34}$$

using a plane strain assumption. In the latter case the out of plane stress σ_{zz} is given by

$$\sigma_{zz} = \frac{E\nu}{(1+\nu)(1-2\nu)}(\epsilon_{xx} + \epsilon_{yy}). \tag{3.35}$$

3.2.3 Principal Stress

An efficient way of displaying the primary force flows is to plot the principal stress, i.e. the minimal and maximal normal stress vector in each point. The principal

stress λ_i and associated principal direction $\hat{\boldsymbol{n}}_i$ of the Cauchy stress tensor $\boldsymbol{\sigma}$ are found by solving the eigenvalue problem

$$\boldsymbol{\sigma}\hat{\boldsymbol{n}}_i = \lambda_i \hat{\boldsymbol{n}}_i. \tag{3.36}$$

Which means that the principal stress vectors are those where the stress vectors only have a normal component. Negative eigenvalues correspond to compression and positive to tension, note that the principal stresses are always orthogonal. In the implementation, the eigenvalues and eigenvectors are evaluated particle wise, and principal stress lines are generated using nearest-neighbour interpolation based on a given step size. Thereby, the continuous principal stress trajectories can be displayed for a selection of particles to guide the user in shape and topology changes based on the primary force flows.

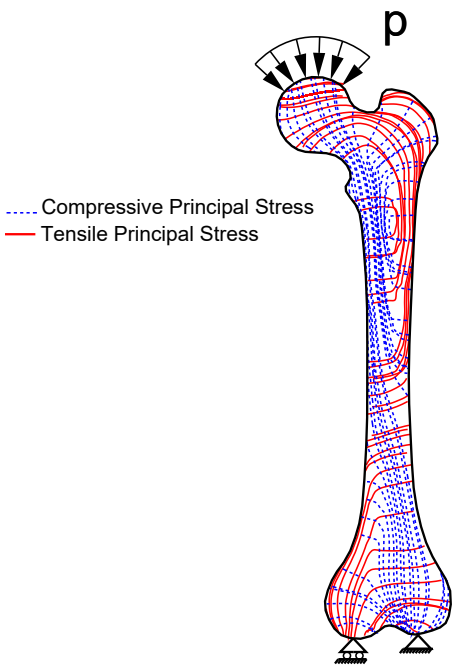


Figure 3.3: Principal stress trajectories of a thigh bone, modelled as an isotropic plane stress continuum using the design tool.

3.2.4 Effective Stress

Effective stress measures are used for quantifying the stress in a single scalar value. This can then be used for assessing material yielding by comparing it to test values achieved by uniaxial tensile testing. Two common effective stress measures are *von Mises* and *Tresca*, both of which are based on the deviatoric part of the stress. The von Mises stress can be determined from the components of the Cauchy stress tensor, for a general stress state in two dimensions it is evaluated as

$$\sigma_{vM} = \sqrt{\frac{1}{2}(\sigma_{xx} - \sigma_{yy})^2 + (\sigma_{yy} - \sigma_{zz})^2 + (\sigma_{zz} - \sigma_{xx})^2 + 3\sigma_{xy}^2}. \tag{3.37}$$

3.2.5 Laplacian Smoothing

As a consequence from that the stress is only computed for discrete particles, stress concentrations may arise as undesired artefacts from the discretization. By applying Laplacian smoothing to the stress state of a particle based on its closest neighbours, this effect can be reduced and a more representative stress field can be obtained. For particle i, with n neighbours, the Laplacian smoothing is given by

$$\bar{\sigma}_i = \frac{1}{n} \sum_{j=1}^{n} \bar{\sigma}_j. \tag{3.38}$$

3.3 Rigid Body Motions

Rigid body motion (RBM) here refers to the states where all particles have the same acceleration $\ddot{\boldsymbol{u}}^{RBT}$ in the case of translations (RBT), or angular acceleration $\ddot{\boldsymbol{\omega}}^{RBR}$ in the case of rotations (RBR). In a dynamic analysis RBM does not necessarily need to be removed if the strain measure is chosen wisely. When displaying the deformation with a scale factor however, even small RBM can interfere with the results and make it difficult to interpret the results. Therefore, the user should be able to neglect such artefacts of the boundary condition setup, why an option of removing RBT and RBR has been added. Finally, since the residual force and kinetic energy are computed from the total acceleration and velocity RBM must be removed if they should converge to zero.

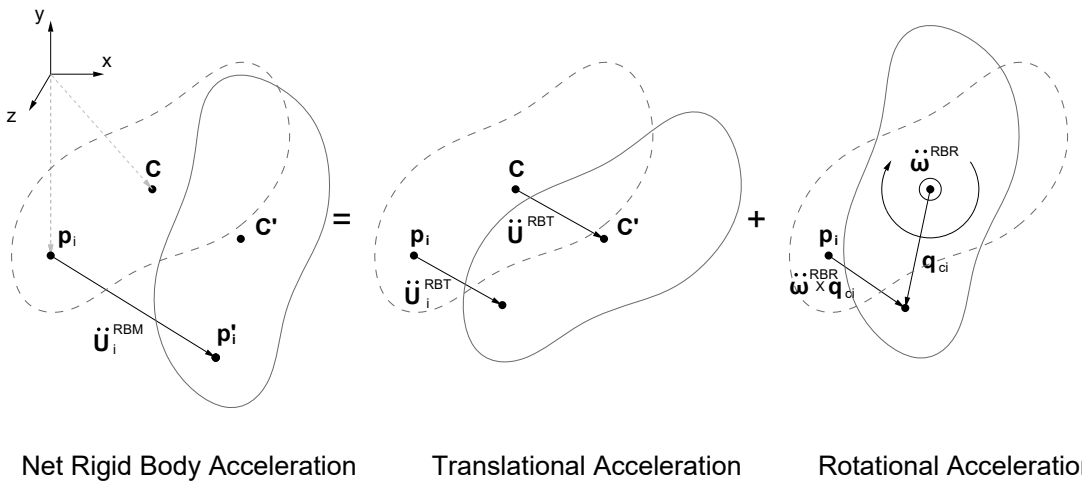


Figure 3.4: Rigid body motion as the sum of translations and rotations.

The RBT is removed by subtracting the average acceleration $\ddot{\boldsymbol{u}}^{RBT} = \sum_n \ddot{\boldsymbol{u}}_n \,/\, m$ of all particles from each particle's acceleration, according to

$$\ddot{\boldsymbol{u}}_i = \ddot{\boldsymbol{u}}_i - \ddot{\boldsymbol{u}}^{RBT}. \tag{3.39}$$

Similarly, the RBR is removed by subtracting the contribution from the average angular acceleration $\ddot{\boldsymbol{\omega}}^{RBR} = \sum_n \ddot{\boldsymbol{\omega}}_n \,/\, n$ from each particle's acceleration, as given by

$$\ddot{\boldsymbol{u}}_i = \ddot{\boldsymbol{u}}_i - \ddot{\boldsymbol{\omega}}^{RBR} \times \boldsymbol{q}_{ci}, \tag{3.40}$$

with $\boldsymbol{q}_{ci}$ being the vector between the mass centre c and particle i. The angular velocity of particle i is given by

$$\ddot{\boldsymbol{\omega}}_i^{RBR} = \frac{\boldsymbol{q}_{ci} \times \ddot{\boldsymbol{u}}_{ci}}{\boldsymbol{q}_{ci} \cdot \boldsymbol{q}_{ci}} = \frac{\boldsymbol{q}_{ci} \times \ddot{\boldsymbol{u}}_i}{\boldsymbol{q}_{ci} \cdot \boldsymbol{q}_{ci}}. \tag{3.41}$$

Where it was used that the relative acceleration $\ddot{\boldsymbol{u}}_i$ is the same as the acceleration of point i since the rotation is about the centre of mass. Note that the angular velocity vector $\ddot{\boldsymbol{\omega}}$ is pointing in the normal direction of the plane of rotation.

3.4 Volume Correction for Non-Uniform Particle Distribution

For a non-uniform particle distribution with varying particle sizes and irregular positions, Equation 2.12 will be a poor approximation. As can be seen in Figure 3.5 each particle no longer occupies a rectangular area, and so a linear scaling of the particle contribution is no longer valid.

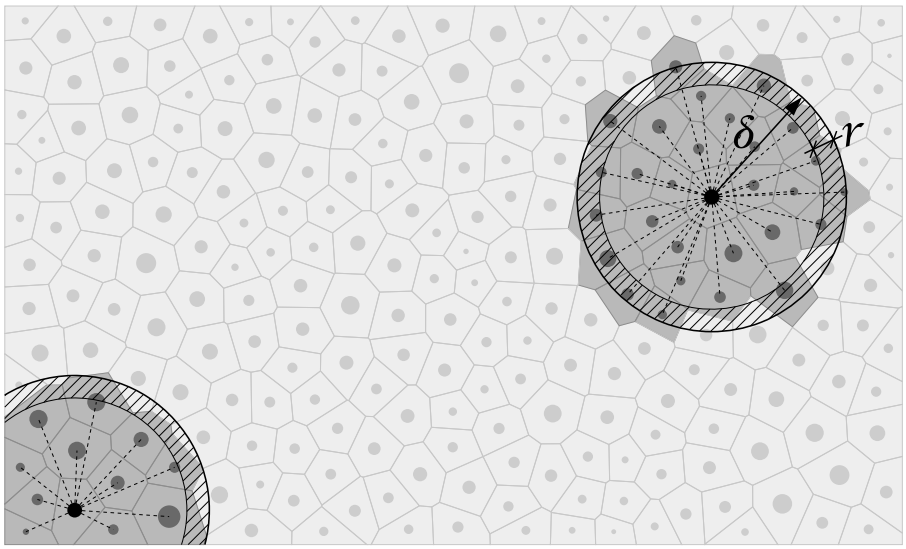


Figure 3.5: Non-uniform particle distribution; families and volume correction regions for two chosen particles.

Since the tool is aimed at the early design stage a small strain assumption can be used. From this it follows that it is sufficient to evaluate the volume correction factor once. The authors propose a particle specific reduction factor that is based on the bond length, the horizon and the radius of the effective particle area. By considering the horizon as flat the distance of the circle which is outside the horizon is $\varsigma_j = r_j + |\boldsymbol{X}_j - \boldsymbol{X}_i| - \delta$. The corresponding area outside the horizon is

$$a_j^o = r^2 \arccos(1 - \frac{\varsigma_j}{r}) - (r - \varsigma_j)\sqrt{r^2 - (r - \varsigma_j)^2}, \tag{3.42}$$

from which the reduction factor can be established as

$$v_j^c = \frac{1 - r^2 \arccos(1 - \frac{\varsigma_j}{r}) - (r - \varsigma_j)\sqrt{r^2 - (r - \varsigma_j)^2}}{r^2 \pi}. \tag{3.43}$$

For a small strain setting the volume correction factor does not need to be updated, since the deformed state can be approximated to be the same as the undeformed.

4

Tool Development and Implementation

This chapter deals with the development of the tool and how the theory is implemented. Firstly, certain key requirements are identified, these will guide the functionalities of the tool. The next step deals with how these requirements can be met by outlining the conceptual functionalities. The chapter ends by describing how the tool is implemented in the CAD software Rhino. This includes both the overall program structure and more detailed software architecture.

4.1 User Requirements

The user should be able to modify the domain both when the simulation is running and when it has reached steady state. During the whole simulation the tool should display the stress field such that the user can make informed updates of the geometry. Specifically, the tool should give the designer the possibilities to:

- Change the shape and size of the body.
- Change the topology of the body by specifying closed curves that defines regions where material should be removed.
- Easily locate the regions with the least stressed particles, which can then be removed parametrically.
- Define grid size, and other parameters used for the simulation.
- Define the extents, magnitudes and directions of the boundary conditions.
- Display the von Mises stress distribution and the principal stress trajectories in the Rhino viewport - the former as a colour plot and the latter as coloured curves. These should continuously update while the simulation is running.
- Obtain relevant data as an output from the analysis.

4.2 Tool Features and Functionality

The tool features and functionalities are based on the requirements presented above. This section aims at formalizing the features of the tool as well as presenting techniques for accomplishing the required functionalities.

4.2.1 Generation of the Initial Design Domain

By providing the tool with a *Curve* object that represents the boundary, a design domain is automatically generated. This curve can be generated manually in Rhino or parametrically in Grasshopper. This way the tool can easily be integrated in a larger analysis while keeping the flexibility for exploration of isolated bodies. Once the boundary curve is provided, a canvas of the domain is created and scaled up to accommodate for an increase of the domain, see Figure 4.1. This canvas constitutes the region in which the geometry can be changed, if the geometry is extended beyond this canvas the simulation must be reset. Even though only particles within the domain will be activated this gives room for the user to expand the domain during the analysis, which is simply done by activating all new particles that come inside the domain. Another reason for choosing to work with a rectangular canvas is that it is easily divided into particle zones, no matter how complicated the initial domain is. The reason for using a zone division of particles is primarily to reduce the computational needs, more on this in Section 4.3.

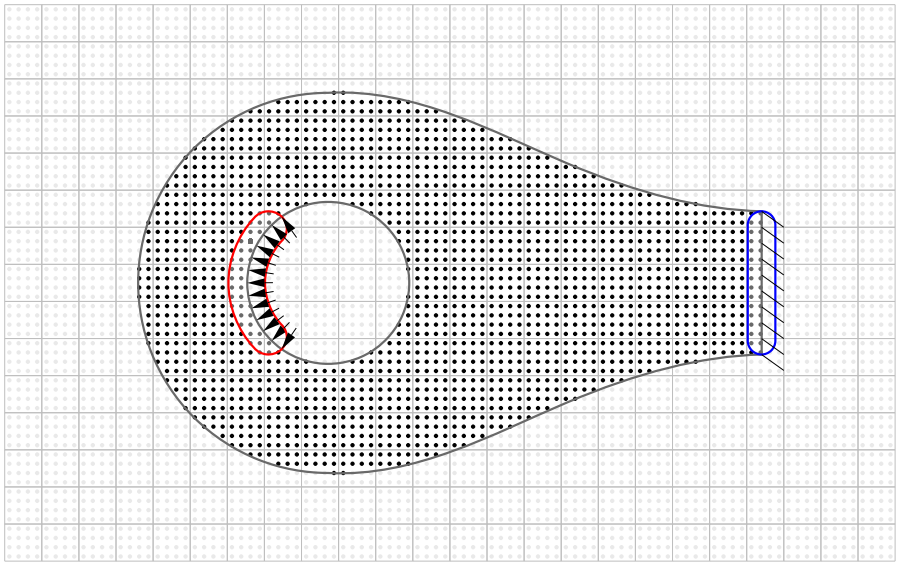


Figure 4.1: Illustration of *active* and *inactive* generated particles with Neumann and Dirichlet boundary condition regions depicted in red and blue respectively

Upon resetting the analysis, the program populates the whole canvas with inactive particles. All particles inside the boundary curve or outside any internal voids are then activated along with all connecting bonds. Active and inactive particles and bonds are then saved in separate lists so that the analysis methods easily can access only the active objects. For a more detailed description, see Algorithm 2 in Appendix B.

4.2.2 Application of Boundary Conditions

The boundary conditions are included in the simulation as outlined in Section 2.2.4. This formulation requires specification of the boundary segment Γ and a magnitude; either a prescribed traction or displacement. Γ is provided to a dedicated boundary condition component (see Table 4.2) along with the magnitude, type and an optional offset parameter. A boundary condition region is created based on the offset distance. This region contains particles that should be assigned the constraint. This region can either have rounded or sharp edges depending on the requirements. The direction of the constraint is by default the normal direction of the boundary. Practically this direction is set to the normal direction of the boundary at the closest point on the boundary. The constraint can be rotated by specifying an angle relative the normal of the boundary. This allows for instance the application of pure shear tractions. The depth the boundary regions are initially set to Δ and δ for Neumann and Dirichlet conditions respectively. For analyses using variable particle size there is an option to change this depth however since Δ has no physical relevance.

4.2.3 Updating the Design Domain

The design domain is defined using one boundary curve and optionally a set of void curves. There are three ways this domain can interactively be altered, see Figure 4.2. Firstly (**A**), by using the void curves to subtract from the domain defined by the boundary - hereby topology changes are possible. Secondly (**B**), by modifying the actual boundary - by using a NURBS curve as input, the control points can be moved and smooth shapes can be obtained. These options are available for geometry input from both Rhino and Grasshopper. The third option (**C**) is to remove the most unloaded particles based on a certain threshold of von Mises stress, see Table 4.5.

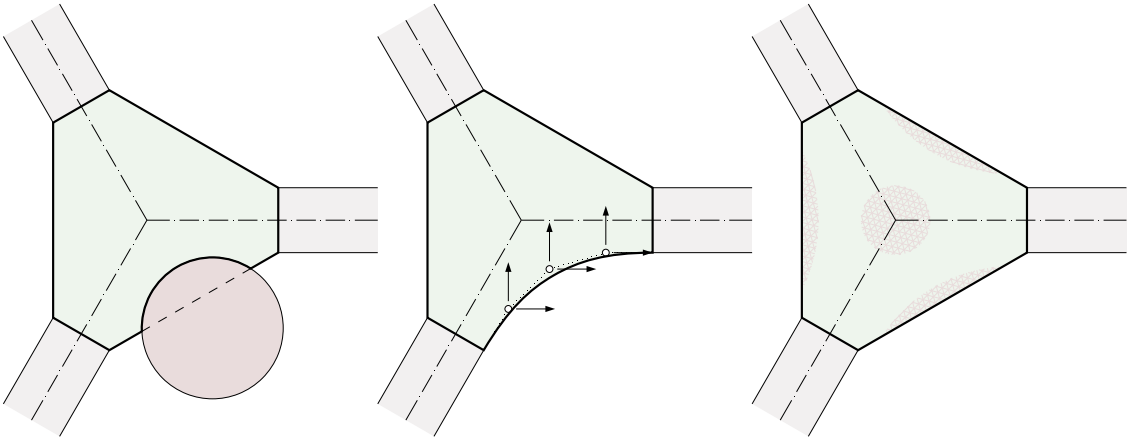


Figure 4.2: Methods for interactive modification of the design domain; (**A**) using voids, (**B**) modifying the boundary and (**C**) remove unloaded material.

When any of the input curves (boundary or voids) are changed, some particles and bonds should change state from being *inactive* to *active*, or vice versa.

Consideration must however be made with regards to the perturbation of internal force distribution that is caused by this. Similarly to how the boundary conditions are applied gradually (see Section subsection 2.2.4), also the activation and deactivation of particles is therefore done in a gradual manner to keep the simulation stable. This process is referred to as *fading*. Those particles that were previously inside and active but are now outside and should be inactive, are set to fading out and the fading counter for that particle is set to 1. In each iteration of the analysis thread this counter is decremented until reaching 0, whereby the state is changed to inactive - and analogously for the particles that are to fade in. After all changes of particle states are done, the bonds are looped over to find all bonds that should be faded in or out. The actual fading is achieved by scaling the stiffness of the bonds. For a more detailed description of the procedure used, see Algorithm 3 in Appendix B.

4.2.4 Particle Shaking

For certain geometries, a regular grid may introduce non-physical stress patterns. This problem is more prominent for a combination of using large particles and geometries with high curvature. There are several remedies for this, one of which is to generate particles based on offsets of the boundary. This approach may however lead to difficulties since the offset will intersect each other for most geometries. For this tool another approach is used. Rather than placing the particles based on the boundary, this approach is based on shaking particles that are placed in a regular grid. The algorithm used for the shaking is the same one used in [21]. This algorithm randomly moves the particles and updates the sizes based on an average distance to neighbouring particles.

4.2.5 Edge Refinement

The modified peridynamics approach allows for varying particle sizes and consequently particle concentrations, or *densities*. By making use of this feature the particle spacing can be made smaller along the boundary and where the stress concentrations can be expected to occur. Thus, it is possible to achieve more accurate results without a significant increase of the overall computational cost (as opposed to the case where the full domain would need a higher particle density). This method of modifying the particle distribution is henceforth referred to as *edge refinement*, and is conceptually shown in Figure 4.3.

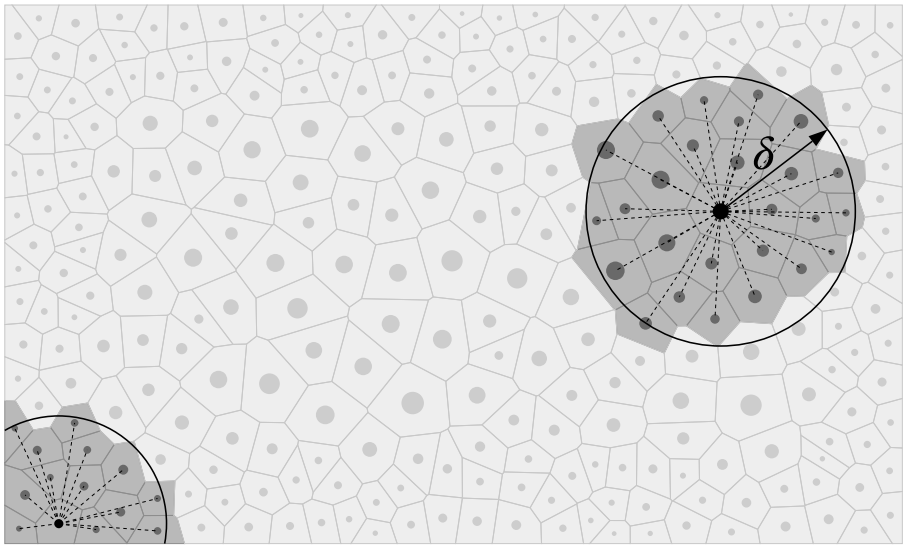


Figure 4.3: Non-uniform particle distribution with applied edge refinement; families for two chosen particles.

4.2.6 Outputs

Since the analysis is dynamic, the stress field will be updated during the transient phase, and after some time it will converge to the steady state stress field for the static solution. The user should be able to remove (or add) regions of the domain during runtime, so the tool must continuously inform the designer of which regions are less loaded. An intuitive understanding of this can be gained either by looking at the von Mises effective stress trajectories, stress components from the Cauchy stress tensor or the principal stress lines - all of which should be possible to display in the Rhino viewport. Furthermore, the arm tensions in the bonds give a clear picture of what forces act on a certain particle on a local level, and these should also be able to display (as compression or tension bonds).

Moreover, the geometry (zones, particles, bonds and principal curvature lines, that is) along with associated colours should be possible to export to Rhino, in order to save and plot the results. Finally, it should be possible to extract data from the analysis, such as geometry and results. These results can be used for further analysis or to validate the result.

4.3 Zoning and Restricted Particle Search

Since it should be possible to update the outer curve and interior voids with little to no respond time, it is of great importance to search for updated particles in only a close vicinity to the changed curve. Each zone is labelled *inside* or *outside* if the zone is completely inside or outside the outer boundary and completely inside or outside all void curves. If the zone is partially inside and partially outside it is labelled *on edge*, depicted in grey in Figure 4.4.

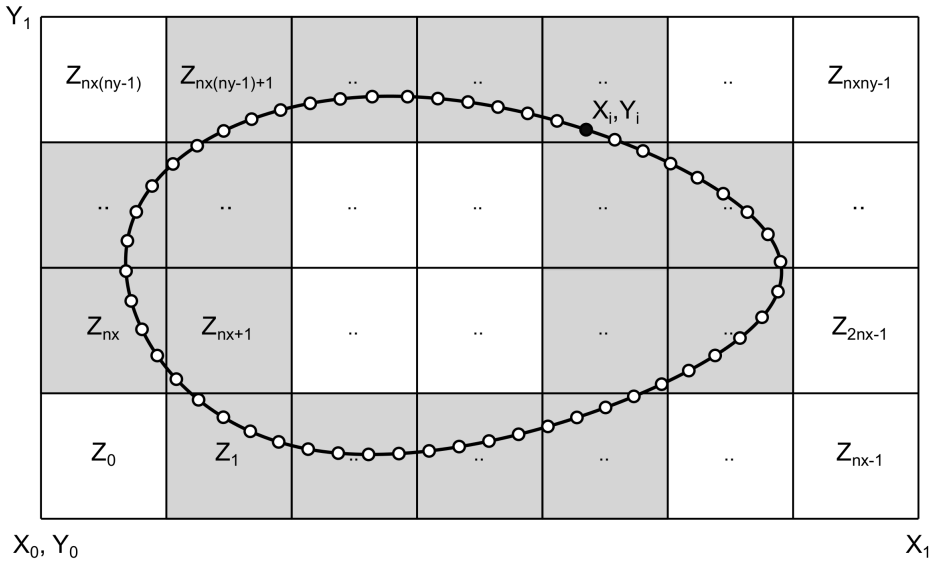


Figure 4.4: Division of boundary curve and corresponding edge zones.

Once a new curve is detected it is divided into small segments. The parent zone Z of each end point of these segments is found using the following mapping

$$Z = \left\lfloor \frac{(X_i - X_0)}{X_1 - X_0} \right\rfloor n_x + \left\lfloor \frac{(Y_i - Y_0)}{Y_1 - Y_0} \right\rfloor n_y n_x, \tag{4.1}$$

where $\lfloor \bullet \rfloor$ is the floor operator, n_x and n_y are the number of zones in x and y direction respectively. Once all edge zones are found, the *Contains* method in the Rhino Common library is applied to the midpoint of the remaining zones to decide if the zone is *inside* or *outside* the boundary curve.

4.3.1 Updating Particle and Bond States

When any of the curves are changed some particles and bonds should change state from being inactive or active to being fading in or out. To avoid checking all particles and bonds, only the zones that have changed inclusion are checked. Those particles that previously were inside but now are outside are set to fading out and the fading counter for that particle is set to 1. In each iteration of the analysis thread this counter is decremented until reaching 0 whereby the state is changed to *inactive*. The actual fading is achieved by scaling the stiffness of the bonds. After all changes of particle states are done the bonds are looped over to find all bonds that should be faded in or out. For a more detailed description of the algorithm used, see Algorithm 3 in Appendix B.

4.4 Software Design

The software is written in the object-oriented language (OOP) C#. The Rhino common API is used for interacting with the Rhino CAD environment, but also as a geometry library for the peridynamic model.

4.4.1 Structuring the Code Using Objects

The key idea of the OOP paradigm is to assign functionality and properties to objects and then specify how these objects interact with each other. In C# the objects are implemented as classes and the functionality as methods. This abstraction makes it easy to model complex interactions since the implementation of certain functionality is internal to the classes. Inheritance can be used when classes share many methods and properties. By putting the common features in a base class and letting the sub classes implement that parent class the common implementation only has to be done once. Classes can be inherited from both abstract and non-abstract base classes, with the difference that the abstract only acts as a template for the child class and cannot itself be instantiated. An abstract class may contain both abstract and non-abstract methods. The non-abstract methods are used for implementing the common functionality of all inherited classes. On the contrary, the abstract methods merely specify what functionality must be implemented in the child classes.
Each physical object in the peridynamics model is represented as an object. Inheritance has been used to make the software easy to extend to 3D. Methods for generating zones and particles in a 3D setting differ from those implemented for 2D. This can in large be attributed to the fact that the geometry is based on surfaces rather than curves. Most other functionality are however shared and is gathered in base classes. Dynamic polymorphism enables the child classes to override methods in the base class. The program can then decide at run-time which child class the instance belongs to and apply the appropriate method. In C# this is done by declaring the base class method as virtual and then overriding the method in the child classes by using the override modifier.

4.5 Partitioning the Code Using Modules

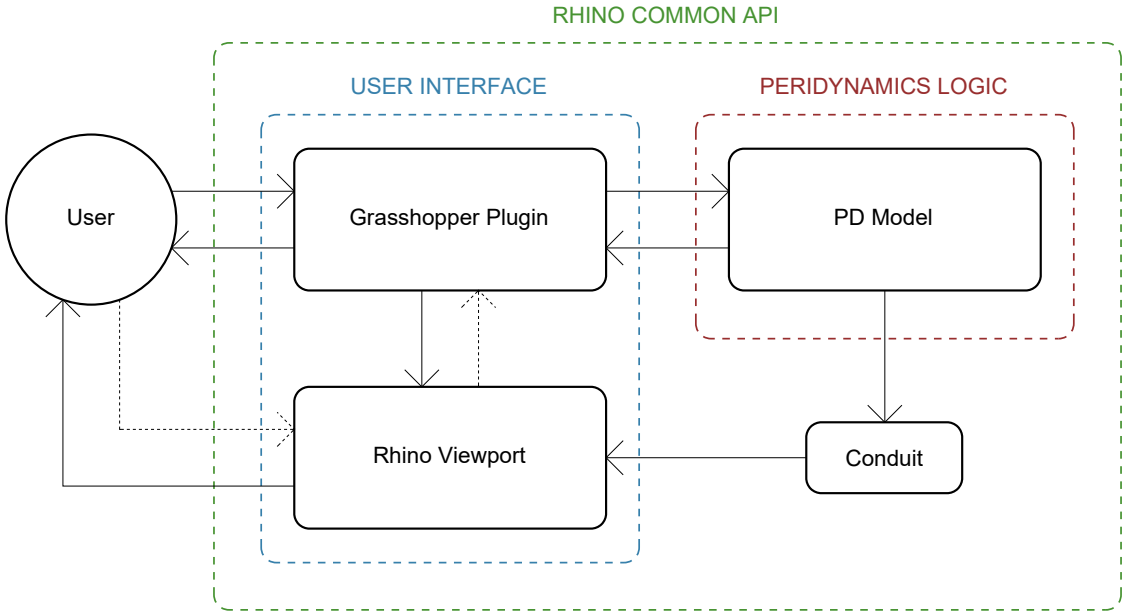


Figure 4.5: Overall software architecture showing how the high-level modules are interacting with each other and the user.

The software is divided into two main modules. The interface module is managing the interaction with Grasshopper and Rhino and the peridynamics model is managing the physics of the peridynamic model. The user can control the simulation by providing necessary information to the Grasshopper components. Based on the information passed to the solver component, appropriate methods in the peridynamics model are executed. The result is then either passed to the Rhino viewport, via the conduit class, or passed to the solver component as a result object. The user can then specify which data to output in the Grasshopper environment by using the results component. In Figure 4.5 the high-level structure is illustrated, note that *Conduit* is a class used for interacting with the Rhino viewport.
In Figure 4.6 the relationships of the classes are illustrated - for a more detailed diagram see appendix Appendix A. As described above the physical model and the user interface are decoupled as to increase the modularity of the code. The interaction between the two modules is handled by the *Main* class. In the next two sections the two modules are described in more detail.

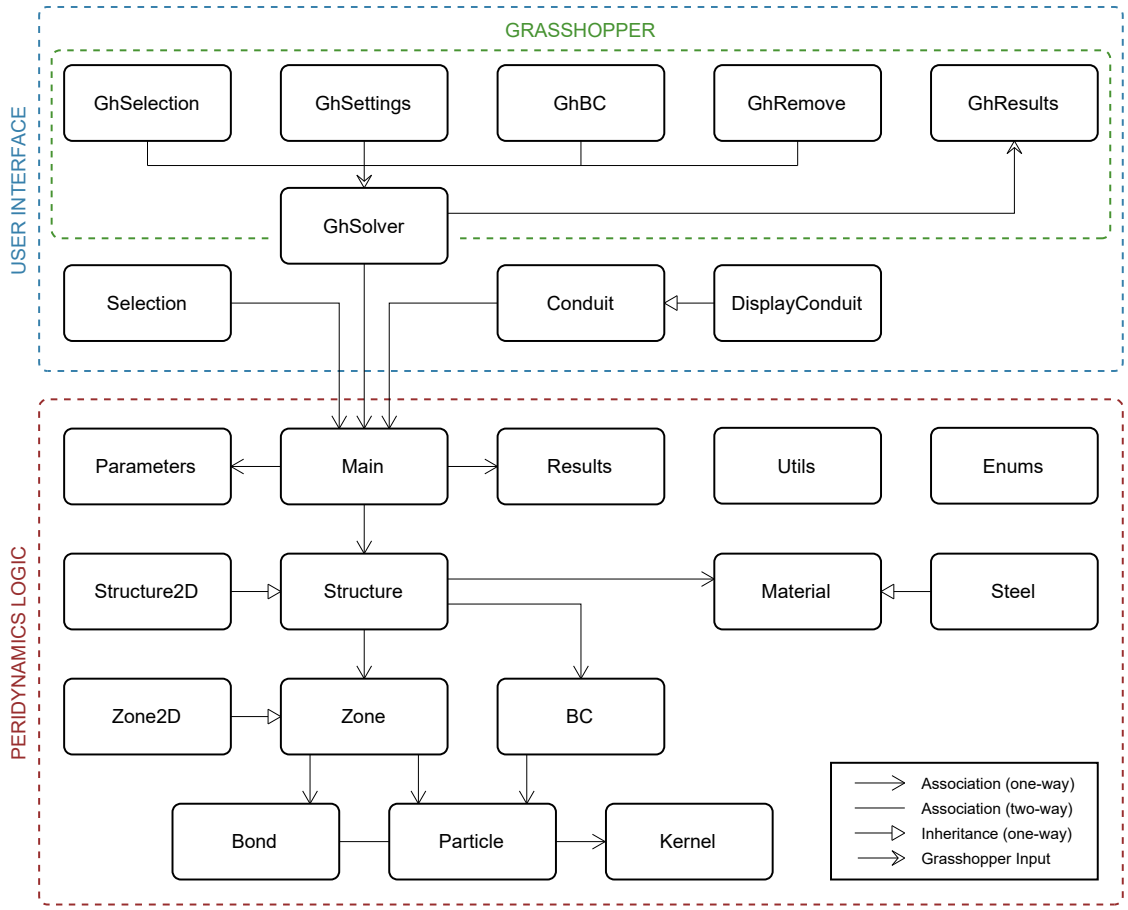


Figure 4.6: Conceptual class diagram showing the overall code structure.

4.5.1 Peridynamics Model Module

The peridynamics logic is contained in a model module, inside the namespace *peridynamics.Model*. The top-level logic in this module is contained in the main class. This class also acts as the bridge to the user interface model. The *Main* class contains methods for setting up and running the analysis. The *Structure* class manages the physics of the analysis and owns a list of *zone* objects. The zone objects are holding lists of *particle* and *bond* objects and methods that act on them. Since the calculations in each time step are performed on both bonds and particles that are completely active and those that are fading in and out, the Structure class holds a list of active bonds and active particles that are updated from the zones lists. This way the program does not need to check for active particles in each time step. When the boundary or void curves are changed these lists get updated according to Algorithm 3 in Appendix B.

The Particle class owns a *Kernel* class that represents the kernel described in Equation 2.17. For the assumption of small deformations, this is only evaluated once and the result is saved in each particle object. The boundary conditions are handled in the *BC* class. Two sub classes inherit from this class, namely *BCDirichlet* and *BCNeumann* which contains specific logic for handling Dirichlet

and Neumann type boundary conditions respectively.

In each time step of the analysis the state of the model is updated using the time stepping scheme described in Section 3.1.1. This includes extrapolating the velocity forward in time by half a time step and using this to estimate the displacement a complete time step forward in time. The next operation is the most computationally expensive in the entire simulation. This is the evaluation of the internal particle forces, caused by the displacement. Since this is to be done on the CPU it must be done sequentially, or divided into just a few parallel threads. For ease of implementation a fully sequential execution is used, see Algorithm 1. Since the average strain of each particle is needed all bonds needs to be looped over one time before the actual force is calculated.

Algorithm 1: Computation of internal force

input : Particle displacements $\boldsymbol{u}_i$
output: Particle internal force $\boldsymbol{f}_i$
for *Bonds in active bonds* **do**
  Calculate the arm strain ϵ_{ij};
  Add the weighted strain contribution to each end particle along with the weighted sum;
end
for *Particle in active particles* **do**
  Divide the strain contributions with the weighted sum according to 2.24;
end
for *Bonds in active bonds* **do**
  Calculate the force flux densities S_i and S_j based on the constitutive model in 2.23;
  Calculate the arm tension T_{ij} and scale it if the bond is to be faded in our out, finally add directionality by multiplying with $\boldsymbol{q}_{ij}$;
  Add the contribution to each end particle, see 2.13;
end

4.5.2 Interface Module

The interaction with the model is contained in a module inside the namespace *peridynamics.Interface*. This module contains the logic for the components that are exposed in Grasshopper. The classes that control the components are denoted with the prefix Gh and inherits from the abstract class *GH_Component*. These classes own methods that control inputs and outputs of the components as well as what is executed when new input is provided. For displaying geometry in the Rhino viewport the class *Conduit* is used. This class inherits from *DisplayConduit*, which is a class provided by the *Rhino.DocObjects* namespace for drawing geometry in Rhino. Finally, to keep track of what should be displayed the Conduit class owns a *Selection* class which holds information about which particles, bonds and zones that currently should be displayed.

4.5.3 Integration in Grasshopper

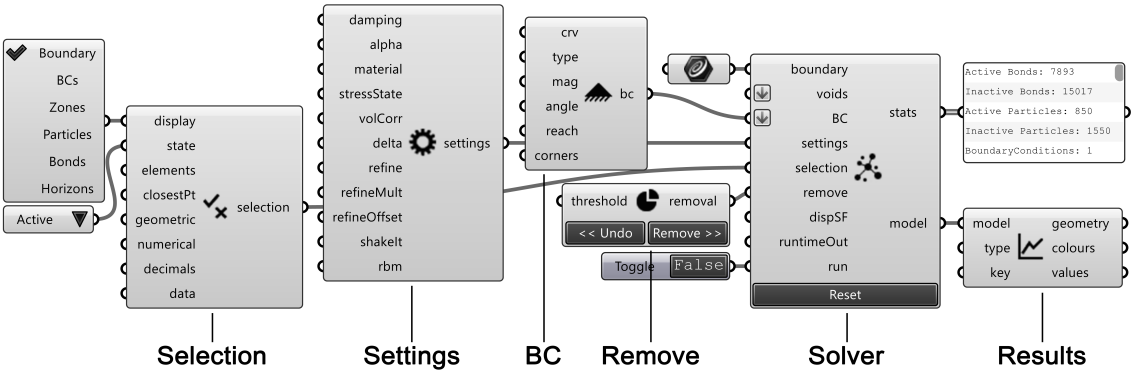


Figure 4.7: Grasshopper plugin components (example setup).

The code is compiled into a Grasshopper assembly file (.gha) and exposed as a toolkit consisting of six components, as seen in Figure 4.7. The core functionality of the software is exposed in the form of a solver component (*Solver*). To that comes four input components (*BC*, *Selection*, *Settings* and *Remove*) and one output component (*Results*). Below, the Grasshopper components are presented one by one.

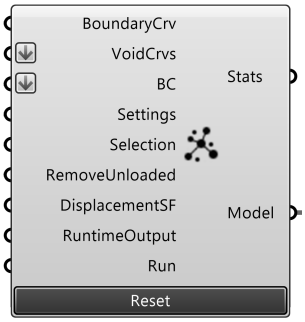
Solver	**Description**
	*Solver* is the executing component within which the whole simulation runs. It displays geometry dynamically based on *Selection*, returns statistics and relevant values during runtime, and generates a model from which relevant results can be obtained with *Results*.

Table 4.1: The Solver Component

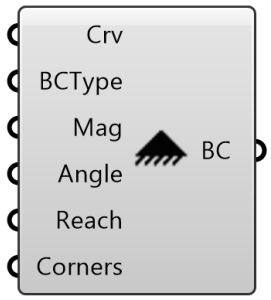
BC	**Description**
	*BC* is short for Boundary Conditions, and this component provides required input to *Solver*. There are two types of BCs, namely prescribed displacement and prescribed traction. Both are defined with curves that are offset to create boundary regions to which particles are assigned.

Table 4.2: The BC Component

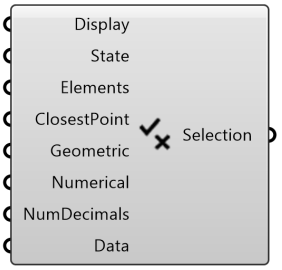
Selection	Description
	*Selection* dictates which geometry, colour plots and numerical values that are to be displayed on the Rhino canvas. It also regulates what timestep data to write out in the Rhino command window.

Table 4.3: The Selection Component

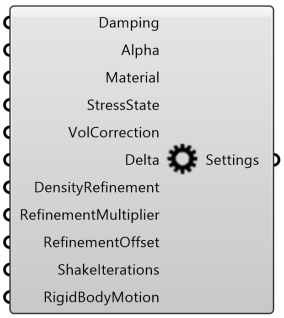
Settings	Description
	*Settings* provides optional input parameters to *Solver*. Maybe most importantly, it regulates the number of particles, and thereby the accuracy of the analysis. Changes to this component requires a reset.

Table 4.4: The Settings Component

Remove	Description
	*Remove* gives the user the opportunity to remove the least loaded particles based on a certain threshold - and undo the move if the result is not desirable. It is an optional input to *Solver*.

Table 4.5: The Remove Component

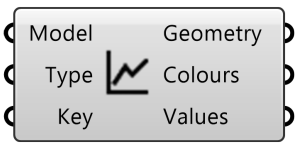
Results	Description
	*Results* selects which output should be mined from *Solver*. Geometries and corresponding colours can be obtained and baked into Rhino, and numerical values can be extracted for all individual elements.

Table 4.6: The Results Component

5
Case Studies

At this stage it remains to validate and show the potential uses of the tool. The validation is done both against analytic and finite element solutions. Once the performance is shown to be satisfactory the tool is applied to the design of truss connections and discontinuity regions.

5.1 Validation

The validation is first done by comparing the displacement field against an analytical solution. Then a validation is done by comparing the displacement and stress field with results obtained from a finite element model.

5.1.1 Isotropic Plate Subjected to Uniaxial Tension

To verify that the modified peridynamics theory is implemented properly and works as expected, the plugin has been used for recreating a benchmark problem presented in [17] for classical peridynamics. Here, the particle grid is uniform and the particles have the same sizes, making it a special case for the modified theory. Further validation is needed to assure that a setup with non-uniform grid and different particle sizes works as intended. The results are compared to analytical results.

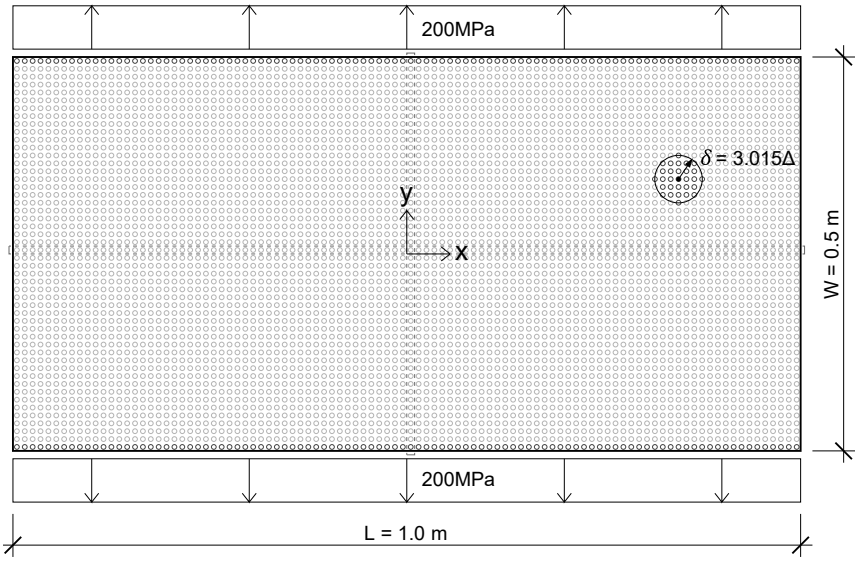


Figure 5.1: Geometry and discretization of the plate (5000 particles).

The test geometry is a rectangular plate subjected merely to uniform uniaxial tension applied on the top and bottom edges. The geometry is presented in Figure 5.1 and in Table 5.1 some of the central properties and parameters are listed - for a more thorough account of the setup, we refer to Chapter 8.3 in [17].

Table 5.1: Parameter setup for case study

Material Parameters		
Young's modulus E	Poisson's ratio ν	Density ρ
200 GPa	1/3	7850 kg/m^3
Loading Conditions		
Displacement constraints	Applied uniaxial tension σ	Analysis type
None	200 MPa	*Plane stress*
Peridynamics parameters		
No. particles in x / y / z	Particle spacing Δ	Horizon δ
100 / 50 / 1	0.01 m	3.015 Δ

In Figure 5.2, the bonds between the particles are displayed. The colours (blue for compression and red for tension) show that the behaviour is as expected for the case of uniaxial tension - the vertical bonds and those with large angle to the horizontal axis are subjected to tension, while the horizontal bonds experience compression as the plate gets contracted.

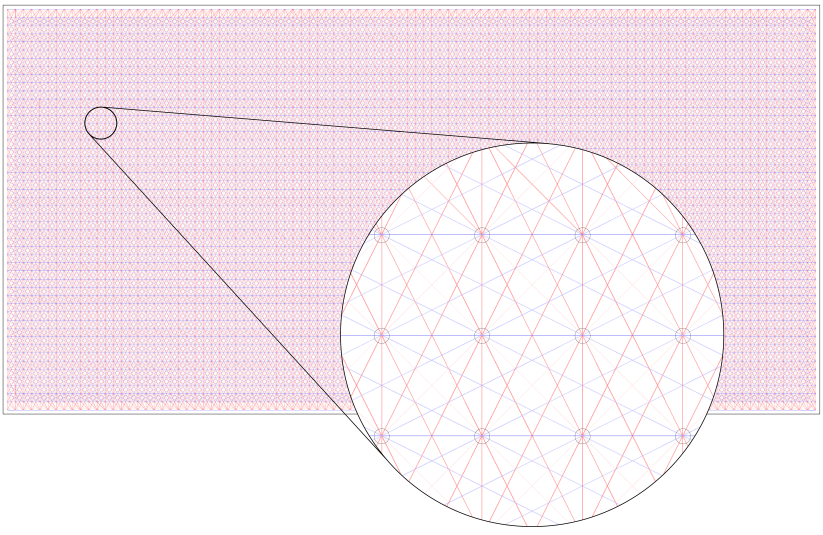
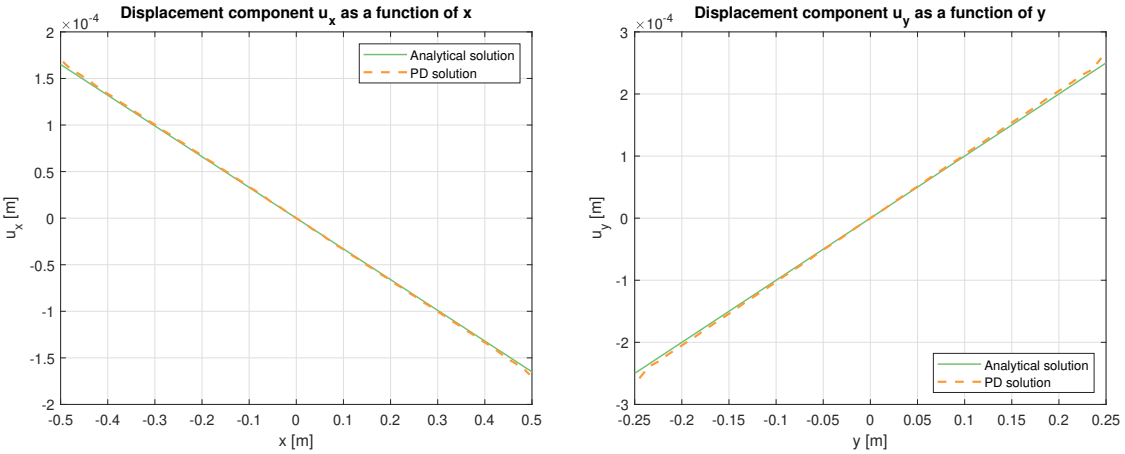


Figure 5.2: Bonds: red for tension, blue for compression (115536 bonds in total).

Figure 5.3 shows two graphs comparing the horizontal and vertical displacements obtained with the plugin to the analytical solutions. The plotted values agree well

to the values measured at the horizontal and vertical center lines of the plate. Furthermore, comparing the displacement in the x-direction with that of the y-direction, it can be concluded that the strains agree with the set value of Poisson's ratio.



(a) Horizontal displacement for y = 0m. **(b)** Vertical displacements for x = 0m.

Figure 5.3: Comparison between analytical and numerical results.

5.1.2 Triaxial Plate Subjected to Compression

To verify that the plugin works for more advanced non-orthogonal load cases, as well as for displacement boundary conditions, a second validation is performed. Here the results are compared to a those produced by a finite element model created in Abaqus, see Figure 5.6a. The geometry of the plate is a truncated equilateral triangle, which is subjected to compressive forces perpendicular to two of the truncated corners, with the third one being fixed. If you like, imagine this to be the design domain for a node with three bars coming together in a truss joint. The key parameters are presented in Table 5.2 and the geometry in Figure 5.4.

Table 5.2: Parameter setup for case study

Material Parameters		
Young's modulus E	Poisson's ratio ν	Density ρ
200 GPa	1/3	7850 kg/m^3
Loading Conditions		
BC1/BC2 applied pressure σ	BC3 displacement	Analysis type
100 MPa	$u_x = 0; u_y = 0$	*Plane stress*
Peridynamics parameters		
Number of particles	Particle spacing Δ	Horizon δ
10630	0.05 m	2.9 Δ

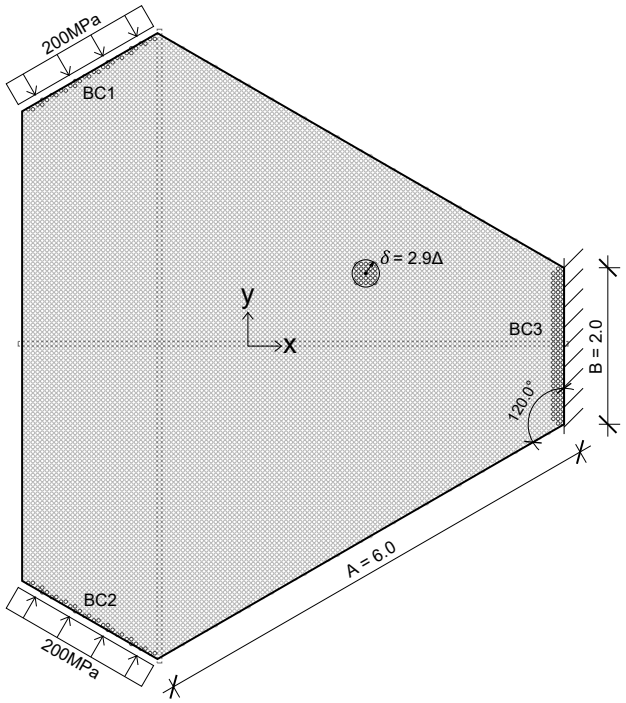
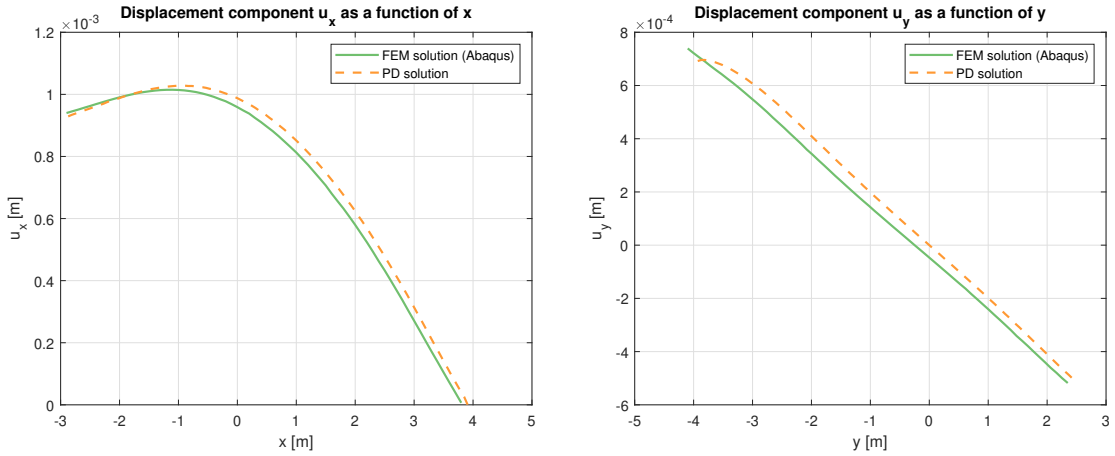


Figure 5.4: Geometry and discretization of the plate (10630 particles).

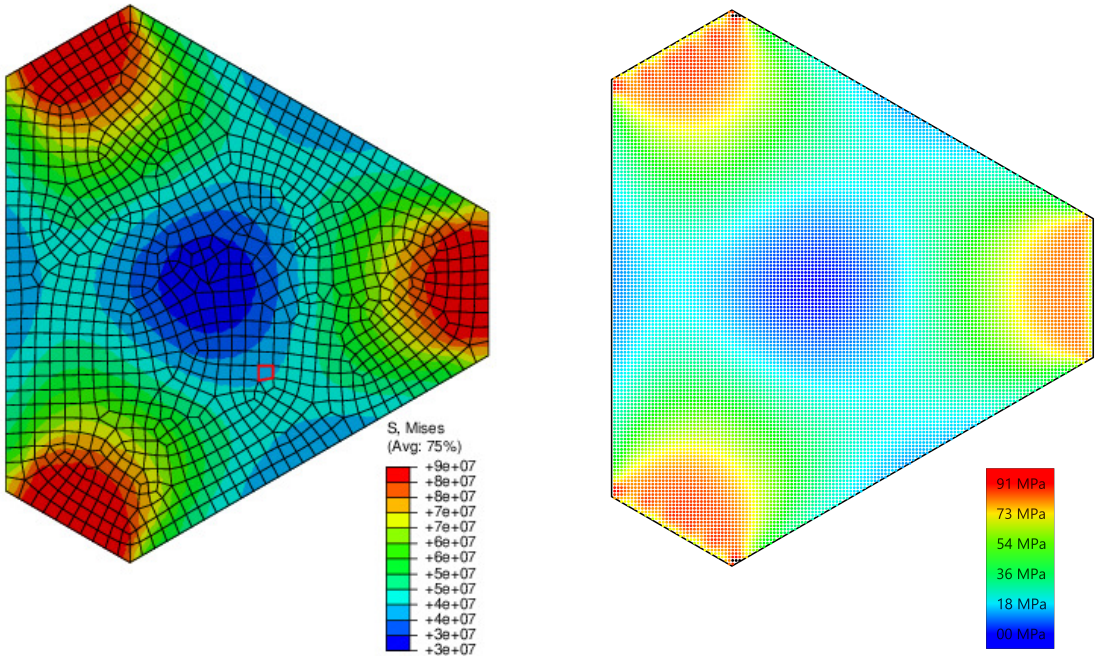
From the graphs in Figure 5.5 it can be concluded that for the results of the plugin agree well with the results from the finite element analysis. One can however note that the displacement results from the plugin are somewhat larger in general. There are multiple possible explanations for this, of which the most probable is combination of the higher resolution of the PD model and the kinematic constraints of the FE model.



(a) Horizontal displacements for y = 0m. **(b)** Vertical displacements for x = 0m.

Figure 5.5: Comparison between displacements obtained using Abaqus and the developed plugin, respectively.

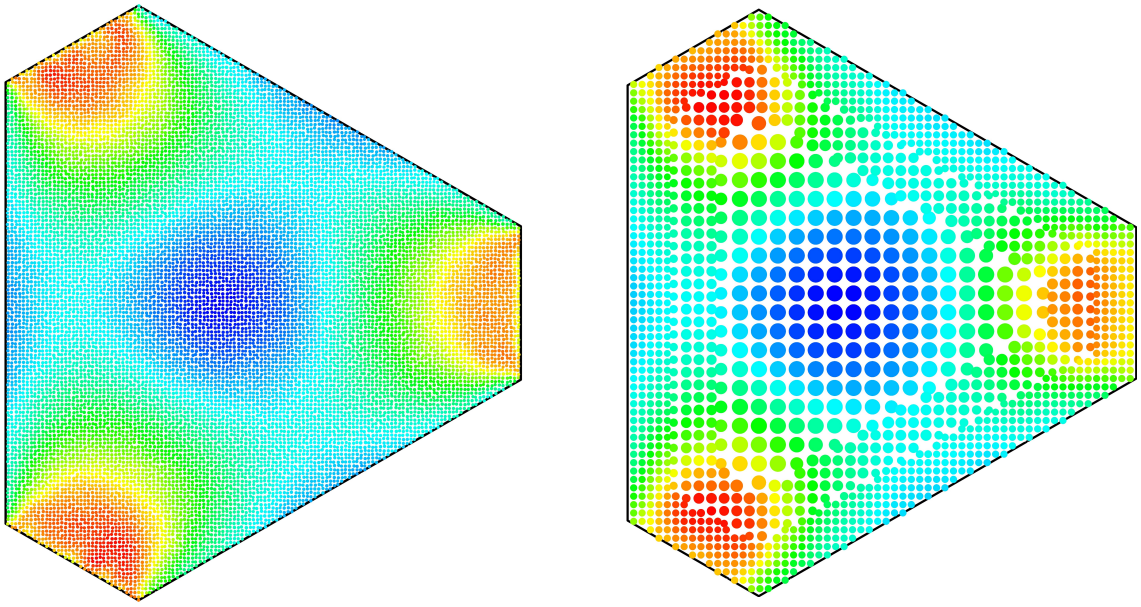
For the validation of the von Mises stress distribution, the rightmost displacement boundary condition is replaced with the same traction boundary condition as the other two boundaries. Since the tractions are self-equilibrated, no displacement conditions are needed on the contrary to the finite element simulation to the left in Figure 5.6a. However, due to the slight asymmetry that is caused by using a uniform orthogonal particle grid on a non-orthogonal shape, there would be a risk of rigid body rotation. This effect is decreasing with increasing number of particles, as is therefor negligible for this setup.



(a) Abaqus result **(b)** Plugin result

Figure 5.6: von Mises stress plots generated with FEM and peridynamics.

Figure 5.6 shows a good agreement between the von Mises stress distributions obtain from Abaqus and the plugin, as well as between the maximum values. In Figure 5.6b, however, the undesired artefacts of using a uniform grid for a non-aligned geometry can be distinguished by comparing the stress concentrations that appear by the non-orthogonal boundary regions to that of the rightmost one. By using Laplacian smoothing, this effect has been significantly reduced, but it is still visible. It could be further reduced by using a non-uniform particle distribution as in Figure 5.7a. The computational cost can also be substantially reduced by using edge refinement, see Figure 5.7b.

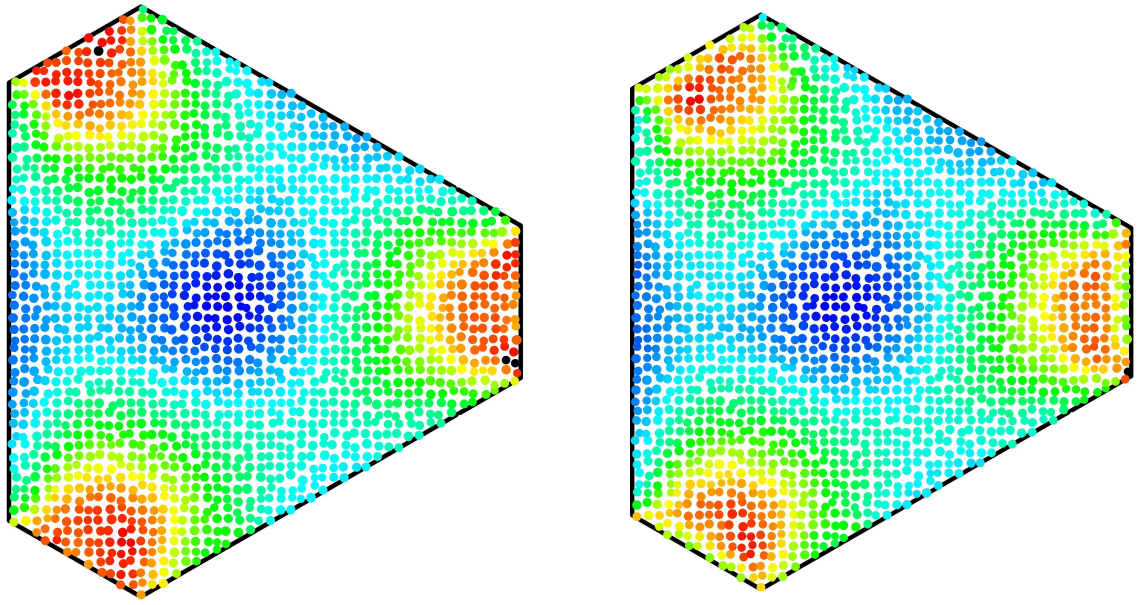


(a) Non-uniform particle distribution.

(b) With applied edge refinement.

Figure 5.7: von Mises stress plots generated with the plugin.

Although the results in Figure 5.6 are promising, it could be improved by applying weights to the least square approximation as described in Section 3.2.1. The effect is briefly investigated by simulating the von Mises stress in the triangular plate. First without weighting the particle contributions using the kernel (Figure 5.8a) and then using weighting the particle contributions (Figure 5.8b). For this specific case the stress at the edges where the loads are applied is poorly captured when weighting the particles. On the other hand are some of peak values eliminated.



(a) Without particle weighting.

(b) With particle weighting.

Figure 5.8: Comparison of von Mises stress field when using particle weighting.

5.1.3 Numerical Stability

Two different approaches for finding a stable combination of mass and time step where discussed in Section 3.1.2. Since the time step derived solely from estimating the speed of sound is independent of the mass, manual calibration of the fictitious mass is needed. The critical mass derived from the highest eigenvalue is on the other hand based on the time step and as a consequence, no manual calibration is needed. Both conditions have been tested, of which the latter is implemented in the tool for the reason stated above. This section aims at verifying that the derived stability criteria in Equation 3.9 results in stable time stepping, even when geometry and settings are changed. Two different setups are considered, the rectangle in Section 5.1.1 and the truncated equilateral triangle in Section 5.1.2. The relative margin to instability is plotted in Figure 5.9 for increasing particle size. In the graph Δ is the particle grid size, Δt the time step and Δt_{crit} the time step for which the solution becomes unstable.

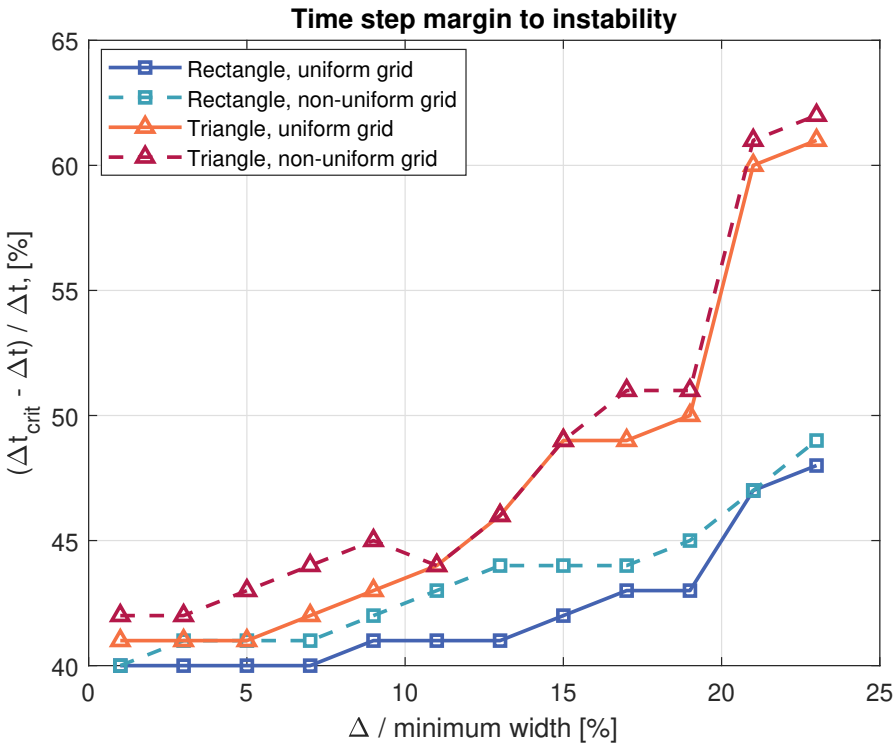


Figure 5.9: Time step size margin to instability for increasing particle size, here expressed by normalizing to the smallest width of the body. The result is plotted both for the uniaxial plate in Section 5.1.1 and the truncated equilateral triangle in Section 5.1.2.

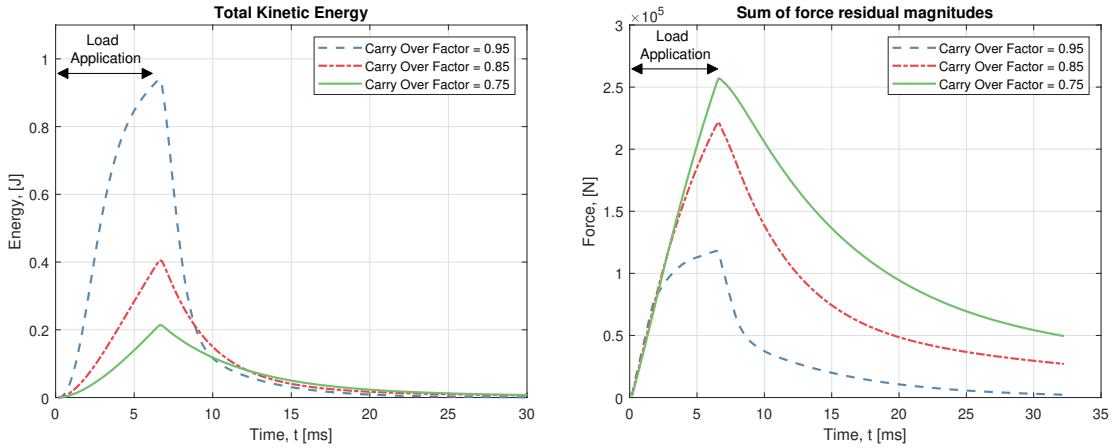
Judging from the results, it seems like the margin converges to around 40%, which is a somewhat conservative figure, indicating that the time step can be scaled up without reaching instability. Based on this, the computed time step is multiplied with a factor of 1.2 in the implementation, as a means of speeding up the convergence.

5.2 Convergence Studies

Both convergence in the time domain and in terms of particle size are investigated. In particular, the convergence rate of displacements components as well as energy measures are qualitatively assessed.

5.2.1 Quasi-Static Convergence

To assess which convergence measure is most appropriate, a comparison is made between the convergence of the total kinetic energy and the sum of all residual force magnitudes in the body. Also different values of the carry over factor used for damping are compared to investigate how the damping is affecting the speed of convergence. The setup used to conduct this study is the same as in Section 5.1.1.



(a) Total kinetic energy. **(b)** Total residual force.

Figure 5.10: Comparison of convergence measures.

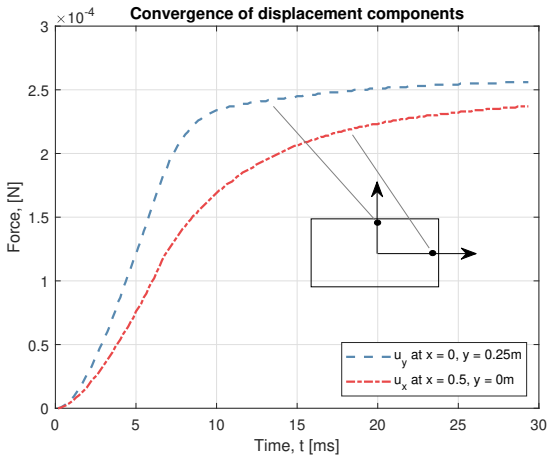


Figure 5.11: Convergence of displacements for two particles.

More insight into the dynamic behaviour of the system can be obtained by studying the convergence of the displacements in two points in the plate, see Figure 5.11. Also,

this is a measure that is directly related to the structural evaluation, which means that the convergence of the chosen measure and the convergence of the displacements must correlate well. The two particles are chosen with regard to the symmetry of the plate; the top one is displaced only in the y-direction and the right one only in the x-direction - see Figure 5.11. As the force is applied at the top and bottom edges in the vertical direction, the top particle is immediately displaced in the same direction, resulting in a clear change when the load application is done (blue curve). In contrast, the mid particle is displaced only in the transverse direction, which is a more gradual and delayed process, as is indicated by its smoother sloop (red curve).

5.2.2 Influence of Particle and Horizon Size

In order to ensure that the plugin behaves as expected, i.e. gives a higher accuracy for a decrease of particle size, a cantilever beam subjected to a vertical point load at the tip is examined (see Figure 5.12 and Table 5.3). Furthermore, the effect of different horizon sizes is included in the analysis. The numerical tip displacement is compared to the analytical solution using Euler Bernoulli beam theory, where

$$u_{analytical} = \frac{FL^3}{3EI}. \tag{5.1}$$

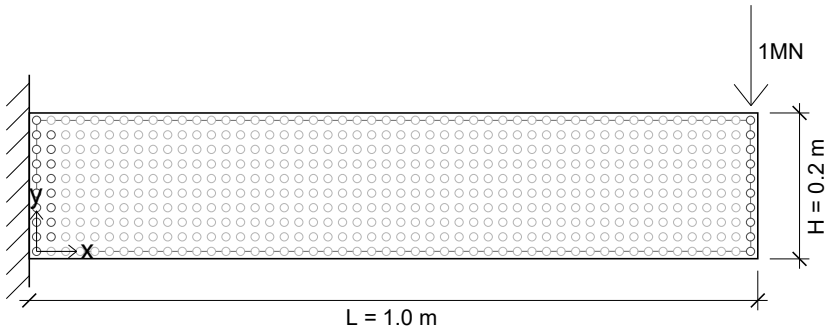
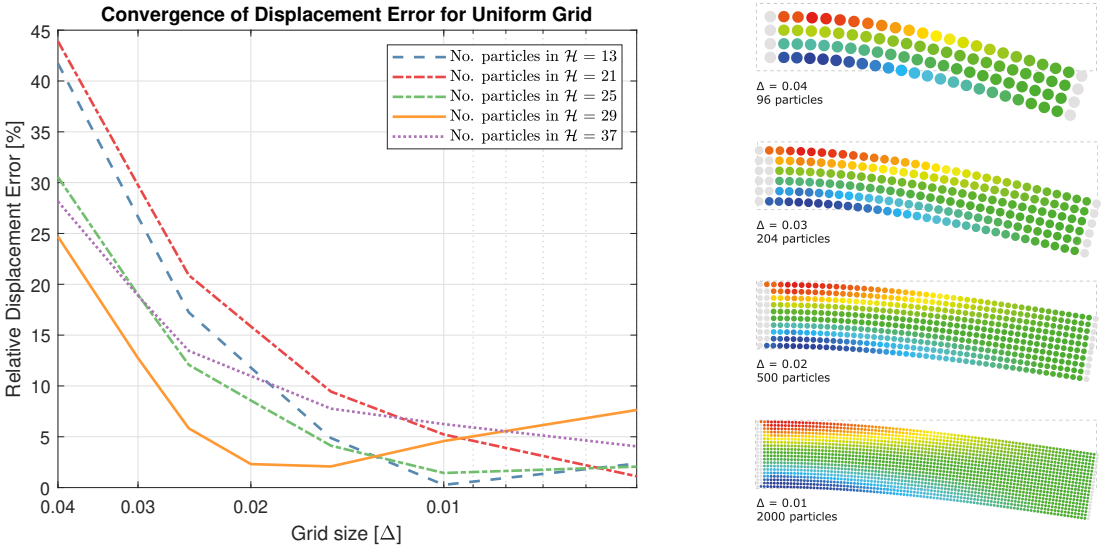


Figure 5.12: Geometry of cantilever beam used in convergence study ($\Delta = 0.02m$).

Table 5.3: Parameter setup for case study

Material Parameters		
Young's modulus E	Poisson's ratio ν	Density ρ
200 GPa	1/3	7850 kg/m^3
Loading Conditions		
Applied force N (right)	Displacement BC (left)	Analysis type
1 MN	$u_x = 0; u_y = 0$	*Plane stress*
Peridynamics parameters		
Number of particles	Particle spacing Δ	Horizon δ
10630	$0.04 - 0.005$ m	$2 - \sqrt{10}\ \Delta$



(a) The tip displacement error as a function of grid size; the error normalized to the analytical solution.

(b) Scaled up deformation and σ_{xx} for different Δ.

Figure 5.13: Convergence study for different particle densities and horizons for a cantilever beam with a uniform particle distribution.

Figure 5.13a shows that the accuracy overall correlates well with the fineness of the grid as one would expect. The values plotted are the absolute values of the relative displacement error, and the reason for some of the graphs starting up again is that the peridynamic beam then deflects more than the Euler Bernoulli beam model. Also this result is reasonable since the Euler Bernoulli beam theory generally overestimates the bending stiffness, especially for high beams. The magnitudes of the values are less important than the convergence of the results. However, for most of the chosen horizons the error is only a few percent, which is well within the limit for what is required in an early design stage and on the conservative side.

The horizon that corresponds to a family of 29 particles yields by far the smallest errors for large grid sizes, and gives the highest deformation and thereby the most conservative results for small grid sizes. It cannot be ruled out that this is not merely a case specific tendency, but further studies would need to be conducted to confirm this. Nevertheless, based on these results, the family size of 29 has been set to the default for a uniform grid.

From Figure 5.13b it is apparent that a higher particle density better fills out, and thereby models, the desired geometry. This is an artifact of the particle creation algorithm, which is zone based rather than boundary based. However, the height and length of the analytical beam has been parametrically adapted to that of the particle grid for all analysed particle distributions, meaning that the overall results are still valid although the actual beam measures vary with the grid size.

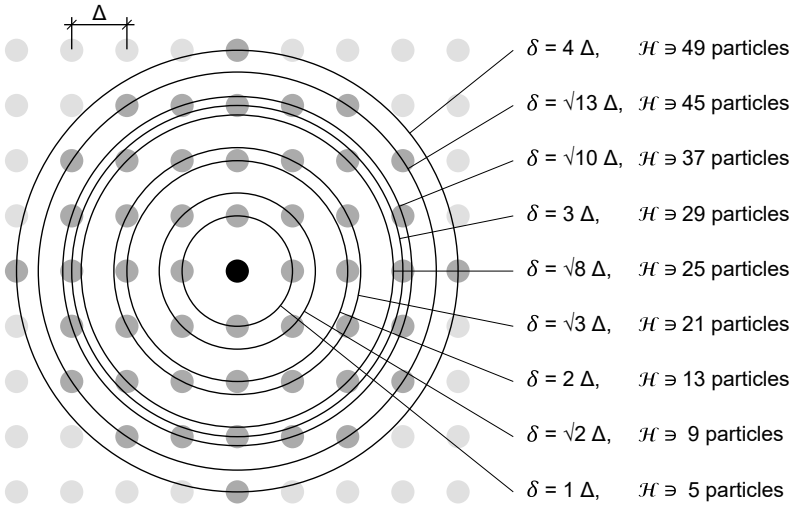


Figure 5.14: Family sizes depending on horizon for a uniform grid.

Due to the symmetry of the uniform grid, there are some clear discrete horizon intervals which yields different family sizes. Figure 5.14 shows these thresholds up to $\delta = 4\Delta$, of which the five middle ones were used in the convergence study (see Figure 5.13a). Thus, to obtain a family size of 29 for a uniform grid, the default value for the horizon has been set to 3Δ. This choice is further motivated by the fact that this number of particles fill a circular domain well. For a random particle distribution, the thresholds follow a smoother pattern with many intermediate layers. Additional convergence studies would be needed to study the effect of the horizon size for such a setup.

5.2.3 Influence of Edge Refinement

For domains with non-orthogonal boundaries, particle packing using a rectangular grid will give poor numerical results due to surface effects and deeper boundary condition regions. As a case study the influence from having a finer particle distribution close to the boundary is investigated. For the circular plate depicted in Figure 5.14, subjected to a prescribed radial displacement Δr, it is apparent that a coarse grid at the boundaries will give a poor approximation of the true axis-symmetric behaviour. By comparing the error and computational time of a coarse and a fine regular grid to a non-uniform grid with the fine particle grid on the edge and the coarse grid on the interior, the advantage of edge refinement is assessed. As a measure of the performance the radial displacement u_r in all non-prescribed particles is computed and compared to the analytical solution

$$u_r(r) = \frac{pr(1-v^2)}{E(1+v)}, \tag{5.2}$$

where p is the corresponding pressure caused by the prescribed displacement Δr and r is the radial coordinate. The relative error is defined as the difference of the displacement normalized to the analytical displacement. Also, a plane stress assumption is made and no volume correction is used.

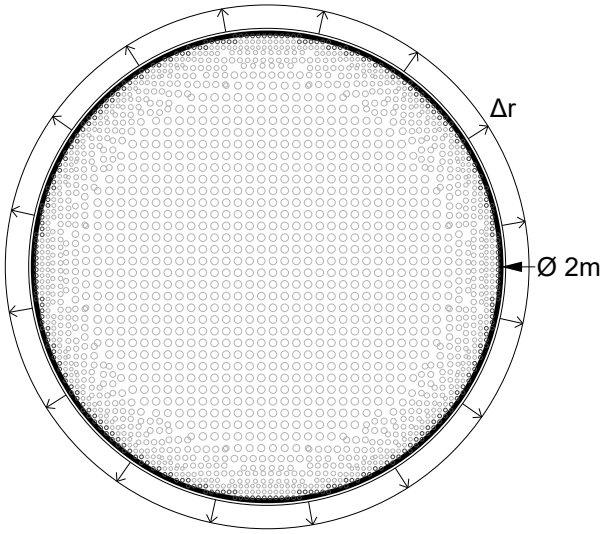


Figure 5.15: Setup of the circular plate.

Parameter	Value
Young's modulus	200 GPa
Poisson ratio	1/3
Refinement offset	4 Δ
Refinement steps	3
Carry over factor	0.95
Δ	15-80 mm
Δr	0.67 mm
α	5.1

Table 5.4: Parameters used in the assessment

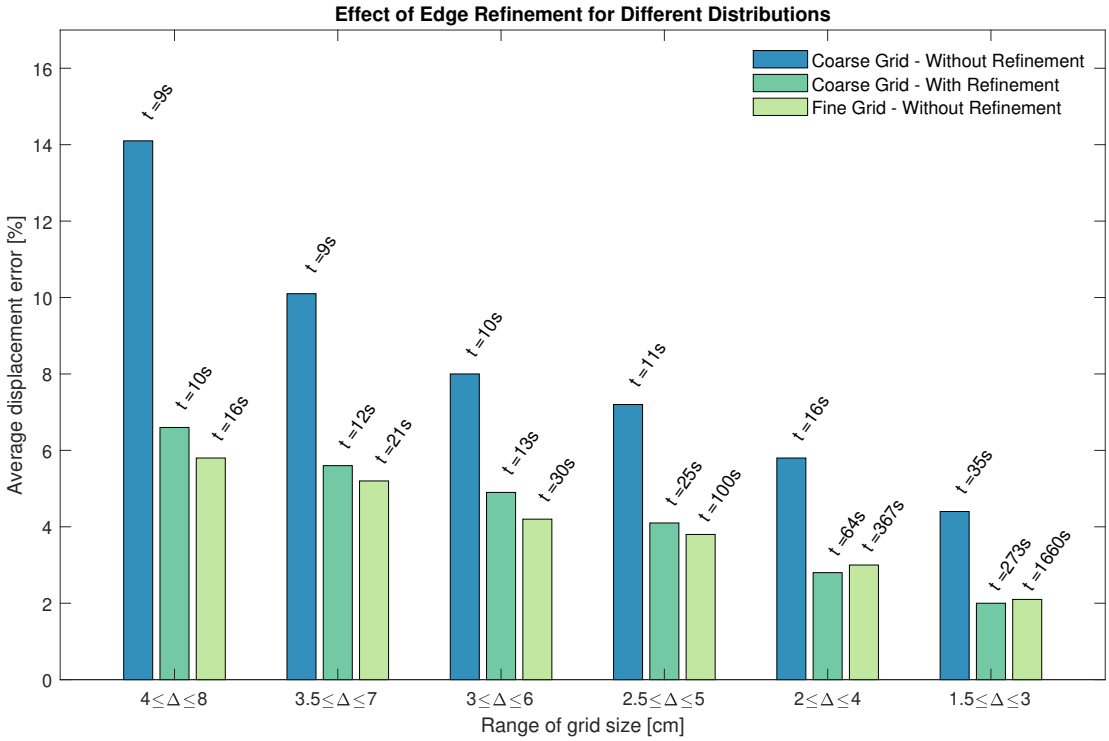


Figure 5.16: Relative average displacement error and simulation time for varying particle distributions. For each range of particle spacing a uniform coarse grid and a fine grid is used and compared to only using the fine grid at the boundaries.

The results from using six different grid ranges are presented in Figure 5.16. It is clear that for this particular setup that edge refinement improves the results significantly without increasing the computational time. As expected, since each

added particle will introduce multiple bonds, and since the computational time is proportional to the number of bonds (see Algorithm 1 in Appendix B), there is a big difference of only adding particles in a small region instead of over the whole domain, yet the accuracy is close to that case.

5.2.4 Influence of Volume Correction

The influence of the volume correction (described in Section 5.2.2) is studied for the same cantilever beam as in Section 5.2.2. Both regular particle distribution, with $\Delta = 0.027$ units of length, and irregular particle distributions are studied. Compared to the setup in Figure 5.12 the load applied here is 8 MN. The result is presented in Table 5.5. As indicated, some improvements could be achieved for certain family configurations for the regular particle distribution. For the irregular distribution it seems to be the opposite, also the simulation is unstable for certain family configurations for the case with volume correction and irregular distribution.

Table 5.5: Comparison of the tip displacement error relative to the analytical solution.

	Regular distribution				Irregular distribution			
Horizon δ / H_a	4.1	4.6	5.1	5.6	4.1	4.6	5.1	5.6
Error, correction on [%]	11.7	19.5	7.2	3.7	24.3	15.2	19.3	30
Error, correction off [%]	21.9	21.9	7.3	0.97	17.1	10.2	17.3	27.0

For uniform grids, volume correction was expected to have a larger effect for small horizons, as each particle contributes more in proportion. This is somewhat confirmed by the results for the regular distribution. The effect is however small due to the character of the kernel function which already decrease the effect from the outermost particles in the family. Due to the small effects and the ambiguous results, and the added computational cost, volume correction is turned off per default. The option is however still available through the Settings component.

5.3 Applications

Two case studies are performed to assess the performance of the tool and to show how it can be used is some common design scenarios.

5.3.1 Design of Truss Connections

To test the usability of the plugin, it has been integrated in a larger Grasshopper workflow that is used for designing a two-dimensional curved truss. A standalone FEM solver is used for computing the forces in the bars. The bar forces and cross sectional properties are then fed to the boundary condition component of the peridynamics plugin. The geometric input is provided partly from within Grasshopper using center lines and a set of parameters, and partly straight from Rhino using the Dynamic Geometry Pipeline from Human®. This way, the shape of the design domain can easily be altered by the user during runtime, either by subtracting material with void curves or by modifying the boundary curve itself. Based on the center lines and cross section of the bars, the Grasshopper script generates lines for the boundary conditions as well as the magnitudes of the boundary traction. For this case study the simplest geometry possible is used as starting domains. For this two-dimensional truss this will be a surface contained inside a polyline spanned between the end points of the boundary lines. Three suitable connections were chosen for further manual shape and topology optimization using the tool. The resulting shapes and corresponding stress fields are illustrated in Figure 5.17.

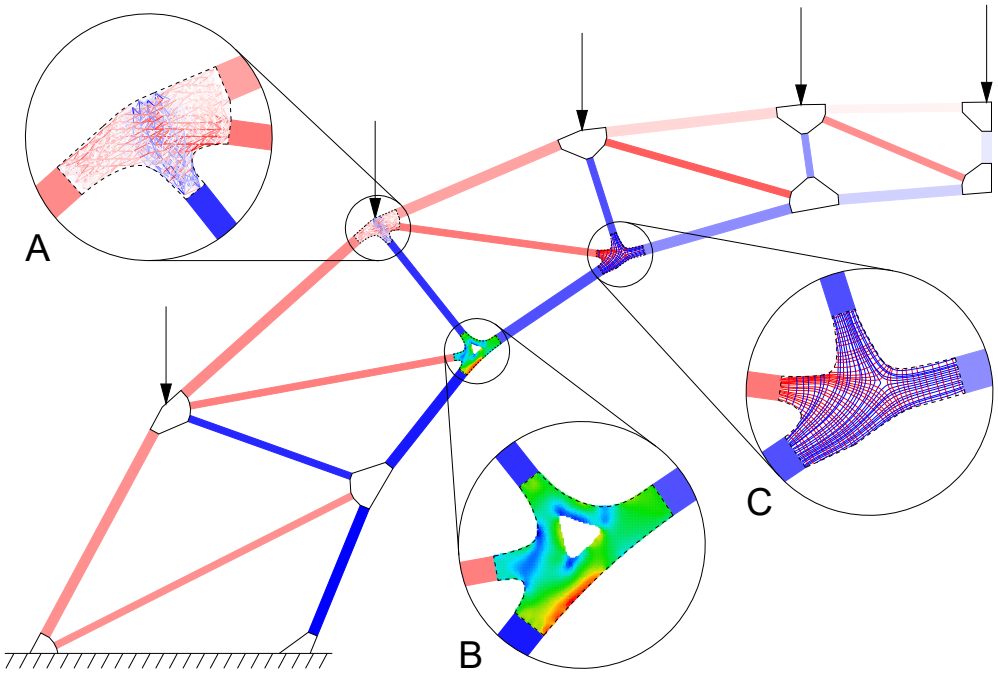


Figure 5.17: Extract of shaped connections in the truss, showing the modified shapes and (**A**) bond forces, (**B**) von Mises stress and (**C**) principal stress lines.

Consider for instance connection C in Figure 5.17. The first stage of the design process is to run the simulation using the initial design domain. In the second stage, closed curves are drawn in Rhino and supplied to the void input of the solver component. These curves are adjusted until a satisfactory stress distribution is achieved. The principal stress trajectories and the von Mises stress field, in this example the von Mises stress and the principal stress lines are used for guiding the design. When a change of any curve is detected the solver fades in or out particles as depicted as stage 2 in Figure 5.18 and 5.19. Finally, if significant changes have been made, i.e. the new boundary has moved outside the edge refinement region, the analysis can be restarted.

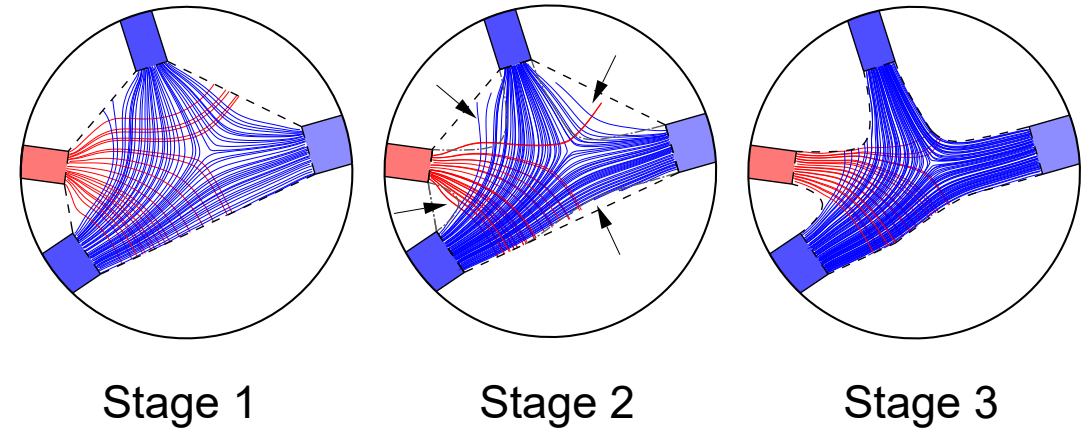


Figure 5.18: Example of how to use the tool to shape one of the connections in the truss using the principal stress trajectories.

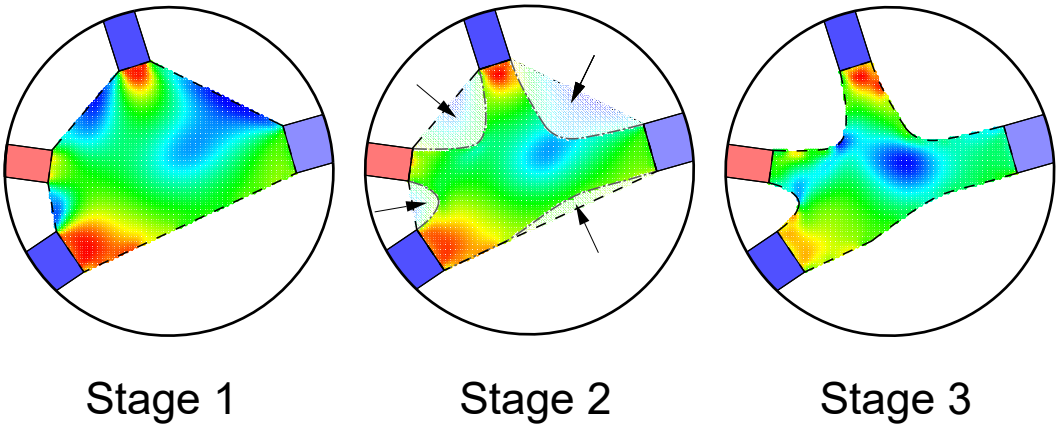


Figure 5.19: Example of how to use the tool to shape one of the connections in the truss using the von Mises stress field.

5.3.2 Discontinuity Regions

Having a good understanding of how the forces are flowing through a structure is crucial for making a good design. This case study aims at exemplifying how the principal stress trajectories can be used as a guide for removing unloaded parts of the design domain and as a consequence save material. As a demonstration, a simply supported beam subjected to a uniform load along with its supporting column heads is modelled using the tool. In Figure 5.20 one can observe how the forces are carried in arch action to the support and that the corner of the beam is basically unloaded. One can also observe the concentration of stress trajectories at the sharp corners in the beam, these areas are experiencing high concentrations of stress and are therefore susceptible to fatigue and fracture.

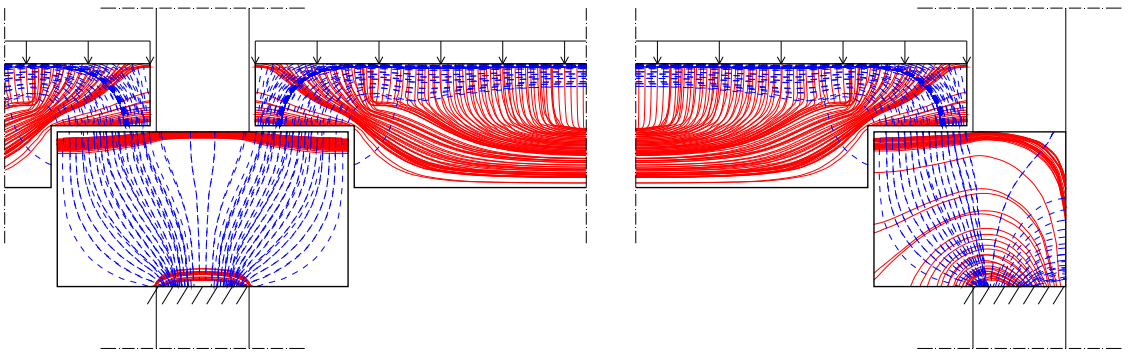


Figure 5.20: Principal stress lines in bulky column heads and slab.

In Figure 5.21, the same members as shown in Figure 5.20 have been reshaped to better fit the force flows. In doing so, a significant amount of material has been saved with a more organic design as the result. Also the areas subjected to high tension forces, and thus the ones in need of reinforcement, are easy to detect. One could immediately identify the potential of using the tool in reinforcement design. In particular the tool could be used as a basis when developing strut and tie models.

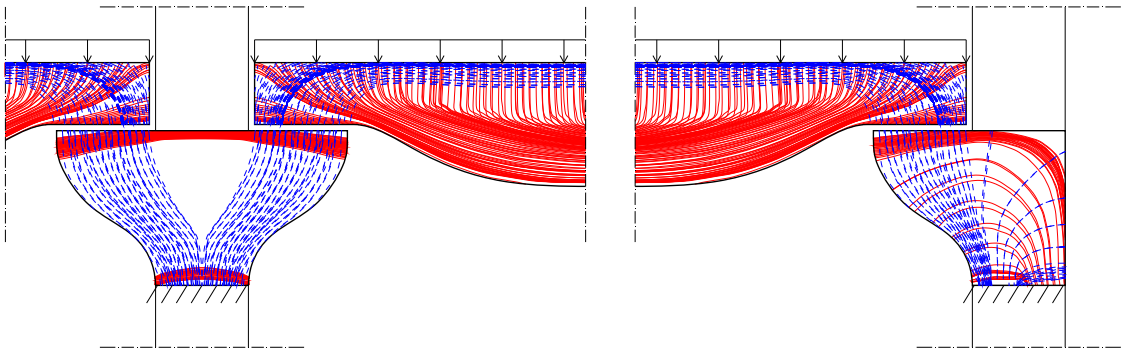


Figure 5.21: Principal stress lines in modified column heads and slab.

6
Discussion

At this stage it is time to conclude the work and discuss the findings. This chapter discuss the performance of the tool and how it relates to the research questions. This is followed by some reflections on the choice of using peridynamics. Lastly, potential areas for further research and development are presented.

6.1 Plugin Performance

After having conducted several case studies and convergence tests, the plugin has proven to perform well both concerning interactivity and accuracy. Both von Mises colour plots and principal stress trajectories seem to be correctly implemented and provide useful feedback on the stress flow. The dynamic relaxation implementation is effective for finding the converged solution. A great effort has been put into deriving a stable time step. It turned out to be well spent time since the user of the tool now does not need to manually calibrate the time step each time a setting is changed. Using a 95% carry over factor for damping has worked well, but to make the simulation even faster an adaptive factor would have to be implemented.

From the outset a variable particle size formulation of peridynamics was chosen as the mechanical model. This was based on the desire to have more densely packed particles at the boundaries and close to stress concentrations. The edge refinement technique implemented shows promising results for the tests that have been conducted. However, despite the reduction of the displacement error in Section 5.2.3, it was difficult achieve a smooth transition between particles of different sizes. This caused concentrations of the von Misses stress, which could be fixed by either improving the edge refinement algorithm, or by deriving a von Mises stress that is less sensitive to large irregularities. The attempt to improve the approximation of the deformation tensor by weighting the particles with the kernel function showed poor results for the limited number of tests performed, see Figure 5.8. Another option could be to use a higher order Taylor approximation as in [9]. As of now, the edge refinement is applied equally to all parts of the boundary. This can be extended to refining areas close to sharp edges or even based on the error using adaptive post-priori refinement techniques.

Another advantage of the formulation is that random particle distribution can be used. As shown in Section 5.1.2 this could be applied to mitigate irregularities in the boundary condition regions. The positive effects are somewhat compensated since irregularities are introduced in other parts of the body. When it comes to volume correction, an improvement of the result was observed for coarse regular grids. On the other hand is the kernel reducing the effects for most practical grid sizes. Considering this and the poor results for the irregular grid, volume correction is turned off as a default.

The power of the tool surely is the way in which it allows the designer to play with a shape and get real-time feedback on how the stress field adapts to geometry changes. The fact that the tool is exposed as a Grasshopper plugin makes it easy to integrate in large scale frameworks and algorithms, as shown with the truss in Section 5.3.1. Thereby, it is also accessible to both architects and engineers. The intention of the tool was to make it easier to design with the structural performance in mind, so that both aesthetical considerations and material efficiency can be integral parts of the design process. The authors believe that the plugin can act as a valuable support in this regard.

6.2 Peridynamics in a Design Process

The choice of using peridynamics was partly based on an interest to explore the possibilities of a particle-based method. In retrospect peridynamics turns out to be flexible for interactive design since particles and bonds can easily be turned on and off. With that said, a similar functionality can probably be accomplished using FEM by scaling the stiffness contributions in the stiffness matrix for degrees of freedom that are outside the body. A disadvantage of using elements instead of particles could however be the difficulty of achieving smooth boundaries when the boundary is changed. On the other hand, one drawback of using a particle method for large scale analyses where high accuracy is demanded is the computational cost. Nevertheless, for an early design stage, the current implementation has proven to be sufficiently fast for most scenarios presented in this thesis.

As described in Section 1.1.2, two of the major advantages of peridynamics are the abilities to model crack propagation and large deformations. None of these features are included in this work, but the framework developed could be used for incorporating fracture simulation. By integrating analysis methods that traditionally are considered advanced in an interactive tool, many phenomenon such as fatigue and fracture can be addressed earlier in the design phase. Thereby, costly adjustments down the line could potentially be avoided.

6.3 Further Developments

In its current state, the plugin is limited to linear elastic material behaviour in a small strain setting. This was the purpose, as it is intended to be used in an early design stage. It should however not be too hard to implement the capacity to handle yielding and fracture, especially not as the framework for deactivation of particles and bonds is already in place.

Another obvious extension would be to implement modelling in three dimensions. The code structure is built to enable such an extension, but it could nonetheless be non-trivial how to extend the curve-based geometry to surfaces and volumes. When it comes to the underlying theory, most of the derivations must be extended to 3D. This should however not be too complicated since the general framework remains the same. On the other hand, the interactivity of the tool could be hard to maintain as it would be demanding on the user to alter the domain, and the geometric results would be harder to visualize. One possible application of the tool if extended to three dimensions would be the design of connections in space frames. Based on the center lines and the cross sections of the elements, an initial design domain and boundary conditions can be generated automatically. The initial design domain could for instance be a ball or a polysurface. In Figure 6.1 one such domain is created for a connection in a space truss.

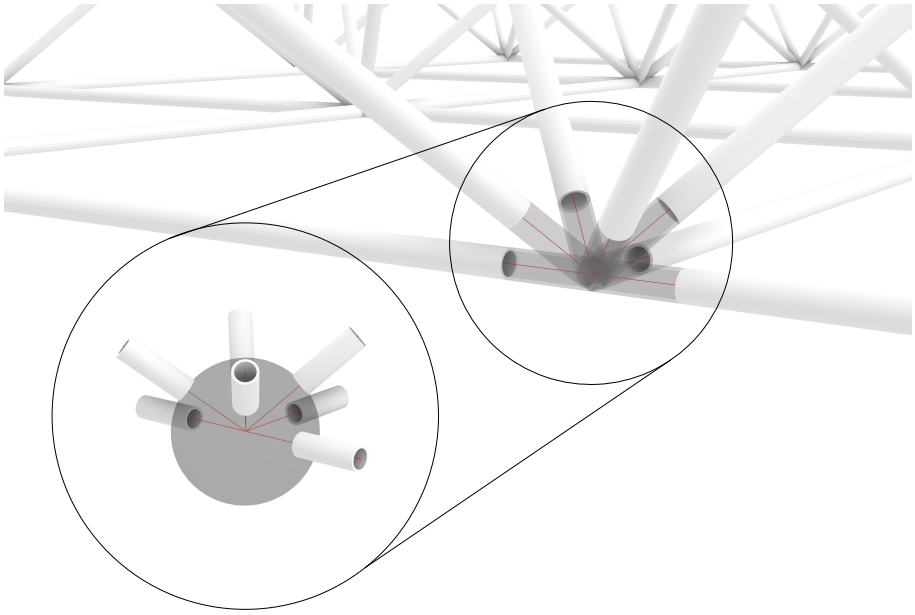


Figure 6.1: Space grid, its associated system lines and an example of an initial design domain.

It would be interesting to explore how the tool and the underlying theory can be extended to composite materials. For this, the interactions between the parts with different properties would need to be studied. The case study on discontinuity regions in Section 5.3.2 suggests that the tool could potentially be useful in the reinforcement design for concrete structures. If the tool is extended to handle steel and concrete interaction, this could be explored further. On the same theme, phase changes in materials could be a subject to study. With the increasing use of 3D-printing, e.g. additive manufacturing for steel, this field of research is very relevant. To allow for the particles to move freely in the melted phase, and then to order themselves as the material solidifies. Also, an extension from only isotropic materials to include the possibility to model anisotropic and orthotropic materials, such as wood, would further widen the range of potential applications.

Although proving more than sufficient performance regarding time consumption and computational cost for a reasonable high number of particles, large domains with a high particle density significantly slow down the process. Certainly, the code is not fully optimized for efficiency, but the real game changer would be to rewrite the code to run on the GPU (Graphic Processor Unit) rather than the CPU (Central Processor Unit). As the computations in each time step can be performed in parallel there could potentially be a huge benefit to gain from using the GPU. Though, this would probably involve a rather large reorganization - or even a complete rewriting - of the code. Nevertheless, this would make the tool even more interactive and powerful.

Finally, the particle generation algorithm should be revised to open up for a more efficient and adaptive refinement process. Now, first the zones are generated based on the boundary curve, and then the particles are generated zone-wise. By reversing that process, i.e. start by generating the particles and then base the zones on the distribution, the particle distribution could be smoother and better mirror the geometry. Moreover, such a framework could potentially better support an adaptive fineness, where the density adopts for domain changes or obtained stress measures.

Bibliography

[1] Altair Engineering, Inc. *OptiStruct*. 2021. URL: https://www.grasshopper3d.com/.

[2] Francesco Buonamici et al. "Generative design: An explorative study". In: *Computer-Aided Design and Applications* 18.1 (2020), pp. 144–155. ISSN: 16864360. DOI: 10.14733/cadaps.2021.144-155.

[3] Stephan C Carlson. *Topology*. 2017. URL: https://www.britannica.com/science/topology.

[4] Dassault Systèmes. *Tosca*. 2021. URL: https://www.3ds.com/products-services/simulia/products/tosca/.

[5] Yadolah Dodge. "Weighted Least-Squares Method". In: *The Concise Encyclopedia of Statistics*. New York, NY: Springer New York, 2008, pp. 566–569. ISBN: 978-0-387-32833-1. DOI: 10.1007/978-0-387-32833-1_422.

[6] DTU. *topOpt*. 2021. URL: https://www.topopt.mek.dtu.dk/.

[7] Paul Filippi et al. *Acoustics : Basic Physics, Theory and Methods*. Elsevier Science & Technology, 1998. ISBN: 9780080498553. URL: https://search.ebscohost.com/login.aspx?direct=true&db=cat07470a&AN=clc.1f55e1c1.7235.41f3.b6cc.45d6992c03ca&site=eds-live&scope=site&authtype=guest&custid=s3911979&groupid=main&profile=eds.

[8] G. C. Ganzenmüller, S. Hiermaier, and M. May. "On the similarity of meshless discretizations of Peridynamics and Smooth-Particle Hydrodynamics". In: *Computers and Structures* 150 (2015), pp. 71–78. ISSN: 00457949. DOI: 10.1016/j.compstruc.2014.12.011. arXiv: 1401.8268.

[9] M G D Geers, R De Borst, and W A M Brekelmans. "Computing strain fields from discrete displacement fields in 2D-solids". In: *International Journal of Solids and Structures* 33.29 (1996), pp. 4293–4307. ISSN: 0020-7683. DOI: https://doi.org/10.1016/0020-7683(95)00240-5.

[10] Mir Ali Ghaffari et al. "Peridynamics with Corrected Boundary Conditions and Its Implementation in Multiscale Modeling of Rolling Contact Fatigue". In: *Journal of Multiscale Modelling* 10.01 (2019), p. 1841003. ISSN: 1756-9737. DOI: 10.1142/s1756973718410032.

[11] Nima Haghdadi et al. "Additive manufacturing of steels: a review of achievements and challenges". In: *Journal of Materials Science* 56.1 (Aug. 2020), pp. 64–107. DOI: 10.1007/s10853-020-05109-0.

[12] Intergovernmental Panel on Climate Intergovernmental Panel on Climate Change. *Climate Change 2014 Mitigation of Climate Change*. Cambridge:

Cambridge University Press, 2014. ISBN: 9781107415416. DOI: 10.1017/CBO9781107415416.

[13] Ali Javili et al. "Peridynamics review". In: *Mathematics and Mechanics of Solids* 24 (11 Nov. 2019), pp. 3714–3739. ISSN: 17413028. DOI: 10.1177/1081286518803411.

[14] C. Lanczos. *The Variational Principles of Mechanics*. Dover Books On Physics. Dover Publications, 1986. ISBN: 9780486650678. URL: https://search.ebscohost.com/login.aspx?direct=true&db=cat07470a&AN=clc.68ace0fd.9063.46db.b261.3f8999bbf068&site=eds-live&scope=site&authtype=guest&custid=s3911979&groupid=main&profile=eds.

[15] Shaofan Li and Wing Kam Liu. "Meshfree and particle methods and their applications". In: *Applied Mechanics Reviews* 55.1 (2002), pp. 1–34. ISSN: 00036900. DOI: 10.1115/1.1431547.

[16] David John Littlewood. *Roadmap for Peridynamic Software Implementation*. Sandia National Laboratories (SNL), Oct. 2015. DOI: 10.2172/1226115.

[17] Erdogan Madenci and Erkan Oterkus. *Peridynamic Theory and Its Applications*. New York, NY: Springer New York, 2014. ISBN: 978-1-4614-8464-6. DOI: 10.1007/978-1-4614-8465-3.

[18] Morten Nobel-Jørgensen et al. "3D interactive topology optimization on hand-held devices". In: *Structural and Multidisciplinary Optimization* 51.6 (2015), pp. 1385–1391. ISSN: 16151488. DOI: 10.1007/s00158-014-1214-8.

[19] David R. Oakley and Norman F. Knight. "Adaptive dynamic relaxation algorithm for non-linear hyperelastic structures Part I. Formulation". In: *Computer Methods in Applied Mechanics and Engineering* 126.1 (1995), pp. 67–89. ISSN: 0045-7825. DOI: https://doi.org/10.1016/0045-7825(95)00805-B.

[20] Jens Olsson. "In Conversation With Simulation". Licentiate thesis. Chalmers university of technology, 2020.

[21] Jens Olsson, Mats Ander, and Chris J. K. Williams. "The Use of Peridynamic Virtual Fibres to Simulate Yielding and Brittle Fracture". In: *Journal of Peridynamics and Nonlocal Modeling* (Apr. 2021). ISSN: 2522-896X. DOI: 10.1007/s42102-021-00051-4.

[22] Peter W. Christensen and Anders Klarbring. *An Introduction to Structural Optimization*. Vol. 153. Dordrecht: Springer Netherlands, 2008. ISBN: 978-1-4020-8665-6. DOI: 10.1007/978-1-4020-8666-3.

[23] J N Reddy. *An introduction to continuum mechanics*. Cambridge University Press. ISBN: 9781107025431. URL: https://search.ebscohost.com/login.aspx?direct=true&db=cat07470a&AN=clc.657cfe00.c1e6.4a85.a2e6.f5a87021766e&site=eds-live&scope=site&authtype=guest&custid=s3911979&groupid=main&profile=eds.

[24] Robert McNeel & Associates. *Grasshopper 3D*. 2021. URL: https://www.grasshopper3d.com/.

[25] S. A. Silling et al. "Peridynamic states and constitutive modeling". In: *Journal of Elasticity* 88.2 (2007), pp. 151–184. ISSN: 03743535. DOI: 10.1007/s10659-007-9125-1.

[26] XIE Engineering Technologies Co. ,Ltd. *Ameba*. 2021. URL: https://www.kancloud.cn/woshiyaoyuan0318/ameba_en/582345.

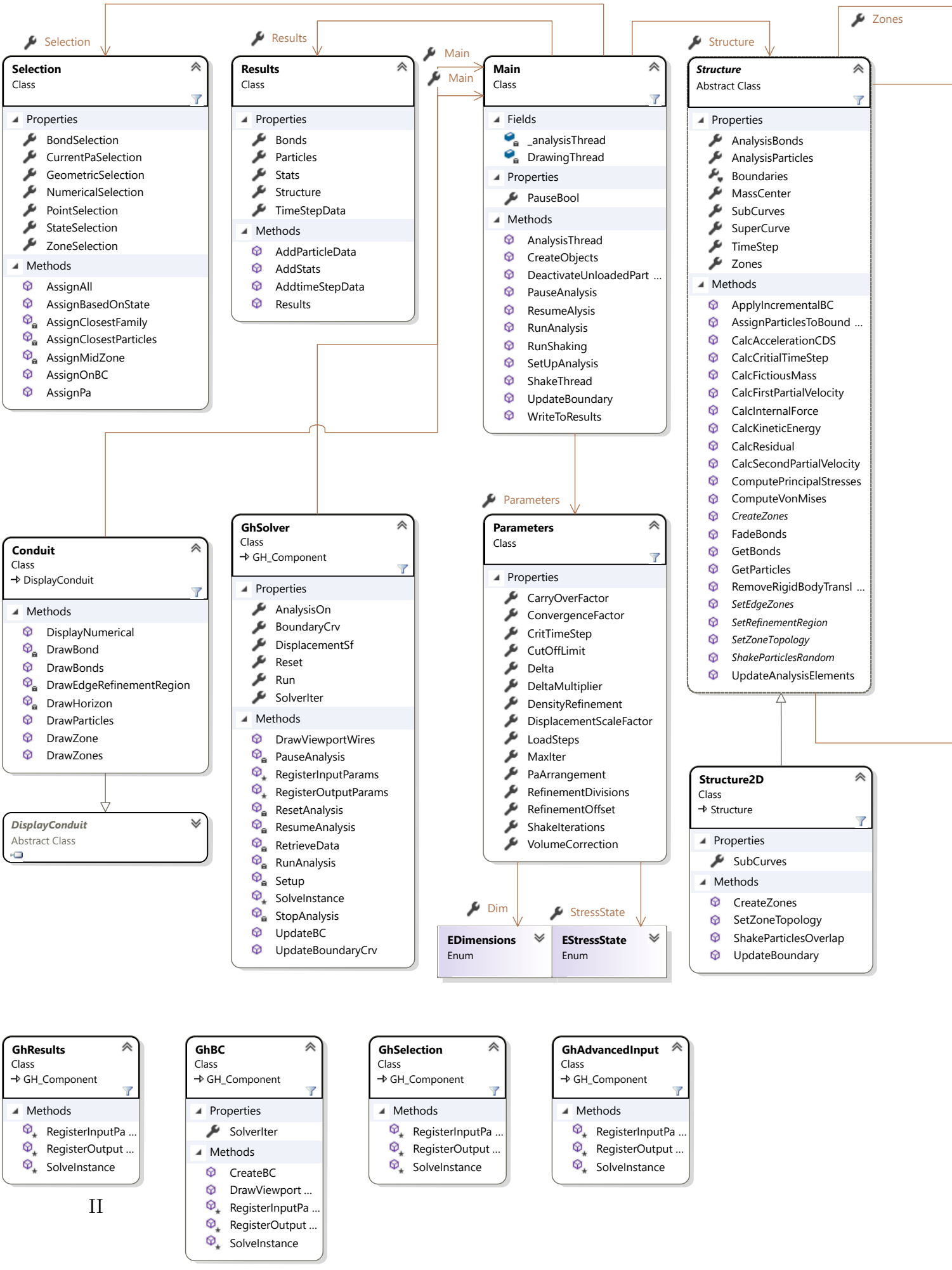
[27] ZJA. *Extended Waal bridge, Nijmegen*. n.d. URL: https://www.zja.nl/en/Verlengde-Waalbrug-Nijmegen.

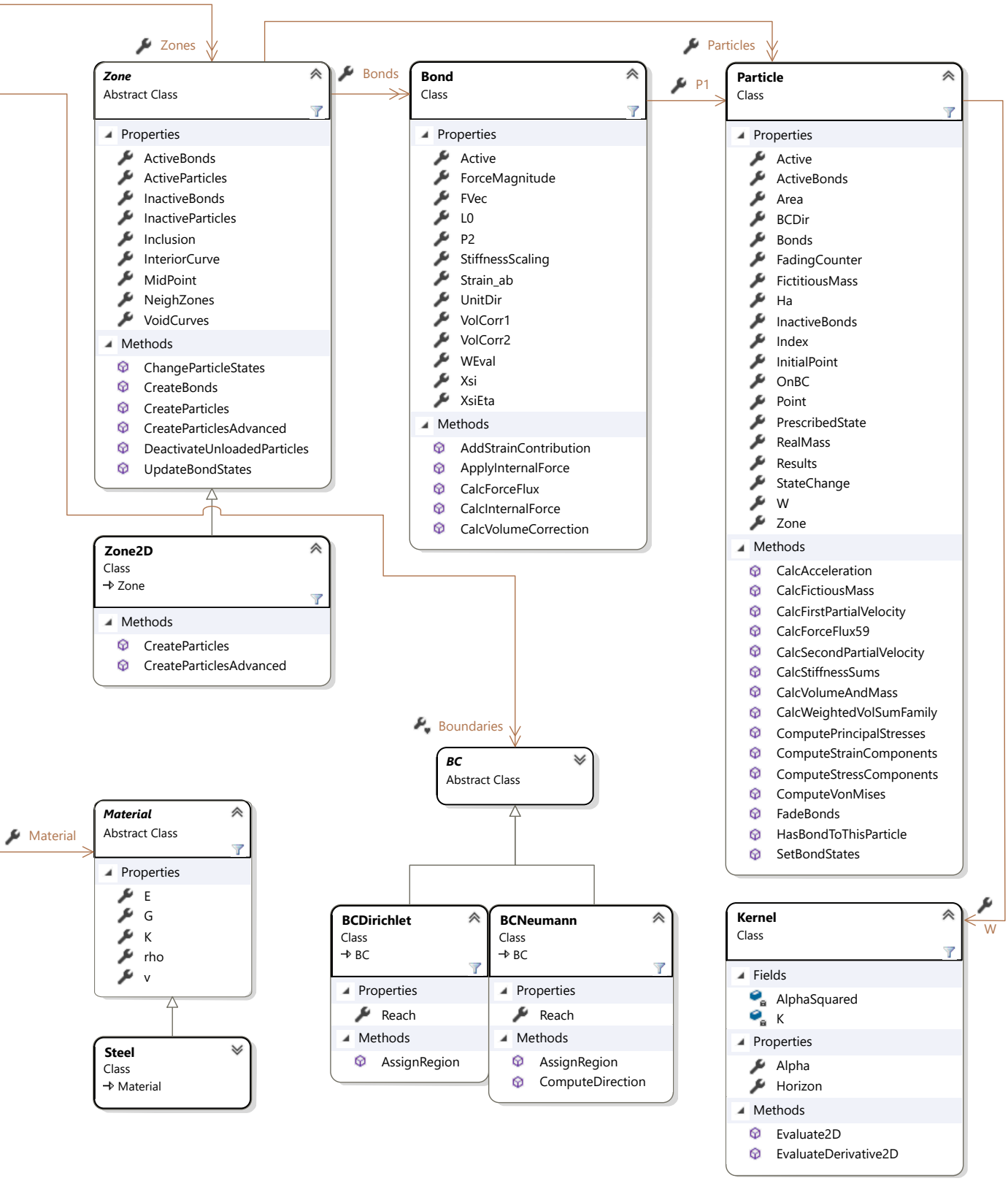
A

Class Diagram

This is a more complete class diagram showing important methods, properties and fields. For a full description of the code, we refer to the documentation

A. Class Diagram




Zones
Zone
Abstract Class
Properties
ActiveBonds
ActiveParticles
InactiveBonds
InactiveParticles
Inclusion
InteriorCurve
MidPoint
NeighZones
VoidCurves
Methods
ChangeParticleStates
CreateBonds
CreateParticles
CreateParticlesAdvanced
DeactivateUnloadedParticles
UpdateBondStates
Bonds
Bond
Class
Properties
Active
ForceMagnitude
FVec
L0
P2
StiffnessScaling
Strain_ab
UnitDir
VolCorr1
VolCorr2
WEval
Xsi
XsiEta
Methods
AddStrainContribution
ApplyInternalForce
CalcForceFlux
CalcInternalForce
CalcVolumeCorrection
Particles
P1
Particle
Class
Properties
Active
ActiveBonds
Area
BCDir
Bonds
FadingCounter
FictitiousMass
Ha
InactiveBonds
Index
InitialPoint
OnBC
Point
PrescribedState
RealMass
Results
StateChange
W
Zone
Methods
CalcAcceleration
CalcFictiousMass
CalcFirstPartialVelocity
CalcForceFlux59
CalcSecondPartialVelocity
CalcStiffnessSums
CalcVolumeAndMass
CalcWeightedVolSumFamily
ComputePrincipalStresses
ComputeStrainComponents
ComputeStressComponents
ComputeVonMises
FadeBonds
HasBondToThisParticle
SetBondStates
Zone2D
Class
Zone
Methods
CreateParticles
CreateParticlesAdvanced
Boundaries
BC
Abstract Class
Material
Material
Abstract Class
Properties
E
G
K
rho
v
Steel
Class
Material
BCDirichlet
Class
BC
Properties
Reach
Methods
AssignRegion
BCNeumann
Class
BC
Properties
Reach
Methods
AssignRegion
ComputeDirection
Kernel
Class
W
Fields
AlphaSquared
K
Properties
Alpha
Horizon
Methods
Evaluate2D
EvaluateDerivative2D

B

Algorithms

In this appendix some of the central algorithms used in the code implementation of the plugin are presented.

Algorithm 2: Particle creation

```
Upon setup;
if no edge refinement then
    for zone in zones do
        create particles based on Δ_input;
    end
else
    for curve in {boundary, voids} do
        create boundary offset pairs and refinement regions;
    end
    for zone in zones do
        if zone is not intersecting boundary, voids or offsets then
            create particles based on Δ_input;
        else
            for region in refinement regions do
                calculate Δ_refined;
                if zone is inside or intersecting then
                    compute potential particle positions based on Δ_refined;
                    for position in particle positions do
                        if position is inside region then
                            create particle based on Δ_refined;
                        end
                    end
                end
            end
        end
    end
end
for zone in zones do
    if zone inside or on border of region bound by boundary and voids then
        for particle in zone do
            if Contained within boundary but outside voids then
                activate particle
            end
            for Bond connected to particle do
                if Both end particles are active then
                    Activate bond
                end
            end
        end
    end
end
```

Algorithm 3: Particle activation and deactivation

```
When boundary curve is changed;
Update which zones are inside boundary;
for zones in active zones do
    if zone inside or on boundary then
        for particle in zone do
            if Inside outer boundary and outside sub boundaries then
                activate particle if not already active;
            else
                deactivate particle if not already active
            end
            for bond connected to particle do
                if both end particles are active then
                    activate bond;
                    set stiffens scaling to 0;
                    flag as fading in and move to the changed particles list
                else
                    deactivate bond;
                    set stiffness scaling to 1;
                    flag as fading out and move to the changed particles list;
                end
            end
        end
    end
end
```

Algorithm 4: Fading particles and bonds

```
To be executed in each analysis iteration;
for Bond in fade in bond list do
    if stiffness factor < 1 then
        increment stiffness factor;
    else
        remove from list;
    end
end
for Bond in fade out bond list do
    if stiffness factor > 1 then
        decrease stiffness factor;
    else
        remove from list;
    end
end
```