UNIVERSITY OF CAMBRIDGE
Department of Engineering

# Modelling the Failure Behaviour of Concrete Structures Using Peridynamics

Author Name: Preben Monteiro Ness

Supervisor: Dr John Orr

May 2019

I hereby declare that, except where specifically indicated, the work submitted herein is my own original work.

*Signed* _____ *date* _____

# Modelling the Failure Behaviour of Concrete Structures Using Peridynamics

Preben Monteiro Ness

May 2019

Department of Engineering
University of Cambridge

King's College

Supervisor: Dr John Orr

# Technical Abstract

Peridynamics is a novel mathematical reformulation of classical continuum mechanics that is particularly well suited to capturing the behaviour of concrete, compared to Finite Element Method approaches. Peridynamics is often mathematically intractable but can be implemented as a computational framework that can be used to numerically simulate the behaviour of structural members under load. This project implemented a bond-based Peridynamics simulation and used it to predict the failure loads of cantilevered reinforced concrete beams.

One of the major challenges associated with Peridynamics is the large computational cost of simulations. This cost depends heavily on the coarseness of the spatial discretisation employed when analysing models. This report investigates how this coarseness affects the reliability of simulation predictions for failure and beam stiffness.

In order to predict the failure load of structural members it is necessary to have a mathematical definition of what constitutes failure within a Peridynamics modelling framework. Two approaches in the scientific literature are presented, and a third approach is proposed by the author. These are referred to as the *velocity decay*, the *force decay* and the *bond damage* convergence criteria, respectively. All three methods are used to predict the failure load of two cantilevered reinforced concrete beams. Performance is assessed by comparing the predictions to failure loads measured in laboratory tests on the beams. The dependence of predictions on tolerance parameters of the convergence criteria is also analysed and compared.

It is found that the velocity decay criterion is highly sensitive to its tolerance parameter, and as a result produces unreliable predictions. The force decay criterion also failed to produce accurate predictions for the beams tested in this implementation. The proposed bond damage criterion gave good predictions of failure loads in all but one of the eight simulations performed. This criterion also appears to be less sensitive to tolerance parameter values than the other two criteria.

Predictions of failure loads as well as beam force-displacement behaviour were found to differ for different spatial discretisation coarseness levels. Four levels of coarseness were used for both beams, ranging from 792 to 6256 total model nodes. A systematic bias in predictions from employing a coarse model was only found for the prediction of failure loads using the force decay criterion. When using this criterion, refining the

spatial discretisation caused the predicted failure loads to decrease.

The proposed criterion for defining structural failure in Peridynamics appears to be a good heuristic alternative to the existing methods. However, further refinement and analysis of sensitivity to tolerance parameters is needed in order to generalise the method as a universal structural failure definition in Peridynamics. To this end, the full software written for this project has been made available by the author.

# Acknowledgements

Firstly I would like to thank my supervisor Dr John Orr for his guidance and support over the course of this project. I would also like thank Dr Gabriel Hattori for many useful discussions on the technical aspects of Peridynamics as well as the details of efficient computational approaches. Finally I would like to thank Mark Hobbs for invaluable help with the software implementation for this project.

# Contents

# List of Figures

# List of Tables

# 1 Notation

### Peridynamics

| | |
|---|---|
| $\boldsymbol{A}$ | Global connectivity matrix |
| $B$ | Set of loaded nodes |
| $\boldsymbol{b}$ | Body force $[N/m^3]$ |
| $C$ | Damping factor $[kg/m^3s]$ |
| $c$ | Bond stiffness $[N/m^6]$ |
| $d$ | Average number of nodes per spatial direction |
| $E$ | Young's modulus $[N/m^2]$ |
| $\boldsymbol{f}$ | Force density function $[N/m^6]$ |
| $G_0$ | Fracture energy release rate $[Nm/m^2]$ |
| $H_{\boldsymbol{x}}$ | Horizon of point $\boldsymbol{x}$ |
| $N_i$ | Set of nodes within the horizon of node $\boldsymbol{x}_i$ |
| $s$ | Bond stretch $[-]$ |
| $s_0$ | Critical bond stretch $[-]$ |
| $\Delta t$ | Time step size $[s]$ |
| $t_{build}$ | Number of time steps to build up load over |
| $t_{max}$ | Maximum number of time steps in a simulation |
| $\boldsymbol{u}$ | Displacement $[m]$ |
| $\dot{\boldsymbol{u}}$ | Velocity $[m/s]$ |
| $\ddot{\boldsymbol{u}}$ | Acceleration $[m/s^2]$ |
| $V_i(\boldsymbol{x}_j)$ | Volume of cell occupied by node $\boldsymbol{x}_j$ within the horizon of node $\boldsymbol{x}_i$ |
| $\boldsymbol{x}$ | Material point (continuous material) |
| $\boldsymbol{x}_i$ | Node number $i$ (discretised material) |
| $\beta$ | Tolerance parameter in the force decay convergence criterion |
| $\delta$ | Horizon radius $[m]$ |
| $\mu$ | Bond state |
| $\nu$ | Poisson's ratio $[-]$ |
| $\rho$ | Material density $[kg/m^3]$ |
| $\tau_D$ | Tolerance parameter in the bond damage convergence criterion |
| $\tau_{\dot{u}}$ | Tolerance parameter in the velocity decay convergence criterion |

**Acronyms and abbreviations**

| | |
|---|---|
| FEM | Finite Element Method |
| HPC | High Performance Computer |
| PD | Peridynamics |
| PDE | Partial Differential Equation |
| RC | Reinforced Concrete |
| XFEM | Extended Finite Element Method |

# 2 Introduction

The traditional approaches employed in the design of reinforced concrete (RC) beams are usually wasteful in terms of material usage and embodied energy [1]. One reason is that in a simple prismatic beam of constant cross section, much of the material is not fully utilised, even at failure. The reasons these designs are still used are numerous: simpler designs are well covered by existing legislative codes governing design. They are often simple to verify as safe, and can typically be analysed without the use of bespoke software.

The analysis of the failure behaviour of RC structures under load is of course of central importance in civil engineering. However, accurately predicting the deflections and failure loads of RC beams is challenging even for simple prismatic geometries. The non-linear behaviour of concrete as well as the importance of fracture and microstructural damage during failure makes many RC beam problems analytically intractable.

Peridynamics (PD) is a mathematical reformulation of the governing laws of continuum mechanics. The central idea of PD is to reformulate the governing equations of continuum mechanics using integral equations [2]. This circumvents the fundamental problem of evaluating derivatives at spatial discontinuities such as cracks, and offers a mathematical framework well suited to analysing concrete structures. For most geometries the governing PD equations have no analytical solution. In practical applications, PD is therefore a numerical and computational modelling framework.

PD models have the attractive property that cracks and microstructural damage require no special mathematical treatment, but arise naturally from the governing equations. However, Peridynamics comes with its own set of challenges. Some of these are shared with other particle models, and some are specific to this mathematical framework. A major problem with PD models is the computational cost of running simulations. Whereas commercial FEM analysis takes on the order of seconds and minutes on a consumer grade computer, equivalent PD models often take on the order of hours to run [3].

## 2.1 Motivation

Throughout this work the term *optimised* or *efficient* design is used to broadly mean a beam design that is in some sense cheaper, lighter or with less embodied energy

than an equivalent prismatic beam. Such optimised designs are typically non-trivial to analyse as they often encompass curved surfaces and tapering cross-sections. Standard Eurocode formulae and approaches can therefore be unreliable ways of approximating the behaviour of such beams. As a result, optimised beam designs can in general only be analysed numerically, typically using Finite Element Methods (FEM).

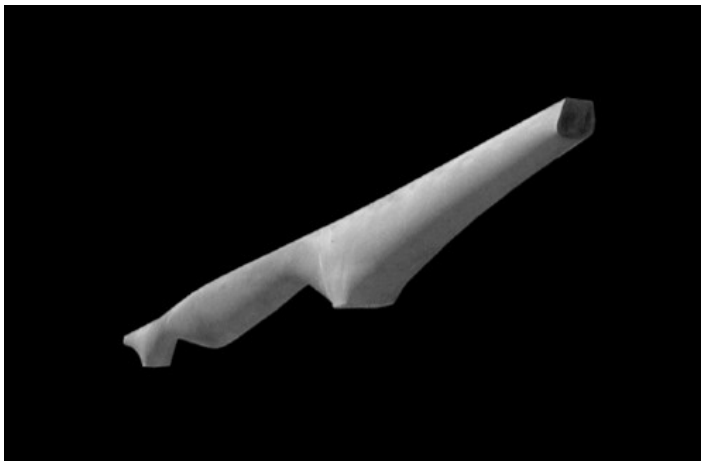


Figure 2.1: Optimised design of RC beam. Credit Mark West and *The Centre for Architectural Structures and Technology* (C.A.S.T.)

This work has examined the use of PD software as a practical tool for beam analysis. In order for PD to be useful in this manner, it is necessary to verify both its physical accuracy and computational practicality. The software developed for this project was written in full by the author and is intended as a versatile implementation of PD analysis, suitable for a large range of experiments [1]. In this work the experiments conducted were on prismatic standardised cantilevered beams. This was chosen as a way to measure the physical accuracy of the model predictions, although the software architecture was written in such a way as to easily handle arbitrary geometries, such as the design shown in Figure 2.1. A natural extension of the investigations described in this work would be applying the software to analyse failure loads of optimised beam designs (see section 8.1 for further details).

## 2.2 Key Areas

In order to assess the physical accuracy of any modelling framework for RC beams it is useful to define certain key characteristic behaviours the model should exhibit. In

[1] Full software implementation available at https://github.com/prebenness/peridynamics

this work the model predictions were assessed in terms of the beams' predicted failure load, failure mode and global stiffness.

The traditional choice for analysing complex concrete structures numerically are FEM packages. However, modelling the complex behaviour of concrete under load is a challenging computational task. The treatment of brittle materials and cracks are problems that are ill suited for traditional FEM software. It is desirable that any alternative model employed has few governing simulation parameters, and that model predictions are insensitive to the exact choice of parameter values or modelling approximations.

In order to perform analysis of failure loads and behaviours, it is necessary to have a strict definition of what constitutes failure. The definition needs to be well defined within the mathematical framework of the model being used and needs to be able to be automatically evaluated as part of the simulation. A good modelling framework hence also needs to include an accurate definition of failure that agrees with experimental observations.

# 3 Modelling of Beam Behaviour

## 3.1 Design Codes

The traditional method for predicting the deflection and failure behaviour of an RC beam is by using codified approximations like the Eurocodes. In order to predict the flexural strength of an RC beam, a simplified rectangular stress block is used to model the internal state of stress of the beam's cross section. The modelling of shear failure, however, is primarily based on empirical results derived from tests on prismatic beams. This means that for non-prismatic or tapering cross sections, the failure loads predicted using the Eurocodes are not necessarily conservative [4].

## 3.2 FEM

FEM software packages are widely used for a range of practical structural analysis applications, as well as for the modelling of heat transfer and fluid flow [5]. In its basic formulation FEM is used to model numerically the evolution of smoothly varying fields, typically governed by one or more partial differential equations (PDEs). The governing equations are solved using a spatial discretisation technique called *meshing*, where the modelled material body is divided into a set of elements of finite size. In structural analysis the governing equations are the PDEs of classical continuum mechanics, and the fields being modelled are typically the stress and displacement fields within the structural member. The governing PDEs for the stress field, in the absence of body forces, in a continuous body internally in equilibrium are given in equations 3.1 and 3.2. Here $\sigma_{ij}$ denotes the stress in the $j$ direction on a plane normal to the $i$ direction [6].

$$\frac{\partial \sigma_{xx}}{\partial x} + \frac{\partial \sigma_{xy}}{\partial y} = 0 \tag{3.1}$$

$$\left\{ \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} \right\} (\sigma_{xx} + \sigma_{yy}) = 0 \tag{3.2}$$

FEM approaches are ill-suited to problems where the field being modelled is not everywhere continuous and differentiable within the modelled body. Since cracks represent discontinuities in the displacement field, the necessary spatial derivatives of the gov-

erning equations are not defined at the crack edge. The problem is therefore unsolvable by direct application of traditional FEM using classical continuum mechanics [7].

Several approaches exist within the framework of FEM to address this shortcoming. Two methods are discussed here: the *Discrete Crack Approach* [8] and the *Smeared Crack Model* [9]. The Discrete Crack Approach redefines the boundaries of the modelled material body during cracking in such a way as to ensure that the crack always lies on a free surface of the body. This method suffers from two major drawbacks. Firstly, cracking is restricted to only occur on edges between elements, limiting the allowable crack propagation paths. Secondly, all element edges that break need to have a set of redundant extra nodes, one for each crack edge. This is to ensure that the continuum boundary can be properly redefined upon breaking. In order to implement this, prior knowledge about likely crack paths is needed. The alternative is to double all element nodes, however this becomes computationally expensive for fine meshes.

The Smeared Crack Model attempts to incorporate material damage from fracture as a distributed weakening of stiffness in the area around the crack. This negates the need to continuously redefine the material body as the crack propagates. However, the behaviour of the crack is highly dependent on the exact meshing employed in the FEM model, and predictions are sensitive to changes in internal system parameters.

Specialised formulations such as the *Extended Finite Element Method* (XFEM) exist, and have been used successfully to simulate brittle material fracture [10]. Nonetheless, no present FEM formulation has been shown to correctly model the full behaviour of concrete in a general framework[11]. In particular, the strain-softening behaviour of concrete has proved difficult to model in an FEM framework. The mathematical framework of Peridynamics seeks to address the shortcomings of FEM outlined here, and is described in the following section.

## 3.3 An Overview of Peridynamic Theory

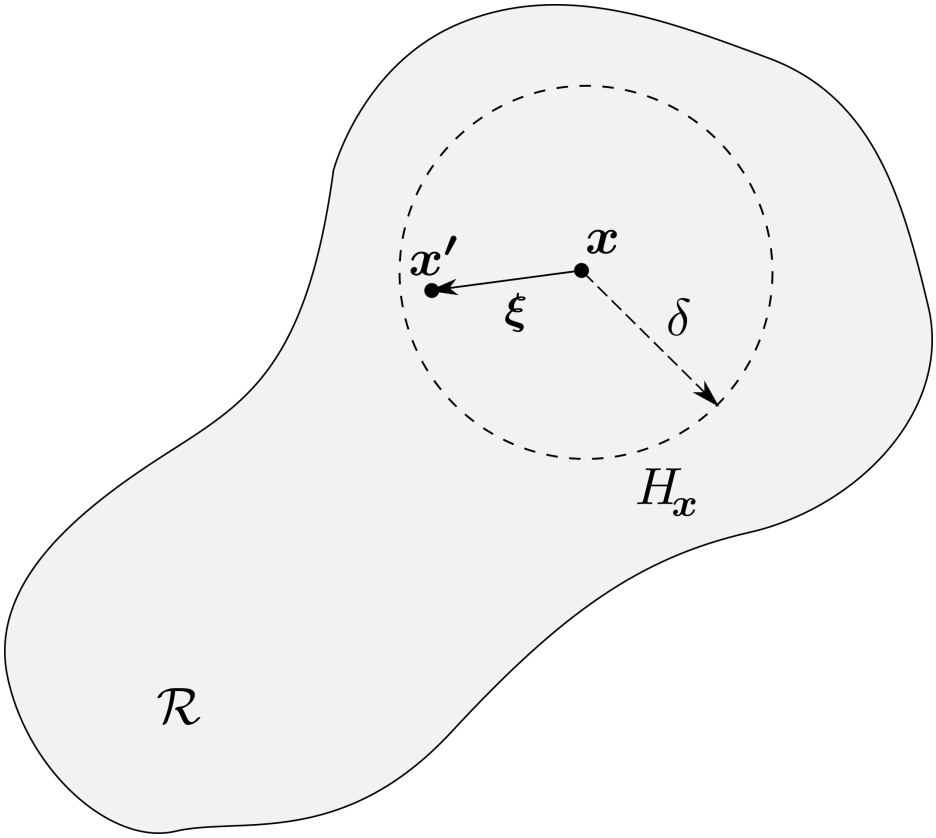



Figure 3.1: Schematic view of a material region and the PD horizon

In order to deal with material discontinuities such as cracks in a mathematically simple way, *Peridynamics* was proposed by Silling (2000) [2]. By reformulating the governing equations of continuum mechanics as integral equations, PD allows for the inclusion of cracks without any additional mathematical treatment. In fact, cracking and microdamage arise naturally in this modelling framework and is central to failure behaviour.

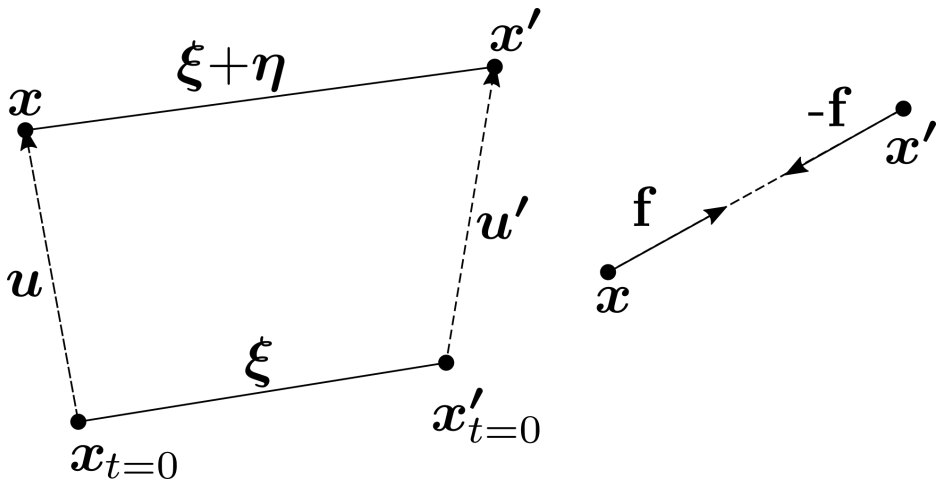



Figure 3.2: Conventional PD notation

The central idea of PD is to describe the behaviour of materials under load without the use of spatial derivatives. Figure 3.1 shows a schematic overview. Every material point $\boldsymbol{x}$ has an associated surrounding region of interaction, $H_x$. This is called the *horizon* of $\boldsymbol{x}$, and is a spherical region of radius $\delta$. Points interact with neighbouring points $\boldsymbol{x'}$ within this horizon through the *force density function* $\boldsymbol{f}(\boldsymbol{u}, \boldsymbol{u'}, \boldsymbol{x}, \boldsymbol{x'})$, where $\boldsymbol{u}$ and $\boldsymbol{u'}$ are the displacements of $\boldsymbol{x}$ and $\boldsymbol{x'}$ from their original positions. $\boldsymbol{f}$ has units of $N/m^6$, and governs the force-displacement behaviour of point pairs. Apart from physical restrictions to ensure $\boldsymbol{f}$ does not violate Newton's third law, there are in principle no limitations on the form of $\boldsymbol{f}$. Different material behaviours can be modelled by changing the dependence of $\boldsymbol{f}$ on the relative displacement of point pairs, to model effects like plasticity and strain hardening.

Regardless of the exact form of $\boldsymbol{f}$, the force density function can be used to formulate the equilibrium equation of PD, shown in equation 3.3. This equation takes the form of Newton's second law and describes the behaviour of a node at position $\boldsymbol{x}$ interacting with neighbouring nodes at $\boldsymbol{x'}$. Here $\rho$ is the material density, $\boldsymbol{u}$ is the node displacement, and $\boldsymbol{b}$ denotes any externally applied body force field. The domain of integration $\mathcal{R}$ is in general taken as the whole of the material, however, in most PD formulations this can be replaced by $||\boldsymbol{x} - \boldsymbol{x'}|| \leq \delta$ as $\boldsymbol{f} = \boldsymbol{0}$ outside this region.

$$\rho(\boldsymbol{x})\ddot{\boldsymbol{u}}(\boldsymbol{x}, t) = \int_{\mathcal{R}} \boldsymbol{f}\Big(\boldsymbol{u}(\boldsymbol{x'}, t) - \boldsymbol{u}(\boldsymbol{x}, t), \boldsymbol{x'} - \boldsymbol{x}\Big) dV_{x'} + \boldsymbol{b}(\boldsymbol{x}, t) \tag{3.3}$$

It can be shown that in the limit of an infinitely fine set of points, the integral equations

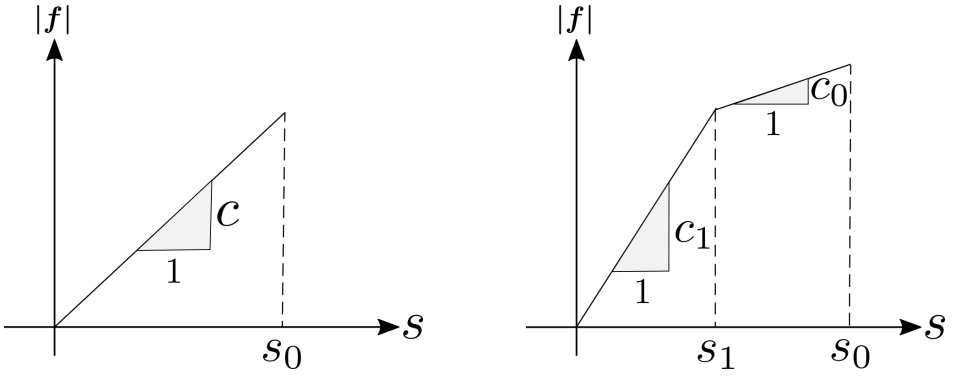
of PD are mathematically equivalent to the classical continuum mechanics formulation [12].

### 3.3.1 Bond Based PD

The form of PD employed in this work is the so called *bond based* model. This is the mathematically simplest form of Peridynamics and was the form proposed in the original PD paper by Silling (2000) [2]. In this form, the magnitude of the force density function $\boldsymbol{f}$ is only a function of the scalar valued strain of the bond between two material points (see Figure 3.3). The relationship is given in equation 3.4, where $c$ is the bond stiffness and has units of $N/m^6$. $\mu$ is a discrete variable which records whether a bond is broken or intact. Initially $\mu = 1$, however if at any point the stretch $s$ exceeds some critical value $s_0$, $\mu$ is set to 0 and the bond is broken. Damage is permanent and a broken bond remains broken for the remainder of the simulation.

$$s = \frac{|\boldsymbol{\xi} + \boldsymbol{\eta}| - |\boldsymbol{\xi}|}{|\boldsymbol{\xi}|} \qquad \boldsymbol{f} = \frac{\boldsymbol{\xi} + \boldsymbol{\eta}}{|\boldsymbol{\xi} + \boldsymbol{\eta}|} cs\mu \tag{3.4}$$

In some aspects bond based PD is limited by its simplicity. It can be shown that the form of $\boldsymbol{f}$ chosen restricts the global material Poisson's ratio $\nu$ to a value of 0.25 for 3D problems. The linear force density function also fails to encapsulate material behaviour such as plasticity. The major advantage of the bond based PD formulation is its comparative computational simplicity. There are many other PD formulations that seek to address the shortcomings of the bond based model, however these are in general more mathematically complex and computationally expensive [13] [14]. These formulations are not addressed further in this work.




(a) Linear elastic force density function (b) Bi-linear force density function

Figure 3.3: Example forms of $\boldsymbol{f}$

### 3.3.2 Defining Material Properties

By employing the linear form of $\boldsymbol{f}$ shown in Figure 3.3a, the function is parameterised by two variables: the stiffness $c$ and the critical stretch $s_0$. By relating the strain energy density within the Peridynamic model to the classical definition of strain energy density, the relationships shown in equation 3.5 can be derived [15]. Here $E$ is the Young's modulus of the material, $G_0$ is the fracture energy release rate and $\delta$ is the Peridynamic horizon.

$$c = \frac{12E}{\pi\delta^4} \qquad s_0 = \sqrt{\frac{5G_0}{6E\delta}} \tag{3.5}$$

## 3.4 Numerical Solutions of Peridynamics

As there are in general no analytical solutions to PD problems, it is therefore necessary to employ numerical techniques, typically through bespoke simulation software.

### 3.4.1 Spatial Discretisation

In order to numerically approximate the volume integral in equation 3.3 the material being modelled is discretised into a set of regions of finite volume. In the discretised material, such regions are referred to as *cells* and the centre of each cell is called a *node*. In the discretised material, node number $i$ is denoted $\boldsymbol{x}_i$. For every node $\boldsymbol{x}_i$ there exists a set a of neighbouring nodes $N_i$ which all lie within the horizon of $\boldsymbol{x_i}$. The force density function $\boldsymbol{f}$ now only needs to be calculated between pairs of neighbouring

nodes, and the integral in equation 3.3 is replaced by a sum over neighbour nodes. This results in the spatially discrete formulation given in equation 3.6, where $V_i(\boldsymbol{x}_j)$ is the volume of the cell occupied by node $j$ which lies inside the horizon of node $i$.

$$\rho(\boldsymbol{x}_i)\ddot{\boldsymbol{u}}(\boldsymbol{x}_i,t) = \sum_{j:\boldsymbol{x}_j \in N_i} \boldsymbol{f}\Big(\boldsymbol{u}(\boldsymbol{x}_j,t) - \boldsymbol{u}(\boldsymbol{x}_i,t), \boldsymbol{x}_j - \boldsymbol{x}_i\Big) V_i(\boldsymbol{x}_j) + \boldsymbol{b}(\boldsymbol{x}_i,t) \tag{3.6}$$

Both regular and irregular node placements are possible. The former arranges cells and nodes in a regular grid pattern. This approach is computationally straightforward, but can cause non-physical effects, such as cracking being constrained by the grid structure. The form of discretisation described here, and used in this work, is known as the *meshfree method* [15].

Computational cost is heavily dependent on the coarseness of the discretisation employed. Letting $d$ denote the average number of nodes per spatial dimension, the number of nodes per unit volume, and hence the total number of nodes in the simulation, grows as $\mathcal{O}(d^3)$. For any given node $\boldsymbol{x}_i$, the number of nodes in the set of neighbours $N_i$ also scales as $\mathcal{O}(d^3)$. Evaluating equation 3.6 for a single node then becomes an $\mathcal{O}(d^3)$ operation. The number of operations per simulation time step hence grows as $\mathcal{O}(d^6)$. This property of the numerical form of the governing equation means that simulations quickly become computationally infeasible for fine discretisation schemes.

### 3.4.2 Viscoelastic Approximations

In the form presented in equation 3.6, there is no energy dissipation in the Peridynamic system. This means that any structural member modelled will behave in an undamped manner and harmonically oscillate indefinitely. Under such conditions, simulations will always fail to converge to the static solutions of interest in RC beam problems.

There are several ways of introducing damping to the simulation, two of which are outlined here [16]. The first option is to make the force density function $\boldsymbol{f}$ depend on the rate of change of stretch $\dot{s}$, hence introducing viscoelastic behaviour. In this work, a computationally simpler approach is employed called *local damping*. Here damping is instead considered at the node level by introducing a damping term to equation 3.6, giving equation 3.7. The damping coefficient $C$ has units of $kg/m^3s$ and is chosen as a simulation parameter.

$$\rho(\boldsymbol{x}_i)\ddot{\boldsymbol{u}}(\boldsymbol{x}_i,t) = \sum_{j:\boldsymbol{x}_j \in N_i} \boldsymbol{f}\Big(\boldsymbol{u}(\boldsymbol{x}_j,t) - \boldsymbol{u}(\boldsymbol{x}_i,t), \boldsymbol{x}_j - \boldsymbol{x}_i\Big) V_i(\boldsymbol{x}_j) + \boldsymbol{b}(\boldsymbol{x}_i,t) - C\dot{\boldsymbol{u}}(\boldsymbol{x}_i,t) \quad (3.7)$$

### 3.4.3 Time Integration

Simulating the evolution of the model from its initial conditions requires some form of numerical time integration scheme. Both *explicit* and *implicit* schemes are possible [17]. In explicit schemes, the value of a function at the $n^{th}$ time step $t_n$ is only dependent on the value of the function and its derivatives at time step $t_{n-1}$ and before. Explicit schemes allow function values to be calculated directly using some pre-determined update formula, but are susceptible to numerical instability if the time step size $\Delta t$ is too large. In implicit schemes the function value at time $t_n$ can depend on function and derivative values at later times. Solving for the state of the system then amounts to solving sets of coupled equations, typically using iterative matrix methods. Implicit schemes are unconditionally stable for linear problems, but come at an increased computational cost.

Explicit time integration has been used in this work, in particular the *forward Euler* method was employed. This gives the relationship between node velocity $\dot{\boldsymbol{u}}$ and acceleration $\ddot{\boldsymbol{u}}$ in equation 3.8. The same update rule is used to calculate node displacements from velocities. Letting the acceleration be a function of the state of the model at the current time step gives the complete numerical PD formulation in equation 3.9.

$$\dot{\boldsymbol{u}}(\boldsymbol{x}_i, t + \Delta t) = \ddot{\boldsymbol{u}}(\boldsymbol{x}_i, t)\Delta t + \dot{\boldsymbol{u}}(\boldsymbol{x}_i, t) \quad (3.8)$$

$$\rho(\boldsymbol{x}_i)\ddot{\boldsymbol{u}}(\boldsymbol{x}_i,t) = \sum_{j:\boldsymbol{x}_j \in N_i} \boldsymbol{f}\Big(\boldsymbol{u}(\boldsymbol{x}_j,t) - \boldsymbol{u}(\boldsymbol{x}_i,t), \boldsymbol{x}_j - \boldsymbol{x}_i\Big) V(\boldsymbol{x}_j) + \boldsymbol{b}(\boldsymbol{x}_i,t) - C\dot{\boldsymbol{u}}(\boldsymbol{x}_i,t) \quad (3.9)$$

To ensure numerical stability for the explicit time integration it is necessary to restrict the size of the simulation time step $\Delta t$. Through stability analysis it can be shown that a conservative bound on the critical time step size is given by equation 3.10 for

linear three dimensional problems [15]. Here $\alpha$ is a safety factor $< 1.0$ to account for non-linear material behaviour, and $c_{ij}$ is the stiffness of the bond between node number $i$ and $j$. Note that the critical time step is a function of the conditions around the chosen reference node $\boldsymbol{x}_i$. The value for all nodes is calculated and the smallest value of $\Delta t_{crit}$ bounds the global simulation time step.

$$\Delta t \ \leq \ \Delta t_{crit}(\boldsymbol{x}_i) \ = \ \alpha \sqrt{\frac{2\rho(\boldsymbol{x}_i)}{\sum_{j:\boldsymbol{x}_j \in N_i} V_i(\boldsymbol{x}_j) c_{ij}}} \qquad \forall \ \boldsymbol{x}_i \tag{3.10}$$

## 3.5 Defining Failure in PD

Although the global behaviour of a modelled beam is described by the mathematical framework laid out in section 3.4, the concept of structural failure is not defined. Several approaches have been proposed in the literature for defining structural failure in a PD simulation. Most approaches define some form of criterion for when a simulation has *converged* to a static solution. If no convergence is seen within some preset number of simulation time steps, the member is deemed to have failed.

The simplest approach is based on the assumption that the modelled body behaves as a harmonic system oscillating about some equilibrium position. If such a position exists the simulation will over time converge to it in the presence of damping [16]. A simple convergence criterion is then given in equation 3.11, where $B$ is the set of loaded nodes, $\tau_{\dot{u}}$ is a simulation parameter, $|B|$ is the number of loaded nodes and $||\ ||_2$ denotes the Euclidean norm of a vector. This criterion measures the average velocity of loaded nodes and will be referred to in this work as the *velocity decay* criterion.

$$\frac{1}{|B|} \sum_{i:\boldsymbol{x}_i \in B} ||\dot{\boldsymbol{u}}(\boldsymbol{x}_i, t)||_2 \ \leq \ \tau_{\dot{u}} \tag{3.11}$$

A more complex convergence criterion is proposed by Huang et al. (2015) [18]. This criterion is defined by a force ratio on some chosen node $\boldsymbol{x}_i$. To this end, the scalar valued function $F(\boldsymbol{x}_i)$ is defined in equation 3.12. By substituting in equation 3.9 and averaging over all loaded nodes, the global convergence criterion is defined in equation 3.13, where $\beta$ is a simulation parameter. This criterion measures the relative magnitude of non-equilibrium forces and will be referred to in this work as the *force*

*decay* criterion.

$$F(\boldsymbol{x}_i) = \frac{||\sum_{j:\boldsymbol{x}_j \in N_i} \boldsymbol{f}\Big(\boldsymbol{u}(\boldsymbol{x}_j,t) - \boldsymbol{u}(\boldsymbol{x}_i,t), \boldsymbol{x}_j - \boldsymbol{x}_i\Big) V_i(\boldsymbol{x}_j) + \boldsymbol{b}(\boldsymbol{x}_i,t)||_2}{||\boldsymbol{b}(\boldsymbol{x}_i,t)||_2} \tag{3.12}$$

$$\frac{1}{|B|}\sum_{i:\boldsymbol{x}_i \in B} F(\boldsymbol{x}_i) = \frac{1}{|B|}\sum_{i:\boldsymbol{x}_i \in B} \frac{||\rho(\boldsymbol{x})_i \ddot{\boldsymbol{u}}(\boldsymbol{x}_i,t) + C\dot{u}(\boldsymbol{x}_i,t)||_2}{||\boldsymbol{b}(\boldsymbol{x}_i,t)||_2} \leq \beta \tag{3.13}$$

## 3.6 Research Focus

There exists a lot of literature demonstrating desirable properties of the underlying mathematics of PD, such as the limiting convergence to classical continuum mechanics. However, few papers have focused on the practical implementation of PD for making predictions about the behaviour of real structural members. Comparing model predictions about RC beams to laboratory tests has therefore been a central focus of the work presented here.

The large computational cost of PD simulations is arguably the single biggest limitation for PD as a practical tool in engineering design. The aim of this work has therefore been to investigate a practical implementation of PD software. The PD model chosen and the approximations employed were selected for computational efficiency. Since the computational cost of simulations depends so heavily on the coarseness of the spatial discretisation scheme employed (see section 3.4.1), an important question is how this coarseness affects model predictions of failure load and beam stiffness.

In order to make predictions on the failure behaviour of beams, it is necessary to define mathematically what is meant by failure. Although there have been some attempts in existing literature to address this, no universally agreed method exists. Investigating different definitions of failure has therefore been important.

**Central questions:**

- Can a coarse spatial discretisation scheme still produce physically realistic model predictions of failure behaviour for RC beams?
- What constitutes a rigorous and general definition of failure in the context of a PD simulation?

# 4 Peridynamics Software Implementation

All the PD software developed for this project was written by the author. The language used was C++ and the structure of the code was based on a previous MATLAB implementation by Mark Hobbs [19]. The author believes that the code produced will serve as a versatile PD implementation suitable for a range of experiments. The implementation seeks to be modular and flexible. The three main stages in the simulation procedure are separated, and a high level breakdown is shown in Figure 4.1. The full code and further notes on implementation and use has been made available on the author's GitHub repository [2].

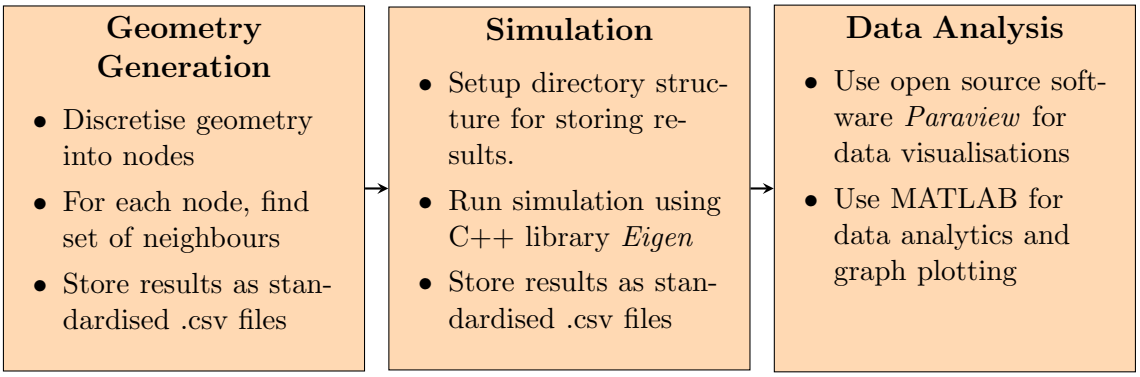



Figure 4.1: High level view of the Peridynamics simulation software developed

The choice of C++ as the modelling language was motivated by the potential for computational efficiency. MATLAB is a high-level interpreted language, whereas C++ is compiled. MATLAB is hence useful for rapid code development and prototyping, however it suffers from computational inefficiencies. For the type of simulation software implemented in this work, C++ could outperform MATLAB by two orders of magnitude in terms of computational time [20]. Additionally, the linear algebra library *Eigen* was used in the writing of the simulation script. Eigen is a free library used widely in numerical and scientific computing research projects, including Google's TensorFlow development and data analysis at CERN's ATLAS particle accelerator [21]. Table 4.1 shows a speed comparisons of a test problem in MATLAB and C++ using Eigen. The problem consists of element-wise multiplication of two large matrices.

[2] https://github.com/prebenness/peridynamics

Table 4.1: Time comparison of element-wise multiplication of two $[D \times D]$ dimensional matrices using MATLAB and the C++ library Eigen. Times in seconds

| **Dimension D** | **MATLAB** | **Eigen C++** | **Speedup Factor** |
|---|---|---|---|
| 100 | $8.7 \times 10^{-3}$ | $9.0 \times 10^{-5}$ | 96.7 |
| 500 | $9.6 \times 10^{-3}$ | $2.7 \times 10^{-3}$ | 3.6 |
| 1000 | $3.4 \times 10^{-2}$ | $1.0 \times 10^{-2}$ | 3.4 |
| 5000 | 2.5 | $2.0 \times 10^{-1}$ | 12.5 |
| 1000 | $1.7 \times 10^{1}$ | $8.7 \times 10^{-1}$ | 19.5 |
| 15000 | $7.7 \times 10^{1}$ | $3.6 \times 10^{1}$ | 2.1 |

## 4.1 Data Structures and Computations

In order to store and manipulate the state of the simulation, a set of data structures were created. The most central are outlined here. The class *sparse matrix* from the Eigen library was utilised extensively. This class allows highly efficient computations using large matrices where only a small subset of entries are non-zero. In this section $N$ denotes the total number of nodes in a simulation.

- **The Connectivity matrix $\boldsymbol{A}$** $[N \times N]$
  The sparse connectivity matrix is defined such that $A_{ij} = 1$ if node $i$ is connected to node $j$ and 0 otherwise. By using built-in methods in Eigen for iterating over the coefficients of the non-zero entries in a sparse matrix, it is possible to efficiently loop over all the bonds in the simulation. For efficient memory use only the upper triangular half of $\boldsymbol{A}$ is stored, since the matrix is by definition symmetric.

- **Node Vectors $\boldsymbol{V}$** $[3 \times N]$
  These dense matrices are collections of column vectors, storing one vector value per node. These are used for example for node displacements, velocities and accelerations. This structure allows the time integration step to be vectorised efficiently as shown in equation 4.1, where $\boldsymbol{V}_d$ are node displacements and $\boldsymbol{V}_v$ are node velocities.

$$\boldsymbol{V}_d(t_{n+1}) = \boldsymbol{V}_d(t_n) + \Delta t \boldsymbol{V}_v(t_n) \tag{4.1}$$

## 4.2 Modelling Choices

This work implemented a bond-based PD simulation. In order to address common hazards in particle modelling in general and PD in particular, some minor modifications were made to the standard PD formulation. These were implemented to increase the numerical stability and physical accuracy of the model, and are outlined here. These corrective measures are based on both on theoretical analysis and empirical findings [3] [22].

- **Load Buildup**

  It has been found that applying external loads fully at the first simulation time step can lead to unphysical beam responses, such as excessive strain wave propagations. An example of this is illustrated in Figure 4.2 which shows a simulation using an early implementation of the author's code. In order to mitigate this hazard, applied loads are built up linearly from time $t = 0$ until $t = t_{build}$, where $t_{build}$ is a simulation parameter.

- **No-Fail Region**

  In order to avoid unphysical failure from tear off at restrained nodes, a so called *no-fail region* is enforced around these nodes. Any bond which is connected to a node that lies within the no-fail region will not break, regardless of whether or not the stretch $s$ exceeds $s_0$.

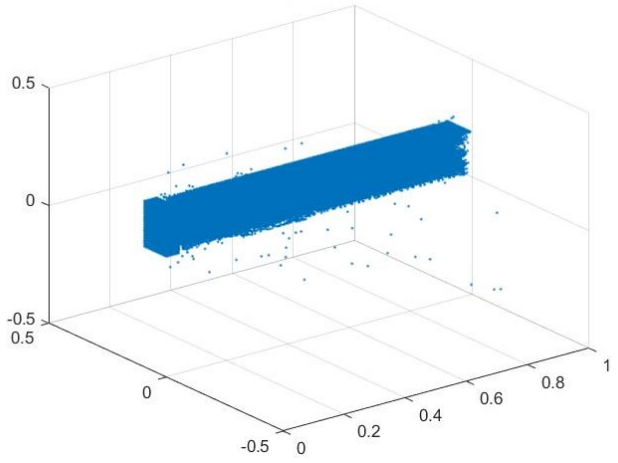



Figure 4.2: Unphysical simulation predictions from strain wave propagation

## 4.3 Simulation Overview

The goal of the simulation was to predict the failure load of modelled RC beams. Figure 4.3 shows a high-level overview of the simulation procedure. There are two main iteration loops, the inner is over time steps for a specific loading, and the outer loop is over test loads. For a given test load, the simulation continues until the beam has reached a criterion for convergence, or a preset number of time steps $t_{max}$ has been exceeded.

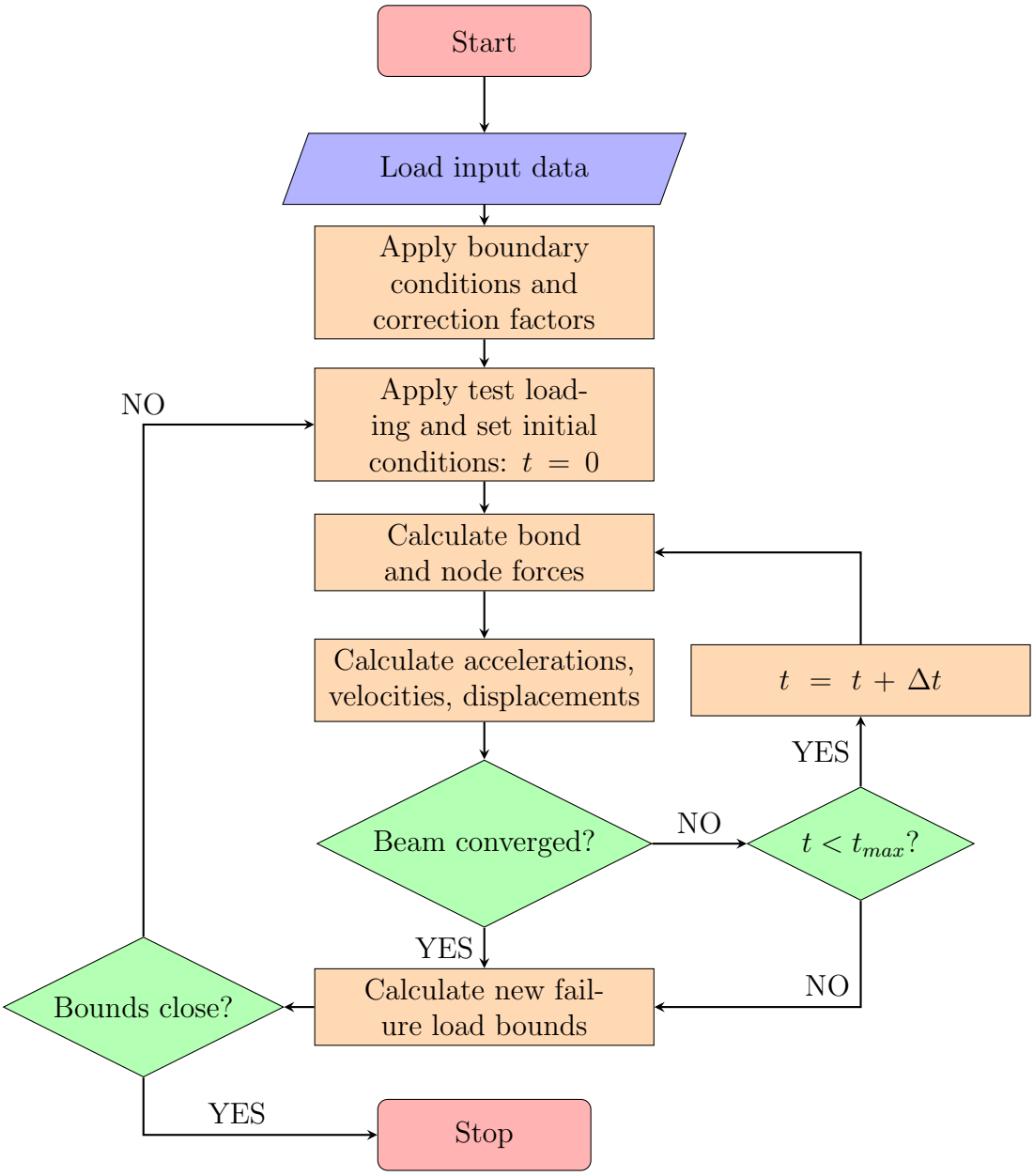



Figure 4.3: Simplified overview of the simulation procedure for finding beam failure loads.

### 4.3.1 Failure Search

The structure of the simulation implemented is such that for any given test load, a binary result of *true* or *false* is returned when assessing failure. This result is returned at the latest when the time step budget is exceeded. The simulation is therefore guaranteed to compute a failure load prediction in a finite amount of time.

To estimate the failure load of a beam, a geometric binary search was conducted, details of which are given in Figure 4.4. The reason for using this particular form of search was that it allows test loads to span a large range of values. This is useful when the variance in failure predictions is large, such as during testing of highly coarse models or unstable convergence criteria.

**Algorithm 1** Geometric Binary Search

```
1:  procedure GeoBin(τ)                          ▷ Specified tolerance τ
2:      U ← U_0                  ▷ Initial values for upper and lower bounds
3:      L ← L_0
4:      T ← √(LU)                    ▷ New test load computed from bounds
5:      if Beam failed then
6:          U ← T
7:      else
8:          L ← T
9:      end if
10:     if U − L > τ then                ▷ Continue until specified tolerance
11:         go to 4
12:     else
13:         return U, L
14:     end if
15: end procedure
```

Figure 4.4: Pseudocode showing the algorithm used for failure load search

### 4.3.2 Failure Criteria

Three different criteria for assessing convergence or failure were implemented in the code. The first two were the velocity decay and the force decay criteria described in section 3.5. The third criterion, referred to as the *bond damage* criterion, is proposed by the author and defined in this section.

The full bond damage criterion is shown in pseudocode in Figure 4.5. $\Delta_{\text{Damage}}$ denotes

the increase in the fraction of broken bonds between the current and the previous time step. $S^*$ and $F^*$ are user-defined parameters, but are taken as $2 \cdot t_{build}$ in this implementation. $||\overline{\dot{\boldsymbol{u}}}||$ is shorthand for the average absolute velocity of loaded nodes, which is the same quantity as is used in the velocity decay criterion in equation 3.11. Finally, $\sum \dot{\boldsymbol{u}} \cdot \boldsymbol{b}$ denotes the sum over all nodes of the dot product of the node velocity and the applied node force vector.

The key idea in this scheme is to only make assessments after bonds have stopped breaking. The algorithm consists of three distinct states defined by the behaviour of the simulated beam. When $\Delta_{\text{Damage}}$ has remained below the tolerance value $\tau_D$ for $S^*$ consecutive time steps the beam is classified as *stable*. The next step is then checking whether node velocities and applied loads are aligned or in opposite directions. If the directions are aligned for $F^*$ consecutive time steps, the beam is classified as *failed* and the algorithm terminates. If the directions are not aligned, the velocity decay criterion is used to check for convergence. Upon classification as *converged* the algorithm terminates.

**Algorithm 2** Bond Damage Convergence Criterion

**procedure** BONDDAMAGE($\tau_D, \tau_{\dot{u}}$) ▷ Tolerance parameters $\tau_D$ and $\tau_{\dot{u}}$
    $D \leftarrow$ *False*
    $F, S, t \leftarrow 0$

    **if** $\Delta_{\text{Damage}} < \tau_D$ **then** ▷ If increase in damage is small
        $S \leftarrow S + 1$
    **else**
        $S \leftarrow 0$
    **end if**
    **if** $S \geq S^*$ **then** ▷ Damage stable for $S^*$ time steps
        $D \leftarrow$ *True* ▷ Beam is *stable*
    **end if**

    **if** $D$ **then**
        **if** $\sum \dot{\boldsymbol{u}} \cdot \boldsymbol{b} < 0$ **then** ▷ Force and velocity in opposite directions
            $F \leftarrow 0$
            **if** $||\bar{\dot{\boldsymbol{u}}}|| < \tau_{\dot{u}}$ **then** ▷ Velocity decay criterion
                **return** *converged* $\leftarrow$ *True*
            **end if**
        **else**
            $F \leftarrow F + 1$
        **end if**
        **if** $F \geq F^*$ **then** ▷ Failing for $F^*$ time steps
            **return** *converged* $\leftarrow$ *False*
        **end if**
    **end if**

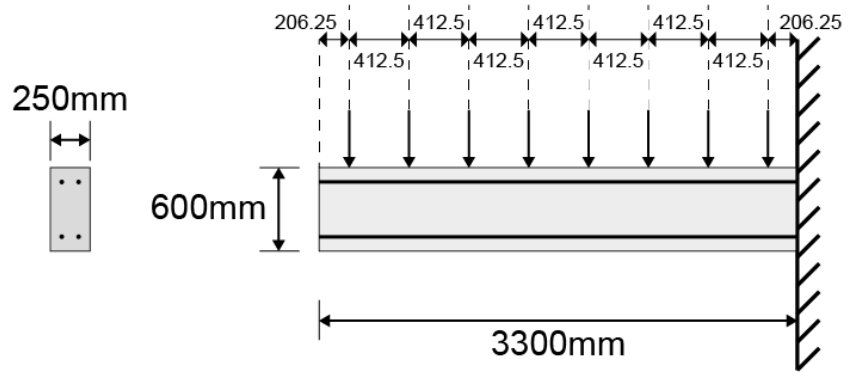
    **if** $t < t_{max}$ **then**
        $t \leftarrow t + 1$
        **call** *SimulateNextTimeStep()*
        **go to** 4
    **else** ▷ If not converged at $t = t_{max}$, beam failed
        **return** *converged* $\leftarrow$ *False*
    **end if**

**end procedure**

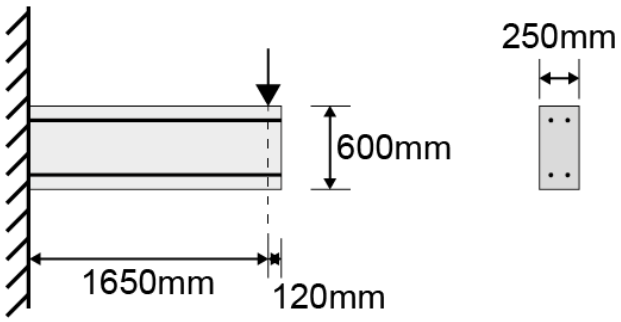
Figure 4.5: Pseudocode for the Bond Damage convergence criterion

# 5 Experiments

In order to investigate the questions identified in section 3.6 the PD simulation predictions were compared with laboratory tests of two cantilevered prismatic RC beams. The beams and loading arrangements are shown in Figure 5.1. These particular tests were picked as comparisons because of the shear dominated failure modes. Each beam was tested twice in the laboratory, using concrete of cylinder strengths 31.1 MPa and 33.5 MPa.




(a) Test 1A, distributed loading




(b) Test 1B, tip loading

Figure 5.1: Laboratory tested beams used as comparisons for simulations

The goal of the experiments were to assess both the dependence of predictions on spatial discretisation coarseness as well as on the chosen convergence criterion. The fraction of broken bonds, as well as displacements and velocities of loaded nodes were

recorded during the simulation. These values were central to the convergence criteria employed and therefore of interest. Additionally, the force-displacement behaviour of the beams were examined.

## 5.1 Experimental Setup

*All simulations and predictions were performed without knowledge of the real failure loads from the laboratory tests.*

The simulations were conducted using the bond-based PD implementation written by the author and described in section 4. The simulations were performed at different levels of discretisation coarseness, with the total number of nodes in the models ranging from 792 to 6256. Unless otherwise stated, the following simulation parameters were used: The build up time $t_{build}$ was set to 500 iterations, and a maximum time step budget $t_{max}$ of 20 000 iterations was allowed. Checks for convergence were performed and data was logged every $50^{th}$ time step.

There is no direct way of setting the strength of the simulated concrete in this simple PD implementation. The ultimate tensile bond stretch value $s_0$ is calculated based on the PD horizon and material properties as described in section 3.3.2. The calculated value was based on a concrete fracture energy release rate of $G_0 = 16.0\ Nm/m^2$ [23]. Whether this corresponds more closely to a concrete cylinder strength of 31.1 MPa or 33.5 MPa is unclear as the precise relationship between these two material quantities is a topic of some uncertainty in the scientific literature [24] [25]. Both laboratory tests are therefore given as comparisons for the simulation predictions in section 6.2. All values given include self weight [3].

[3]The simulation does not include gravity, but the laboratory and simulation loads both denote the total sum of forces applied to the beam, and are comparable

# 6 Results

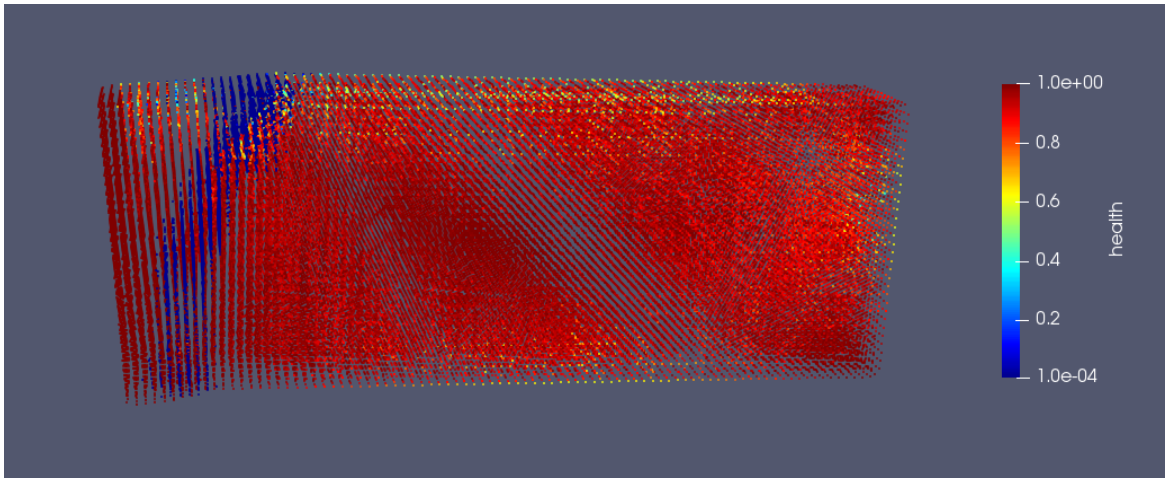



Figure 6.1: Failure mode of beam 1B tested at 400 $kN$. Points represent model bonds, and broken bonds are coloured dark blue

Figure 6.1 shows an image from the simulation of beam 1B loaded at 400 $kN$ using the 6256-node model. This test loading resulted in structural failure, and the state of model bonds are shown. The health rating indicates how close a bond is to failure, with 1.0 indicating $s = 0.0$ and 0.0 indicating $s = s_0$ [4].

## 6.1 Dynamic Response

All graphs presented in this section show the data logged during the simulation of beam 1B at a spatial discretisation level of 6256 nodes. The beam was loaded at various test loads according to the geometric binary search described in section 4.3.1. Tests that were deemed to cause structural failure are plotted in red and tests where the beam survived the loading are plotted in green. Failure was assessed in this section using the bond damage convergence criterion.

The displacement and velocities of loaded nodes were recorded. Figure 6.2 plots displacements and Figure 6.3 plots velocities throughout the simulation. In Figure 6.3b the evolution of the average loaded node velocity during the initial part of the simulation is shown in detail. Note that the beam loading in all cases is applied in the negative $z$-direction, most plotted values are therefore negative.

[4]Due to numerical inaccuracies in the simulation, for bonds with a health rating of zero the value stored is $1.0 \times 10^{-4}$

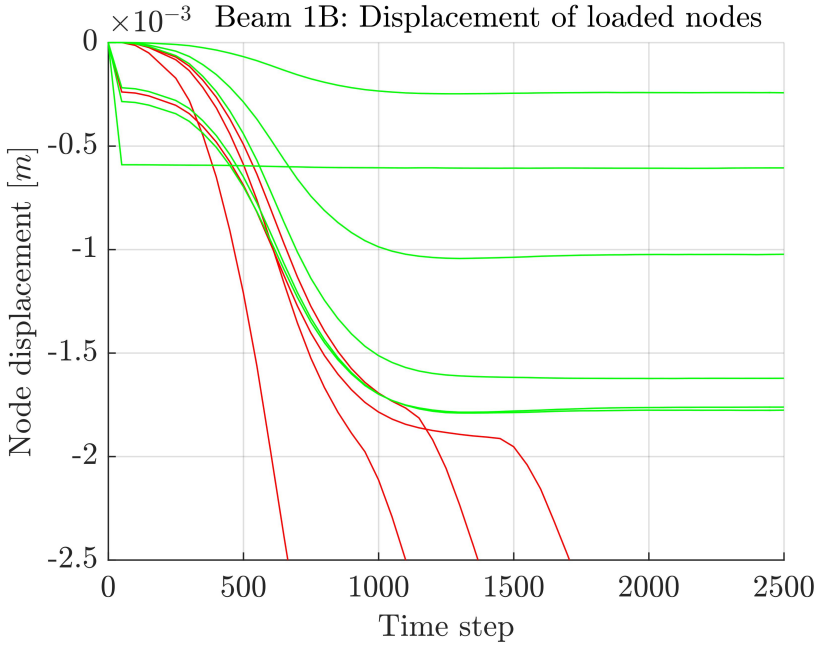
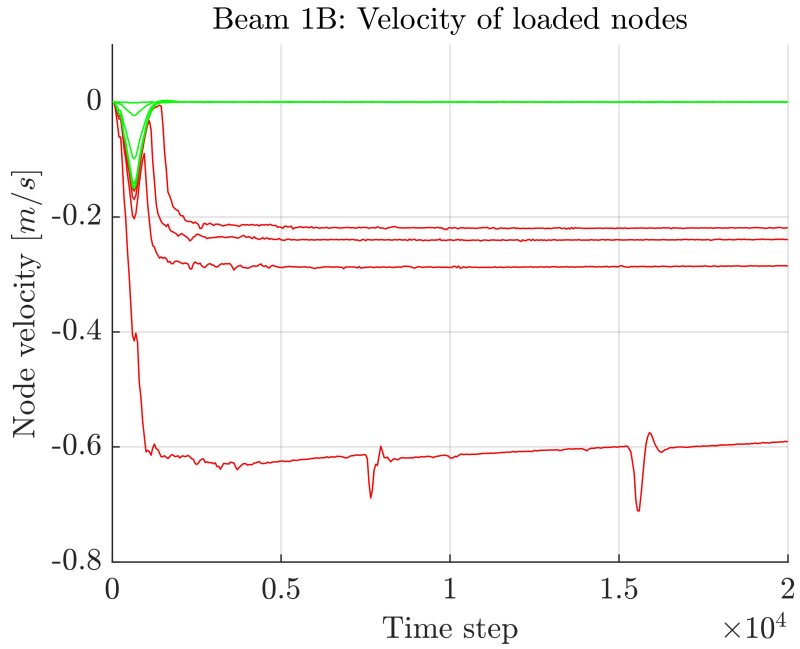
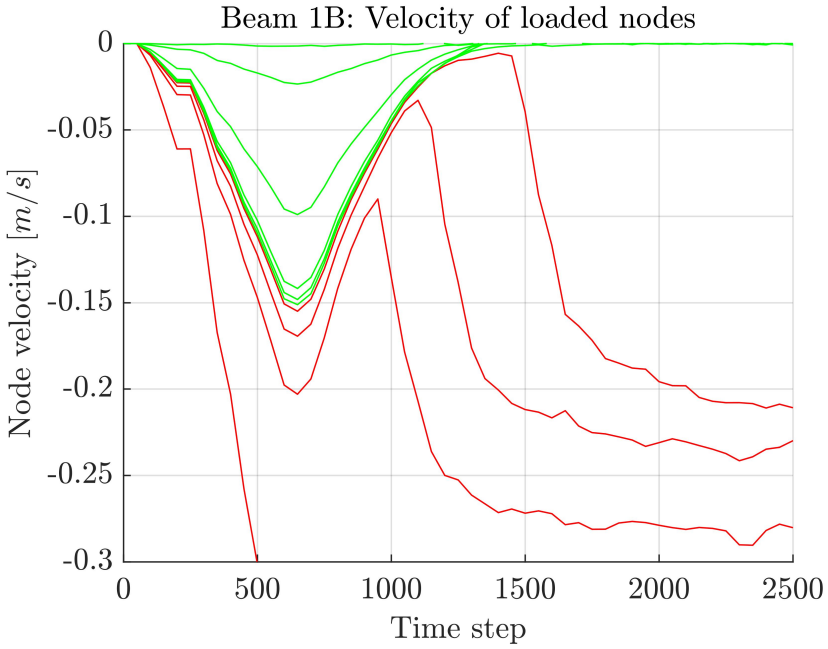



Figure 6.2: Node displacements during the first 2500 time steps of the simulation for beam 1B




(a) Average velocity of loaded nodes during the entire simulation of beam 1B



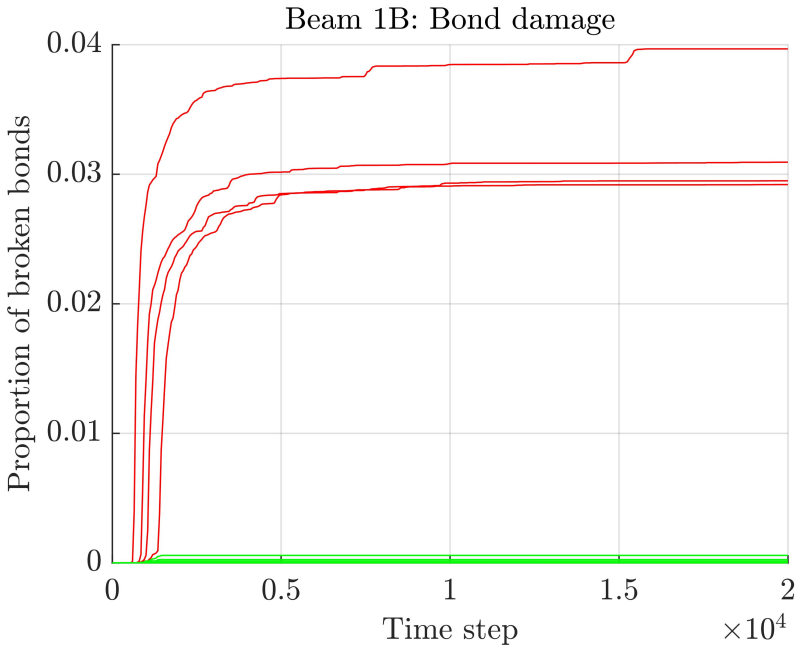

(b) Detailed view of the first 2500 simulation time steps

Figure 6.3: Average loaded node velocity for beam 1B

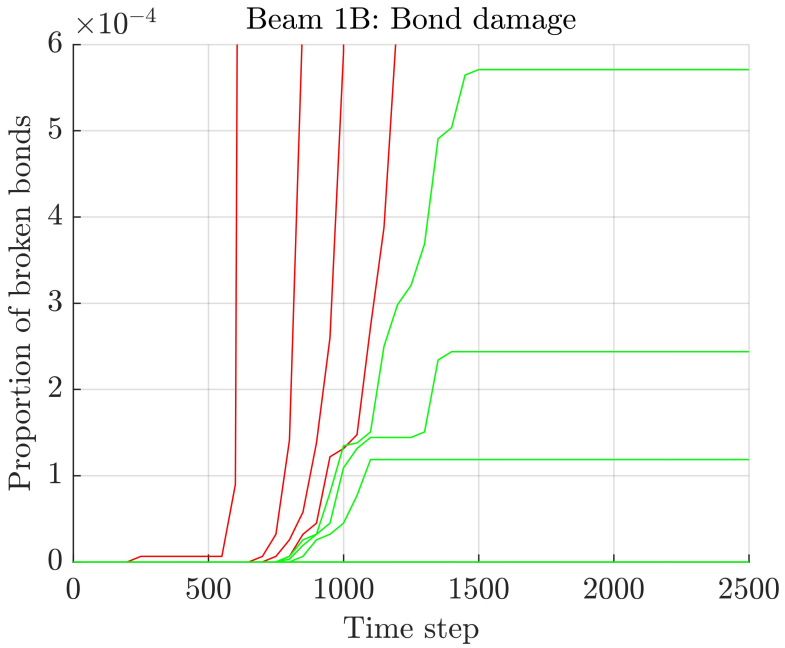
### 6.1.1 Damage and Failure

The proportion of total bonds broken since the start of the simulation was used as a metric to describe failure in the bond damage criterion. Figure 6.4 Shows the evolution

of this quantity as the simulation progressed. Figure 6.4b shows in more detail the development during the first part of the simulation.




(a) Proportion of broken bonds during the entire simulation of beam 1B




(b) Detailed view of the first 2500 simulation time steps

Figure 6.4: Proportion of broken bonds during simulation of beam 1B

### 6.1.2 Force versus Displacement

For the tests that converged to a static solution, the average magnitude of the final displacement of the loaded nodes was recorded. This is plotted as a function of the magnitude of the applied load for beam 1B in Figure 6.5, at varying discretisation coarseness levels.

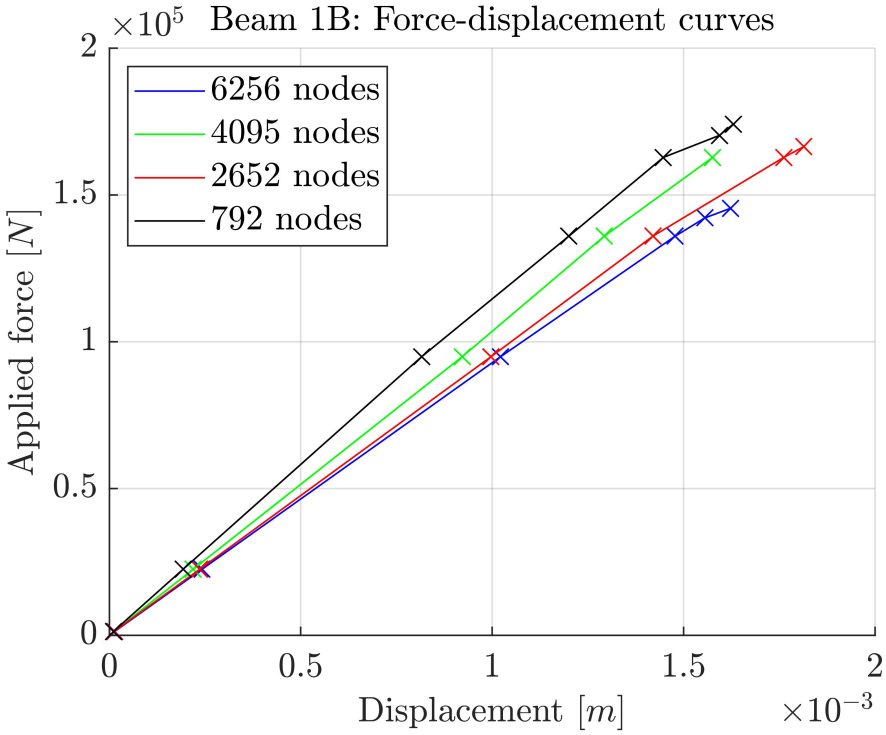



Figure 6.5: Force-displacement behaviour of beam 1B at varying discretisation coarseness levels

## 6.2 Prediction of Failure Loads

Using the three convergence criteria described in sections 3.5 and 4.3.2, the failure loads of beams 1A and 1B were predicted. The maximum iteration number $t_{max}$ was set as 5 000.

### 6.2.1 Velocity Decay

Using the initially proposed criterion of decaying velocities, the failure loads of the two beams were assessed. The convergence tolerance was set to $\tau_{\dot{u}} = 0.5$, and the results are shown in Table 6.1. In order to examine the sensitivity of the predictions

to the tolerance parameter of the convergence criterion, a range of different values were tested. The 2970-node model of beam 1A was used for this investigation and the results are shown in Table 6.2.

Table 6.1: Predicted failure load ranges in $kN$ as a function of number of nodes in the simulation. The velocity decay convergence criterion was used

| **TEST 1A** | | **TEST 1B** | |
|---|---|---|---|
| **Num. nodes** | **Pred. failure load** | **Num. nodes** | **Pred. failure load** |
| 952 | 961.4 - 983.3 | 792 | 203.7 - 208.4 |
| 2970 | 878.7 - 898.7 | 2652 | 182.1 - 186.2 |
| 4392 | 840.1 - 859.2 | 4095 | 190.5 - 194.8 |
| 6048 | 961.4 - 983.3 | 6256 | 190.5 - 194.8 |
| **Actual failure load** $f_{ck} = 31.1$ MPa | **174** | | **132** |
| **Actual failure load** $f_{ck} = 33.5$ MPa | **190** | | **154** |

Table 6.2: Sensitivity analysis of the velocity decay convergence tolerance. The 2970-node model of beam 1A was used for the simulations. Loads are displayed in $kN$

| **Tolerance** $\tau_{\dot{u}}$ | **Pred. failure load** |
|---|---|
| $7.2 \times 10^{-1}$ | 1203.9 - 1231.2 |
| $5.0 \times 10^{-1}$ | 878.7 - 898.7 |
| $7.2 \times 10^{-2}$ | 312.3 - 319.5 |
| $7.2 \times 10^{-3}$ | 116.1 - 118.8 |

### 6.2.2 Force Decay

Predictions of failure loads were repeated using the force decay criterion with a tolerance parameter of $\beta = 0.5$.

Table 6.3: Predicted failure load ranges in $kN$ as a function of number of nodes in the simulation. The force decay convergence criterion was used

| **TEST 1A** | | **TEST 1B** | |
|---|---|---|---|
| **Num. nodes** | **Pred. failure load** | **Num. nodes** | **Pred. failure load** |
| 952 | 7440.3 - 7609.5 | 792 | 1725.2 - 1764.4 |
| 2970 | 6800.3 - 6955.0 | 2652 | 785.3 - 803.1 |
| 4392 | 5942.0 - 6077.2 | 4095 | 686.2 - 701.8 |
| 6048 | 5680.7 - 5809.9 | 6256 | 500.9 - 512.3 |
| **Actual failure load** $f_{ck} = 31.1$ MPa | **174** | | **132** |
| **Actual failure load** $f_{ck} = 33.5$ MPa | **190** | | **154** |

### 6.2.3 Bond Damage

Finally, failure predictions were made using the proposed bond damage criterion. Parameter values were set as $\tau_{\dot{u}} = 0.5$ and $\tau_D = 10^{-5}$. The outlier prediction for the 4392-node model of beam 1A was repeated using the parameters $\tau_{\dot{u}} = 5.0$ and $\tau_D = 10^{-4}$, and $t_{max} = 10\ 000$ but the same failure predictions were obtained.

Table 6.4: Predicted failure load ranges in $kN$ as a function of number of nodes in the simulation. The bond damage convergence criterion was used

| **TEST 1A** | | **TEST 1B** | |
|---|---|---|---|
| **Num. nodes** | **Pred. failure load** | **Num. nodes** | **Pred. failure load** |
| 952 | 194.8 - 199.2 | 792 | 174.1 - 178.0 |
| 2970 | 142.2 - 143.4 | 2652 | 166.4 - 170.2 |
| 4392 | 2.3 - 2.3 | 4095 | 162.7 - 166.4 |
| 6048 | 166.4 - 170.2 | 6256 | 145.4 - 148.7 |
| **Actual failure load** $f_{ck} = 31.1$ MPa | **174** | | **132** |
| **Actual failure load** $f_{ck} = 33.5$ MPa | **190** | | **154** |

# 7 Discussion

The distribution of bond failures shown in Figure 6.1 indicates that the simulation accurately captures the shear based failure mode of the tested beams. Although this simple simulation does not include plasticity, and hence not steel yielding, the qualitative aspects of failure appear to be well simulated. Another shortcoming of the simulation is that there are no restrictions stopping node collisions, meaning that different material regions can pass through each other. This is of course unphysical, but eliminating the checks on allowed node positions makes the simulation computationally cheaper.

## 7.1 Failure and Convergence

**Velocity Decay**

The sensitivity analysis reported in Table 6.2 suggests that the failure load predictions using the velocity decay criterion are highly dependent on the tolerance parameter $\tau_{\dot{u}}$. This is of course undesirable as predictions should depend on the behaviour of the modelled beam, and not on the details of the failure metric used. Moreover, even when a constant value of $\tau_{\dot{u}} = 0.5$ is used across all simulations, the accuracies of the failure load predictions differ greatly between beam 1A and 1B. As demonstrated in Table 6.1, the load ranges predicted for beam 1A are around 5 times as large as the predictions for beam 1B.

One important reason why this convergence criterion can be unreliable is because the simulation ends as soon as the average absolute node velocity goes below some threshold value. Looking at the red curves in Figure 6.3b, which indicates structural failure under the bond damage criterion, there is a clear trend of decaying velocities for most tests. Between roughly time steps 600 and 1500, three of the tests which later go on to fail, all fall below an absolute node velocity of $0.1m/s$. Using the velocity decay criterion with a tolerance above $\tau_{\dot{u}} = 0.1$ hence leads to overestimated failure loads for this simulation.

**Force Decay**

The force decay criterion did not provide reliable or accurate predictions in these experiments. Although the criterion was proposed based on theoretical considerations of equilibrium in particle models, this was found to be the worst performing failure

metric. The criterion was originally proposed in Huang et al. 2015, with a recommended tolerance value of $\beta = 5 \times 10^{-7}$ [18]. However, the values obtained in Table 6.3 were based on a value of $\beta = 0.5$. Smaller values of $\beta$ were tried, but this resulted in worse predictive performance, with few beam tests converging at all.

One possible explanation is that for small applied loads, and hence small values of $\boldsymbol{b}(\boldsymbol{x}_i, t)$, the evaluation of equation 3.12 becomes numerically unstable. In such situations it is possible that lowering the applied test load *increases* the value of the left hand side in equation 3.12, hence making the test less likely to be deemed as having converged to an equilibrium solution. If this is the case, the geometric binary load search will fail and the predicted failure load will tend to the value initially set as the global lower bound. Another possibility is that the time step budget of $t_{max} = 5000$ was not large enough to reliably assess convergence. This could also stem from numerical errors in the simulation. If accrued floating point errors are of similar magnitude to node velocities and accelerations in later stages of the simulation, the convergence criterion will never be met for small values of $\beta$.

**Bond Damage**

Table 6.4 shows that using the bond damage criterion for convergence gives good predictions of failure loads for all but one of the simulations. It is hard to say with certainty why the predictions for test 1A were so poor when using the 4392-node model. It is possible that this particular discretisation caused instability in the model, through physical or numerical effects. Changing the parameters of the no-fail zones and the load build up described in section 4.2 might bring the predictions in line with the rest of the simulations.

### 7.1.1 Limitations of the Proposed Method

Although the initial results presented in this report are promising for the proposed bond damage criterion, further investigations are needed to properly assess its shortcomings.

In its current form, any test that is assessed as converged under the proposed criterion will have been deemed as converged under the velocity decay criterion. This means that choosing an extremely low tolerance value $\tau_{\dot{u}}$ will result in underestimating failure loads. A point to note is that the converse is not necessarily true: picking large values of $\tau_{\dot{u}}$ does not automatically overestimate failure loads. In this sense the bond damage

criterion is conservative.

Because the proposed criterion includes a test for beam vibrations by checking whether node loads and velocities are aligned or not, this method will not work if the beam is overdamped. This means that employing the bond damage criterion places limitations on the value of the damping coefficient $C$ used in the simulation.

## 7.2 Discretisation Coarseness

A major focus of this work has been to investigate the dependence of simulated beam behaviour on the coarseness of the spatial discretisation. It is promising that even using fewer than 1000 nodes to simulate the tested beams yields reasonable predictions of failure loads using the bond damage criterion. At the range of coarseness levels tested there does not appear to be an identifiable trend relating the value of the predicted failure load to the number of nodes in the simulation when the velocity decay or bond damage criteria were used. However, looking at Table 6.3 there is a clear trend that using a finer discretisation leads to a lower predicted failure load when the force decay criterion is employed. Due to the systematically unreliable nature of the predictions using this criterion it is hard to assess if this trend continues for models with a larger number of nodes.

For the velocity decay and bond damage criterion, using a coarse model does not appear to bias the simulation to overestimate or underestimate failure loads. However, it is possible that by increasing the number of nodes used further, predictions will tend to some fixed limiting value across all criteria.

### 7.2.1 Global Stiffness

It is hard to conclude based on the data presented in Figure 6.5 what effect discretisation coarseness has on global beam stiffness. The model using 4095 nodes is stiffer than the coarser model using only 2652 nodes, however for the 792-node and the 6256-node models this relationship is reversed. This indicates that the coarseness does not systematically bias the model to be more or less stiff, although the stiffness variation between different models is significant. More experiments are needed to further assess this relationship.

All force-displacement curves in Figure 6.5 also display some form of strain softening

before failure. This is a characteristic concrete behaviour and is challenging to model using FEM approaches [11]. In order to assess more carefully what effect discretisation coarseness has on this behaviour, displacement controlled tests are needed. The simulations performed here were force controlled and did not include any unloading.

# 8 Conclusions

This work proposed a new way of defining failure in Peridynamics simulations and used this approach to predict the failure loads of cantilevered RC beams. Simulations were conducted at different levels of spatial discretisation coarseness, and based on the investigations conducted, the following conclusions are presented:

- The proposed bond damage convergence criterion provides a good heuristic alternative to the established methods for defining structural failure.
- The predictions of failure loads using this method appear to be more robust to changes in the parameters of the criterion compared to the other two tested methods.
- The velocity decay convergence criterion is highly dependent on the tolerance parameter $\tau_{\dot{u}}$ used, and in the form presented here produces unreliable predictions of failure load.
- The coarseness employed in the spatial discretisation of an RC beam model affects the simulation predictions of failure load and global beam stiffness. However, there is no systematic bias to overestimate or underestimate values using the discretisations examined in this work when the velocity decay or bond damage convergence criteria are used.

## 8.1 Future Work

There are several possible avenues of further investigation based on the work presented in this report, central suggestions are:

- Exploring the dependence of model predictions on spatial discretisation further by using models with a larger range of node counts. This will likely require the use of High Performance Computers (HPCs), as well as the suggested software improvements outlined in section 8.1.2.
- Investigating the predictive performance of the model and the proposed bond damage convergence criterion on more complex beam designs, such as the optimised design shown in Figure 2.1.

- Identifying and characterising the situations under which using the proposed convergence criterion results in poor predictions of beam behaviour.

### 8.1.1 Towards a Universal Failure Criteria

The findings presented in this report are promising for the use of the bond damage criterion for assessing structural failure in PD simulations of RC beams. However, the limitations identified in section 7.1.1 need to be addressed further. The criterion has two associated tolerances that are set as parameters. The sensitivity of predictions to varying parameter values needs to be explored. Additionally, it was observed in the 4392-node model of beam 1A that using this criterion gave predictions that were extremely low. This outlier simulation was also repeated with larger tolerance parameters, but the predictions were unchanged. This behaviour was only observed for this one particular model and should be investigated further.

### 8.1.2 Software and Numerical Efficiency

Finally, some suggestions for improvements in the implemented simulation software written for this project are presented.

**Computing Neighbour Distance**

One of the most computationally expensive steps of the bond based PD simulation is computing the distance between neighbouring nodes. A method for potentially making this computation more efficient when using high-performance linear algebra libraries such as Eigen was proposed by the author and is outlined here. This approach was however not included in the final simulation software.

Let $\boldsymbol{V}^x = [x_0, x_1, \ldots, x_N]^T$ denote the $[N \times 1]$ column vector which contains the $x$-coordinate of all nodes in the simulation, where $N$ is the number of nodes. Now define the matrix $\boldsymbol{\Lambda}^x$ as shown in equation 8.1, by stacking $\boldsymbol{V}^x$ $N$ times in the row-direction. Finally, evaluate the matrix $\boldsymbol{H}^x$ defined in equation 8.2, where $\odot$ denotes element-wise multiplication[5], $\boldsymbol{A}$ is the connectivity matrix as defined in section 4.1 and $\boldsymbol{\Lambda}^{xT}$ denotes the transpose of the matrix $\boldsymbol{\Lambda}^x$. $H^x_{ij}$ is now the distance from node $i$ to node $j$ in the $x$-direction. $\boldsymbol{H}^y$ and $\boldsymbol{H}^z$ are computed in a similar manner.

[5] $\odot$ is the Hadamard product of two matrices

$$\boldsymbol{\Lambda}^{x}_{[N\times N]} = \left[\boldsymbol{V}^{x}, \boldsymbol{V}^{x}, \ldots, \boldsymbol{V}^{x}\right] \tag{8.1}$$

$$\boldsymbol{H}_{x} = \boldsymbol{\Lambda}^{x} \odot \boldsymbol{A} - \boldsymbol{\Lambda}^{xT} \odot \boldsymbol{A} \tag{8.2}$$

**Parallelisation**

At every time step in the simulation, there are independent values which are computed for either all bonds or all nodes in the model. This makes the PD simulations computationally costly, but also means that running computations in parallel can speed up the simulation considerably. For efficiently written code, the expected speedup is close to linear in the number of threads. However, improper implementation of concurrent software might result in performance degradation. [26]. A parallel implementation of the PD simulation in this work was initially explored, but was ultimately not implemented.

# References


[1] John Orr, Antony Darby, Timothy Ibell, and Mark Evernden. Optimisation and durability in fabric cast 'double t' beams. In *Second International Conference on Flexible Formwork (icff2012)*. University of Bath, 2012.

[2] Stewart A Silling. Reformulation of elasticity theory for discontinuities and long-range forces. *Journal of the Mechanics and Physics of Solids*, 48(1):175–209, 2000.

[3] Florin Bobaru, John T Foster, Philippe H Geubelle, and Stewart A Silling. *Handbook of peridynamic modeling*. CRC press, 2016.

[4] Michael P Collins, Evan C Bentz, Phillip T Quach, and Giorgio T Proestos. The challenge of predicting the shear strength of very thick slabs. *Concrete International*, 37(11):29–37, 2015.

[5] Junuthula Narasimha Reddy. An introduction to the finite element method. *New York*, 1993.

[6] Ellis H Dill. *Continuum mechanics: elasticity, plasticity, viscoelasticity*. CRC press, 2006.

[7] WH Gerstle, N Sau, and N Sakhavand. On peridynamic computational simulation of concrete structures. *Special Publication*, 265:245–264, 2009.

[8] D_ Ngo and Alex C Scordelis. Finite element analysis of reinforced concrete beams. In *Journal Proceedings*, volume 64, pages 152–163, 1967.

[9] Rene de Borst, Joris JC Remmers, Alan Needleman, and Marie-Angèle Abellan. Discrete vs smeared crack models for concrete fracture: bridging the gap. *International Journal for Numerical and Analytical Methods in Geomechanics*, 28(7-8):583–607, 2004.

[10] Dibakar Datta. Introduction to extended finite element (xfem) method. *arXiv preprint arXiv:1308.5208*, 2013.

[11] SV Chaudhari and MA Chakrabarti. Modeling of concrete for nonlinear analysis using finite element code abaqus. *International Journal of Computer Applications*, 44(7):14–18, 2012.

[12] S. A. Silling and R. B. Lehoucq. Convergence of peridynamics to classical elasticity theory. *Journal of Elasticity*, 93(1):13, 2008.

[13] Thomas L Warren, Stewart A Silling, Abe Askari, Olaf Weckner, Michael A Epton, and Jifeng Xu. A non-ordinary state-based peridynamic method to model solid material deformation and fracture. *International Journal of Solids and Structures*, 46(5):1186–1195, 2009.

[14] Erdogan Madenci, Mehmet Dorduncu, Atila Barut, and Nam Phan. Weak form of peridynamics for nonlocal essential and natural boundary conditions. *Computer Methods in Applied Mechanics and Engineering*, 337:598–631, 2018.

[15] Stewart A Silling and Ebrahim Askari. A meshfree method based on the peridynamic model of solid mechanics. *Computers & structures*, 83(17-18):1526–1535, 2005.

[16] Bahattin Kilic. Peridynamic theory for progressive failure prediction in homogeneous and heterogeneous materials. 2008.

[17] David John Littlewood, Timothy Shelton, and Jesse David Thomas. Estimation of the critical time step for peridynamic models. Technical report, Sandia National Lab.(SNL-NM), Albuquerque, NM (United States), 2013.

[18] Dan Huang, Guangda Lu, and Pizhong Qiao. An improved peridynamic approach for quasi-static elastic deformation and brittle fracture analysis. *International Journal of Mechanical Sciences*, 94:111–122, 2015.

[19] Mark Hobbs. Computational modelling of reinforced concrete structures with peridynamics. First year phd report, University of Cambridge, 2018.

[20] Tyler Andrews. Computation time comparison between matlab and c++ using launch windows. 2012.

[21] Eigen library homepage. http://eigen.tuxfamily.org/index.php?title=Main_Page. Accessed: 2019-05-22.

[22] QV Le and F Bobaru. Surface corrections for peridynamic models in elasticity and fracture. *Computational Mechanics*, pages 1–20, 2018.

[23] BB Sabir. Fracture energy and fracture toughness of concrete. *Magazine of Concrete Research*, 46(169):237–243, 1994.

[24] Surendra P Shah. Fracture toughness for high-strength concrete. *Materials Journal*, 87(3):260–265, 1990.

[25] M Mazloom and H Salehi. The relationship between fracture toughness and compressive strength of self-compacting lightweight concrete. In *IOP Conference Series: Materials Science and Engineering*, volume 431, page 062007. IOP Publishing, 2018.

[26] Intel Corporation. Avoiding and identifying false sharing among threads. https://software.intel.com/en-us/articles/avoiding-and-identifying-false-sharing-among-threads. Accessed: 2019-03-12.

# A Risk Assessement Retrospective

The risks identified at the start of this project were related to extensive computer use. The identified risks were repetitive strain injury, eye strain and sitting for long periods of time. In retrospect this was an accurate assessment as the project consisted of writing and running software on a laptop. Doing more of the work at the available desktop computers at the Engineering Department could have been a good way of mitigating some of the risks identified.