Simulation of Fibre Pull-out Using a Graphics Processing Unit Accelerated Discrete Element Model

By

Sven Dressler

A dissertation presented in the fulfilment of

the requirements for the degree of

Master of Engineering

in

Mechanical Engineering

Supervised by Prof. N. Wilke

at the

Department of Mechanical Engineering

Faculty of Engineering, the Built Environment and Information Technology

University of Pretoria

Pretoria

August 2020

Synopsis

The stress transfer and debonding behaviour of reinforcing fibres embedded in a matrix are important properties defining the behaviour of the composite. A typical minimal example of the interaction between a fibre and a matrix is a fibre pull-out experiment. In this experiment, a partially embedded fibre is extracted from a concentric cylinder of the matrix material. Physical, numerical, and analytical investigations have been conducted of such fibre pull-out problems. This research is a further numerical investigation of a fibre pull-out problem using a Discrete Element Model (DEM). A DEM has advantages over other numerical methods (such as Finite Element Models (FEM)) such as easily incorporating fracture formation.

A DEM code was developed using the interpreted programming language Python and employed the parallel computing strengths of a Graphics Processing Unit (GPU) to improve the speed of the code. The developed code was found to be two orders of magnitude faster than similar code running in serial on a Central Processing Unit (CPU). The chain fountain and the interaction of a sphere and plane were modelled to demonstrate its capabilities. The code is best suited for problems with bonded particles and where additional contacts do not occur during the simulation.

A fibre pull-out problem was modelled, and the results were compared to research published by others. Two phases of the fibre pull-out were modelled: stress growth (the interface between the fibre and matrix is intact), and debonding (where the interface yields and fails). It was found that the DEM could accurately recover the stresses when compared to research undertaken by others. The load/displacement relationship for the fibre pull-out displayed dynamic effects which were eliminated by using an arc-length control method. The resulting load/displacement relationship differed from published results. The results differed due to the simulations undertaken here being able to eliminate dynamic effects while the published results included the dynamic forces which occur in the final stages of fibre debonding.

The developed code shows utility for future investigations into fibre debonding problems. Other problems which can also be investigated using this code include the chain fountain, flexible sheets and materials which fail in a brittle manner.

Acknowledgements

I wish to acknowledge with gratitude the following individuals:

My supervisor, Prof. N. Wilke, for his intellectual support as well as the time he invested in the undertaking of this project.

My parents and sister, for their endless faith in my capabilities, and their encouragement and support.

My wife and children, for patience, understanding and encouragement

"Habe nun, ach! Philosophie,
Juristerei und Medizin,
Und leider auch Theologie
Durchaus studiert,
mit heißem Bemühn.
Da steh ich nun, ich armer Tor!
Und bin so klug als wie zuvor;…"

Johann Wolfgang von Goethe

Contents

List of Figures

List of Tables

List of Abbreviations

CPU	**Central Processing Unit**
DEM	**Discrete Element Model**
FEM	**Finite Element Model**
GPU	**Graphics Processing Unit**
GP-GPU	**General Purpose Computing on Graphics Processing Units**
LSDP	**Linear Spring Dash-Pot**
XFEM	**Extended Finite Element Model**
NRMSE	**Normalized Root Mean Square Error**

List of Symbols

m	**Particle Mass**
u	**Particle Location**
F	**Force**
J	**Polar Moment of Inertia**
W	**Particle angular velocity**
M	**Moment**
V	**Relative velocity**
K_n	**Normal bond stiffness**
K_t	**Tangential bond stiffness**

β	**Non-viscous damping factor**
ε_{zz}	**Strain in z-direction**
σ_{zz}	**Stress in z-direction**
σ_{rr}	**Stress in the direction normal to the fibre**
$\sigma_{\theta\theta}$	**Stress in the radial direction**
E	**Youngs's modulus**
G	**Shear modulus**
v	**Poisson's ratio**
αT	**Thermal expansion coefficient**
τ_{rz}	**Shear stress**
γ_{rz}	**Shear strain**
ω	**Axial displacement**
a	**Fibre radius**
b	**Matrix radius**
α	**Shear lag parameter**
Co	**Location of bond centroid**
U	**Particle displacement**
Δt	**Timestep**
n	**Normal vector**
Ω	**Angular displacement**
I	**Moment of Inertia**
T	**Torque**
ω_{max}	**Highest natural frequency**
δ	**Shear slip**
D	**Damage**

1 Introduction

Fibre-reinforced composites are a broad class of modern materials which seek to combine the stiff and tensile strength properties of individual fibres with the compressive strength of the binding material (henceforth termed the matrix). Examples of such composites are fibre reinforced concrete (FRC), fibreglass and carbon fibre composites.

Load-carrying within a fibre reinforced composite is characterised by the transfer of force between the matrix and the embedded fibres. In conventional composites the matrix is less stiff than the fibres with lower resistance to tensile forces, under tensile loading that exceeds the matrix tensile resistance, the matrix cracks. The load is then transferred through the interface between the fibre and matrix and transferred across the crack via axial force in the fibres. The load transfer properties and strength of the interface are, together with the strength of the fibre and the matrix, essential properties that define the strength of the composite material.

Modelling (numerical, physical, and analytical) of the interface under load can inform properties when designing composite materials. As an example: knowing the maximum force that the interface can resist for a particular length fibre also informs the maximum axial force that the fibre would need to resist (such that the fibre does not break before the interface fails).

Physical modelling of the fibre/matrix interface under load has previously been undertaken using a pull-out test. For this test, a fibre embedded in a matrix is extracted under a tensile load. The axial displacement and force relationship provides information on the shear strength of the interface. The stress distribution along the interface can, however, not be recovered from this test. Numerical models have been developed using primarily Finite Element Models (FEM) to model pull-out tests. Analytical solutions have been developed, which, together with FEM, have provided confidence in our understanding of load transfer behaviour for an intact interface. The analytical models have, however, not been able to easily incorporate the failure of the interface under the same analytical framework. FEM, being developed for continuous materials, does not natively incorporate fracture. In this research, we investigate an alternative numerical method to model the load transfer of an intact interface and the failure of the interface under increased loading.

The Discrete Element Method (DEM) is a numerical technique which models force transfer between independent particles. Interaction between particles can be via cohesive or non-cohesive contact models. Cohesive contact models with strength criteria allow the modelling of stresses and failure of materials under loading. This research investigates whether DEM is suitable to successfully model the force transfer to and the failure of a fibre-matrix interface during a fibre pull-out test.

Initially, a commercial DEM package was investigated to be used for this investigation. It was however found that the available packages were too inflexible for the intended use, the intended modelling being atypical of regular DEM use. A DEM code was therefore developed using the Python programming language. The execution speed of the developed code was improved by leveraging the parallel computational strengths of Graphics Processing Units (GPU) and exploiting the inherently parallel nature of DEM. The developed code is an innovative result, using a high-level language (Python) for such numerical modelling exploiting a GPU has not (to the authors' knowledge) been undertaken

before. The performance of the DEM is investigated, and the improvement from GPU parallelization is reported.

Several example problems which are well suited to the developed code are also shortly investigated and discussed. Two bond models were employed for the DEM, a moment transferring model (parallel bonds) and a basic model which only transfers force between particles (LSDP). Damping is an essential component of DEM and can unexpectedly influence behaviour. Different damping methods were also investigated, and non-physical behaviour was displayed for local non-viscous damping, a well-known damping method, applied under rigid body motion.

The developed DEM is used to investigate the pull-out of a fibre from a cylindrical matrix for both an intact interface and an interface undergoing failure. The accuracy of the numerical model is compared to published results. The published results include an analytical model and a numerical (FEM) model of the problem. The effect of fibre/matrix stiffness, DEM particle size, fibre length, interfacial stiffness and bond model is investigated and compared to the analytical solution. During this investigation, it became clear that the failure behaviour reported in the published paper did not agree with the behaviour recovered in the DEM modelling. A novel arc-length control method was employed to track the failure behaviour for the entire pull-out curve. This final stage of the failure curve has not been reported in numerical modelling of fibre pull-out (to the authors' knowledge).

1.1 Structure of this Dissertation

The research conducted here can be subdivided into two sections: i) the development of the DEM code and verification thereof and ii) the investigation of the fibre pull-out problem. The initial development and primary investigation in this dissertation are presented as two self-contained chapters, namely:

- **Chapter 3** presents the development of the DEM code, code performance and several verification example problems.
- **Chapter 4** presents the results of the fibre pull-out test using the developed DEM. Results are compared to analytical and numerical results presented by Chen and Yan (2015).

In the spirit of their self-contained nature, each chapter offers an independent literature review and conclusions aligned with the particular investigation under consideration. From chapters 3 and 4, two journal papers will be prepared to be submitted for review.

- **Chapter 2** offers a short literature review that combines vital research and background that covers Chapter 3 and 4, while
- **Chapter 5** offers concluding remarks, findings and future work that combines Chapter 3 and 4.

2 Literature Review

2.1 Stress Transfer of Embedded Fibres

The mechanisms whereby forces are transferred between a fibre embedded in a matrix and the matrix are well understood and widely studied (Nairn, 1997, 2000; Nairn *et al.*, 2001; Avery, 2016; Hsueh, 1988; Naaman *et al.*, 1989; Chen and Yan, 2015). The design of fibre reinforced concrete, traditional reinforcing, soil anchors and foundation piles all benefit from the resolution of the stress transfer between the fibre (or linear feature) and the matrix.

The stress field for the elastic behaviour of a fibre/matrix stress problem can be derived using a shear lag approach (Nairn, 1997). The term “shear lag” originates from the study of the design of T, I and box beams (Reissner, 1946). The shear lag theory is valid only for an embedded fibre with no debonding along the interface. Chen and Yan (2015) made an extension to include the debonding method. Building on this, further research was undertaken by Guo and Zhu (2015) and Heidarhaei, Shariati and Eipakchi (2019) to investigate debonding in fibre-reinforced composites.

Fibre pull-out of partially embedded fibres occurs in three phases (Naaman *et al.*, 1989; Chen, Beyerlein and Brinson, 2009), stress growth phase, debonding phase, and friction phase (Figure 2-1). During the stress growth phase, the interface between the matrix and fibre is intact, and the displacement/force relationship is linear. The debonding phase is characterised by the progressive failure of the interface and non-linear displacement/force relationship. Once the interface has entirely failed, stress transfer is only possible via friction between the fibre and matrix.

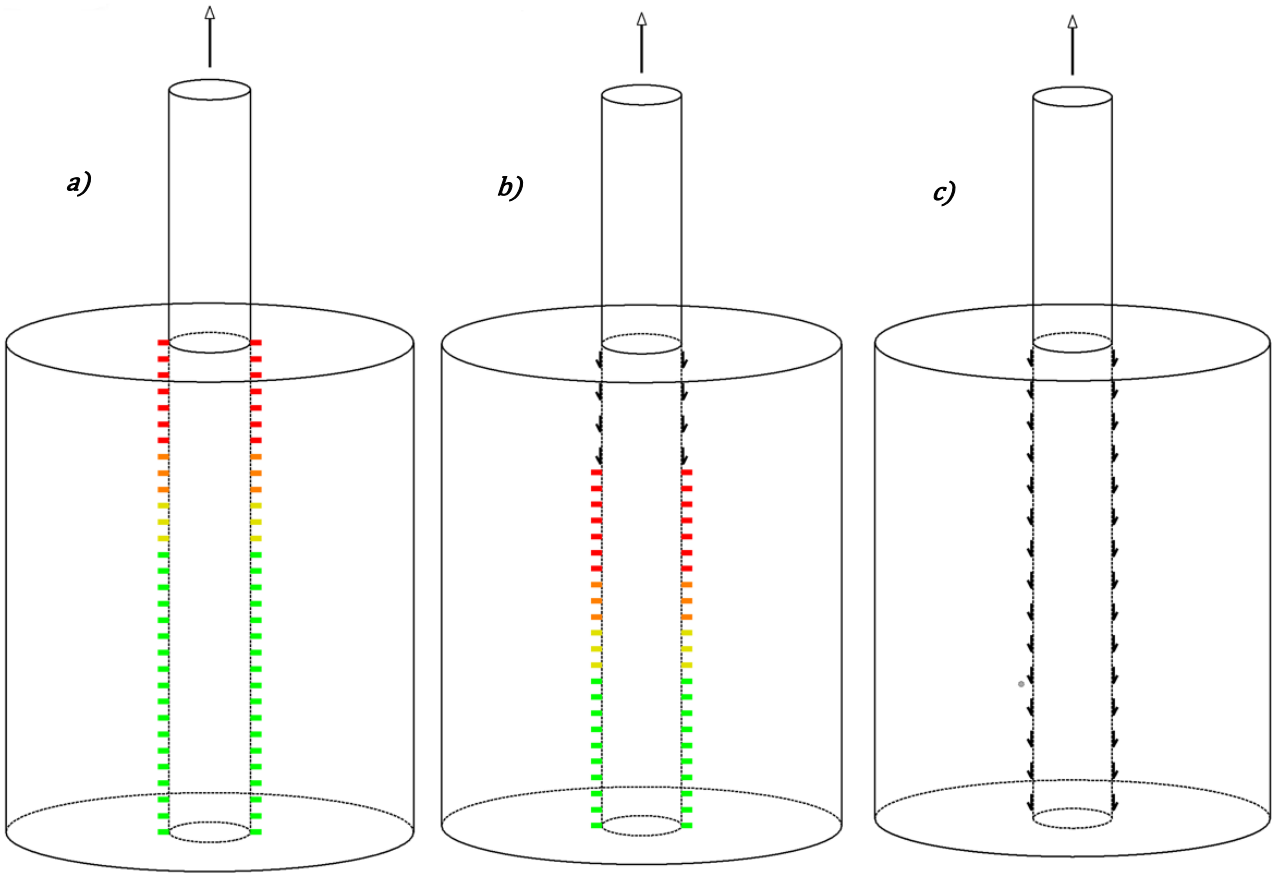

Figure 2-1: Pull-out behaviour of an embedded fibre in a matrix. Cohesive bonds between the matrix and fibre are represented with coloured lines (red = higher shear force). ***a)*** *Stress growth phase,* ***b)*** *debonding phase and* ***c)*** *friction phase.*

2.2 Numerical Modelling of Embedded Fibres

The stress growth and debonding phases of fibre pull-out have been modelled using computational continuum approaches such as the finite element method (FEM) (Denneman *et al.*, 2011; Jansson, 2011). FEM can utilize bonding elements to make allowance for fracture (Kang *et al.*, 2014). A discrete damage zone model has been used to simulate the formation of fractures in FEM (Liu, Duddu and Waisman, 2012). FEM treatments of fracture require either predefined fracture locations or dynamic re-meshing (Moës, Dolbow and Belytschko, 1999). Extended FEM or XFEM can resolve discontinuities without requiring the mesh to align to the boundaries of discontinuities. XFEM has been used to investigate pull-out of rebar and fibre matrix interfaces achieving good agreement with experimental results (Bouhala *et al.*, 2013; Orlando and Benvenuti, 2016).

Discontinuum approaches, such as the Discrete Element Method (DEM), provide another possibility to model fibre reinforcing (Yang *et al.*, 2010). DEM is most well-known in modelling the behaviour of granular materials (Potyondy and Cundall, 2004) by modelling numerous individual particles interacting. It is, however, possible to model a continuum using DEM by applying bonds between the individual particles (Leclerc *et al.*, 2017). Fracturing can then be simulated by merely

applying a failure criterion to these bonds. DEM has proven to be well suited to the modelling of material where fracturing has a significant influence on the material strength, e.g. concrete (Monteiro Azevedo, 2003; Potyondy and Cundall, 2004; Azevedo and Lemos, 2006; Yang *et al.*, 2010). The fracture behaviour of composites has been widely investigated using DEM (Yang *et al.*, 2010; Sheng *et al.*, 2010; Potapov, Faucher and Daudeville, 2012; Wolff *et al.*, 2013; Koval, Danh and Chazallon, 2014; Sadek *et al.*, 2014; Maheo *et al.*, 2015; Ismail, Yang and Ye, 2016; Zhang and Xie, 2017; Leclerc *et al.*, 2017; Marcon *et al.*, 2017; Wang *et al.*, 2017; Shang *et al.*, 2018). Figure 2-2 depicts the discussed numerical methods graphically.

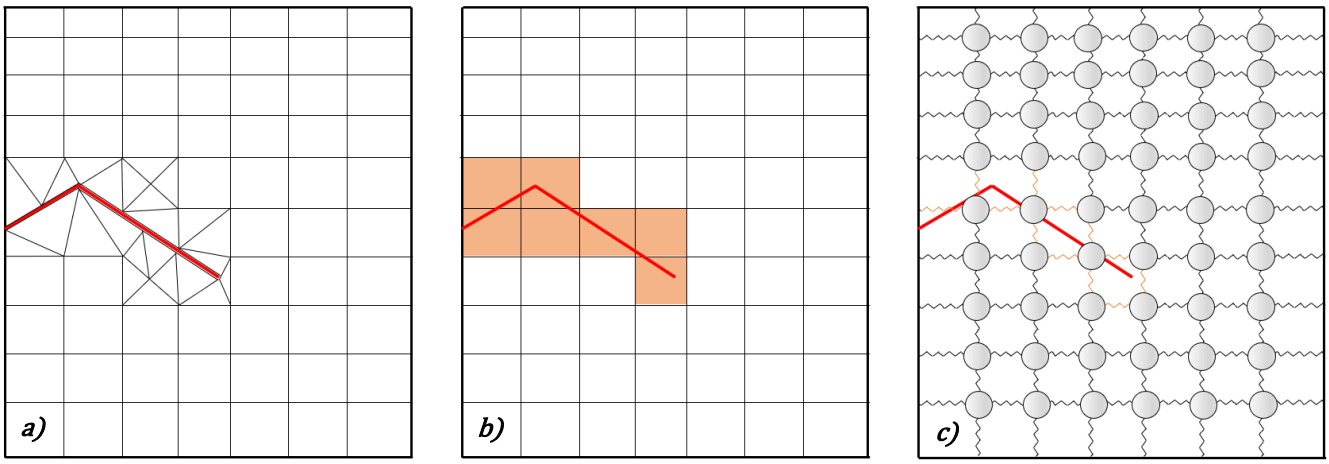

Figure 2-2: Example of some numerical methods for simulating the formation of a fracture (red): ***a)*** *FEM with re-meshing,* ***b)*** *XFEM, elements with enriched functionality to take account of discontinuities are highlighted and* ***c)*** *DEM, bonds experiencing damage are highlighted.*

2.3 Parallelization of DEM on GPUs

General Purpose Graphics Processing Units (GPGPU) are well suited to the parallel nature of DEM (Amada *et al.*, 2004; Govender *et al.*, 2015; Qi *et al.*, 2015). Recent advances in the design of computer hardware such as GPUs have allowed significant computational power to be leveraged to solve computationally difficult problems (Ghorpade, 2012).

Due to the well-suited nature of GPUs for tackling parallel problems, many studies have been undertaken in academia using GPUs for computation. Research has been undertaken in neural network machine learning algorithms (Zhang *et al.*, 2017), n-body problems (Elsen *et al.*, 2006) and Finite Element Models (Fu *et al.*, 2014). NVIDIA introduced the programming model known as CUDA in 2007 to simplify the development of programs on GPUs (Nickolls, Buck and Garland, 2008).

Python is an interpreted programming language which has increased in popularity in engineering, science and data science communities in recent years (Oliphant, 2007; Millman and Aivazis, 2011). For various reasons, standard Python programs do not execute quickly when compared to compiled languages such as C, C++ and Fortran (Lam, Pitrou and Seibert, 2015). Several modules have been developed for Python which are built around compiled and optimized functions written primarily in C.

Examples of such Python modules are SciPy (written for optimization and image processing tasks), Numpy (handling of vast arrays of data) and Numba (van der Walt and Aivazis, 2011; Lam, Pitrou and Seibert, 2015; Virtanen *et al.*, 2019). Numba is a high-performance python compiler that provides tools to compile functions written in Python and Numpy using the industry-standard LLVM compiler (Lam, Pitrou and Seibert, 2015). Numba also supports execution on the parallel architecture of GPUs (Millman and Aivazis, 2011; van der Walt and Aivazis, 2011; Lam, Pitrou and Seibert, 2015). In particular, Numba supports NVIDIA GPUs through Numba CUDA or AMD GPUs through Numba ROCm, making GPU compute readily available.

3 A GPU Based Discrete Element Model for Bonded Particles – Development and Examples

Discrete element modelling has become an often-employed technique for the simulation of systems with many interacting bodies such as granular material. Interaction models between particles can be non-cohesive or cohesive. Cohesive interactions between particles can be used to simulate continuous material. Discrete element models are well suited to implementation on parallel computational architecture. Graphics processing units are inherently designed for parallel computation, and recent advances in the architecture and compiler design have allowed general computation to be undertaken on GPUs. This study details the implementation of DEM written in Python and leveraging the parallel nature of GPUs for computational speed up. The program is written relying heavily on the Numba module which allows the compilation of Python syntax for execution on a GPU. The purpose of the code is for the simulation of the pull-out of a fibre embedded in a matrix and other similar problems. The problem is simulated with bonds between all interacting particles. Non-reversible bond damage is simulated, and each bond must, therefore, be stored and bond damage updated at each time step. The paper describes the implementation of collision detection, particle force determination and equation of motion integration written for execution on GPU. The data structure and memory use are described. The method used to apply boundary conditions is described. The performance of the developed code is investigated by comparison with similar codes, using Numpy and Numba Python modules, written for serial execution on CPU only. It was found that the developed code was 1000 times faster than the Numpy+Python implementation and 4 times faster than the Numba+Python implementation for force determination and equation of motion integration. Collision detection was 900 times faster compared to Numpy+Python but performed slower compared to Numba+Python.

3.1 Introduction

The use of computational parallelization techniques for solving numerical problems has become widespread with the advent of improved hardware and systems architecture. Many problems in engineering and the sciences are well suited for parallelization. Once such problem is the solution of the Discrete Element Method (DEM), a computationally demanding technique which the resolves the interaction of a collection of interacting particles (Govender, Wilke and Kok, 2015; Qi *et al.*, 2015).

Graphics Processing Units (GPUs) are a ubiquitous component in modern personal computers, used to render 3D graphics efficiently. The GPU architecture is designed to be able to execute multiple simultaneous computational threads. Recent advances in the design of computer hardware such as GPUs have allowed significant computational power to be leveraged to solve computationally difficult problems (Ghorpade, 2012).

The parallelization of DEM can be undertaken using four primary methods. Which method is best suited to the problem to be solved is dependent on the available hardware. The different methods used are:

- Multi-CPU methods, using the Message Passing Interface (MPI) to facilitate communication between the processors (Dowding, Dmytryshyn and Belytschko, 1999; Maknickas *et al.*, 2006; Shigeto and Sakai, 2011). A possible technique for parallelization is to assign a subgroup of elements (usually defined by a region) in the simulation to each available CPU. Each CPU then performs the necessary computations on the assigned elements in serial. Various methods exist to deal with elements at the interface between assigned regions.
- Multi-thread CPU methods, these methods take advantage of the multicore and multithread nature of modern CPUs (Shigeto and Sakai, 2011). Each element in the simulation is assessed using a unique thread which executes concurrently with other possible threads.
- GPU methods, these take advantage of modern GPUs which offer a large number of simultaneously executable threads and large amounts of fast memory (Amada *et al.*, 2004; Ma *et al.*, 2011; Xu *et al.*, 2011; Zheng, An and Huang, 2012).
- CPU-GPU heterogeneous architecture methods seek to combine the strengths of GPU and CPU parallelization techniques (Yue *et al.*, 2014).

It should be noted that only one of the listed papers can handle interactions that are not only non-cohesive. This paper presents a code that was developed to run a cohesive bonded DEM using the Python programming language and GPU acceleration.

Writing programs to execute using GPUs was done initially using platforms DirectX and OpenGL. This language is difficult to write for non-graphics applications and was an obstacle to the widespread use of GPUs within the scientific computing community. NVIDIA introduced the programming paradigm known as CUDA in 2007 to simplify the development of programs (Nickolls, Buck and Garland, 2008). Python modules have also been developed which allow python scripts to be executed using the parallel architecture of GPUs (Millman and Aivazis, 2011; van der Walt and Aivazis, 2011; Lam, Pitrou and Seibert, 2015).

3.1.1 Discrete Element Method

The Discrete Element Method (DEM) was first proposed by Cundall (Cundall, 1971; Cundall and Strack, 1979) to investigate the behaviour of granular assemblies. DEM employs discrete elements (such as discs, spheres or polyhedra) which interact only at their points of contact. The two phases in the application of DEM consist of the application of a force-displacement law at contact points and the application of Newton's second law to the elements themselves. Deformation of the particles themselves is assumed to be sufficiently small (compared to the deformation of the overall assembly) that they can be assumed to be rigid. Applying Newton's second law to the translational and rotational motion of individual particles results in (Rojek *et al.*, 2012):

$$m_i \ddot{\bar{u}}_i = \overline{F_i} \tag{1}$$

$$J_i \dot{\overline{W}}_i = \overline{M_i} \tag{2}$$

Where m_i, $\bar{u}_i$ and $\overline{F_i}$ are the mass, displacement, and resultant particle force of the *i*th element. J_i, $\overline{W_i}$ and $\overline{M_i}$ are the particle moment of inertia, angular velocity, and resultant moment of the *i*th element. The forces and moments acting on the *i*th particle can be described according to:

$$\overline{F_i} = \overline{F_i^{ext}} + \sum_{j=1}^{n_i^c} \overline{F_{ij}^{contact}} + \overline{F_i^{damp}} \tag{3}$$

$$\overline{M_i} = \overline{M_i^{ext}} + \sum_{j=1}^{n_i^c} \overline{M_{ij}^{contact}} + \overline{M_i^{damp}} \tag{4}$$

Where $\overline{F_i^{ext}}$ and $\overline{M_i^{ext}}$ are any external forces and moments acting on the particle. $\overline{F_{ij}^{contact}}$ and $\overline{M_{ij}^{contact}}$ are the force and moment due to interaction between the *i*th and *j*th element and n_i^c is the number of particles interacting with the *i*th particle. $\overline{F_i^{damp}}$ and $\overline{M_i^{damp}}$ are damping forces and moments which are applied to reduce spurious oscillations in the system.

Various contact models can be used to calculate forces at the contact points as a function of the particle overlap distances and velocities. The calculated forces are accumulated onto the individual particles participating in the interaction. Contact models can be tailored to represent non-cohesive (no tension forces, such as between individual sand grains) and cohesive (such as between particle representing rock) material (Potyondy and Cundall, 2004).

An explicit numerical scheme is employed to integrate over particle accelerations twice in time to recover particle velocities and displacements successively. The integration is undertaken over a given timestep which is equal for all particles in the simulation. The explicit numerical scheme is conditionally stable with a maximum timestep above which the model becomes unstable. Figure 3-1 shows a depiction of the steps inherent in a DEM for cohesionless contacts.

The DEM formulations used for this study were the parallel bond and linear spring and dash pot (LSDP) formulations (Cundall and Strack, 1979; Potyondy and Cundall, 2004; Rojek *et al.*, 2012). For each pair of bonded particles, a bond is formed which transmits bending moments (for parallel bonds), normal forces (for parallel and LSDP), shear forces (for parallel and LSDP) and torsional moments (for parallel bonds) between the particles. Forces are calculated according to the inter-particles tangential and normal strains. A damage law can be applied to these bonds to simulate bond strengths. An example of the calculation steps for DEM for bonded particles is shown in Figure 3-2.

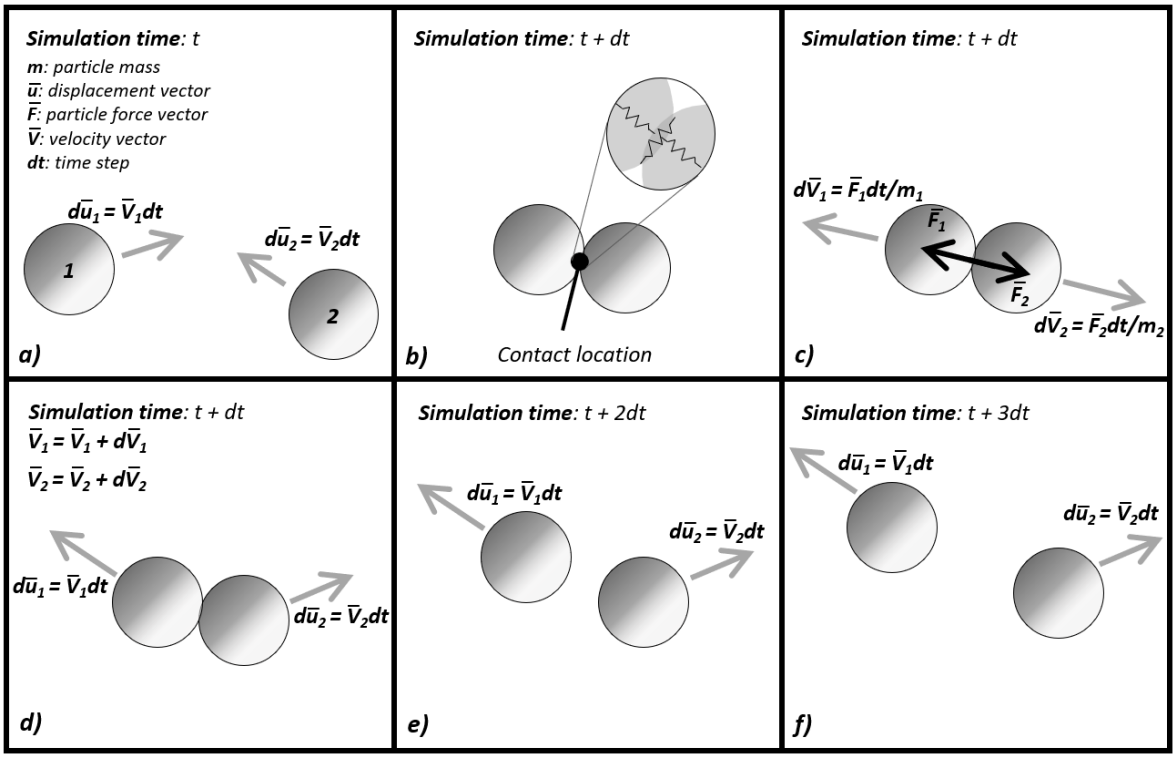

Figure 3-1: Steps in a Discrete Element Model: ***a)*** *Initial position and velocity of particles, no contact. Velocity is used to determine new particle locations,* ***b)*** *Collision detection is run and a contact location is identified, overlap between particles is determined,* ***c)*** *forces at the contact location are identified and the change in particle velocity is determined,* ***d)*** *particle velocities are updated,* ***e)*** *particle positions are updated, no further contact is detected, no velocity update,* ***f)*** *particle positions are updated, no further contact is detected, no velocity update.*

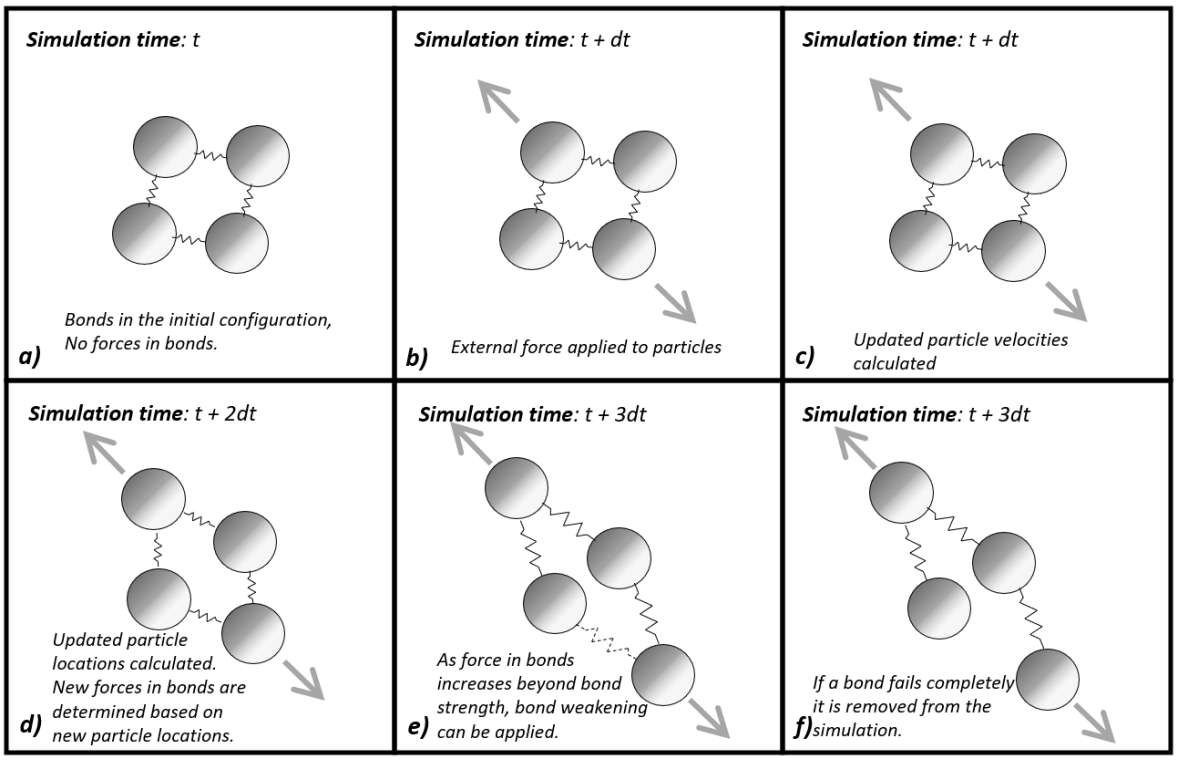

Figure 3-2: Example calculation for four bonded particles.

3.1.2 Parallel Computing

Most computers programs are written to use a serial computation method. Steps in the algorithm are executed sequentially with each step relying on the previous steps. Specific common computing problems such as depth-first-search algorithms are inherently sequential, and they cannot be efficiently solved in parallel (Greenlaw, Hoover, and Ruzzo, 1995). Adding additional computing pipelines that are executed simultaneously does not increase the speed of execution if each pipeline must wait for a previous one to complete.

Specific problems are well suited to executing in a parallel manner with multiple computing pipelines which are executed simultaneously. Problems which lend themselves well to such approaches require little interaction between computing pipelines. The individual computing pipelines can be set up at the start of the algorithm and executed simultaneously. The efficiency of the program is reduced if pipelines need to communicate between themselves during the execution of the code. The speed-up of a program written in parallel is described by (Greenlaw, Hoover, and Ruzzo, 1995):

$$\frac{Best\ Sequential\ Time}{Number\ of\ Processors} \leq Parallel\ Time \tag{5}$$

Notably, a computing problem to be written in parallel may need to incur an overhead to cast the problem to multiple processors. It can be seen from the equation that problems

which are well suited to parallel execution have a small serial start-up cost relative to the parallel portion of the execution. DEM lends itself well to execute in parallel.

3.1.3 General-Purpose Computing on Graphics Processing Units

Graphics Processing Units are special-purpose cards that are used in the computing environment to handle graphics operation such as image generation, resizing, recolouring and 3D rendering (Nickolls and Kirk, 2009). The GPU functions as an additional computing unit (the device) which runs independently but controlled by the CPU and associated infrastructure (the host). A schematic of how the GPU and CPU and interact is shown in Figure 3-3. The GPU has on-chip RAM and computational nodes designed primarily for the mostly parallel task of image generation. The host sends image data to the GPU which processes the data and returns it to the host. The GPU operates under the Single Instruction Multiple Data (SIMD) methodology. The GPU can be envisaged as multiple computing pipelines which are executed at once. All the threads receive the same instructions, and each thread can be assigned a unique ID which is used to access data help in on-chip memory. Thus all threads will execute the same instructions but with different data (such as different pixels in an image for image processing tasks) (Chen, 2009).

Image data is often stored with integer data describing the colour of each pixel. For this reason, older GPUs did not include architecture to efficiently carry out computation on data types stored as larger precision float numbers (such as single or double-precision floating point numbers). As the computational abilities of GPUs started being used to tackle non-graphics problems higher precisions were required (Owens *et al.*, 2008). Recent graphics cards can handle larger data types efficiently (Nickolls and Kirk, 2009) recent trends, however, have been to optimize performance for single and half precision floating point numbers due to increasing use of GPUs for machine learning where higher precision is not needed (Ho and Wong, 2017).

Due to the well-suited nature of GPUs for tackling parallel problems many studies have been undertaken in academia using GPUs for computation. Research has been undertaken in neural network machine learning algorithms (Zhang *et al.*, 2017), n-body problems (Elsen *et al.*, 2006) and Finite Element Models (Fu *et al.*, 2014).

The cards are designed for rapid uploading and downloading of image data to the host system. The smallest computing unit on a GPU is termed a thread. Threads are executed on the device according to the Single Instruction Multiple Data (SIMD) architecture. Each thread executes a given set of instructions. For simple algorithms that do not include branching of the execution instructions all threads on the device execute the same instruction simultaneously. For algorithms that include branching code (such as “switch” or “if” statements) the diverging instruction sets need to be executed independently and in serial. This can lead to a loss in computational efficiency (Nickolls and Kirk, 2009; Ghorpade, 2012).

Threads are grouped together into “warps” of 32 threads. Access to memory on a GPU is organized into requests 32 addresses long. Should the memory requests be coalesced (the locations accessed in memory are next to each other), the requested data can be returned in a single request cycle. As the individual memory request locations move further apart the requests are serialized again, leading to in-efficiencies.

Threads are also grouped together into blocks. Threads in a block are assigned a pool of memory that they can all access and modify. Each thread is assigned a unique numerical ID which can be interpreted as to its location on the device. This ID is used by each thread to access memory locations.

Different types of memory are available on a GPU. They differ according to how much space is available, how fast they can be accessed and the availability of the memory to different threads. The different memory types are:

- **Global memory** describes the memory area that can be accessed from all threads in all blocks. It is the slowest memory on the device but has the largest memory space. Global memory can be read and written from both the host and the device.
- **Shared memory** is available to each block of threads. Threads can access shared memory faster than they can access locations in global memory. Should multiple threads try to access the same location in shared memory, the accesses are serialized which decreases the computational speed.
- **Constant memory** is read-only memory available to all threads on the device. It is read-only memory which can only be written to from the host (the CPU). Access to constant memory is fast, but at the cost of reduced storage space. Constant memory reads can be broadcast across all reads. Should all threads be attempting to access the same location in constant memory, the read requests are broadcast across all requesting threads very efficiently.
- **Register memory** is memory available to each thread; it is limited in size but is the fastest memory available on the device.
- **Texture memory** is read-only memory of similar usage as constant memory.

Memory locations can be written to by many different threads simultaneously, and this creates the possibility for race conditions to occur. A race condition occurs when several computational threads modify a location in memory without considering the possibility that other threads may be trying to modify the same memory location.

A typical race condition on a GPU may occur when several threads are incrementing a counter. Each thread requests the present value of the counter from memory. The thread then increments the value and writes it back into the counter's location in memory. It is possible that between the read and write requests, other threads may access the value of the counter to increment it, resulting in incorrect values being written into the counter.

Race conditions can be overcome on a GPU by using atomic functions. An atomic function does not allow other threads to access the memory location being written to until the read and write operation is completed. The read and write processes are, therefore, serialized, this can result in a significant reduction in the speed of the code execution.

Moving data from normal PC RAM (Host RAM) to device RAM is a slow process which is avoided if possible (Govender, Wilke and Kok, 2015). The programming paradigm for running DEM on GPU is then to i) load all the data onto the device at the start of the analysis and ii) control all the phases of the algorithm on the device. This requires the use of algorithms which

lend themselves to the capabilities of the GPU (in particular the broad phase collision detection algorithm).

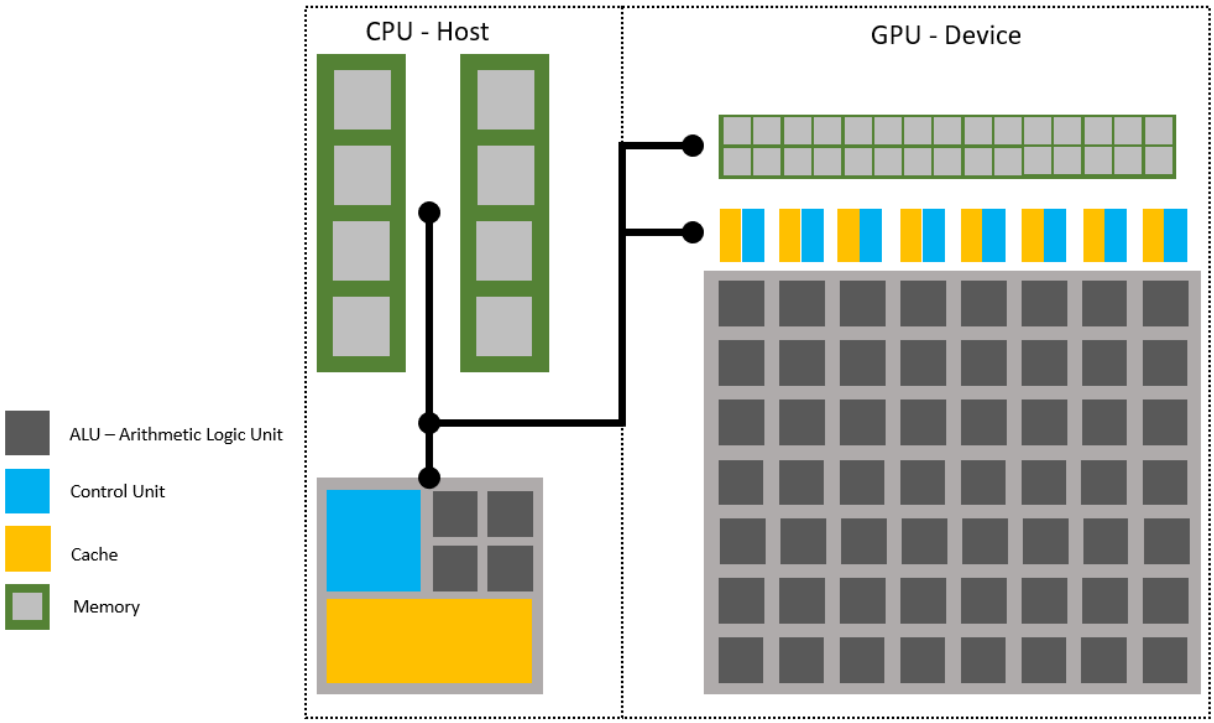

Figure 3-3: GPU processing pipeline.

3.1.4 Python Modules

Python is an interpreted programming language which has increased in popularity in engineering, science and data science communities in recent years (Oliphant, 2007; Millman and Aivazis, 2011). For various reasons, standard Python programs do not execute quickly when compared to compiled languages such as C, C++ and Fortran (Lam, Pitrou and Seibert, 2015). Several modules have been developed for Python which are built around compiled and optimized functions written primarily in C. Examples of such Python modules are SciPy (written for optimization and image processing tasks), Numpy (handling of vast arrays of data) and Numba (van der Walt and Aivazis, 2011; Lam, Pitrou and Seibert, 2015; Virtanen *et al.*, 2019). Numba is a high-performance python compiler that provides tools to compile functions written in Python and Numpy using the industry-standard LLVM compiler (Lam, Pitrou and Seibert, 2015). Numba also supports execution on the parallel architecture of GPUs (Millman and Aivazis, 2011; van der Walt and Aivazis, 2011; Lam, Pitrou and Seibert, 2015). In particular, Numba supports NVIDIA GPUs through Numba CUDA or AMD GPUs through Numba ROCm, making GPU compute readily available.

The functions to be compiled need to be written using a syntax that is allowed by the compilers. Significant speedups have been noted using Numba when compared to pure python. As the GPUs have started to be used for solving more general computing problems a programming language better suited for researchers was developed. Nvidia has developed the CUDA programming framework which allows the rapid development of programs leveraging GPU hardware written in C++ or C.

Numba has included a compiler which allows specially written Python functions to be executed on NVIDIA GPUS using the CUDA programming framework. An example of a Python function written for execution on GPU is shown below (Figure 3-4):

```
import numpy
from numba import cuda

Host_A = numpy.array([1.1, 2.1, 3.1, 4.1])
Host_B = numpy.array([1.1, 2.1, 3.1, 4.1])

Device_A = cuda.device_array(Host_A.shape(), dtype=numpy.float32)
cuda.to_device(Host_A, to=Device_A)
Device_B = cuda.device_array(Host_B.shape(), dtype=numpy.float32)
cuda.to_device(Host_B, to=Device_B)

@cuda.jit
def Add_on_GPU (D_A, D_B):
    i = cuda.grid(1)
    if i < D_A.shape[0]:
        D_A[i]+=D_B[i]

Add_on_GPU[1,4](Device_A, Device_B)
Output = Device_A.copy_to_host()
```

Figure 3-4: Python function written for GPU execution.

3.2 Methodology

A program was written using the python programming language to carry out a DEM simulation of a collection of bonded particles. Numba was used to optimize the program for execution on a GPU. The program and data structure are discussed in further detail below.

3.2.1 General Program Structure

The developed program is comprised of four stages. During the initial stage, the model comprising all the interacting particles is generated. The particle and bond property tables are generated and sent to the GPU. The next stage is to generate bonds between adjacent particles. A collision detection algorithm is employed to populate the list of particle bonds with all possible interacting particles. The next two phases calculate the forces in the bonds and integrate the equations of motion. The boundary conditions are applied during the equation of motion integration. The collision detection phase only needs to be repeated if the maximum particle displacement in the system is large enough that new particle interactions could occur. At each stage in the program, relevant parameters are updated and stored on the GPU so that they can be used for the next phase in the program. Data can be copied onto the host (traditional CPU and RAM) for tracking system histories. The overall program structure is shown in Figure 3-5.

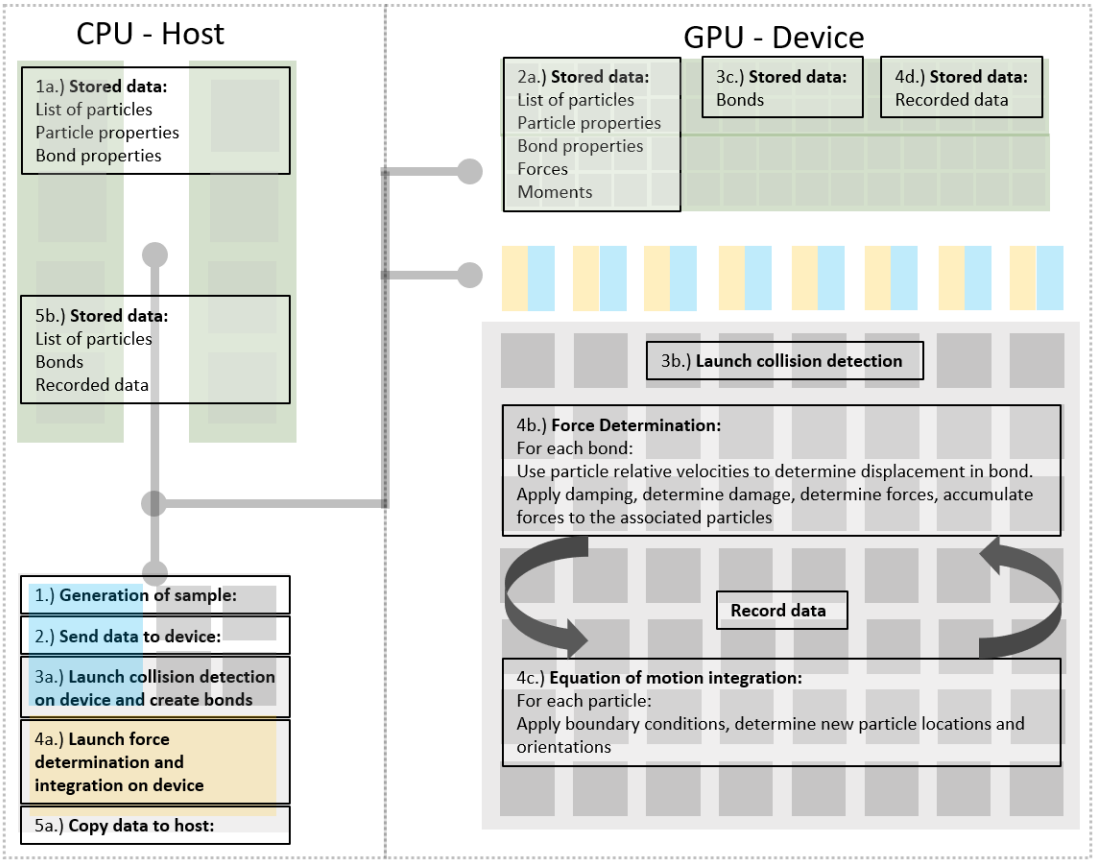

Figure 3-5: Developed program structure.

The main loop in the program consists of the force determination – equation of motion integration loop which is executed at each timestep. The detailed pseudocode of these two steps is shown in Figure 3-6.

```
for each timestep in Simulation do
    ---------------------------------------
    Force Determination
    for each B in Bonds do
        read bond properties
        read particles p1 and p2 in B
        determine relative velocity (Vrel) between p1 and p2
        normal component of dispacement U_N = Vrel · n_N
        tangential component of dispacement U_T = Vrel · n_T
        apply damage law if required modifying K_N and/or K_T
        normal force F_N = U_N · K_N
        tangential force F_T = U_T · K_T
        apply damping if required modifying F_N and F_T
        moment M = F_T × r
        F_p1 = F_p1 + F_N + F_T
        F_p2 = F_p2 - F_N - F_T
        M_p1 = M_p1 + M
        M_p2 = M_p2 + M
    end
    ---------------------------------------

    Integration
    for each p in Particles do
        read particle properties, location and velocity
        read F_p and M_p
        apply integration scheme
        update particle properties, location and velocity
    end
    ---------------------------------------
end
```

Figure 3-6: Pseudocode of Force Determination and Integration steps.

3.2.2 Data Structure

How data can be stored on a GPU depends on what kind of memory the data is being stored in. General global memory on the GPU can store various types of data such as arrays or objects. Constant memory, however, cannot store objects but is designed to store arrays only. Particle and bond information was stored as a custom Numba datatype, which is similar to the structs available in C or C++. Structs can be understood as objects with no associated methods. Storing data in this format simplifies code and ensures that information related to a specific particle or bond is stored close together. An indication of the data structure is shown in Figure 3-7.

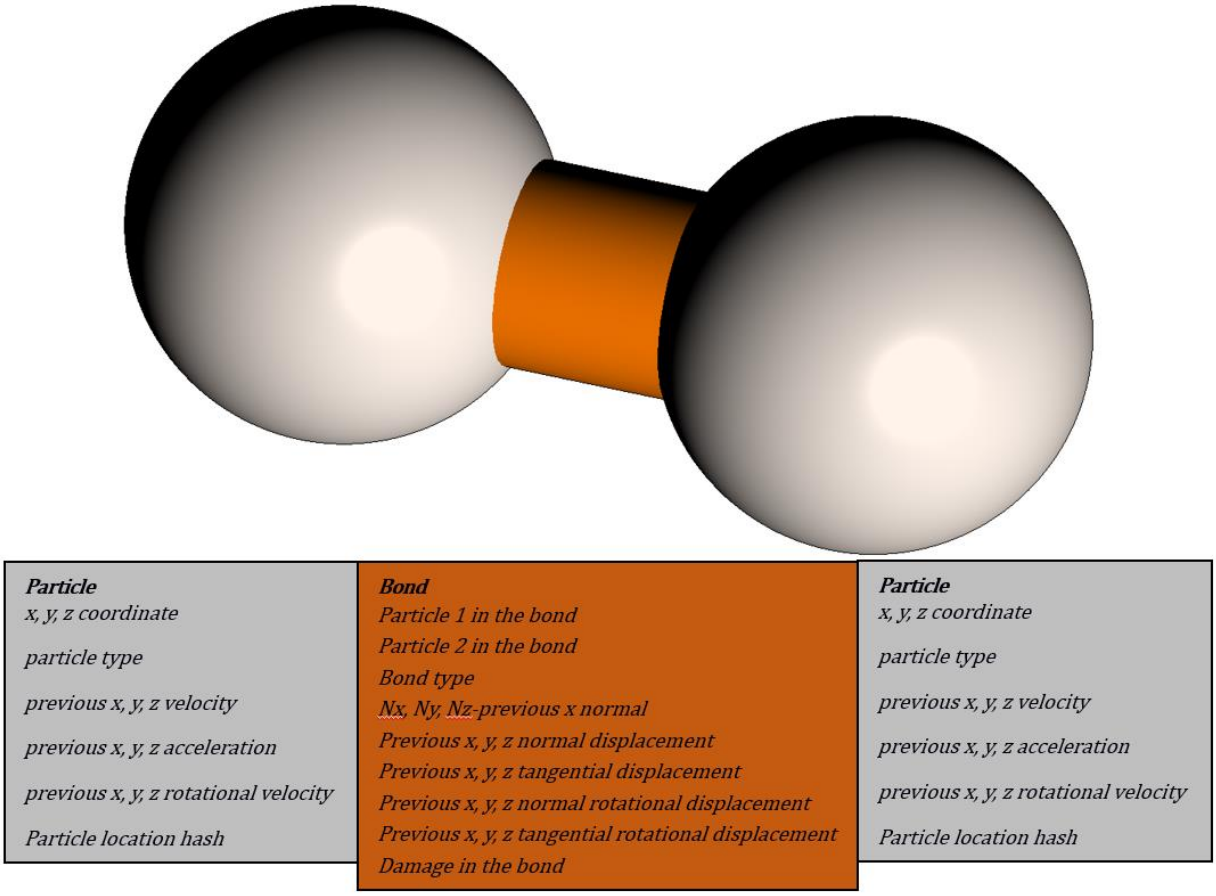

Figure 3-7: Two bonded particles, the particles are in grey and the bond is orange.

For the DEM code different arrays were stored on the device. Six arrays storing the forces (**Fx**, **Fy**, **Fz**) and moments (**Mx**, **My**, **Mz**) accumulated for each particle for each time step. Each array is a vector of floats with the same length as the number of particles. This array is stored in global memory. Arrays containing individual particle and bond variables are stored in global memory.

An array (**B_prop**) containing bond properties for each type of bond is created. This array is stored in constant memory. Information stored in the bond property array is listed below:

- Bond area
- Bond normal stiffnesses
- Bond Tangential stiffnesses
- Bond moment of inertia (for parallel bonds)
- Bond polar moment of inertia (for parallel bonds)
- Bond yield slip
- bond failure slip
- Additionally, the numerical index in the array serves as the bond type numerical ID

An array, (**P_prop**) that contains particle properties for each type of particle in the simulation. This array is stored in constant memory. This array stores the following information:

- Particle radius
- Particle mass
- Particle boundary conditions
- Additionally, the numerical index in the array serves as the particle type numerical ID

The two types of particle in the bond are used to calculate the index in the bond types array. The index is calculated from the two particles types as:

$$index = 10a + b \tag{6}$$

where a = the smallest of the two types of particles in the bond

b = the largest of the two types of particles in the bond

Therefore, a bond with particle types 1 and 2 has a bond type 12, and a bond between two particles with type 1 has bond type 11.

3.2.3 Collision Detection

The identification of interacting particle pairs is the first phase of a discrete element simulation. As the number of particles (N) in the simulation increases the number of possible interacting particles to be investigated (n) increases according to:

$$n = \frac{N^2}{2} \tag{7}$$

Checking each pair of particles for possible interaction would result in computational difficulties as the number of particles increases. Nearest neighbour search is a standard problem in computer science with many well-known algorithms. Reducing the number of required checks to identify possible collisions is the primary goal of broad phase collision detection. During this phase, possible interparticle interactions are determined. This phase can represent the largest computational resource of the algorithm.

The algorithm used for broad phase collision detection is a spatial hashing algorithm. This method divides the problem area into a regular grid. Each particle in the problem set is within one of the cells in the regular grid. Furthermore, each cell in the regular grid can be assigned a unique identifier, a hash. While traditional hash tables can be used a simple hash function can be used and is shown below.

$$hash = zi * wx * wy + xi * wz + yi \tag{8}$$

$$wx = \frac{Sample\ maximum\ x - \ Sample\ minimum\ x}{Cellsize} \tag{9}$$

$$wy = \frac{Sample\ maximum\ y - \ Sample\ minimum\ y}{Cellsize} \tag{10}$$

$$wz = \frac{Sample\ maximum\ z - \ Sample\ minimum\ z}{Cellsize} \tag{11}$$

$$xi = integer\left(\frac{x - Sample\ minimum\ x}{Cellsize}\right) \tag{12}$$

$$yi = integer\left(\frac{y - Sample\ minimum\ y}{Cellsize}\right) \tag{13}$$

$$zi = integer\left(\frac{z - Sample\ minimum\ z}{Cellsize}\right) \tag{14}$$

Using a simplified hash function as described, allows for specific advantages in the storage and access of memory locations. The hash is calculated for each particle in the sample and stored in the struct of the particle. The IDs of the particles with a particular hash are written into a vector. The next step in the algorithm is then, for a given particle, to read the hash and compare it to the particles with the same hash as well as those in the surrounding cells. Figure 3-8 illustrates how the model space is divided into cubes, for a particle in the green cube it would be compared to particles in the green and orange cubes (not all shown) only.

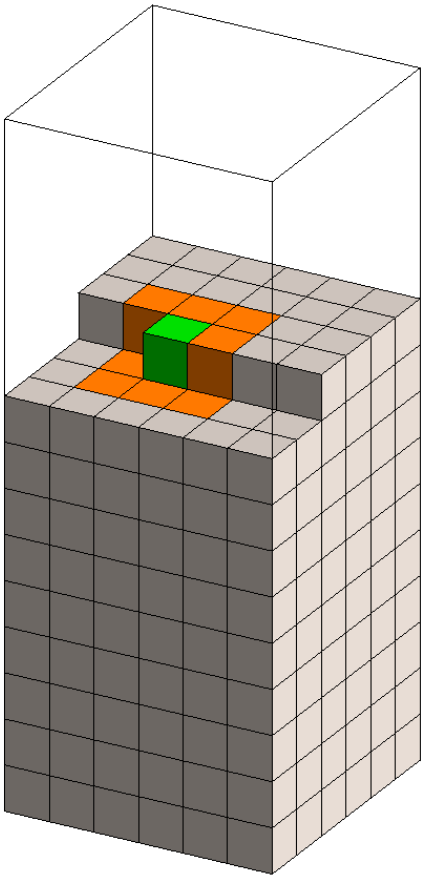

Figure 3-8: Broad phase collision detection. The full modelled area is within the rectangular prism. The modelled area is subdivided into cubic subdivisions. Each particle is assigned to one of the subdivisions (green cube) and possible collisions are checked with green cube and orange cube (not all shown).

A significant algorithmic speed-up can be achieved by not running the broad phase collision detection algorithm for every time step. The potential detected collisions are stored and used for several time steps before they are recalculated.

If the algorithm is executed as described each pair of interacting particles will result in two bonds being stored in the list of bonds. As an example, consider a bond of two particles (A and B) with individual IDs (given by their index location in the list of particles) of 12 and 25: Two bonds would be identified here, one with particle A = 12 and particle B = 25, and another with particle A = 25 and particle B = 12. This can be avoided by ensuring that the target particle has a numerical ID higher than the numerical ID of the base particle. The result in the example above is that the bond where particle A = 25 and particle B = 12 would not be listed as a bond.

In this phase, the exact interparticle distances for the pairs of particles identified in the broad phase collision detection are determined. The exact interparticle distance is either determined using the Euclidean distance between the particles or determined by integrating the relative particle velocities. The interparticle distances and relative velocities are used as inputs into the constitutive model to determine the forces to be applied to each particle.

3.2.4 Particle Force Determination

The sum of forces for each particle is determined and used together with the particle masses to determine particle accelerations. The particle accelerations are then integrated twice over a timestep to determine new particle locations and velocities.

Once the forces in each bond are determined they are added to the accumulated forces for the particles in the bond. Since each bond is analysed by a different thread on the GPU a possibility exists that different threads could attempt to update the same particles forces at the same time. It is, therefore, possible that race condition could occur which results in incorrect forces being written into the particle. The possible race condition can be avoided by using atomic functions. Atomic functions ensure that only one thread can modify a location in global memory at a time. The accesses will be serialized when different threads attempt to simultaneously modify the same location in memory. Atomic functions which result in the serialization of multiple threads can lead to a slowdown in the code execution.

Modern compilers optimize the atomic functions by accumulating values in each warp in shared memory (memory assigned to each block) and summing the totals to global memory after all threads in the warp have finished executing. The result is reduced atomic add operations to global memory.

3.2.5 Damping

Physical systems will always include a dissipative component which tends to reduce the kinetic energy of the system. If a DEM interaction is simulated with no dissipation the system will not reach an equilibrium but will oscillate continuously. If many bonded particles are simulated with no damping, the response will be unstable.

Dissipation in the system can be introduced in four ways (Cundall and Strack, 1979; Potyondy and Cundall, 2004): Friction, local viscous, global viscous and global non-viscous damping. Frictional dissipation is introduced as a constant force which acts in the opposite direction as the relative motion in the bond.

Viscous damping applies a force to the particles proportional (α) to the mass (m) and relative velocity ($\overline{V}$) between the interacting particles.

$$\overline{dF_{viscous}} = -\alpha m\bar{V} \tag{15}$$

It can be shown that for a 1-D system of linear springs a critical damping factor exists for which the system will converge to equilibrium in minimum time. Different critical damping factors apply to Degrees of Freedom (DOF) about translation and rotation. The critical damping factors are given by the equations below:

$$D_t^{crit} = 2\sqrt{m_p k_n} \tag{16}$$

$$D_r^{crit} = 2r^2\sqrt{\frac{2m_p k_t}{5}} \tag{17}$$

D_t^{crit} and D_r^{crit} are the critical damping factors for translational and rotational DOF's respectively. k_n and k_t are the particles' normal and tangential stiffnesses respectively, and m_p is the particle mass.

Global viscous damping applies a velocity-dependent force to all particle in the system irrespective of inter particle interactions. This form of damping acts on rigid body motion.

Non-viscous damping applies a damping force proportional (β) to the particle acceleration and in a direction dependent on the velocity ($sgn(\overline{V})$) of the particle (Azvedo, 2003; Potyondy and Cundall, 2004). The force aims to increase unbalanced forces ($\overline{F}$) that decrease particle velocities and decrease unbalanced forces that increase particle velocities. An advantage of this method is that rigid body motion is not damped.

$$\overline{d\overline{F}_{non-viscous}} = -\beta sgn(\overline{V})\overline{F} \tag{18}$$

The final force to be applied to the particle is then determined from:

$$\overline{F_{damped}} = \bar{F} + \overline{d\overline{F}_{non-viscous}} + \overline{dF_{viscous}} \tag{19}$$

3.2.6 Equation of Motion Integration and Boundary Conditions

Boundary conditions are applied by first multiplying the calculated particle velocity with either 0 (if a boundary condition is to be applied) or 1 (if the particle is free with no boundary conditions). The next step is to add the desired velocity in the equation (0 for free particles or for fixed particles and non-zero for non-zero boundary velocities).

The boundary conditions are stored in constant memory on the GPU for maximum access speed. An array in constant memory cannot easily be modified from the host while the program is running. Some scenarios such as investigations into hysteresis require the boundary condition to change during the simulation. Modifying the boundary conditions during a simulation was undertaken by launching different kernels with boundary conditions hardcoded for different particle types.

3.2.7 Contact Models

Two contact models were implemented in the developed code: A linear spring model with simple linear springs transferring normal and tangential forces between spherical particles and a parallel bond model which can be viewed as the linear spring model with an additional beam element in parallel which can transfer bending moments and torques between two particles.

The primary difference between the two contact models is that the parallel bond model can transfer bending moments (Figure 3-9). This is best illustrated by simulating a simply supported horizontal element at equilibrium under a uniformly distributed load. The linear spring contact model should result in the element approximating a catenary curve while the parallel bond model would result in the deflection expected of a uniformly loaded beam.

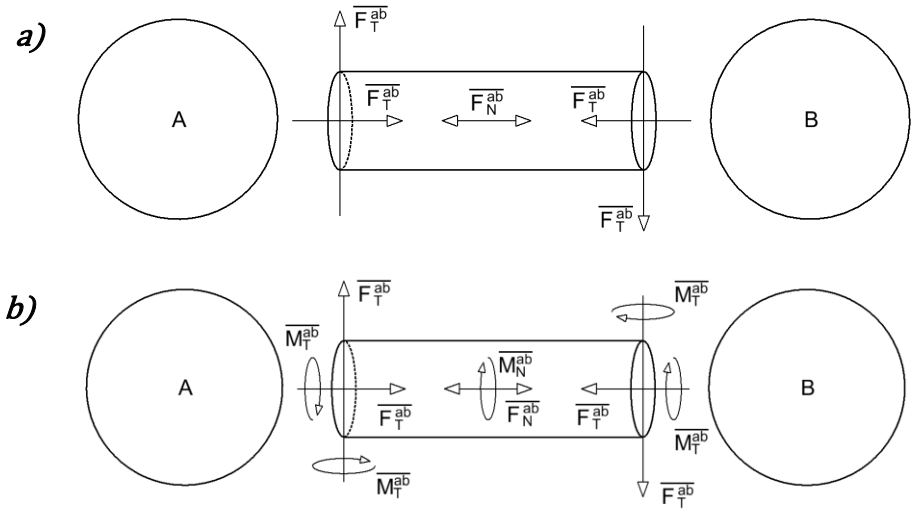

*Figure 3-9: Contact models implemented, **a)** linear spring model with no moment transfer between particles, **b)** parallel bond model with moment transfer.*

3.2.8 Beam and Catenary

The difference between the two bond models developed can be well demonstrated by simulating a chain of particles hanging under self-weight. A vital aspect of all DEM simulation is to ensure that the bond parameters used are calibrated against the physical model to be simulated. This section demonstrates an example of such a calibration.

The hanging beam example demonstrates how a calibration methodology could be employed to adjust bond parameters to match analytically predicted values. The simulated problem is a string of 50 particles with radius 1m which hangs under an imposed load of 1N/m. The particles are bonded via parallel bonds as described in section 2.7.

The bond parameter of **Kn** (the normal stiffness) is calibrated such that the chain of particles approximates the behaviour of a cylindrical beam with radius of 1m and Youngs Modulus of 1e9Pa. Calibration was carried out using a line search algorithm. An initial guess of **Kn** is assumed and the maximum sag of the simulated chain of particles was determined. An initial step size was assumed to modify **Kn** by, at each iteration the maximum sag is determined and **Kn** modified using the bisection method once the correct value of **Kn** is bracketed.

The final calibrated beam position and the analytical solution are plotted in Figure 3-10. The steps in the calibration process are shown in Figure 3-11. The analytical value can be rapidly approached, and care needs to be taken that the string of particles has achieved equilibrium before the maximum sag is determined.

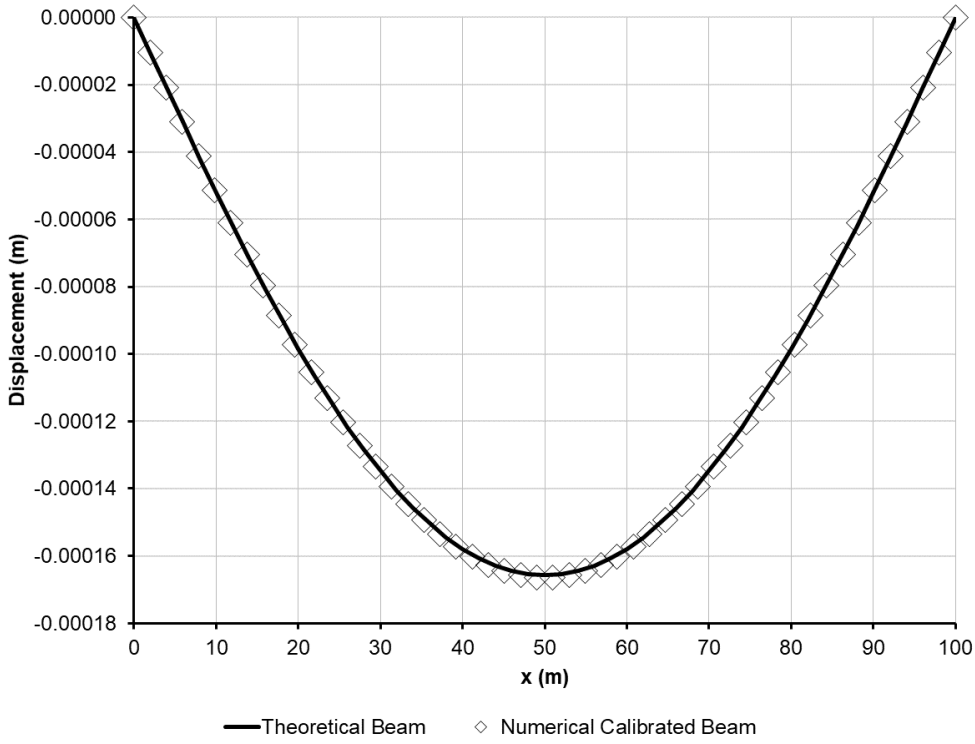

Figure 3-10: Numerically predicted particle locations compared to an analytical beam.

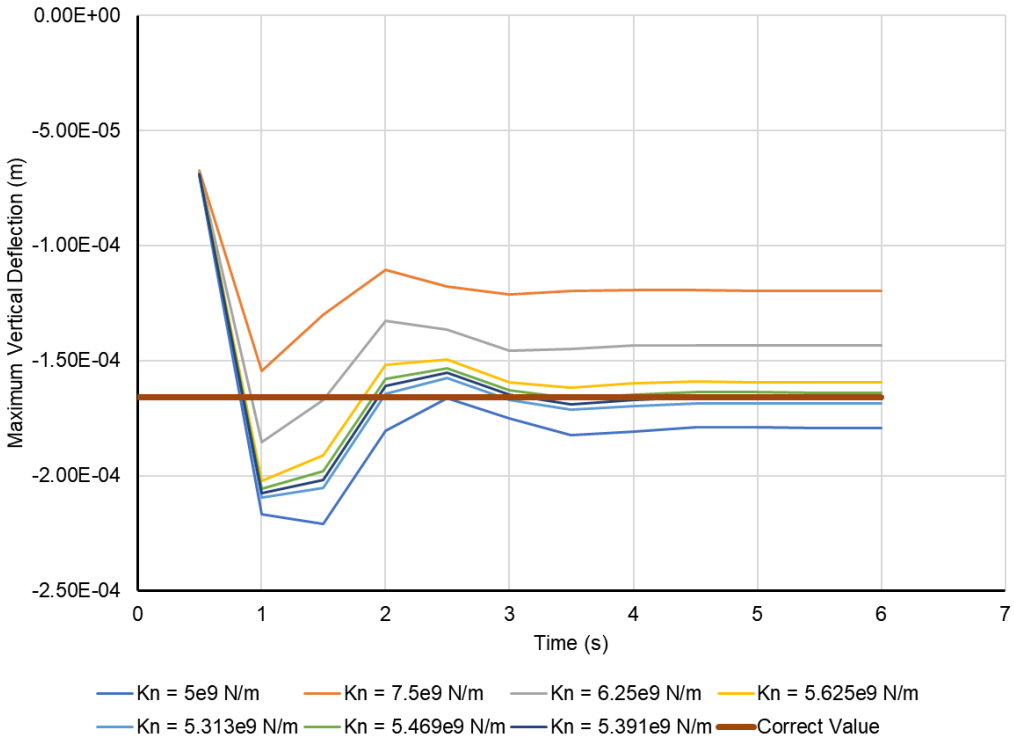

Figure 3-11: Convergence of the maximum deflection to the analytical result during calibration.

For comparison of parallel bonds with linear spring dash pot type bonds the same string of particles is simulated using LSDP bonds. The result should be particle positions which approximate a catenary curve. The final particle positions, as well as the analytical catenary curve with the same horizontal tension, are shown in Figure 3-12.

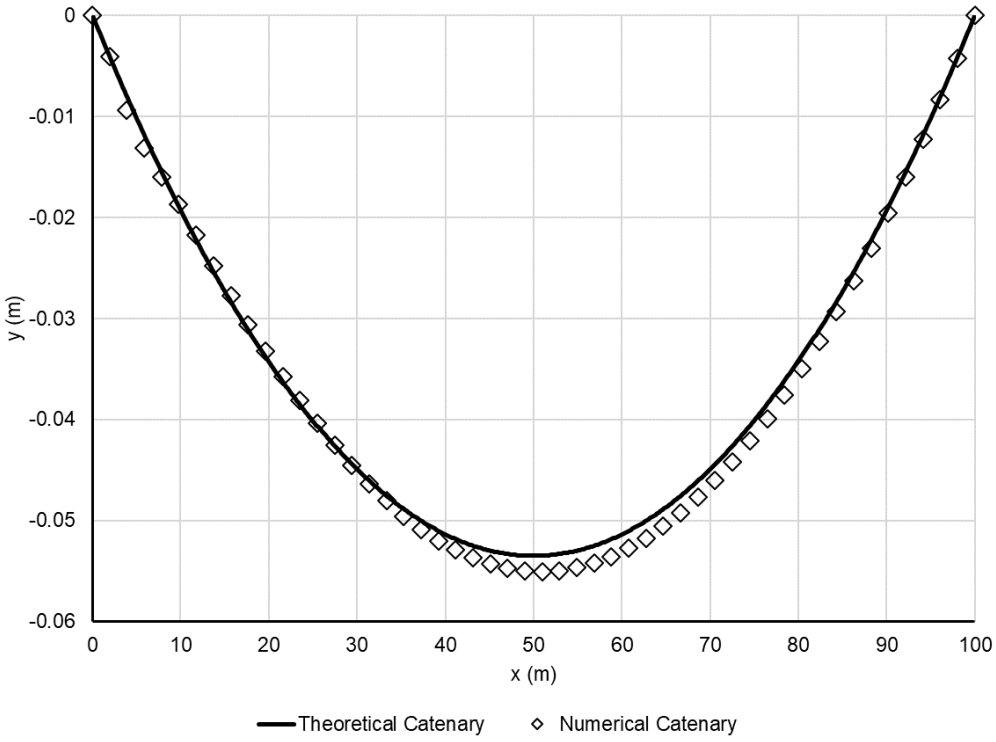

Figure 3-12: Numerically predicted particle locations compared the analytical catenary with the same horizontal tension.

3.2.9 Bond Breakage

Bonds between particles can be simulated to weaken and break as loads or displacements between particles increase. All interactions are modelled as interacting in a linear elastic manner while bond loads are below a certain threshold. Once the threshold is exceeded the bond could either yield or fail.

If the bond is modelled to fail once the ultimate load has been reached the bond can simply be deleted from the list of bonds to be investigated. No historical information of the bond needs to be stored. If the bond is modelled to yield and soften once a threshold load is reached the history of the bond needs to be stored.

To store the history of the bond, it is necessary to create a bond object which contains the particles forming the bond, the damage parameter and strain in the bond. This creates a new array that needs to be stored on the device. The history of the bond is stored as a damage variable between 0 (no damage) and 1 (completely broken). The damage in the bond is applied to the bond normal and tangential stiffness.

3.3 Results

The developed code is investigated here for performance relative to similar non-GPU based code. Some examples of investigations that are well suited to the code's performance characteristics are also presented.

3.3.1 Performance

The developed code was used to simulate the axial loading of a cylinder of particles generated with a hexagonal close-packed structure. The improvement in the code execution speed, when compared to standard non-optimized Python, was compared. Three different code optimization methods were used for the same code and the execution speed for different numbers of particles was measured using the "time" Python library. The used optimization methods were:

- No optimization, data is stored in Numpy arrays as custom Numpy data types. This code is executed in serial on the CPU.
- Numba CPU optimization, data is stored in Numpy arrays as custom Numpy data types. Numba's "Autojit" function is used to compile the functions using the LLVM compiler. This code is executed in serial on the CPU.
- Numba GPU optimization, data is stored in Numpy arrays as custom Numpy data types. Numba's "CUDA" compiler is used to compile specially written functions for execution on GPU.

The execution times for two different phases in the program were timed for 100 iterations and the average reported as the execution speed. The two program phases there were timed are:

- Bond creation and collision detection that includes the creation and storing of bonds in the list of bonds as well as identifying interacting particle pairs.
- Force determination and equation of motion integration. Analysis of the bonds to determine the force in the bonds and accumulation of forces to the participating particles. Determination of particle locations at the next time step by the integration of the equation of motion.

A cylinder of particles with hexagonal close packing was simulated (Figure 3-13). One end of the cylinder was kept fixed while the other end was displaced at a constant velocity. Different cylinder radii were used to display the effect of the number of particles in the simulation.

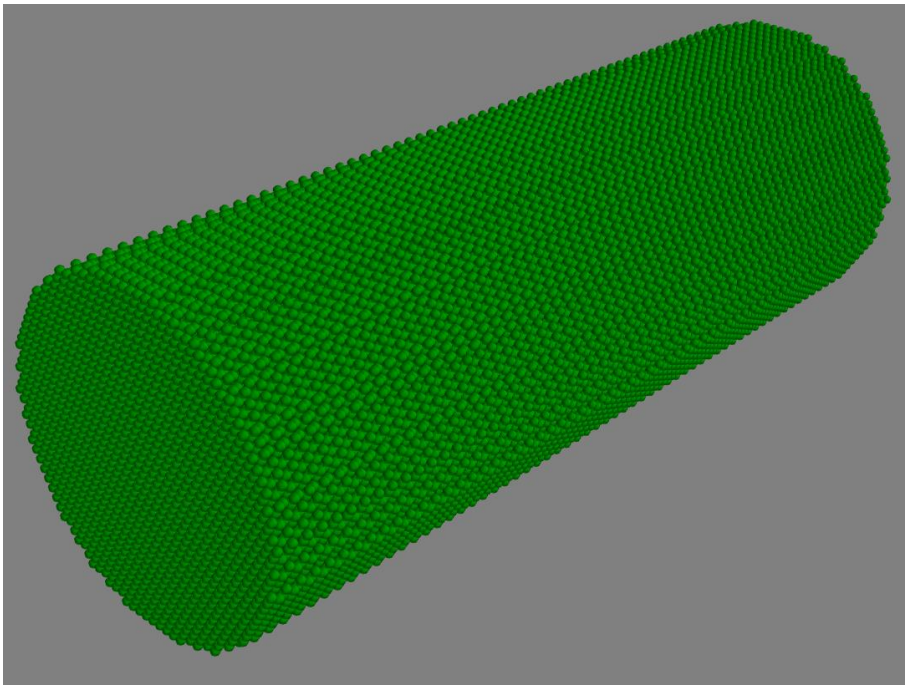

Figure 3-13: Hexagonal close-packed sample.

The average execution times for the collision detection phase are shown below (Figure 3-14). It can be seen that the pure Python code with no optimization is significantly slower than both Numba optimization methods. It is interesting to note that the sequentially executed Numba optimized CPU code is faster than the GPU optimized code. A possible reason for this is that the GPU code could lead to multiple global memory collisions (a memory collision leads to the memory requests being serialized). The GPU collision detection algorithm also includes multiple memory accesses from a single thread which requires multiple clock cycles to complete. A future study should seek to improve the collision detection speed on GPU. The CPU optimized code executed between 600 and 900 times faster than pure Python while the GPU optimized code executed 50 to 60 times faster than pure Python.

The code developed for this study focused on simulating problems where all particles are bonded together, and no new contacts are formed during the simulation. The contact list is therefore generated once at the beginning of the simulation.

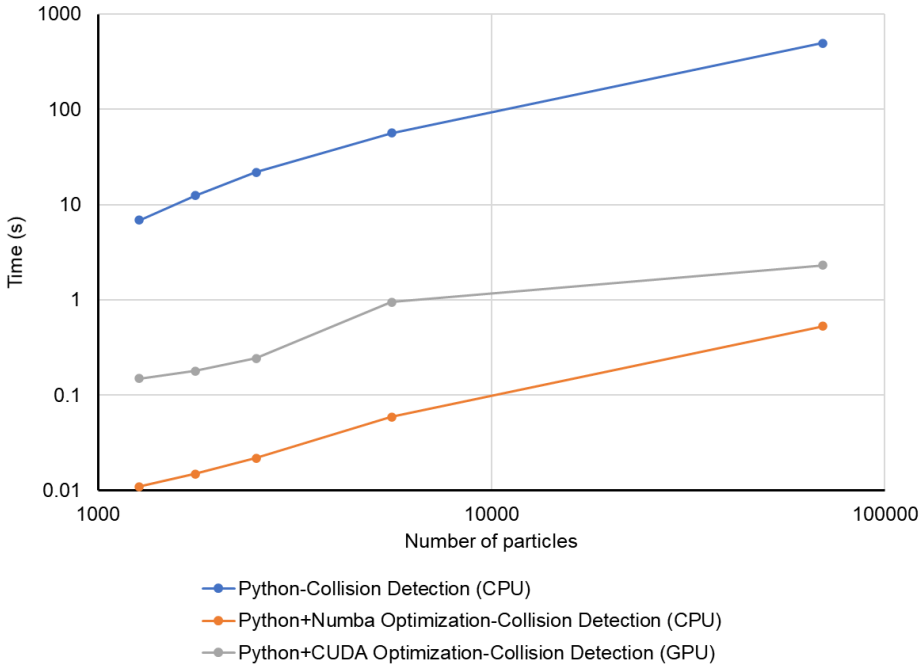

Figure 3-14: Execution times for the collision detection phase of the model. Different code optimization methods and the number of particles are compared.

The average execution times for the force determination and integration are shown in Figure 3-15. The Numba optimized code executed significantly faster than the non-optimized Python code (between 1000 and 1500 times faster for CPU optimized and between 1000 and 5500 times faster for GPU). It is notable that the CUDA code executes in a fixed amount of time independent of the number of particles while the number of particles is less than 6000 while the serial codes execute slower as the number of particles increases. This behaviour can be understood from the way instructions are executed on the GPU. Under the Single Instruction Multiple Data (SIMD) paradigm, all threads are executed on the GPU simultaneously with instruction broadcast across them. If more threads are required than are available on the GPU (or more resources such as memory are required) then the sets of threads are executed sequentially. The result is that if for any number of launched threads less than the total possible number of threads on the GPU, the execution time is the same. On a Nvidia K2200 graphics card, as was used in this study, there is a maximum of 10 240 threads available, assuming sufficient resources are available. The number of threads launched for each kernel is either equal to the number of particles (collision detection and equation of motion integration) or the number of bonds (force determination). It can be seen from Figure 3-15 that once the number of particles exceeds the available threads, the execution time increases as the thread sets are executed sequentially.

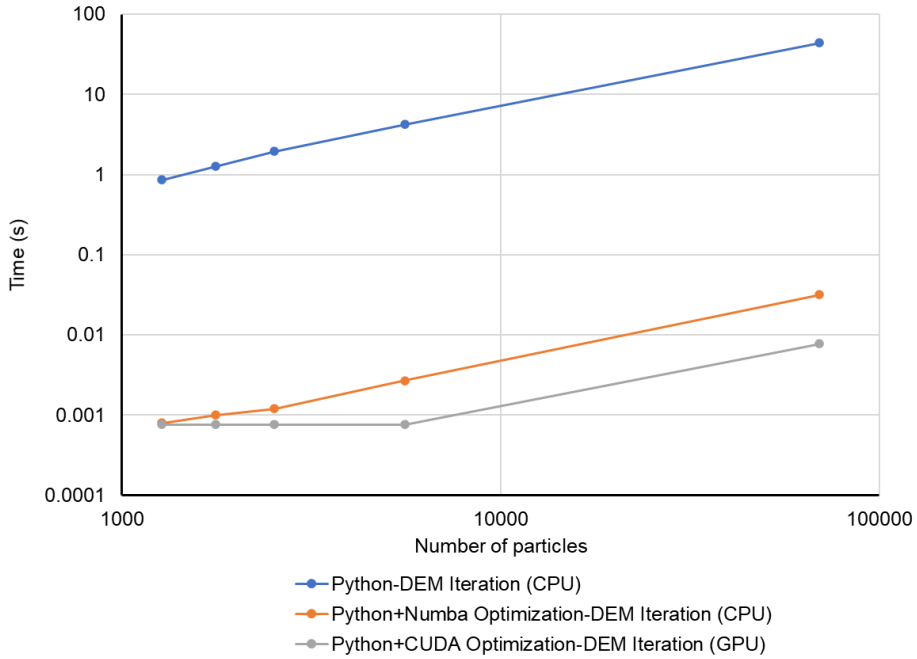

Figure 3-15: Execution times for the force determination and integration for one iteration of the model. Different code optimization methods and the number of particles are compared.

3.3.2 Further Examples

From the performance investigation the code performs best when the broad phase collision detection can be minimized. Problems that fit this criterion would exhibit little change in the set of particles in contact with each other during the simulation. Barring the trivial example of no particle contacts several exciting examples of such situations exist which are presented here.

The use of the developed code is demonstrated in this section by exploring three examples: A chain fountain, elastic interaction of a sphere and plane and a fibre pull-out from a matrix. These examples seek to explore different components of the developed program.

- The simulation of the chain fountain problem displays the ability of the model to capture the dynamics of a rapidly moving system. The interaction of particles with a boundary is also shown.
- The indentation contact investigation explores the linear elastic behaviour of many bonded particles interacting with a surface. This investigation also employed the application of body forces to the individual particles.
- The fibre pull-out model investigates both the application of de-bonding models as well as extensive calibration to match analytical predictions. The application of complex simulation control to track failure paths is also demonstrated.

3.3.2.1 Chain Fountain

The chain fountain or “Mould effect” occurs when a string of metal beads self-siphons via gravity (Figure 3-16). The effect can be demonstrated by filling container (such as a beaker) with a chain of metal beads, a free end of the chain is then lifted over the edge of the container. If the length of the hanging portion of the chain is greater than the effective portion within the container the remaining chain will be pulled out of the container by the hanging portion. Several physical experiments have demonstrated that the chain of beads rises higher than the edge of the container, seemingly violating the expected behaviour. This effect was first noted by Steven Mould (Mould, 2013) and investigated by Biggins and Warner (2014). Several explanations have been offered for this behaviour: Biggins explains it as being due to the dynamics of a string of flexibly linked rigid elements, while Flekkøy, Moura and Måløy (2018) claim that this behaviour is more complex and depends on the properties of both the chain and the container. An essential property of the chain of beads used in these investigations is that the chain resists being bent into a small radius. Different bond types between DEM particles can be used to model a chain of particles with varying amounts of bending resistance.

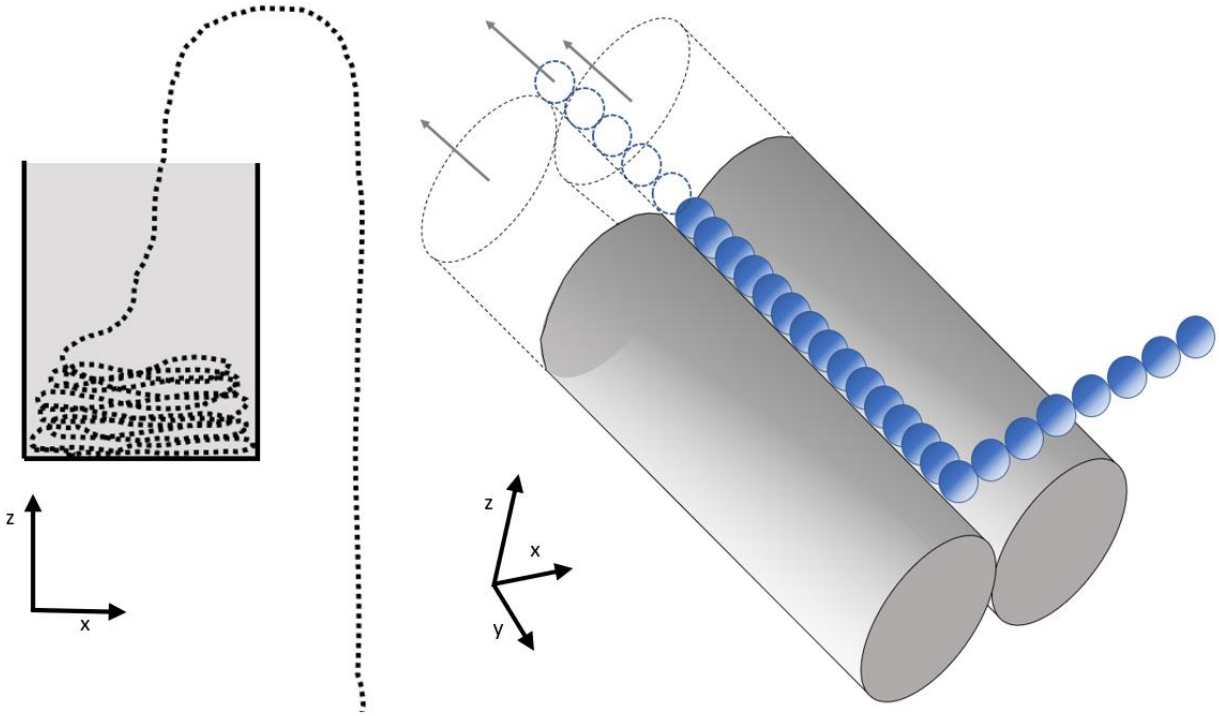

Figure 3-16: Schematic representation of the chain fountain effect which occurs during the self-siphoning of a chain of beads.

The chain fountain is explored here by constructing a minimal experiment which should produce the expected behaviour. A string of particles resting between two cylinders with one end of the string siphoning over one of the cylinders was used as the minimal experiment (Figure 3-17). Parallel and LSDP bonded particles were used to simulate the particles. Particles with a radius of 1m, ***Kn*** of 1e9Pa and density of 1000kg/m^3 were used in the simulation.

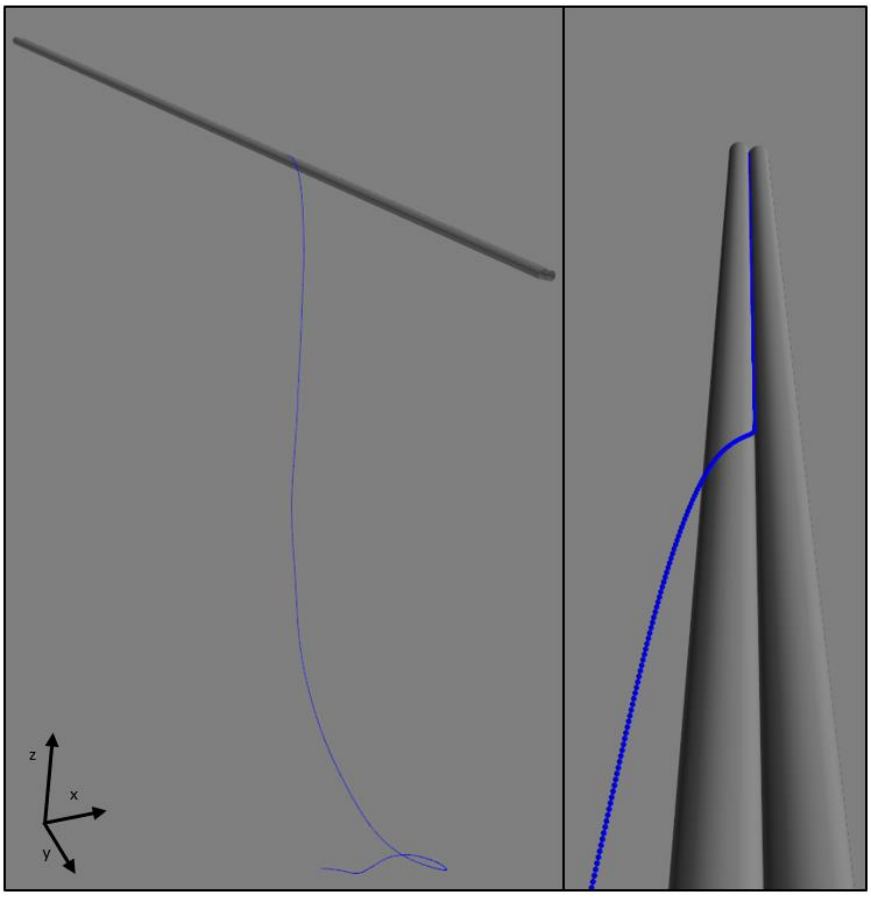

Figure 3-17: 3D models of the minimal experiment used to investigate the chain fountain.

The behaviour for parallel bonds and LSDP bonds should differ significantly. While LSDP bonded elements can be bent to high curvatures due to the lack of bending stiffness between elements, parallel bonded elements, however, resist being bent into tight curves. The string of beads at different times is shown in Figure 3-18 for parallel bonds and LSDP bonds. The vertical reaction force between the cylinders and the string of particles at the location at which the string picks up from the cylinders is shown for, parallel bonds in Figure 3-19, and LSDP in Figure 3-20. It can be seen that for the LSDP bonded string there is no excess reaction force at the pick-up point while for parallel bonds an excessive force is generated at the pick-up point due to the string of particles resisting high curvature. It can also be seen that the magnitude of the force at the pickup point increases as the time of simulation (and therefore the length of the hanging portion of the chain) increases, the research by Biggins and Warner (2014) suggests this.

For the minimal problem investigated in this example no notable elevation of the chain of particles over the edge of the cylinder was noted. The difference in the force between the chain and the cylinders at the pick-up point for the different bond models does, however, suggest that the bending stiffness of the chain strongly influences this force. Further modelling using the developed code could be undertaken for varying bending stiffnesses and chain lengths to investigate the effect further. The minimal example presented here is a promising experimental set-up to isolate the effects resulting in the chain fountain as popularly portrayed.

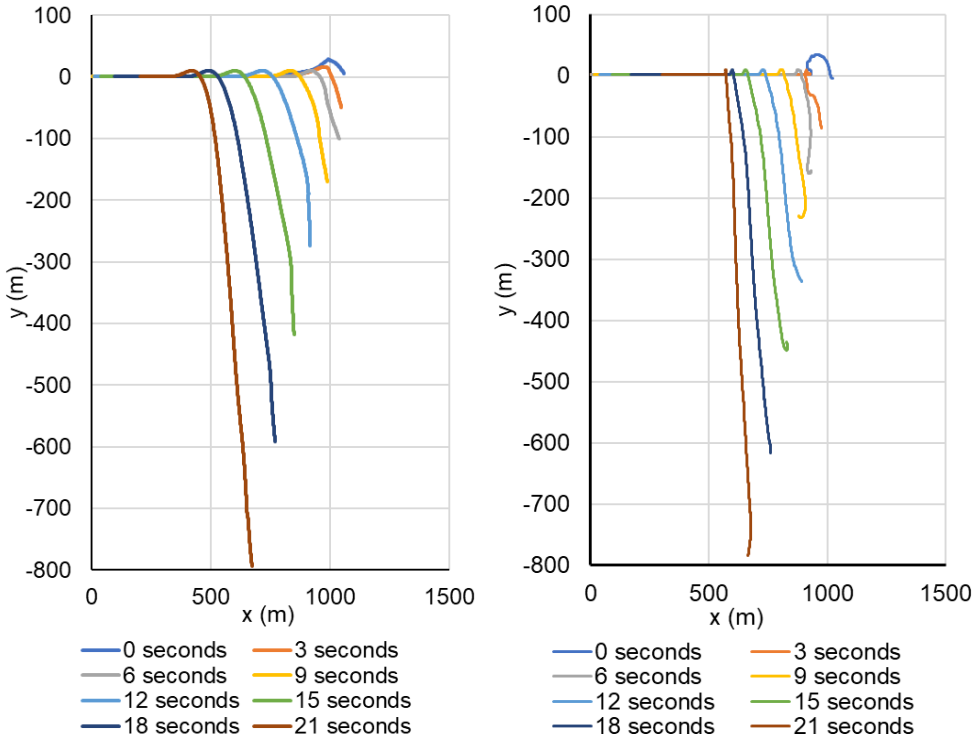

Figure 3-18: Chain locations with time for parallel bonded particles (left) and LSDP bonded particles (right).

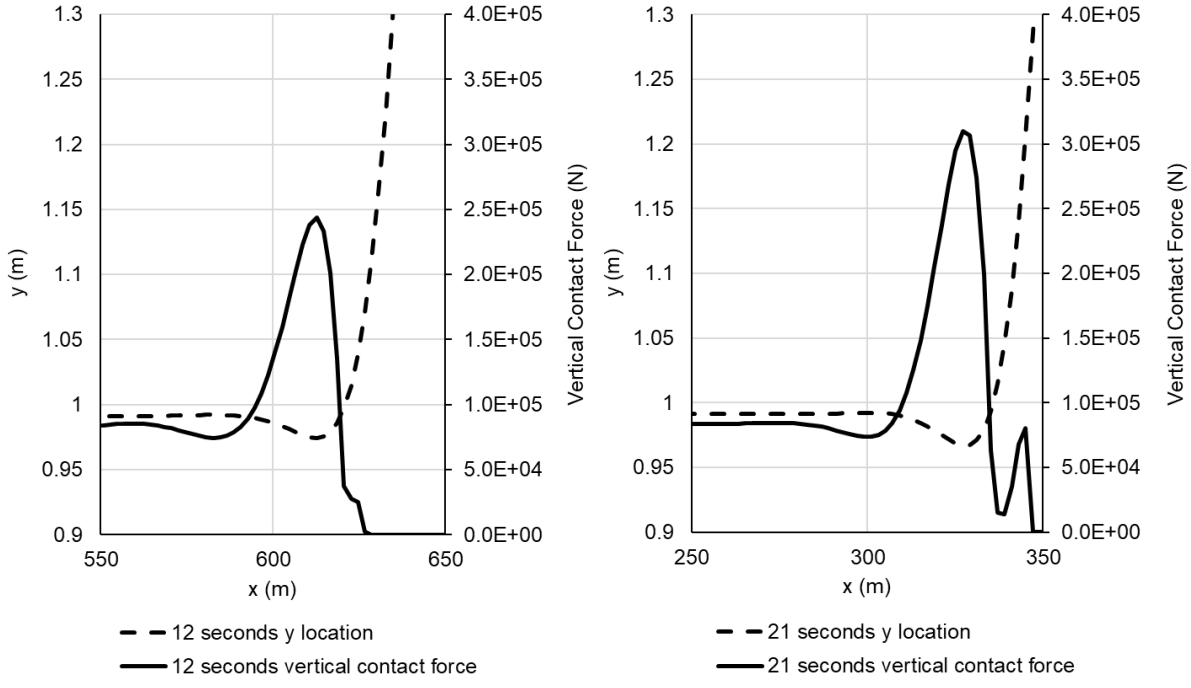

Figure 3-19: Vertical surface reaction force and chain location plotted against distance at the point where the chain leaves the surface it is resting upon at 12 second (left) and 21 seconds (right), parallel bonded particles.

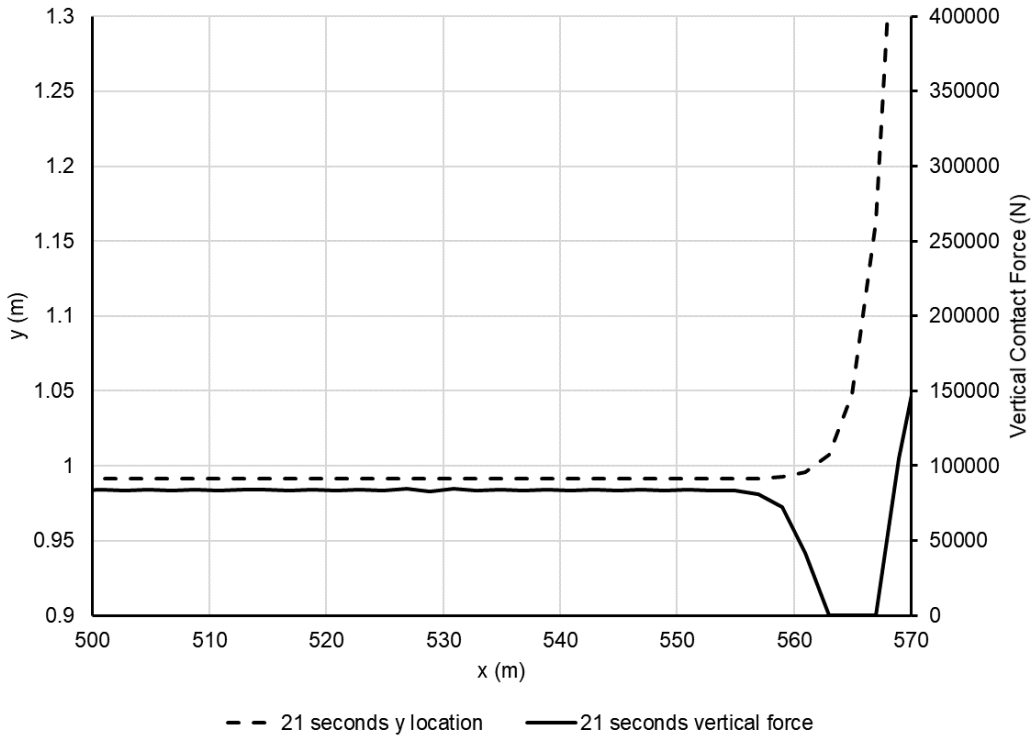

Figure 3-20: Vertical surface reaction force and chain location plotted against distance at the point where the chain leaves the surface it is resting upon at 21 seconds, simply bonded particles.

3.3.2.2 Indentation Contact

The contact behaviour of a sphere and a plane is a well-studied problem in contact mechanics with Hertz providing the first of many treatments of the problem (Hertz, 1881). In this example a sphere of elements contacts a plane under its own weight (Figure 3-21). Analytical solutions are known which describe the contact area and indentation depth between the plane and the sphere (Villaggio, 1996).

The analytical and numerical results are normalized over an interval by dividing by the maximum load and the maximum indentation or contact area. The results are shown in Figure 3-22. Figure 3-23 depicts the evolution of particles in the sphere which are in contact with the plane as the force increases.

It can be seen from the results that the indentation depth prediction is closer to the analytical solution that the contact area prediction. The contact area is determined by fitting a convex hull to all the particles in contact with the plane. Due to the discrete nature of the contact points, this method of estimating the contact area could be inaccurate.

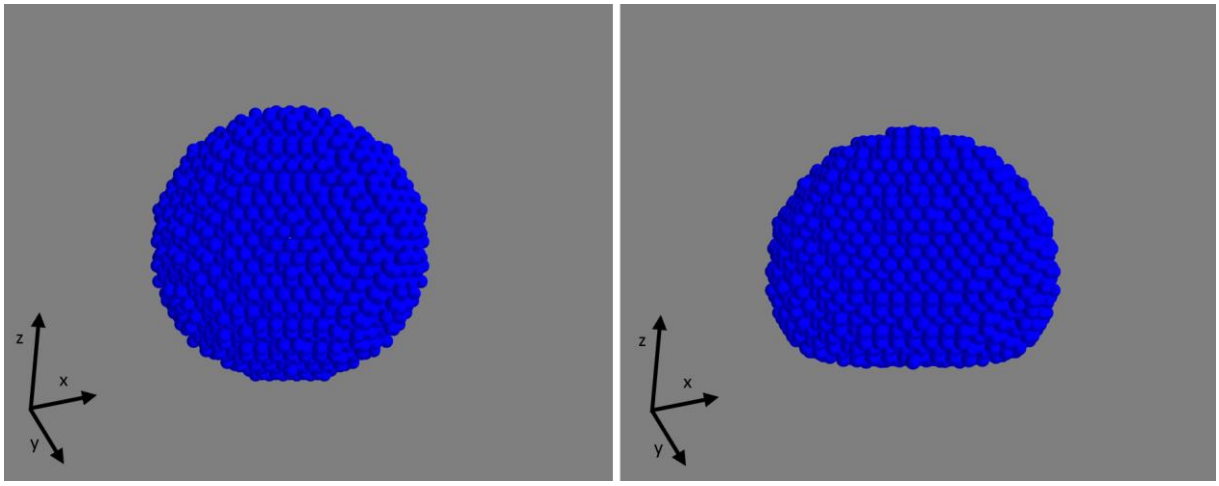

Figure 3-21: Sphere of bonded particles, free (left) and resting on a horizontal plane (right).

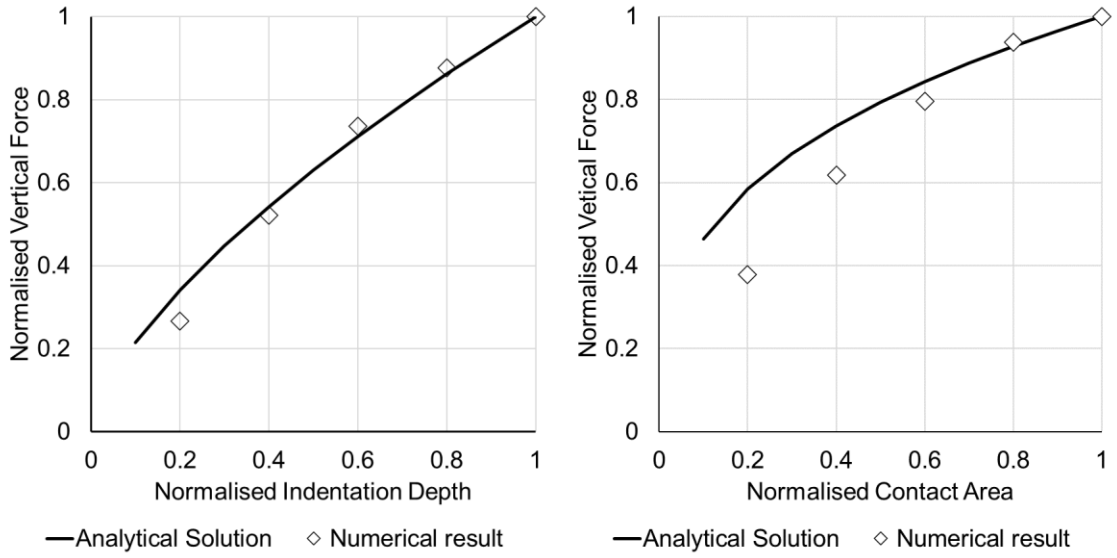

Figure 3-22: Numerically predicted indentation depth (left) and surface contact area (right) compared to the analytical result.

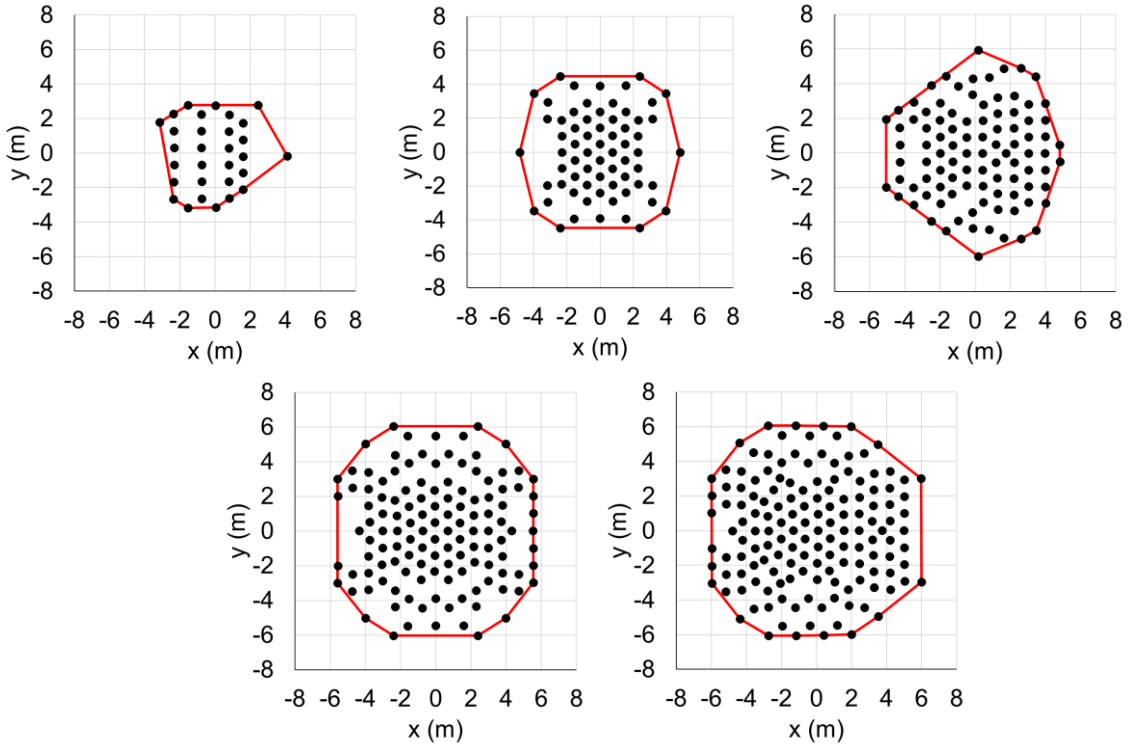

Figure 3-23: Particles in contact with the horizontal plane as the load increases, the fitted convex hull is shown.

3.3.2.3 Fibre debonding

Thus far all examples have dealt with bonds with no failure behaviour. The addition of bond breakage allows problems exhibiting fracture and de-bonding to be explored. The example simulated in this paper is the extraction of a stiff fibre from a more pliable matrix. The matrix is fixed at one end and the fibre is extracted away from the fixed end (Figure 3-24). The bonds at the interface between the fibre and matrix obey a bilinear softening rule which is a function of the shear displacement at the interface.

Calibration of the bond parameters was done in two stages. In the initial stage the matrix and fibre bonds were calibrated for ***Kn*** and ***Ks*** to produce the required Youngs Modulus and Poissons ratio using a similar approach to the line search employed for example 1. The next phase of the calibration focused on ensuring the fibre embedded in the matrix behaved as predicted analytically during the linear elastic phase of the fibre pull-out. The area of the interfacial bonds between the fibre and the matrix was scaled such that the total area of the interfacial bonds is equal to the expected value for a fibre with equivalent embedded length and diameter. The ***Kn*** and ***Ks*** parameters for the fibre bonds are then further modified such that the analytically predicted linear elastic phase of the fibre pull out behaviour is matched. The second step is required to account for the large particle size relative to the fibre diameter.

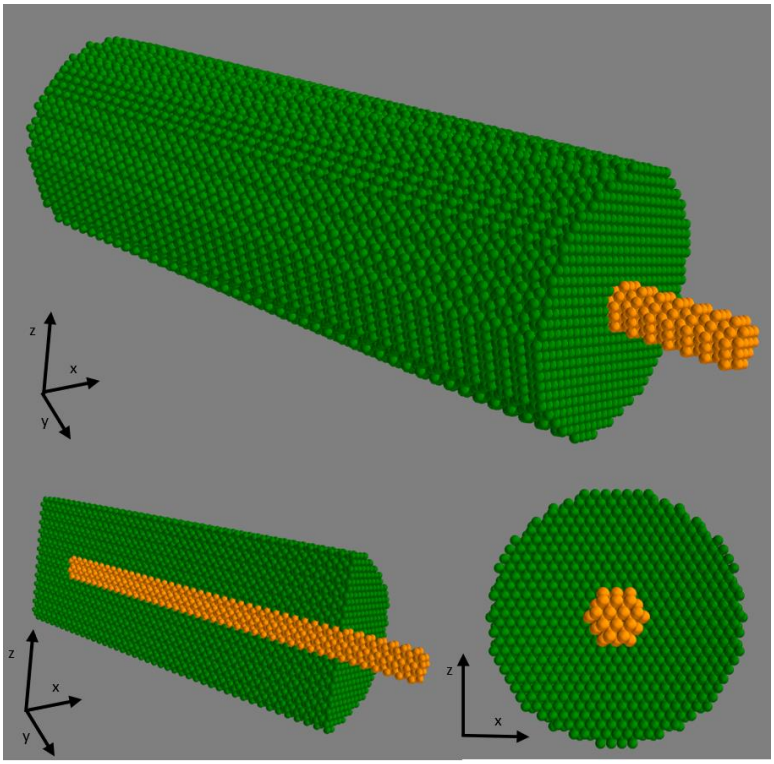

Figure 3-24: 3D models (cross section on lower left) of the simulated fibre (orange) embedded in a matrix (green), particle diameter = 0.5m.

The interfacial bond shear force magnitudes during the debonding of the fibre are plotted along the length of the fibre for different extents of debonding in Figure 3-25. The analytically predicted shear forces are also plotted. There is good agreement between the analytical prediction and the numerical result from the DEM model.

It can be shown that the de-bonding force/displacement behaviour follows a compound path and requires an arc-length limiting algorithm to extract the full stress-strain path. Arc-length limiting algorithms are techniques applicable to numerical methods such as FEM and cannot be applied to DEM. To mimic the effect of an arc-length limiting algorithm a loading/unloading approach was used. The loading/unloading algorithm employed limited the amount of damage that could occur along the interface before the fibre and matrix are returned to their starting position before being loaded again. This approach proved to be useful in following the full force/displacement curve for this problem (Figure 3-26).

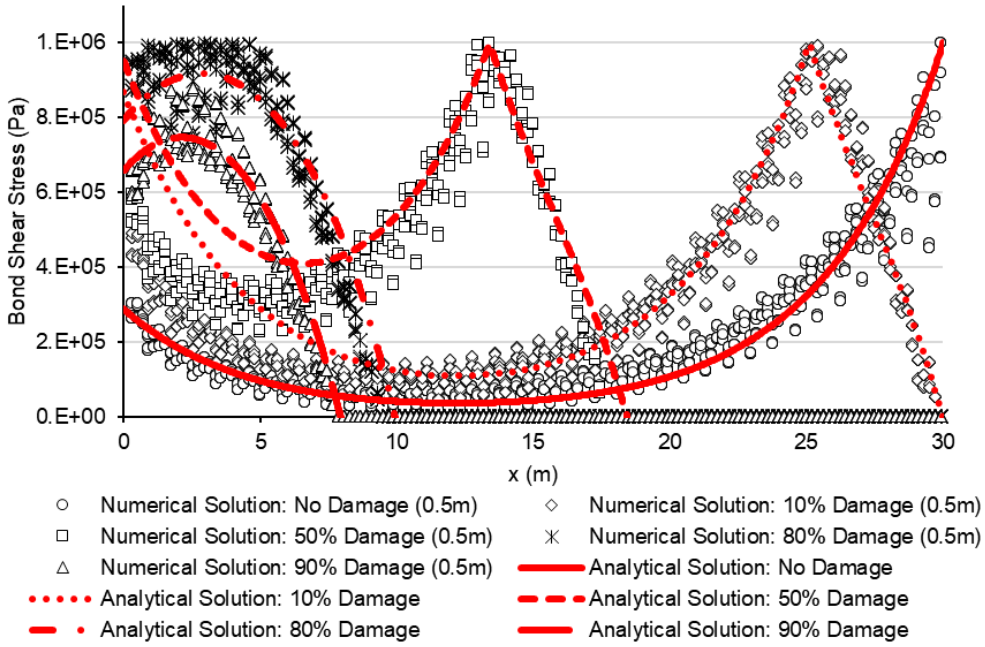

Figure 3-25: Bond shear stresses (in the z-direction) plotted against the bond centroid location along the interface length for a particle diameter of 0.5m. The bond shears are plotted for five points during the pull-out curve. The analytically predicted shear forces are also plotted.

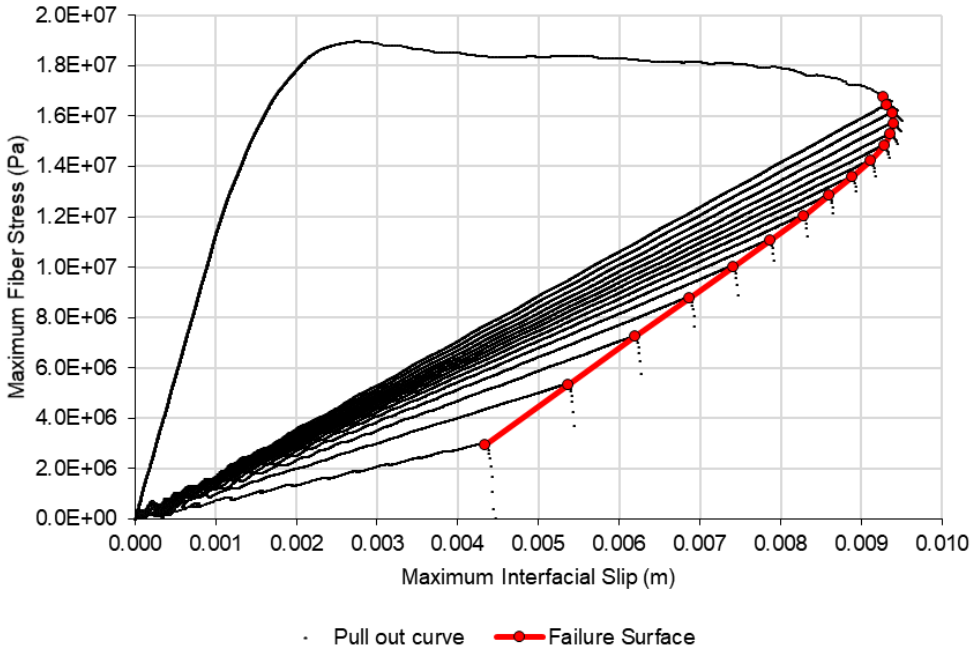

Figure 3-26: Full failure surface. The failure surface is defined as the minimum combination of interfacial stress slip and fibre stress which results in the remaining bonds accruing damage. It is apparent that the previous history of damage to the interface plays a role in the shape of the failure surface. Beyond a maximum interfacial slip further damage to the interface occurs at reducing load and slip

3.4 Conclusion

Graphics Processing Units (GPUs) provide a computational framework to execute computations in parallel efficiently. Specific problems such as Discrete Element Models (DEM) are well suited to optimization by parallelization. The Python programming language together with the Numba module was used to develop a GPU accelerated double-precision DEM code. The DEM was developed to investigate systems of bonded particles which undergo little relative motion. New collisions are not expected to form during the running of the simulation.

Collision detection was executed on GPU using a spatial hashing approach for broad phase collision detection. Particle force determination and equation of motion integration was undertaken on GPU. The performance of the developed code was investigated by comparing execution speeds for three different versions of the code: pure Python, Python optimized with Numba, and Python optimized for execution on GPU. It was found that Numba optimization performed the fastest in the collision detection phase, but the GPU optimized code performed the fastest for force determination and integration.

Example problems that are well suited to the strengths of the developed code were modelled. The chain fountain (the self-siphoning effect of a flexible chain) was investigated. A minimal experimental setup which should produce similar behaviour to a full-scale physical experiment was used. The expected excess reaction force at the chain pick-up point was recovered for a chain bonded with a parallel bond model and was absent for the LSDP bonded chain. Further investigation is recommended to investigate this effect further. The contact force and the area between a sphere composed of bonded particles and a plane was simulated. The analytically predicted behaviour compared well with the simulated model.

The DEM code developed for this study was used to investigate the debonding behaviour of an embedded fibre. Some of the results of that investigation are presented here. The complete results of that study will be presented in a further paper.

4 Investigation into the use of Discrete Element Models for simulating uniaxial pull-out of an embedded fibre

The combination of brittle material with ductile fibres can produce competent composites. The fibres transmit tensile forces across cracks that form in the brittle matrix at relatively low tensile strains. The fibre reinforcing, therefore, acts to both increase the maximum stress a structural section can support and improve the post maximum stress behaviour from brittle to ductile failure. Concrete is the most common construction material but suffers from a low tensile strength which necessitates the use of reinforcing. Typical concrete reinforcing consists of relatively large diameter steel bars orientated according to the expected loads. Advances in fibre production methods and a reduction in cost has led to an increase in the use of smaller, thinner fibres in concrete in the form of Fibre Reinforced Concrete (FRC). An essential aspect of defining the effectiveness of fibre reinforcing is resolving the behaviour of the interface between the fibre and the matrix as the load being transmitted between the matrix and fibre increases. Once the behaviour of the interface is well understood the length, diameter and shape of the fibres can be optimised to maximize their effectiveness as reinforcing members.

The interface behaviour for simple fibres is understood analytically, and several models exist that can predict the stresses in the interface. Numerical models using finite element methods (FEM) have been used to investigate this problem in a more general way. FEM, being inherently a description of a continuum, does not elegantly describe the debonding process that occurs during the debonding of fibres from the surrounding matrix. Discrete Element Methods (DEM) describe continuous and discontinuous materials as the interaction between multiple independent particles. Many different models have been developed to describe the force-displacement relationship between particles within the DEM framework. These models range from complex to simple. For this study two different DEM contact models are compared to investigate the model complexity that is required to describe fibre/matrix interface stresses accurately. A simple model (a linear spring model that only transmits normal and tangential forces) and a more complex model (parallel bonds which transmit normal and tangential forces, moments, and torsion) were used. Two stages of fibre pull-out were modelled independently using a GPU accelerated DEM simulator developed by the author: a fully bonded stage and the de-bonding stage. It was found that both models were able to simulate all stages when compared to analytical solutions. No improvement to the model behaviour was evident from using a complex contact model; for this reason, a simpler, faster contact model should be used to analyse this problem.

4.1 Introduction

The increased utilization of short-fibre composites in various industries has spurred research into the mechanisms of force transfer between fibre and matrix as well as failure in fibre reinforced materials. The fibre reinforced composite material can fail in three primary modes: i) fibre failure, ii) interface failure, and iii) matrix failure (Naaman *et al.*, 1989). Which of these modes is dominant for a given composite is a function of the matrix, fibre, interfacial strength, stiffness parameters, fibre geometry and loading.

To effectively design a fibre reinforced composite, it is necessary to understand which failure mode is dominant. An optimal outcome for toughness would be to maximize the energy dissipation of the composite. By adding short-fibre load-bearing elements into a cementitious matrix, the toughness of the composite is enhanced (Aslani and Nejadi, 2011; Jd, 2014). The primary mechanism increasing the toughness of fibre reinforced material is spanning of cracks by the fibres. The spanning action increases the energy needed to propagate cracks in the material.

Fibres also modify the pre-cracking behaviour of the matrix by increasing the effective Youngs Modulus of the composite. The load transfer characteristics for an individual fibre are not symmetric. Maximum transfer of load into the fibre occurs for loading parallel to the long axis of the fibre. By increasing the fibre length, the strength of the composite can be improved. However, the maximum load to fibre length relation stagnates as the fibre length increases. Therefore, by increasing the fibre length beyond a specific length does not improve the maximum load of the composite.

The primary mode of force transfer between a smooth fibre and matrix is through shear stresses developed at the interface. The magnitude of the interface shear stress is a function of the slip between the fibre and the matrix. Since most composites have significantly different stiffnesses between the matrix and fibres, the stress distribution on the interface is non-linear.

For partially embedded fibres, the stiffness ratio between the fibre and matrix dictates whether the maximum interface shear stress occurs at the embedded end of the fibre or the entry point of the fibre into the (Chen and Yan, 2015). Fibre pull-out of partially embedded fibres occurs in three phases (Naaman *et al.*, 1989; Chen, Beyerlein and Brinson, 2009), as shown in Figure 4-1 a) stress growth phase, Figure 1-1 b) debonding phase, and Figure 1-1 c) friction phase. During the stress growth phase, the interface between the fibre and matrix is entirely intact. Loads are transferred from the fibre to the matrix predominantly in a linear elastic manner for brittle matrices. As the interface stress increases, it reaches the debonding failure limit, beginning the onset of the debonding phase. The debonding phase is characterized by the onset and development of a failure zone from either one of the ends or both ends of the fibre. The interface damage is irreversible, and the load-displacement relation starts to deviate from the typical linear relationship. Once the failure zone has developed over the entire interface, i.e. completely debonded the particle from the matrix, forces are only transferred between the matrix and fibre by friction when pulled along the fibre direction.

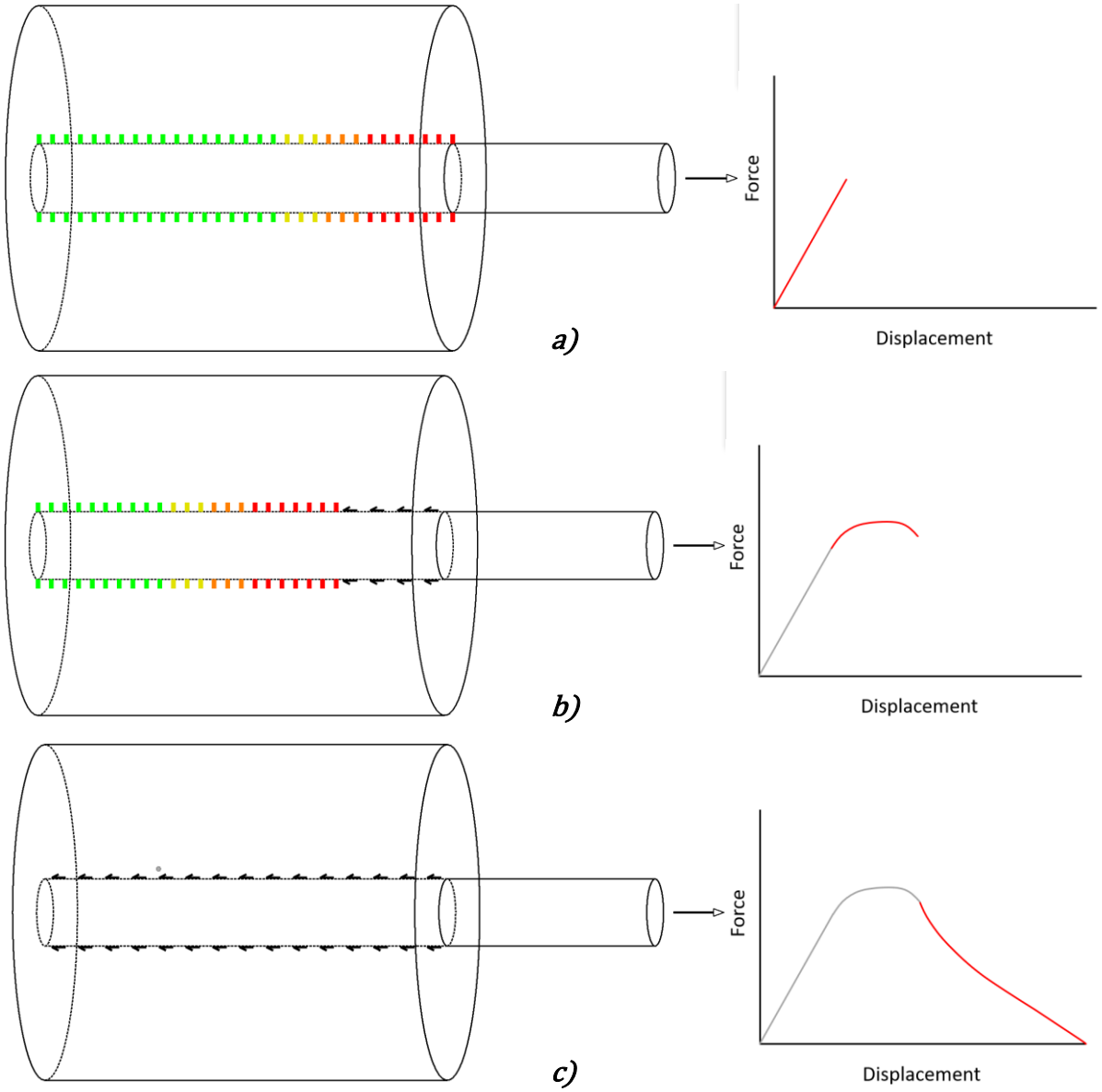

Figure 4-1: Pull-out behaviour of an embedded fibre in a matrix. Cohesive bonds between the matrix and fibre are represented with coloured lines (red = higher shear force). a) Stress growth phase: force between fibre and matrix is only transmitted via bonds. A typical scenario is that the maximum interfacial shear force occurs towards the free end of the fibre. b) Debonding phase: the maximum shear stress in the vicinity of the interface is exceeded, and those bonds break. c) Friction phase: all the bonds on the interface have broken, and force is only transmitted via friction between the fibre and matrix along the tensile direction.

Analytical solutions that describe the stresses in a system of a single embedded fibre have been developed (Chen and Yan, 2015). The shear lag approach can be used to describe the elastic stresses. Experimental confirmation of the shear lag theory is given by (Nairn, 2000). The shear lag theory is valid only for an embedded fibre with no debonding along the interface. An extension to include the debonding mechanism was made by (Chen and Yan, 2015). The standard theory was modified by superimposing a stress distribution, defined as a function of the slip in the interface, which represented the damage along the interface. This method has been used in further research on composite debonding (Guo and Zhu, 2015;

Heidarhaei, Shariati and Eipakchi, 2018, 2019; Lau *et al.*, 2018; Arain *et al.*, 2019; Budarapu *et al.*, 2019).

The stress growth and debonding phases of fibre pull-out have been modelled using computational continuum approaches such as the finite element method (FEM) (Denneman *et al.*, 2011; Jansson, 2011). FEM utilizes bonding elements to make allowance for fracture (Kang *et al.*, 2014). A discrete damage zone model can be used to simulate the formation of fractures in FEM (Liu, Duddu and Waisman, 2012). In FEM this can be achieved by connecting two domains of the finite element mesh using discrete spring elements. The discrete spring elements transfer forces between nodes of the mesh. By decomposing the shear and normal strains in an element, different modes can be simulated. A fracture occurs in one of three modes: Mode 1 (Tension), Mode 2 (Shear) and Mode 3 (Tearing). The debonding along a fibre for pull-out occurs as a mode 2 failure (Chen and Yan, 2015). Most approaches incorporating fracture into a FEM model make a priori assumptions concerning the orientation and location of the fractures in the material (Zhan Y. und Bui, 2014). Approaches in FEM which explicitly model fracture growth resort to computationally expensive remeshing to capture rapidly varying stresses at the crack tip and conform the mesh to the crack surface. Most FEM models do not make allowance for a collision between different areas of the meshed area. To incorporate the interaction between different areas of the mesh, the interaction is modelled using spring elements. Tracking a large number of potential interactions requires a robust collision detection method which reduces the computational advantages of FEM. FEM is, therefore, less suited to scenarios where evolving contact scenarios are present when compared to a method where collision detection is central to the solutions approach.

Extended FEM or XFEM is a modification to the classical FEM approach which seeks to model fracture while eliminating the need for remeshing. XFEM can resolve discontinuities without requiring the mesh to align with the boundaries of such discontinuities. XFEM has been used to investigate pull-out of rebar and fibre matrix interfaces achieving good agreement with experimental results (Bouhala *et al.*, 2013; Orlando and Benvenuti, 2016).

Discontinuum approaches, such as the discrete element method (DEM), provide another possibility to model fibre reinforcing (Yang *et al.*, 2010). DEM is most well-known in modelling the behaviour of granular materials (Potyondy and Cundall, 2004) by modelling numerous individual particles interacting. It is, however, possible to model a continuum using DEM by applying bonds between the individual particles (Leclerc *et al.*, 2017). Fracturing can then be simulated by merely applying a failure criterion to these bonds. DEM has proven to be well suited to the modelling of material where fracturing has a significant influence on the material strength, e.g. concrete (Monteiro Azevedo, 2003; Potyondy and Cundall, 2004; Azevedo and Lemos, 2006; Yang *et al.*, 2010). The fracture behaviour of composites has been investigated using DEM (Yang *et al.*, 2010; Sheng *et al.*, 2010; Potapov, Faucher and Daudeville, 2012; Wolff *et al.*, 2013; Koval, Danh and Chazallon, 2014; Sadek *et al.*, 2014; Maheo *et al.*, 2015; Ismail, Yang and Ye, 2016; Zhang and Xie, 2017; Leclerc *et al.*, 2017; Marcon *et al.*, 2017; Wang *et al.*, 2017; Shang *et al.*, 2018). The stress distribution along the fibre/matrix interface has, however, not been investigated, or compared to analytical solutions. Many DEM models of composite materials do not accurately resolve the interfaces between materials but seek to reduce the number of particles required by averaging material properties. Research has also been limited due to the computational demands of DEM. This study seeks to investigate further the use of DEM specifically for the resolution of the debonding behaviour of a single fibre embedded in a matrix. The stress distribution along the interface before and during debonding can have

important implications for the design of fibre reinforced composites. Confirming that a DEM model can recover the expected interfacial stresses would allow more complex embedded fibre problems to be investigated.

Two popular models for the modelling of bonded particles in DEM are the linear-spring dash-pot (LSDP) (Monteiro Azevedo, 2003) and parallel bond models (Potyondy and Cundall, 2004; Herman, 2015) presented in Section 3. This study investigates the sensitivity of the modelled debonding behaviour to the chosen contact model in Section 5.3.3. As the size of the particles used in DEM decreases the behaviour of the DEM model approaches that of a continuum (Cundall, 1984). It is therefore essential to understand the influence of particle size on the required accuracy. The effect of particle size is, therefore, investigated in Section 5.2.1 and Section 5.3.4, and compared to analytical solutions presented in Section 2. The results of the simulation for the fully bonded and debonding phase are verified against analytical solutions in Section 5. The frictional phase is not considered in this study. For all simulations, the same packing is used (hexagonal close packing). The interface is modelled using shear only elements obeying a bilinear softening law as described by Liu, Duddu and Waisman, (2012). The total area of the bonds on the interface is scaled to match a continuous fibre, as discussed in Section 4. The fibre and the matrix are modelled using either LSDP or parallel bond bonded elements. DEM allows for the investigation of both the bonded and debonding phases of failure without changing the modelling approach. The uniaxial pull-out of a large fictitious fibre proposed by Chen and Yan, (2015), is considered as shown in Figure 4-2.

The compute in this study was accelerated by developing and implementing a GPU based DEM framework for bonded particle applications in the Python programming language. The code executes on General Purpose Graphics Processing Units (GPGPU) philosophy. GPGPU is well suited to the parallel nature of DEM (Amada *et al.*, 2004; Govender, Wilke and Kok, 2015; Qi *et al.*, 2015).

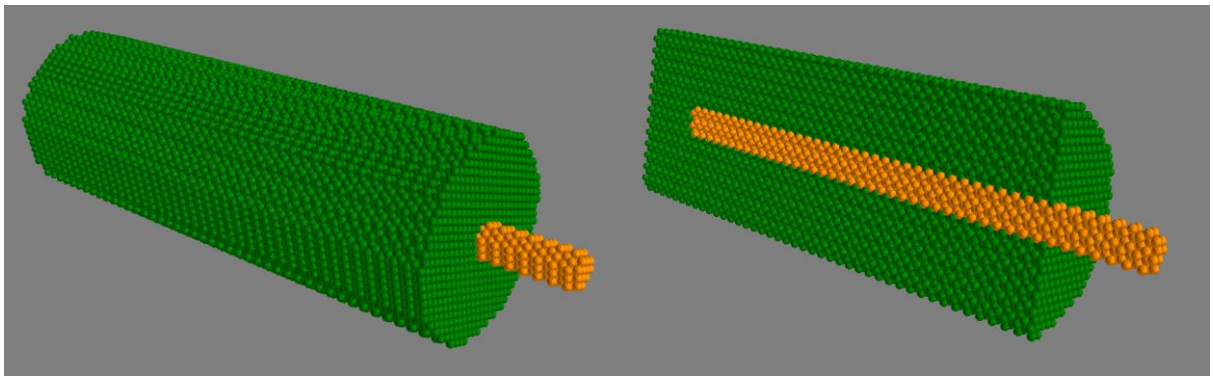

Figure 4-2: 3D models (cross section on right) of the simulated fibre (orange) embedded in a matrix (green), particle diameter = 0.5m.

The results of the simulation show that the simulation is not sensitive to the choice of the contact model used. As LSDP requires the least computational effort it is recommended to be used for this sort of simulation. It was also found that the result is sensitive to the size of the particles used in the simulation. The size of the particle should be at most 25% of the fibre diameter to capture the correct form of the pull-out curve.

Future study is recommended to investigate the effect that different packings would have on the results. Breakage of the matrix and fibre, as well as interaction between multiple

fibres, should also be investigated. The Python module used to write the program used in this study has limitations; a language such as C++ would be better suited for future work.

4.2 Analytical Solution

The mechanisms whereby forces are transferred between a fibre embedded in a matrix are well understood and widely studied (Nairn, 1997, 2000; Nairn *et al.*, 2001; Avery, 2016; Hsueh, 1988; Naaman *et al.*, 1989; Chen and Yan, 2015). The design of fibre reinforced concrete, traditional reinforcing, soil anchors and foundation piles all benefit from the resolution of the stress transfer between the fibre (or linear feature) and the matrix.

The stress field for the elastic behaviour of a fibre/matrix stress problem can be derived using a shear lag approach. The term “shear lag” originates from the study of the design of T, I and box beams (Reissner, 1946).

A typical model used when investigating the mechanics of a fibre pull-out problem is one of a cylindrical fibre embedded within a cylindrical matrix (Figure 4-3).

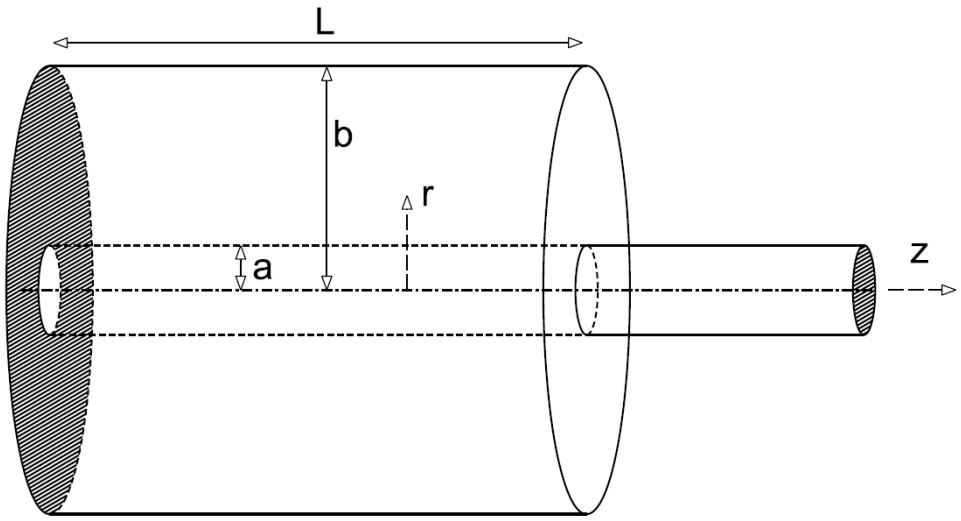

Figure 4-3: The standard problem assessed of a fibre embedded in a cylindrical matrix element. The left end of the matrix is fixed while the right end of the fibre is displaced to the right.

The constitutive relationship of a fibre embedded in a matrix can be described with the following equations (Chen and Yan, 2015). The fibre and the matrix are assumed to deform according to:

$$\varepsilon_{zz} = \frac{\sigma_{zz}}{E_A} - \frac{v_a}{E_A}(\sigma_{rr} + \sigma_{\theta\theta}) + \alpha_A T \tag{20}$$

$$\gamma_{rz} = \frac{\tau_{rz}}{G_A} \tag{21}$$

Where ε_{zz}, σ_{zz}, σ_{rr} and $\sigma_{\theta\theta}$ are the strain in the z-direction, the stress in the z-direction, stress in the direction normal to the fibre and stress in the radial direction respectively. τ_{rz} and γ_{rz} and the shear stress and shear strain. E_A, G_A, v_A and α_A are the fibre or matrix tensile modulus, shear modulus, Poissons ratio and thermal expansion coefficient respectively. In addition to these constitutive relationships the following axial equilibrium relationships must be met:

$$\frac{\partial \sigma_{zz}}{\partial z} + \frac{1}{r}\frac{\partial (r\tau_{rz})}{\partial r} = 0 \tag{22}$$

It can be shown that the fundamental shear lag assumption assumes the following relationships for the z strain and shear strain (Nairn 1996):

$$\varepsilon_{zz} = \frac{\partial \omega}{\partial z} \tag{23}$$

$$\gamma_{rz} = \frac{\partial \omega}{\partial r} \tag{24}$$

Where ω is the axial displacement of the fibre. To resolve the stresses in the fibre and matrix, assumptions need to be made about the form of the functions describing the shear stress distribution in the fibre and matrix.

$$\tau_{rz}^{fiber}(r,z) = \tau_a(z) I(r) \tag{25}$$

$$\tau_{rz}^{matrix}(r,z) = \tau_a(z) O(r) \tag{26}$$

$\tau_a(z)$ is the axial shear stress at the interface and $I(r)$ and $O(r)$ describe the shear stresses in the fibre and matrix as a function of r (r is the distance from the centre of the fibre). The shear stress at the centre of the fibre and the outer surface of the matrix must be zero. Furthermore, the shear stresses at the outer edge of the fibre, and the inner edge of the fibre are identical. Different assumptions have been made concerning the form of these relationships. Assuming no shear stress in the fibre results in $I(r) = 0$. Chen and Yan (2015) suggest the following functions for $I(r)$and $O(r)$:

$$I(r) = \frac{r}{a} \tag{27}$$

$$O(r) = ab\frac{\left(\frac{b}{r} - \frac{r}{b}\right)}{(b^2 - a^2)} \tag{28}$$

The stress at the interface between the fibre and the matrix is modelled as being a function of the slip between the two materials. There is, therefore, a discontinuity at the interface described according to the following equation:

$$w^{fiber}(a,z) - w^{matrix}(a,z) = \delta(z) \tag{29}$$

It was shown by Chen and Yan (2015) that the governing shear lag equation could be described as follows:

$$\frac{d^2\sigma_f(\zeta)}{d\zeta^2} = \alpha^2\left[\sigma_f(\zeta) - \gamma\sigma_p - \sigma_T\right] \tag{30}$$

ζ is a scaling factor defined as z/a, $\alpha$and $\gamma$are constant coefficients of the equation and are functions of the material properties and geometry as well as the assumed functions describing the radial distribution of shear stress in the fibre and matrix. σ_f, $\sigma_p$and $\sigma_T$are the stress in the fibre, loading stress at the fibre free end and the stress induced by temperature respectively. This second-order differential equation of ζ can be solved under the following boundary conditions:

$$\sigma_f(\zeta = 0) = \epsilon\sigma_p \tag{31}$$

$$\sigma_f(\zeta = l) = \sigma_p \tag{32}$$

The first boundary condition describes the stress at the embedded fibre end as a function of the stress at the free (loaded) fibre end. The parameter ϵ describes how much stress is transferred into the matrix at the embedded end. The second boundary condition fixes the stress at the free (and loaded) end to be equal to the applied stress. The final solutions of the differential equation have the form (Chen and Yan, 2015):

$$\sigma_f = \sigma_p\varphi(\zeta, l, \alpha) \tag{33}$$

$$\tau_a = \frac{\sigma_p}{2}\omega(\zeta, l, \alpha) \tag{34}$$

$$\delta = \frac{\sigma_p}{2K_0}\omega(\zeta, l, \alpha) \tag{35}$$

Where

$$\varphi(\zeta, l, \alpha) = \frac{sinh(\alpha\zeta)}{sinh(\alpha l)} - \epsilon\frac{sinh\big(\alpha(\zeta - l)\big)}{sinh(\alpha l)} + \gamma\left[1 - \frac{sinh(\alpha\zeta) - sinh\big(\alpha(\zeta - l)\big)}{sinh(\alpha l)}\right] \tag{36}$$

$$\omega(\zeta, l, \alpha) = \varphi'^{(\zeta,l,\alpha)} = \alpha\left[\frac{cosh(\alpha\zeta)}{sinh(\alpha l)} - \epsilon\frac{cosh\big(\alpha(\zeta - l)\big)}{sinh(\alpha l)} - \gamma\frac{cosh(\alpha\zeta) - cosh\big(\alpha(\zeta - l)\big)}{sinh(\alpha l)}\right] \tag{37}$$

The form of the interfacial shear stresses as described by shear lag theory is a function of the fibre/matrix stiffnesses and geometry. The maximum shear stress occurs close to the point where the fibre exits the matrix or close to the embedded end depending on these parameters. If the fibre is stiffer than the matrix, the maximum interfacial shear will occur at the free end with the inverse true if the matrix is stiffer than the fibre. The distribution of interfacial shear stresses for different fibre/matrix Youngs Moduli with the free end on the right are plotted in Figure 4-4.

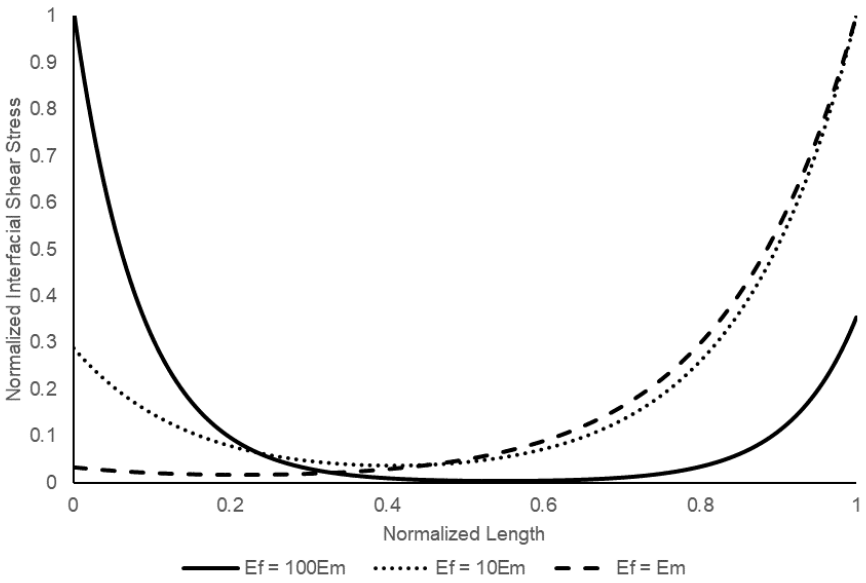

Figure 4-4: Analytically predicted shear stress for different fibre/matrix Youngs Moduli combinations.

The shear lag theory does not make provision for debonding between the fibre and the matrix. Chen and Yan (2015) have incorporated the debonding behaviour by superimposing a second stress system in areas where the fibre and matrix are in a debonding stage. The central assumption of this approach is that the debonding of the fibre and the matrix does not occur suddenly but exhibits a yielding behaviour before complete failure occurs. An example of this approach is shown in Figure 4-5. The length of the embedded fibre is 30m and is being extracted towards the right. The magnitude of shear at a point in the interface is the product of the slip between the matrix and fibre and an interfacial stiffness parameter K (1e9 here). The interface begins yielding at a slip value of 0.001m and fails at 0.003m. This results in a yield shear stress of 1e6 Pa (0.001x1e9).

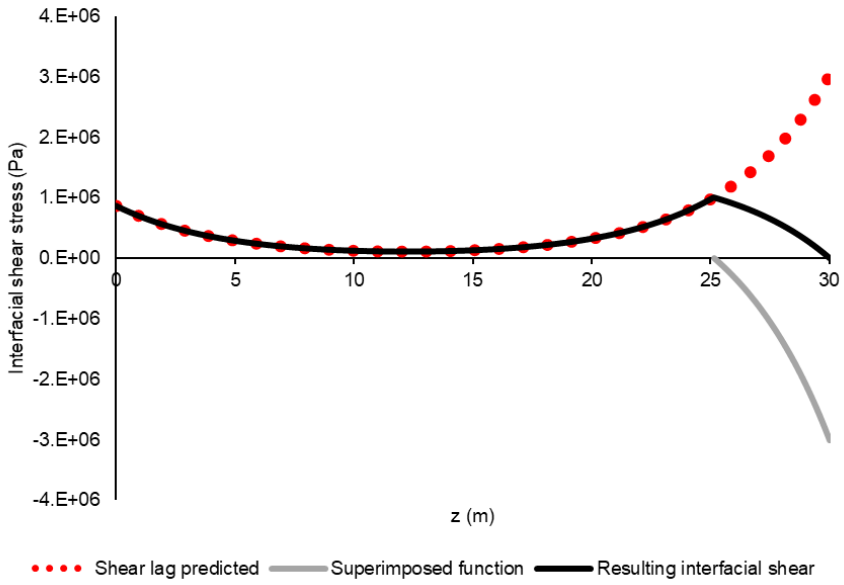

Figure 4-5: The superposition method to incorporate interfacial damage into the shear lag theory description of fibre pull-out.

4.3 Discrete element formulation

The Discrete Element Method (DEM) was first proposed by Cundall (1971 and 1974) to investigate the behaviour of granular assemblies. DEM employs discrete elements (such as discs, spheres or polyhedra) which interact only at their points of contact. The two phases in the application of DEM consist of the application of a force-displacement law at contact points and the application of Newton's second law to the elements themselves. Deformation of the particles themselves is assumed to be sufficiently small (compared to the deformation of the overall assembly) that they can be assumed to be rigid. Applying Newton's second law to the translational and rotational motion of individual particles results in (Rojek *et al.*, 2012):

$$m_i \ddot{\bar{u}}_i = \overline{F_i} \tag{38}$$

$$J_i \dot{\overline{W_i}} = \overline{M_i} \tag{39}$$

Where m_i, $\bar{u}_i$ and $\overline{F_i}$ are the mass, displacement, and resultant particle force of the *i*th element. J_i, $\overline{W_i}$ and $\overline{M_i}$ are the particle moment of inertia, angular velocity and resultant moment of the *i*th element. The forces and moments acting on the *i*th particle can be described according to:

$$\overline{F_i} = \overline{F_i^{ext}} + \sum_{j=1}^{n_i^c} \overline{F_{ij}^{contact}} + \overline{F_i^{damp}} \tag{40}$$

$$\overline{M_i} = \overline{M_i^{ext}} + \sum_{j=1}^{n_i^c} \overline{M_{ij}^{contact}} + \overline{M_i^{damp}} \tag{41}$$

Where $\overline{F_i^{ext}}$ and $\overline{M_i^{ext}}$ are any external forces and moments acting on the particle. $\overline{F_{ij}^{contact}}$ and $\overline{M_{ij}^{contact}}$ are the force and moment due to interaction between the *i*th and *j*th element and n_i^c is the number of particles interacting with the *i*th particle. $\overline{F_i^{damp}}$ and $\overline{M_i^{damp}}$ are damping forces and moments which are applied to reduce spurious oscillations in the system.

Various contact models can be used to calculate forces at the contact points as a function of the particle overlap distances and velocities. The calculated forces are accumulated onto the individual particles participating in the interaction. Contact models can be tailored to represent non-cohesive (no tension forces, such as between individual sand grains) and cohesive (such as between particle representing rock) material (Potyondy and Cundall, 2004).

An explicit numerical scheme is employed to integrate over particle accelerations twice in time to recover particle velocities and displacements successively. The integration is undertaken over a given timestep which is equal for all particles in the simulation. The explicit numerical scheme is conditionally stable with a maximum timestep above which the model becomes unstable.

The DEM formulations used for this study were the parallel bond and linear spring and dash pot (LSDP) formulations (Cundall and Strack, 1979; Potyondy and Cundall, 2004; Rojek *et al.*, 2012). For each pair of bonded particles, a bond is formed which transmits bending moments (for parallel bonds), normal forces (for parallel and LSDP), shear forces (for parallel and LSDP) and torsional moments (for parallel bonds) between the particles. Forces are calculated according to the inter-particles tangential and normal strains.

4.3.1 Bond displacement determination

At the beginning of each time-step the displacements between each pair of particles are determined by integrating the translational and rotational equations of motion twice. At each time step the relative velocity of two contacting particles (Figure 4-6) is determined according to (Potyondy and Cundall, 2004):

$$\overline{V_R^{ab}} = \overline{V^b} + \overline{W^b} \times \left(\overline{Co^{ab}} - \overline{P^b}\right) - \left(\overline{V^a} + \overline{W^a} \times \left(\overline{Co^{ab}} - \overline{P^a}\right)\right) \qquad (42)$$

Where:

$\overline{V_R^{ab}}$ is the relative velocity between particles a and b.

$\overline{V^b}$ is the velocity of particle b.

$\overline{W^b}$ is the angular velocity of particle b.

$\overline{Co^{ab}}$ is the location of the midpoint of the contact between particles a and b.

$\overline{P^b}$ is the location of particle b.

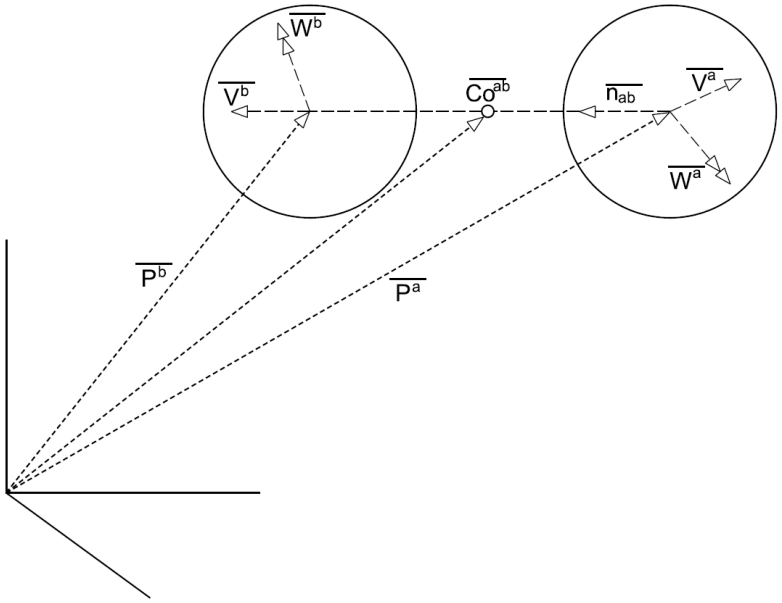

Figure 4-6: An example of two bonded elements.

The displacement increment in the last time step is determined according to the integration scheme employed (forward Euler method shown here):

$$\overline{dU^{ab}} = \overline{V_R^{ab}}\Delta t \tag{43}$$

Where:

$\overline{dU^{ab}}$ is the displacement increment between particles a and b.

Δt is the timestep.

The displacement increment is then partitioned into normal and tangential components.

$$\overline{dU_N^{ab}} = \overline{n_{ab}}\left(\overline{dU^{ab}} \cdot \overline{n_{ab}}\right) \tag{44}$$

Where:

$\overline{dU_N^{ab}}$ is the normal displacement increment between particles a and b.

$\overline{n_{ab}}$ is the normal vector between a and b.

The tangential displacement increment is given by:

$$\overline{dU_T^{ab}} = \overline{dU^{ab}} - \overline{dU_N^{ab}} \tag{45}$$

In the parallel bond model, it is also possible to transmit relative rotations (as bending moments) through the bonds. The relative angular velocity can be calculated from:

$$\overline{W_R^{ab}} = \overline{W^b} - \overline{W^a} \tag{46}$$

The increment of angular displacement can be calculated from:

$$\overline{d\Omega^{ab}} = \overline{W_R^{ab}}dt \tag{47}$$

The increment of angular displacement can also be decomposed into portions operating normal and tangentially to the bond direction:

$$\overline{d\Omega_N^{ab}} = \overline{n_{ab}}\left(\overline{d\Omega^{ab}} \cdot \overline{n_{ab}}\right) \tag{48}$$

$$\overline{d\Omega_T^{ab}} = \overline{d\Omega^{ab}} - \overline{d\Omega_N^{ab}} \tag{49}$$

4.3.2 Force determination – parallel bonds

The parallel bond model determines forces between interacting particles as being transmitted by a beam element. The bond can transmit normal and shear forces as well as moments and torques. Each force and moment can include a velocity-dependent dissipative component that can be visualised as a dashpot. The force transfer mechanisms for the parallel bond model is shown in Figure 4-7 the dashpots representing damping are not included.

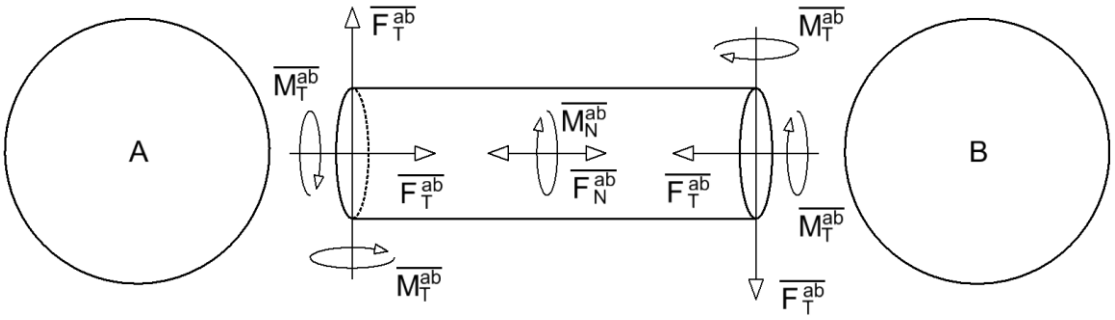

Figure 4-7: Schematic diagram of stress transfer for the parallel bond model.

The force increments for the normal force vector is given by

$$\overline{\Delta F_N^{ab}} = k_N^{ab} A^{ab} \overline{dU_N^{ab}}, \tag{50}$$

while the tangential force vector is given by

$$\overline{\Delta F_T^{ab}} = -k_T^{ab} A^{ab} \overline{dU_N^{ab}}. \tag{51}$$

The moment vector normal to the bond direction is given by

$$\overline{\Delta M_N^{ab}} = -k_T^{ab} J^{ab} \overline{d\Omega_N^{ab}}, \tag{52}$$

The moment vector tangential to the bond direction is given by:

$$\overline{\Delta M_T^{ab}} = -k_N^{ab} I^{ab} \overline{d\Omega_T^{ab}}. \tag{53}$$

The moment generated by the tangential force vector is given by:

$$\overline{\Delta M_{FT}^{b}} = \Delta F_T^{ab} \times \left(\overline{Co^{ab}} - \overline{P^{b}}\right), \tag{54}$$

$$\overline{\Delta M_{FT}^{a}} = \Delta F_T^{ab} \times \left(\overline{Co^{ab}} - \overline{P^{a}}\right), \tag{55}$$

k_N^{ab} and k_T^{ab} are the normal and tangential stiffness of the bond. The area, moment of inertia and the polar moment of inertia of the bond are given by, respectively

$$A^{ab} = \pi R^2 m \tag{56}$$

$$J^{ab} = \frac{1}{2}\pi R^4, \tag{57}$$

$$I^{ab} = \frac{1}{4}\pi R^4, \tag{58}$$

with R the average radius of the bond given by

$$R = \frac{R_a + R_b}{2}. \tag{59}$$

4.3.3 Force determination – LSDP

The most straightforward force-displacement law for interacting particles is a linear normal and tangential stiffnesses in DEM, represented by linear springs. A linear dashpot models dissipation for each transmitted force. Figure 4-8 depicts the force transfer mechanisms for the LSDP model (dashpots are omitted).

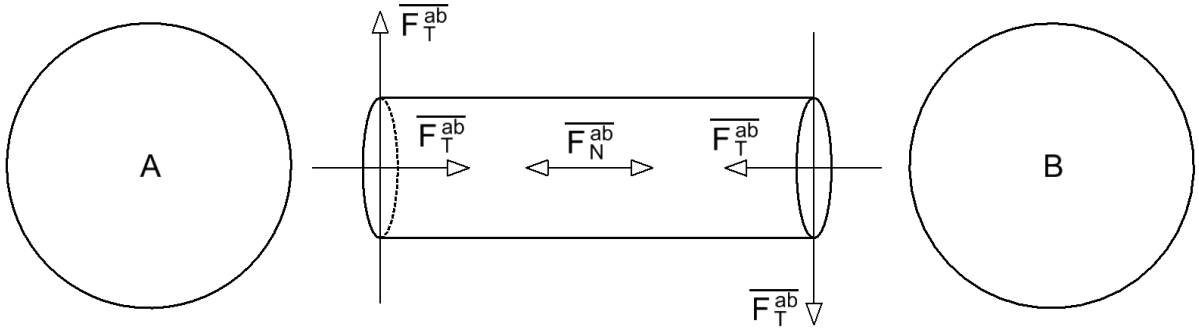

Figure 4-8: Schematic diagram of stress transfer for the linear spring dash-pot model.

Normal and tangential strain increments are calculated the same way as for the parallel bonds. The normal force increment vector is then given by:

$$\overline{F_N^{ab}} = k_N^{ab}\overline{U_N^{ab}} \tag{60}$$

The tangential increment force vector is given by:

$$\overline{F_T^{ab}} = -k_T^{ab}\overline{U_N^{ab}} \tag{61}$$

The moments applied to each particle are given by:

$$\overline{M_{FT}^{b}} = F_T^{ab} \times \left(\overline{Co^{ab}} - \overline{P^b}\right) \tag{62}$$

$$\overline{\mathrm{M_{FT}^{a}}} = F_T^{ab} \times \left(\overline{\mathrm{Co^{ab}}} - \overline{\mathrm{P^a}}\right) \tag{63}$$

Where:

$\overline{Co^{ab}}$ is the location of the midpoint of the contact between particles a and b.

$\overline{P^b}$ is the location of particle b.

4.3.4 Contact model discussion

The primary difference between the two contact models is that the parallel bond model can transfer bending moments. To illustrate the implications of this, consider a simply supported

horizontal string of particles at equilibrium under a uniformly distributed load in the vertical direction, e.g. gravity. The LSDP contact model results in the string of particles approximating a catenary curve, while the parallel bond model results in the deflection expected of a uniformly loaded beam, as shown in Figure 4-9. Differences between the DEM and analytical solution are due to the finite particle size.

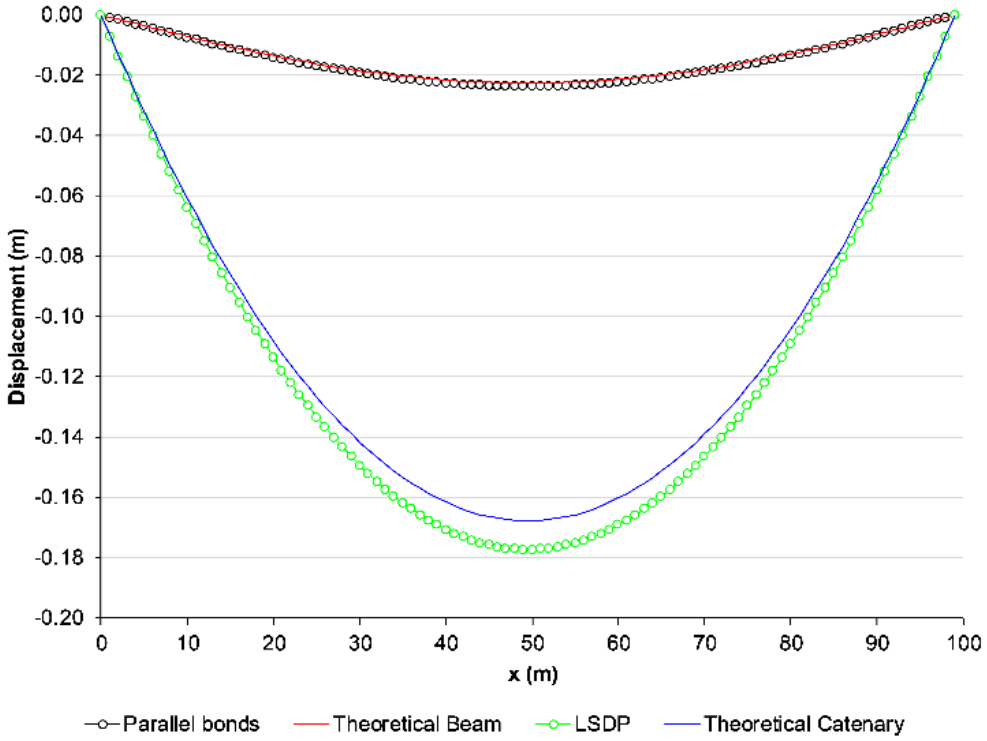

Figure 4-9: Theoretical and simulated equilibrium positions for a simply supported string of elements.

Consider a pair of particles just making contact. A constant counter clockwise rotation of 0.1 radians/s is applied to the first particle on the left located at x=y=0. The behaviour of the LSDP and parallel bond models are distinctly different, as shown in Figure 4-10 and Figure 4-11, respectively. For the LSDP model, the second particle rotates in the opposite direction, while for the parallel bonds model, the second particle rotates in the same direction. For both models, the rotation of the second particles is shown in Figure 4-12. Both models seek to minimize the relative velocity at the interface between the particles. The parallel bond model achieves this by moving the centroid of the second particle around the centroid of the first particle. The second particle rotates in the same direction as the first particle. The LSDP model achieves this by rotating the second particle in the opposite direction of the first particle while the centroid position remains unchanged.

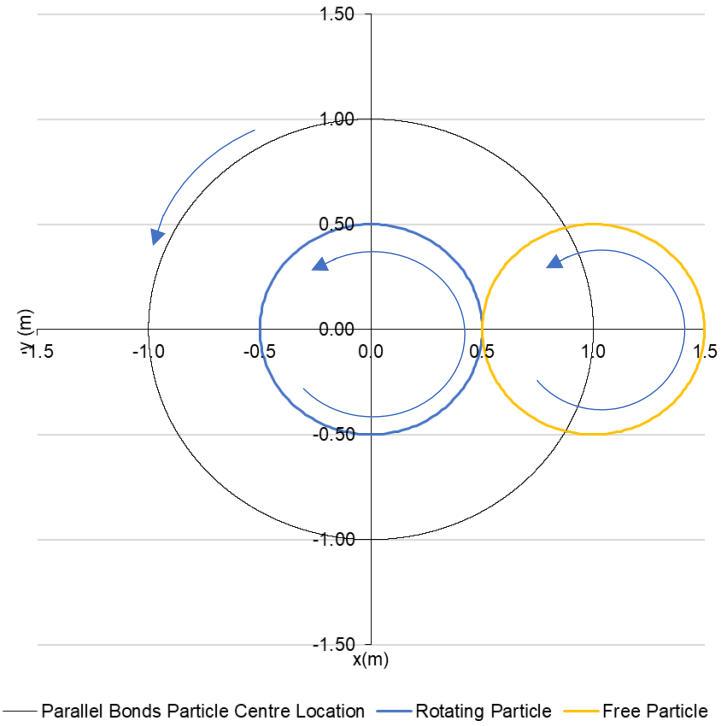

Figure 4-10: Movement of two particles as one is rotated that are bonded with a parallel bond. Rotation direction and movement direction are indicated with blue arrows.

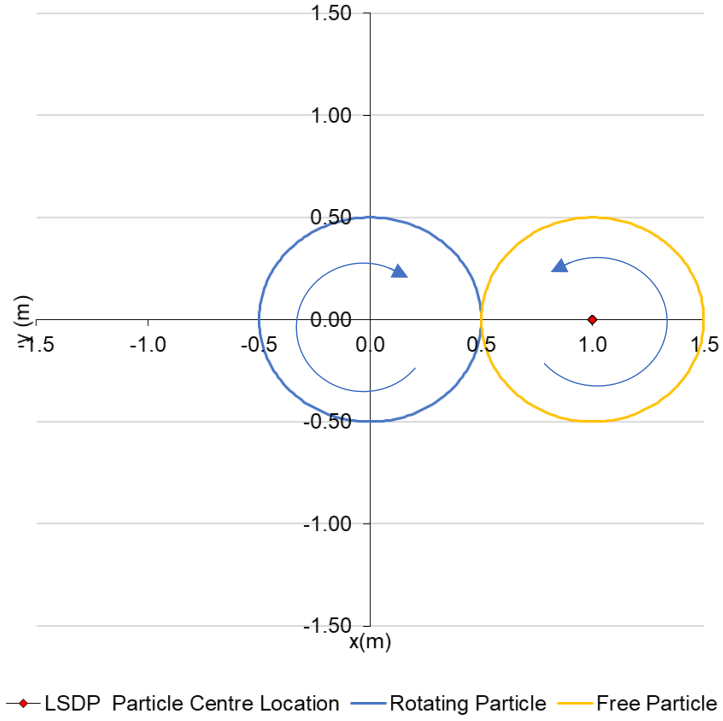

Figure 4-11: Movement of two particles as one is rotated that are bonded with an LSDP bond. Rotation direction and movement direction are indicated with blue arrows.

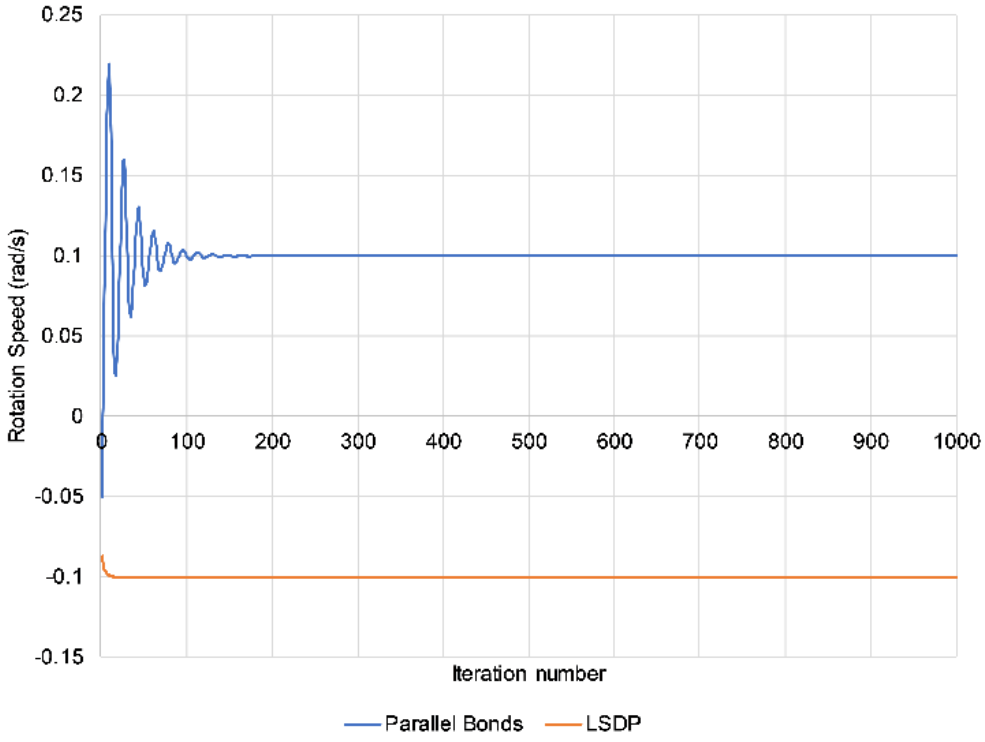

Figure 4-12: Rotational velocity for the free particle for LSDP and parallel bonded particle pairs. Notably, the rotational velocities are in opposite directions.

4.3.5 Integration

Each particle accumulates forces and torques due to all the contacts involved. The time integration scheme used for this study was the velocity Verlet scheme (Hofmann, 2011).

On time-step accelerations are determined based on the summed particle forces and torques according to the following equation.

$$\ddot{\bar{u}}^t = \frac{\bar{F}}{m} \tag{64}$$

$$\bar{u}^{t+1} = \bar{u}^t + dt\left(\dot{\bar{u}}^{t-\frac{1}{2}} + \ddot{\bar{u}}^t dt\right) \tag{65}$$

$$\bar{u}^{t+1} = \bar{u}^t + \dot{\bar{u}}^t dt + \frac{1}{2}(\ddot{\bar{u}}^t dt^2) \tag{66}$$

$$\dot{\bar{u}}^{t+1} = \dot{\bar{u}}^{t-1} + \frac{1}{2}(\ddot{\bar{u}}^t + \ddot{\bar{u}}^{t-1})dt \tag{67}$$

$$\dot{\bar{u}}^{t+\frac{1}{2}} = \dot{\bar{u}}^{t-\frac{1}{2}} + \ddot{\bar{u}}^t dt \tag{68}$$

Angular velocities are determined from the particle total torque and moment of inertia.

$$\dot{\bar{\omega}}^t = \frac{\bar{T}}{I} \tag{69}$$

$$\bar{\omega}^{t+\frac{1}{2}} = \bar{\omega}^{t-\frac{1}{2}} + \dot{\bar{\omega}}^t dt \tag{70}$$

In the leapfrog method, the particle locations and accelerations are known on timesteps while velocities are known between timesteps.

4.3.6 Critical time-step

DEM simulations are conditionally stable with a maximum stable time step (Δt_{cr}). For a bonded particle system, the maximum timestep can be defined from the highest natural frequency of the bonded particle system (ω_{max}). The natural frequency can be defined from the highest angular eigenfrequency (Azvedo, 2003):

$$\Delta t_{cr} = \frac{2}{\omega_{max}} \tag{71}$$

The maximum stable time step for two particles is a function of the mass of the particles and interparticle stiffness according to the following equation:

$$\Delta t_{cr} = \frac{2}{\sqrt{\frac{K}{m}}} \tag{72}$$

In the simulation, the stable timestep might be shorter than the theoretical value due to particles being bonded to multiple particles. Šmilauer and Chareyre, (2015) provide a methodology to determine the critical time step for multiple bonded particles. The maximum natural frequency for a system of bonded particles occurs when two particles move in opposite directions and can be given by:

$$\Delta t_{cr} = \sqrt{2}\sqrt{\frac{m_i}{K_i}} \tag{73}$$

The effective stiffness for each particle in the bond can be determined from all the bonds the particle is involved in according to:

$$K_{iw} = \sum_j \left(K_{Nj} - K_{Tj}\right) n_{jw}^2 + K_{Tj} \tag{74}$$

For: $w \in \{x, y, z\}$

Where:

K_{iw} is the effective stiffness in direction *w* for particle *i*.

K_{Nj} is the normal bond stiffness for bond *j* that the particle is involved in.

n_{jw} is the normal vector of bond *j* in direction *w*.

The timestep for this study was chosen based on the theoretical value. Stability was investigated by calculating the kinetic energy of a cylindrical sample of 100 bonded particles under a small displacement. For timesteps larger than the stable timestep, the kinetic energy of the system can be shown to increase without bounds.

4.3.7 Damping

Bonded discrete element models are susceptible to oscillations due to the non-dissipative nature of the bonded models, binary contact resolution between particles, and explicit time integration. Dissipative elements, such as dampers damp these oscillations. Dampers are modelled as additional forces that act on the particles. In DEM, damping can be introduced at the bonds acting proportionally to the relative particle velocities (such as viscous damping), or local damping proportional to individual particles' acceleration (such as non-viscous damping).

The final damping force acting on the particle is given by

$$\overline{F_{damped}} = \bar{F} + d\overline{F_{non-viscous}} + d\overline{F_{viscous}} \tag{75}$$

Where $\bar{F}$ is the sum of forces (body and contact) acting on the particle, $\overline{dF_{viscous}}$ and $d\overline{F_{non-viscous}}$represent the viscous and non-viscous damping contributions, respectively. In the absence of damping, any perturbation introduced into the system would persist indefinitely.

4.3.7.1 VISCOUS DAMPING

Viscous damping applies a force to the particles proportional to the relative velocity between the interacting particles.

$$\overline{dF_{viscous}} = -\alpha m\bar{V} \tag{76}$$

Critical damping provides the fastest way to achieve zero amplitude for a damped oscillator. The critical damping factor for the 1-D linear spring-mass-damped oscillator is given by:

$$D_t^{crit} = 2\sqrt{m_p k_n} \tag{77}$$

$$D_r^{crit} = 2r^2\sqrt{\frac{2m_p k_t}{5}} \tag{78}$$

D_t^{crit} and D_r^{crit} are the critical damping factors for translational and rotational DOF's respectively. k_n and k_t are the particles' normal and tangential stiffnesses respectively, and m_p is the particles' mass.

4.3.7.2 NON-VISCOUS DAMPING

Non-viscous damping applies a damping force proportional to the particle acceleration (which is analogous to the unbalanced force on the particle) and in a direction dependent on the velocity of the particle (Azvedo, 2003). The force aims to increase unbalanced forces ($\overline{F}$) that

decrease particle velocities ($\overline{V}$) and decrease unbalanced forces that increase particle velocities as follows:

$$\overline{dF_{non-viscous}} = -\beta sgn(\overline{V})\overline{\mathrm{F}} \tag{79}$$

The ratio of the unbalanced force that is applied as damping is β, $sgn(\overline{V})$ returns the sign of the velocity. An advantage of this method is that rigid body motion is not damped.

4.3.7.3 NUMERICAL INVESTIGATION

We investigate the effect of viscous and non-viscous local damping on i) a pair of particles released under tension, and ii) a chain of 50 connected particles undergoing rigid body movement.

For two particles under tension, the first particle is fixed in position while the second particle is moving away at a constant velocity before being released. The viscous damping coefficient is chosen based on the ratio of the critical damping, while the non-viscous damping coefficient is chosen based on the ratio of the damping force to the resultant force on the particle. A coefficient of 0.7 implies that 70% of the resultant force is applied as the damping force.

The velocity for the second particle is shown in Figure 4-13. Both damping approaches rapidly reduce the velocity of the second particle and return the two particles to rest. The envelope of the non-viscous damping is larger than the viscous damping, and the velocities take slightly longer to get damped.

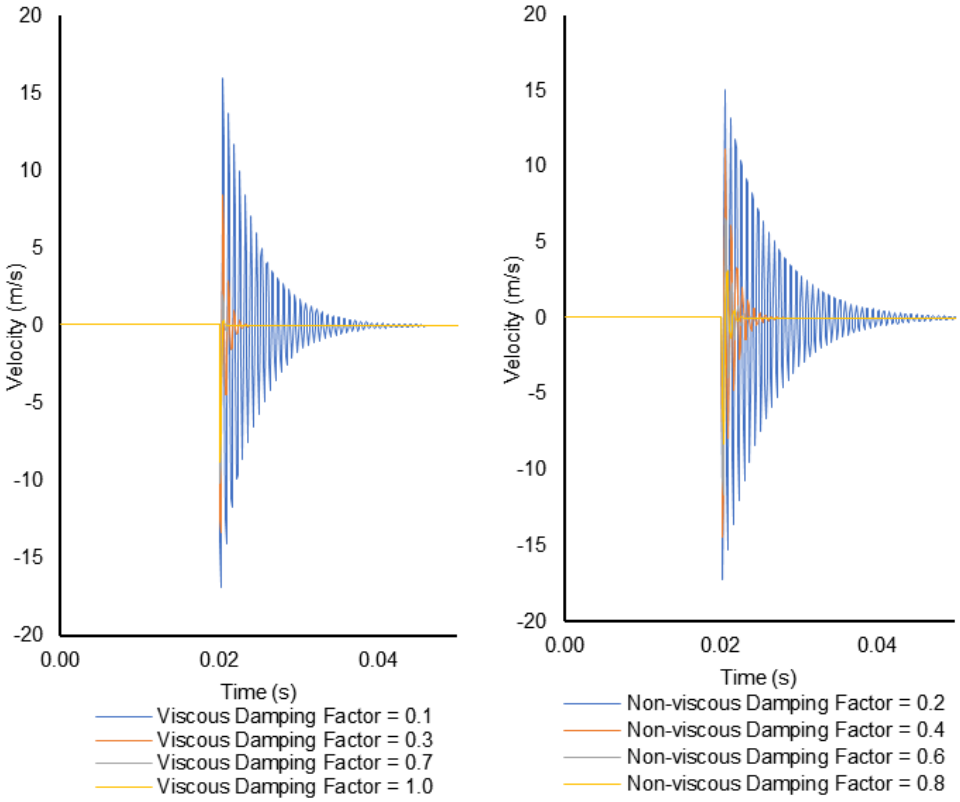

Figure 4-13: Velocity of the free particle for two particles with viscous damping. The free particle is displaced at 0.1m/s for 0.02s and then released. Different magnitudes of the damping coefficient (as a ratio of the critical damping magnitude) were investigated.

Consider the second problem, a chain of 50 connected undergoing rigid body motion. Constant velocity is applied to one end of an initially at rest particle chain. The total kinetic energy all 50 particles, and the unbalanced force in the end particle is reported for various damping coefficients. The total kinetic energy of the system and the expected solution is shown in Figure 4-14. The kinetic energy of the viscously damped system approaches the expected total kinetic energy for all magnitudes damping coefficients. In turn, the kinetic energy of the non-viscously damped system tends to overestimate the total kinetic energy.

The end force for viscous and non-viscous damping is also shown in Figure 4-14. All magnitudes of the damping coefficient for viscous damping approach the correct value of 0N. Non-viscous damping estimates large opposing end forces that are proportional to the unbalanced force each particle is experiencing.

Hence, non-viscous damping produces non-physical results for a group of bonded particles undergoing rigid body motion.

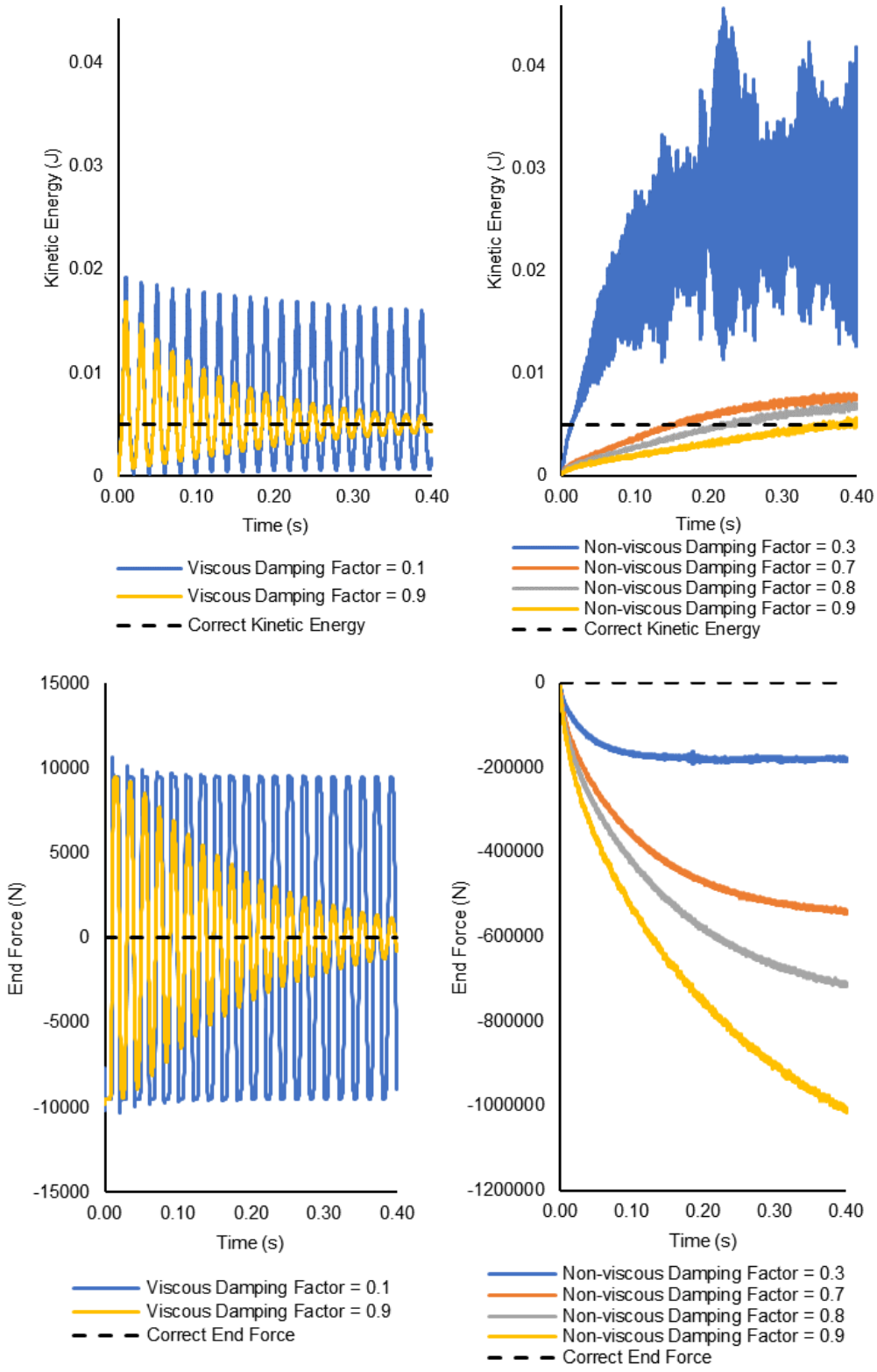

Figure 4-14: Top row: Kinetic energy of a chain of 50 particles where an end particle is displaced at a constant velocity with viscous (left) and non-viscous (right) damping employed. Bottom row: End force for a chain of 50 particles where an end particle is displaced at a constant velocity with viscous (left) and non-viscous (right) damping employed.

4.4 Methodology

4.4.1 DEM code

A DEM simulation package for bonded particles was developed and implemented using the Python 3.7 programming language. This framework takes full advantage of General Purpose Computing on Graphics Processing Units (GP-GPU) using the Numba module (Lam, Pitrou and Seibert, 2015). All calculations were computed using double precision compute, hence, requiring higher-end GPU cards for efficient compute (Nvidia, 2010).

As the present study does not include the behaviour once the fibre has fully debonded, it is not necessary to carry out broad phase collision detection repeatedly during the running of the simulation. Hence, additional acceleration was achieved by only determining contacting pairs at the beginning of the simulation and stored on the GPU. A regular hash table (Govender, Wilke and Kok (2015)) was used to speed-up the broad-phase collision detection on the GPU. GPU kernels were developed to apply boundary conditions, estimate contact forces, integrating the equations of motion and evolving bond softening and breakage.

4.4.2 Bond Softening

A cohesive damage law as described by Liu, Duddu and Waisman, 2012; Chen and Yan, 2015; Leclerc *et al.*, 2017 was used in this simulation. This model is shown in (Figure 4-15). The model is bilinear and describes the reduction in bond stiffness once the yield slip (δ_0) in the bond has been reached. The cohesive bonds in this study were only applied to the bonds in the interface between the fibre and the matrix. In keeping with the study of Chen and Yan (2015) the bonds were only permitted to transfer shear forces and the measure of the shear slip in the bond was used in the softening law. The softening law is non-reversible, and the degree of damage accrued in each bond, therefore, needs to be recorded at each timestep. The shear slip ($\delta(z)$) at the interface is calculated as the difference of the matrix and fibre axial displacement ($w^m(a,z)$ and $w^f(a,z)$ respectively) (Equation 80). K_0 is the shear stiffness of the bond and describes the shear stress generated by a given slip for the scenario when no damage in the bond has occurred (Equation 81). As the bond accrues damage the value of K_0 is reduced according to the damage evolution relationship (Equation 82).

$$\delta(z) = w^f(a,z) - w^m(a,z) \tag{80}$$

$$K_0 = \frac{\tau_0}{\delta_0} \tag{81}$$

$$D = \frac{\delta_1(\delta^{max} - \delta_0)}{\delta^{max}(\delta_1 - \delta_0)} \tag{82}$$

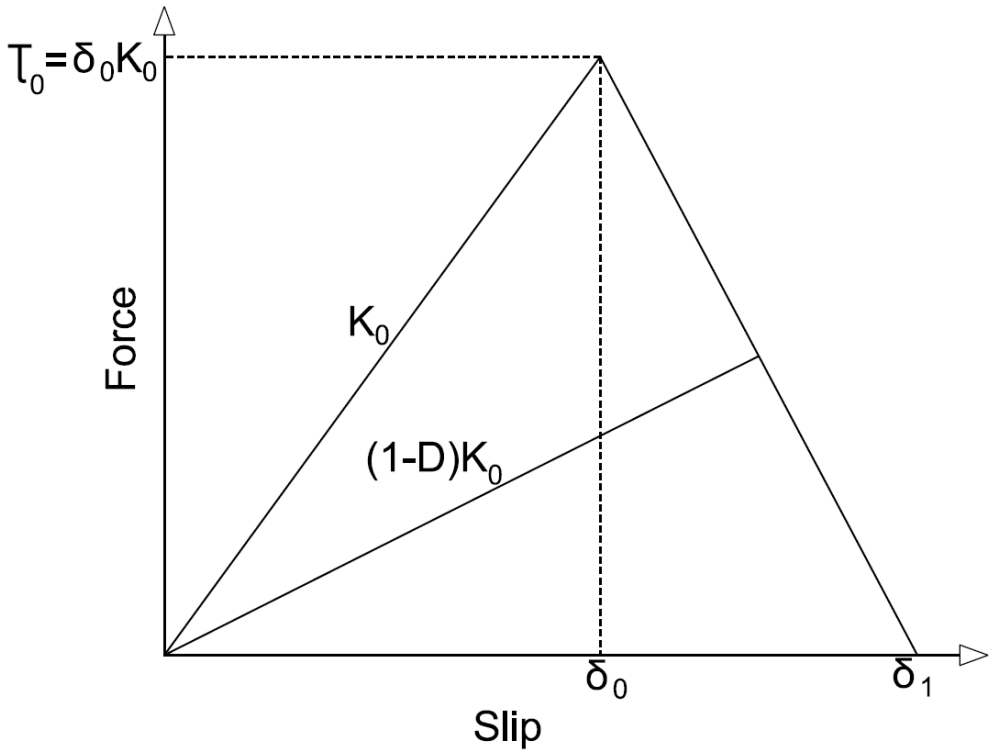

Figure 4-15: Bilinear bond softening model employed in this research.

4.4.3 Sample Packing

For this investigation, a single particle packing (of a uniform particle size) was employed. This approach has the following weaknesses:

- All regular packings display biased loading directions, should bond weakening be employed preferential crack directions are evident,
- Regular packings have directional contact preferences, and this may present as isotropic stiffness behaviour. This behaviour should be reduced as sample size increases, and
- size monodispersed packings have a lower maximum density than size polydispersed particle packings.

Regular size monodispersed packings have the following advantages:

- Packings can very quickly be generated of any dimensions,
- Calibration results can accurately be interpolated for different values of Youngs Modulus and Poissons ratio, and
- Storing a single particle size has computational advantages in the form of reduced storage requirements.

This investigation is focussed on DEM in general and comparing LSDP and parallel bond models used for modelling fibre pull-out. For comparison using regular packings and a single

particle size should be sufficient. Examples of the models used in the numerical investigation are shown in Figure 4-16 (the dimensions match those used by Chen and Yan (2015)).

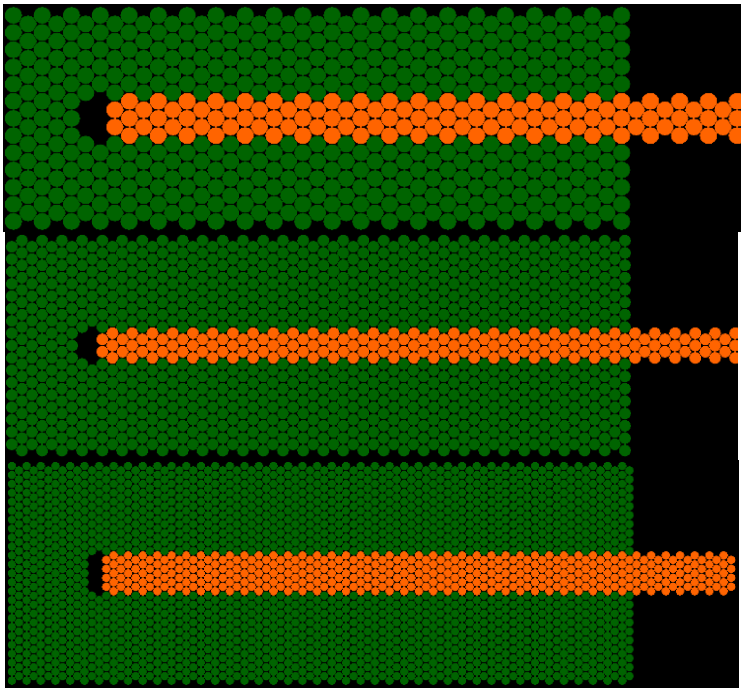

Figure 4-16: Cross-sections through the centre of the model for different particle diameters: 1m (top), 0.7m (middle), 0.5m (bottom). The fibre (orange) length bonded to the matrix (green) is 30m, and the fibre diameter is 2m. Note that the embedded end of the fibre is not bonded to the matrix.

4.4.4 Calibration

When starting a simulation, it is necessary to calibrate the inter-particle stiffness parameters to recover the macroscopic material behaviour such as the Youngs modulus and Poissons ratio (Syed, 2017). The macroscopic properties of an ensemble are a function of the inter-particle stiffnesses as well as the particle packing. Calibration of the bond stiffnesses was undertaken in two steps:

1. Determination of the bond stiffnesses such that the elastic behaviour of a cylinder of particles (chosen as 6m radius and 30m long to match the matrix geometry used by Chen and Yan (2015) matches those of the intended matrix and fibre.
2. The fibre bond stiffnesses are modified again such that the embedded fibre/matrix system displays the expected behaviour in the linear elastic phase. This step compensated for the reduced number of particles through the width of the fibre compared to the matrix.

For **Step 1** calibration, the microscopic bond parameters are recovered by estimating the macroscopic response for a cylinder of particles under tension (an example of the sample used is shown in Figure 4-17). A particle packing in equilibrium and under load was constructed by displacing one end of the sample while fixing the other end. Once the desired displacement

was achieved, both ends of the packing were fixed and iterated until equilibrium (defined here as the difference in end forces being smaller than 10 N).

Once the sample is at equilibrium, the Poissons ratio and sample Young's modulus can be recovered from the sample strain (perpendicular and parallel to the loading direction) and the end force, respectively. Bond parameters (specifically the bond normal and tangential stiffnesses) can then be chosen to return the required Poissons ratio and Young's modulus. Specifically, the Poissons ratio for a packing depends on the ratio between the normal (**Kn**) and tangential (**Kt**) stiffnesses, the packing itself and the particle size. For example, geometrical properties limit a two-dimensional close-packed lattice of equal spheres to a Poisson's ratio of 0.25. The Poisson ratio was calibrated by modifying the ratio **Kt**/**Kn** for each bond type (matrix-matrix, fibre-fibre, and matric-fibre). In DEM, the Young's modulus is primarily a function of the normal stiffnesses between particles. For general particle packings, the particle size distribution and packing density influence the Young's modulus of the particle ensemble.

The relationships between **Kn** and the Young's modulus and **Kt/Kn** and the Poissons ratio are approximately linear. The relationship **Kt/Kn** and are Youngs modulus are weakly coupled, making independent sequential parameter estimation possible through iteration. The calibration procedure is, therefore:

- Initial guess for **Kn** and **Kt/Kn**
- Using a line search algorithm estimate **Kt/Kn** such that $v_{measured} = v_{required}$
- Using a line search algorithm estimate **Kn** such that $E_{measured}=E_{required}$
- Repeat steps 2 and 3 until changes in **Kn** and **Kt/Kn** are less than 0.1%.

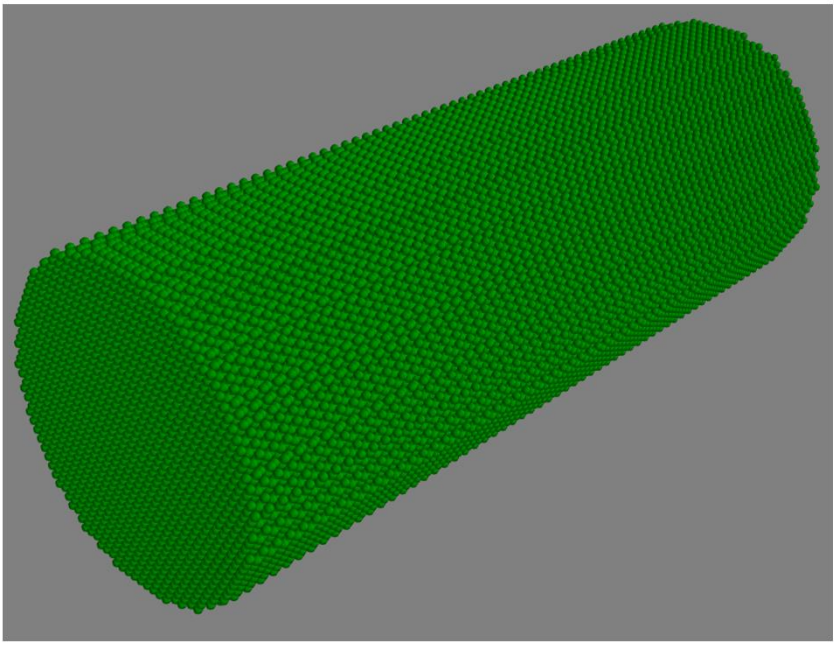

Figure 4-17: An example of a sample used for calibration.

The problem under consideration models a thin fibre in a relatively sizeable surrounding matrix. The stiffness of the modelled fibre is influenced by the diameter of the fibre relative

to the particle size. **Step 2** calibration was undertaken on the embedded fibre. The stress in the end of the fibre for a certain degree of interfacial slip is known from the analytical solution of the system. The fibre normal stiffness (**Kn**) was modified such that the system behaviour exactly matched the analytical solution for the linear elastic phase of the experiment. An embedded fibre length of 30m with a radius of 1m in a matrix with a radius of 6m was used. The end of the fibre was displaced by 0.001m (a distance which ensured that the system would remain in the linear elastic zone). Once the difference in the end forces (at the fibre and matrix end) reduced to less than 10N the interfacial slip at the point where the fibre enters the matrix was measured. The value of the fibre **Kn** was modified (and the model rerun) such that the difference between the measured and expected slip was less than 0.1%. The value of **Kt**/**Kn** determined in step 1 was not changed.

All calibration steps were carried out for the range of particle sizes employed in this investigation (0.5m, 1m and 2m diameters). The calibrated bond parameters are shown in Table 4-1 for a fibre Youngs Modulus of 50GPa and a matrix Youngs Modulus of 5GPa. It can be seen from the calibration results that particle diameters 0.5m and 1.0m have similar bond stiffnesses (given by AE/L for normal and tangential stiffnesses). For a particle diameter of 0.7m, the tangential bond stiffness is significantly less than for the other diameters. This may be due to 0.7m not being an even multiple of the fibre diameter of 2m, and this results in a significantly narrower fibre than for the other particle diameters.

It is sometimes necessary to calibrate other parameters for a simulation, such as particle mass or bond strength parameters. For this study the particle mass is not important as the simulation is undertaken under a quasi-equilibrium state where dynamic forces are not large. The bond strength parameters do not need to be calibrated as the bond shear strength behaviour has been provided in the analytical model. Should the bond strength parameters need to be calibrated the approach would be similar to that carried out here. Once the linear elastic parameters are calibrated the strength parameters would be calibrated against known behaviour.

Table 4-1: Calibrated bond and particle parameters.

Particle Diameter = 0.5m						
Bond Type	**Bond Normal Stiffness (N/m)**	**Bond Tangential Stiffness (N/m)**	**Bond Area (m^2)**	**Particle Mass (kg)**	**Akn/L**	**Akt/L**
Fibre-Fibre	7.40E+10	7.40E+08	0.196	392.7	1.5E+10	1.5E+08
Matrix-Matrix	7.40E+09	7.40E+07	0.196	392.7	1.5E+09	1.5E+07
Fibre-Matrix	-	1.00E+09	0.054	392.7	-	5.4E+07
Particle Diameter = 0.7m						
Bond Type	**Bond Normal Stiffness (N/m)**	**Bond Tangential Stiffness (N/m)**	**Bond Area (m^2)**	**Particle Mass (kg)**	**Akn/L**	**Akt/L**
Fibre-Fibre	6.70E+10	6.70E+07	0.385	1077.6	1.8E+10	1.8E+07
Matrix-Matrix	6.70E+09	6.70E+06	0.385	1077.6	1.8E+09	1.8E+06
Fibre-Matrix	-	1.00E+09	0.132	1077.6	-	9.4E+07
Particle Diameter = 1.0m						
Bond Type	**Bond Normal Stiffness (N/m)**	**Bond Tangential Stiffness (N/m)**	**Bond Area (m^2)**	**Particle Mass (kg)**	**Akn/L**	**Akt/L**
Fibre-Fibre	4.17E+10	1.25E+09	0.784	3141.6	1.6E+10	4.9E+08
Matrix-Matrix	4.17E+09	1.25E+08	0.784	3141.6	1.6E+09	4.9E+07
Fibre-Matrix	-	1.00E+09	0.197	3141.6		9.9E+07

4.4.5 Bond area

In the analytical model and physical experiments, the force between the fibre and the matrix is transferred across an interface. The interface area can be calculated by assuming a cylindrical fibre. In the considered DEM the interface is modelled as a collection of discrete bonds. The individual areas of the DEM bonds in the interface were scaled such that the sum of bond areas would equal the expected interfacial area of a cylindrical fibre with the same dimensions as the scenario being simulated.

4.4.6 Scale

The purpose of this study is to investigate the applicability of the developed DEM for modelling fibre pull out. The published results by Chen and Yan (2015) that were used to judge the accuracy of the results were based on a model with dimensions in the meter range. This is significantly larger than typical fibre reinforcing, which may have diameters and lengths in the millimetre range. Larger scale models allow larger computational timesteps which is better suited for this study.

The scale of the model simulated is significantly larger than realistic physical models. The ability of the developed model to simulate realistic scale problems is not investigated as part of this study. It is, however, a reasonable assumption that scaling laws could be applied to the model as is to recover accurate modelling of smaller-scale problems while using the same particle sizes.

4.4.7 Experimental Control

Consider the uniaxial pull-out test conducted in this study. The fibre can be pulled out by controlling

i) the applied force and measuring displacement (load control),

ii) the applied displacement and measuring the force (displacement control), or

iii) a combination of the applied force and distance, substantially limiting the allowed step on the force-displacement graph per time step (such as arc-length control (Leon *et al.*, 2011)).

Figure 4-17 depicts three force-displacement graphs. Curve A signifies softening behaviour, Curve B hardening behaviour and Curve C softening behaviour with some displacement instability. Displacement control can be used to estimate Curves A and B. Load control can only be used to estimate Curve B, while arc-length control can be used to estimate all three curves.

The investigations undertaken in this study were conducted using displacement control. It was, however, found that the load-displacement curve followed a path that would require an arc-length limiting control method (such as for curve C). Arc-length limiting control methods cannot, however, be applied in DEM.

For explicit time integration approaches, such as DEM, the estimation of the load-displacement curve C requires an alternative approach. In this study, the change in damage is limited using prescribed displacement control. The increment of damage that is allowed to occur along the interface is limited. Once the allowed increment of damage has occurred, the system is completely unloaded with no damage allowed before displacement is applied again. The system is unloaded to ensure that no elastic energy is stored in the system, which would lead to sudden failure of the interface once the system is reloaded and damage allowed to accrue.

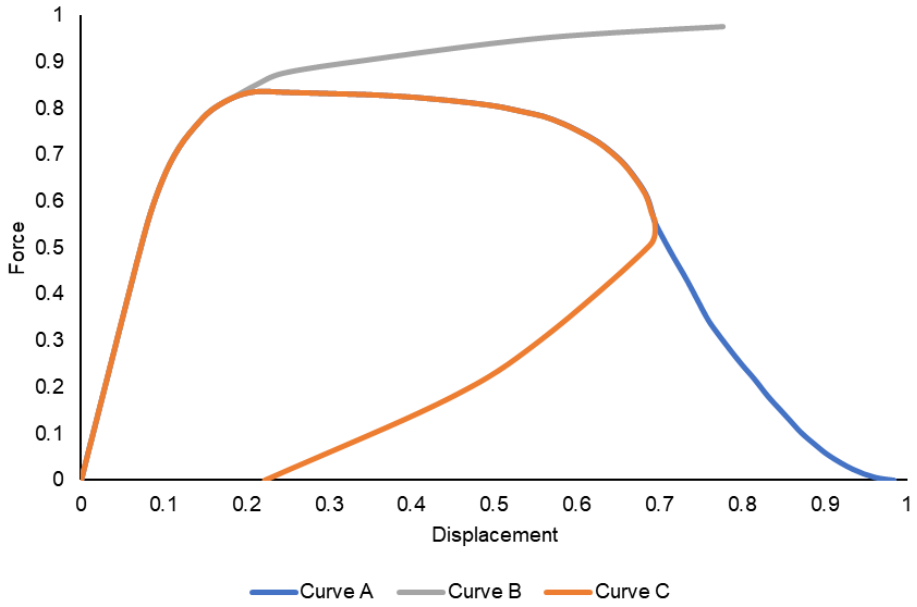

Figure 4-18: Force/displacement curves for different types of experimental control.

4.4.8 Final model parameters

The calibrated bond properties, as given in Table 4-1 were used in the simulation for parallel and LSDP type bonds. The damping method used was the viscous damping with a damping coefficient of 0.7. The time step varied between 1e-5 and 1e-6 seconds depending on the particle size and bond stiffness. All other parameters used in the investigation are given in the results section.

4.5 Results

4.5.1 Modelled phases

This study focusses on the stress growth phase and debonding phase that occurs during fibre pull-out using the LSDP and parallel bonds models. The bond models are compared in Section 5.3.1. It was noted that there is no significant difference between the two contact models. Unless otherwise stated the LSDP model was used to carry out the investigations.

In the stress growth phase, all bonds between the fibre are intact, and the relationships between interface slip and end force are linear. The studies carried out in this phase focusses on the interfacial shear forces for different geometric and material parameters. The results for the stress growth phase are presented in Section 4.5.2.

Investigations for the debonding phase focus on the interfacial shear force and interface slip - end stress relationship for the entire debonding process. The results for debonding phase are presented in Section 4.5.3.

The accuracy of the numerical results compared to the analytical solution is determined by calculating the Normalised Root Mean Square Error (NRMSE), normalised by the range of the data.

4.5.2 Stress growth phase

A fibre pull-out was simulated using LSDP and parallel bond models and the results analysed. For all simulations, the fibre was extracted to the point at which the first interfacial bond begins yielding. The same yield shear slip was used for all simulations: 1e-3m. The results presented show the bond shear stresses at the bond centroid locations for the interface bonds. The axial stress in the fibre is determined by accumulating the bond shear stresses along the fibre length as the interfacial bonds can only transmit shear stresses. The interfacial slip is measured at the right-most extent of the interface between the fibre and matrix.

The geometry of the modelled fibre is shown in Figure 4-3. In all scenarios, the fibre is extracted towards the right. Four studies are conducted in which the influence of particle size, stiffness ratios, fibre length and interfacial stiffness are investigated. The parameters used in the simulations are shown in Table 4-2.

Table 4-2: Typical problem parameters

Parameter	Magnitude	Unit
Particle Diameter	0.5 (varies)	Meter
Matrix Youngs Modulus	5e9 (varies)	Pascal
Fibre Youngs Modulus	50e9	Pascal
Matrix Poissons Ratio	0.3	Dimensionless
Fibre Poissons Ratio	0.2	Dimensionless
Fibre Length	30 (varies)	Meter
Fibre Radius	1 (varies)	Meter

Matrix Radius	6	Meter
Interfacial bond K value	1e9 (varies)	Pascal
Interfacial bond yield slip	1e-3	Meter
Extraction Speed	0.01 (varies)	Meter/Second
Timestep	1e-5	Second

4.5.2.1 Effect of particle size – stress growth phase

The fibre pull-out is simulated at an extraction speed 0.01m/s using three particle sizes, with radii of 1m, 0.7m and 0.5m until the first bond in the interface yields.

Since each particle size was individually calibrated, it is expected that the different particle sizes result in the same stress distribution along the fibre. It is expected that larger particle sizes would result in higher interfacial bond stress variance as bonds for larger particles represent more substantial areas of the interface.

The results of the interfacial bond shear stress for each bond between the fibre and the matrix are shown in Figure 4-18. The interfacial bond shear stress magnitudes and form along the length correspond well with the analytical solution for different particle sizes. Particles with radius 0.5m and 1m show the smallest error compared to the analytical solution. The 0.7m radius particle has the largest error; this may be due to 0.7 not being a multiple of the fibre diameter of 2m. This could result in a difference in the modelled fibre diameter and the required diameter.

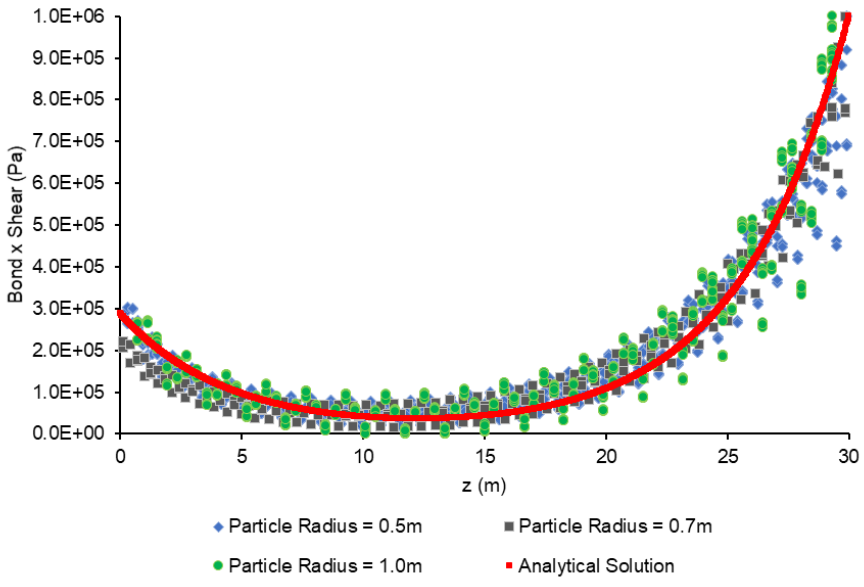

Figure 4-19: Comparison of interfacial bond shear stresses for different particle sizes. Radius = 0.5m NRMSE = 0.055, Radius = 0.7m NRMSE = 0.0711, Radius = 1m NRMSE = 0.0512.

4.5.2.2 EFFECT OF VARYING STIFFNESS RATIOS

The stiffness ratio between the matrix and fibre has a significant effect on the stress distribution along the interface. As the difference between the fibre stiffness and matrix stiffnesses increases, the slip between them and the change in slip with distance along the interface should increase. Particles that form part of the interface form bonds in the direction of the embedded and free end of the fibre. As the gradient of interfacial slip increases, a larger scatter of bond stresses results, as bonds associated with a single particle span a broader range of interfacial slip. The results are presented in Figure 4-19 and Figure 4-20 and are compared to the analytical solution.

Figure 4-19 (**a**) presents the interfacial bond shear stress when the stiffness of the matrix is larger than the fibre (with a matrix to fibre stiffness ratio of 10:1). The DEM solution underpredicts the analytical solution at the maximum shear stresses, while the DEM solution overpredicts the lower shear stresses in the middle of the fibre. A consequence of the discrete nature of the particles. The overall agreement between the DEM and analytical solutions is within 0.0048 NRMSE.

Figure 4-19 (**b**) presents the interfacial bond shear stress when the stiffness of the matrix and the fibre are the same. Good agreement with the analytical solution is achieved. The DEM solution, in this case, underpredicts the interfacial shear along the length of the fibre. The maximum interface stress occurs at the point where the fibre exits the matrix. The NRMSE between the DEM and the analytical solution is 0.047. While the error is more substantial than for the case where the fibre is less stiff than the matrix this may be due to higher shear stresses over the length of the interface (compare Figure 4-19 where the interfacial shear stress is low for most of the interface.).

Figure 4-19 (**d**) presents the interfacial bond shear stress when the stiffness of the matrix is less than the fibre (with a matrix to fibre stiffness ratio of 1:100). The DEM solution underpredicts the analytical solution at the maximum shear stresses, while the DEM solution overpredicts the lower shear stresses in the middle of the fibre. This is due to the discrete nature of the particles. The maximum interface stress occurs at the embedded end of the matrix. The NRMSE between the DEM and the analytical solution is 0.111. While it can be seen that the DEM predicts the correct form of the interfacial shear, there is a more extensive spread of shear stresses.

The model parameters that will be used to investigate the debonding phase result in a fibre to matrix stiffness ratio of 10:1. From Figure 4-19 (**c**) it can be seen that the NRMSE for this geometry and parameters is 0.049, this is similar to the NRMSE for equal matrix and fibre stiffnesses. This gives confidence that the simulation will be accurate over the intended fibre to matrix stiffness ratio of 10:1.

Figure 4-20 compares the estimated fibre axial stress for the fibre-matrix stiffness ratios against the analytical and expected result. The axial stress is in general underpredicted along the fibre length when the fibre stiffness is greater or equal to the matrix stiffness with the inverse true if the matrix is stiffer than the fibre. The best agreement is achieved when the fibre stiffness is less or similar to the matrix stiffness, while the worst agreement is obtained when the fibre is significantly stiffer than the matrix.

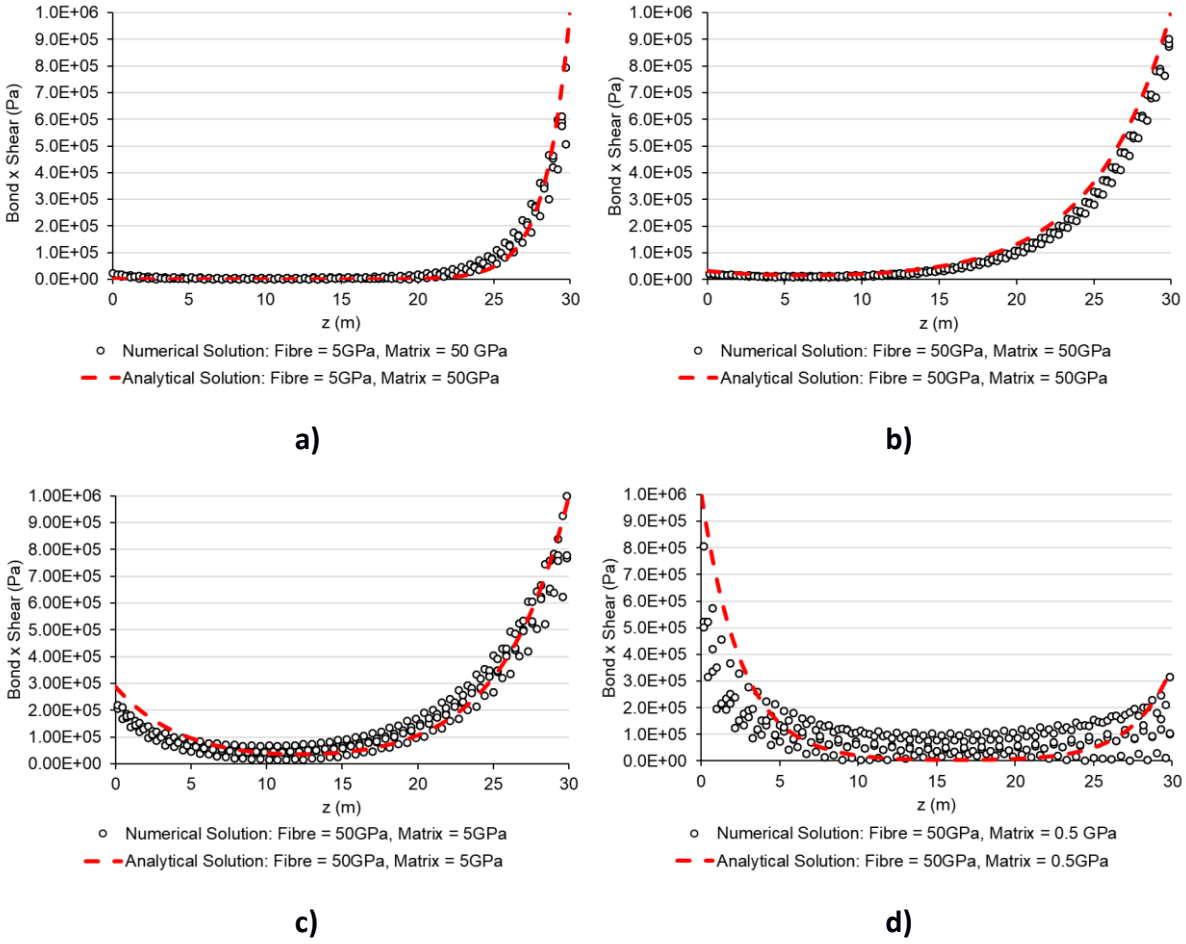

Figure 4-20:Bond shear stresses (in the x direction) plotted against the bond centroid location along the interface length for four different fibre/matrix stiffness ratios. The analytically predicted shear stresses are also plotted. NRMSE for the figures are: ***a)*** *= 0.0048,* ***b)*** *= 0.047,* ***c)*** *= 0.049 and* ***d)*** *= 0.111*

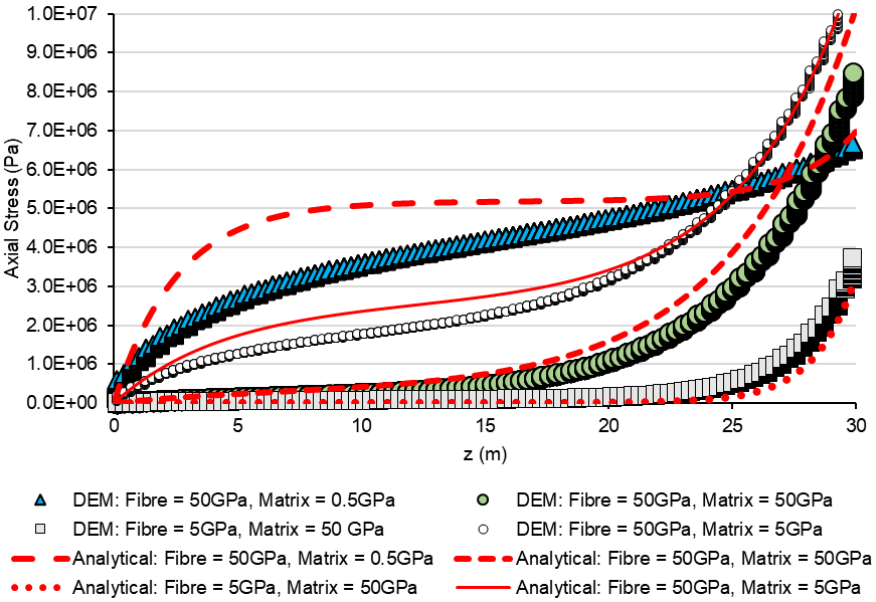

Figure 4-21: Fibre axial stress plotted against location along the fibre length for different values of fibre and matrix Youngs Moduli. The analytically predicted axial stresses are also plotted. NRMSE Matrix:Fibre 10:1 - 0.0733, NRMSE Matrix:Fibre 1:1 - 0.0733, NRMSE Matrix:Fibre=1:10 - 0.036, NRMSE Matrix:Fibre=1:100 - 0.159.

4.5.2.3 EFFECT OF FIBRE LENGTH

To investigate the influence of embedded fibre length on the stress distribution, we considered fibre lengths between 10m and 40m long were modelled using the parameters tabulated in Table 4-2. The shape of the stress distribution is not expected to change as the length of the fibre increases. For the parameters used here, the maximum shear stress will occur where the fibre exits the matrix.

The results are presented in Figure 4-21 and Figure 4-22. DEM slightly underpredicts the bond shear stress and axial fibre stress at the embedded end of the fibre. However, the overall agreement with the analytical results is suitable for all investigated fibre lengths (interfacial shear stress and axial fibre stress NRMSE between DEM and analytical solution is in the range of 0.02 – 0.05). As the length of the fibre increases, the minimum shear stress along the fibre length reduces. Low interfacial shear stress indicates low relative slip between the fibre and the matrix. This occurs for long fibres where the length is large enough to transfer the load to the matrix.

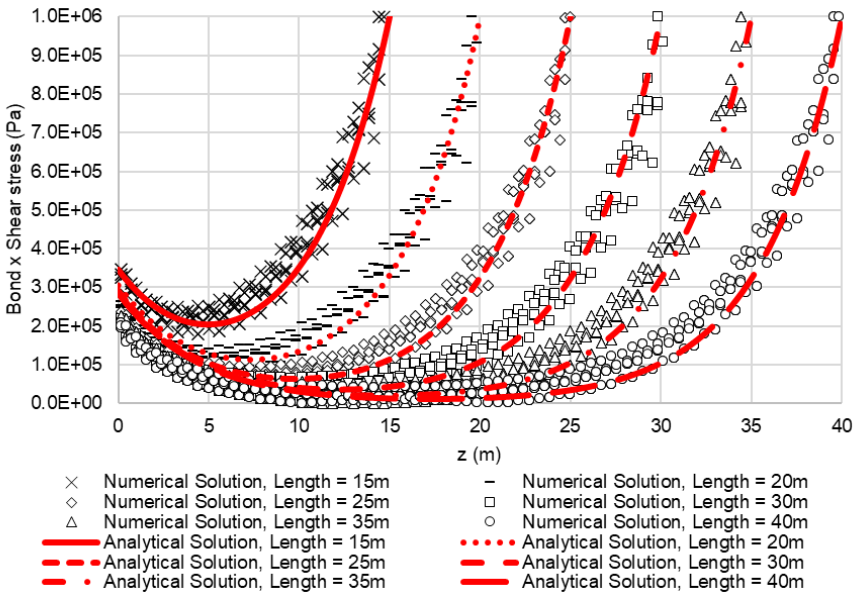

Figure 4-22: Bond shear stresses (in the z-direction) plotted against the bond centroid location along the interface length for different fibre lengths. The analytically predicted shear stresses are also plotted. NRMSE: 15m - 0.05, 20m - 0.05, 25m - 0.045, 30m - 0.

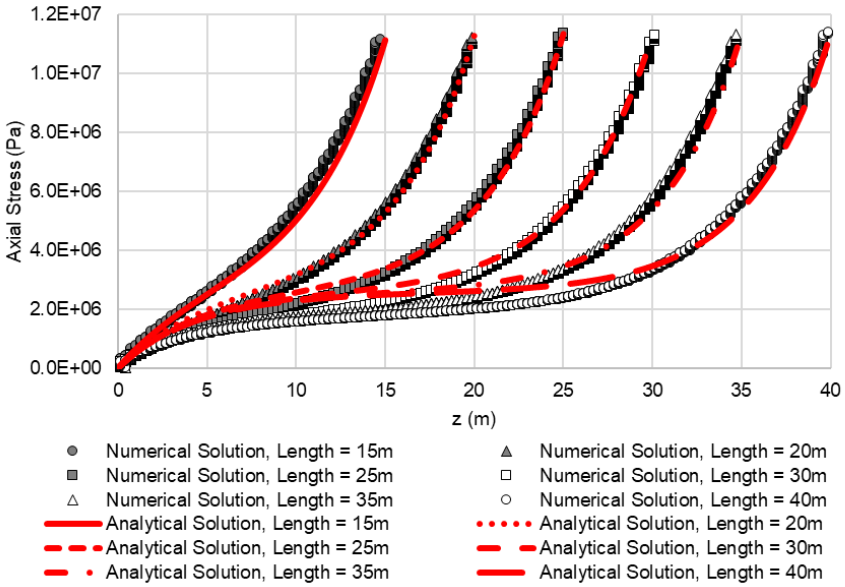

Figure 4-23: Fibre axial stress plotted against location along the fibre length for different values of fibre length. The analytically predicted axial stresses are also plotted. NRMSE 15m - 0.032, 20m - 0.020, 25m - 0.026, 30m - 0.036, 35m - 0.038, 40m - 0.046.

4.5.2.4 EFFECT OF INTERFACIAL STIFFNESS

The shear stiffness of the interface between the fibre and the matrix has a significant effect on the shape of the interfacial shear stress. The interfacial stiffness relates the magnitude of the slip along the interface to the magnitude of the shear stress. The ability of the DEM to model the effect of varying the interfacial stiffness is investigated here.

Two interfacial stiffnesses were investigated, namely 1e7 and 1e10 Pa. The yield slip for the bonds was maintained at 1e-3m. The implication is, therefore, that the bonds will yield at different stresses. The matrix and fibre stiffnesses are 5 and 50 GPa, respectively. The results are presented in Figure 4-23 and Figure 4-24. The maximum bond shear stress for an interfacial stiffness of 1e10 Pa (Figure 4-24) is three orders of magnitude greater than for an interfacial stiffness of 1e7 Pa (Figure 4-23) as would be expected (since both investigations show results at the same maximum interface slip).

From the results, for the lower interfacial stiffness the range of interfacial bond shear stresses is less (less than one order of magnitude) than for the higher interfacial stiffness (approximately seven orders of magnitude). The deviation between the analytical solution for the bond shear stress is also larger for the higher interfacial stiffness, which also demonstrates larger scatter around the analytical solution. A higher interfacial stiffness implies that smaller interparticle displacements across the interface will result in higher forces being transmitted between the fibre and the matrix. To produce the same maximum interfacial slip, as was done here, the fibre will have to be extracted as a higher axial force for the larger interfacial stiffness. The strains within the fibre and matrix will therefore be larger for the higher interfacial stiffness as a larger overall force is being carried by the fibre/matrix system. These larger strains (within the matrix and fibre) are possible what leads to the larger spread of interfacial shear stresses seen in Figure 4-24. These large strains leading to spurious results may be alleviated by using smaller particles, or a range of particle sizes and a random packing (this would eliminate directions of preferred force transfer inherent to regular packings). Further study is necessary to confirm this. Similarly, the higher interfacial stiffness results in a more considerable difference between the analytical and numerically predicted axial fibre stress.

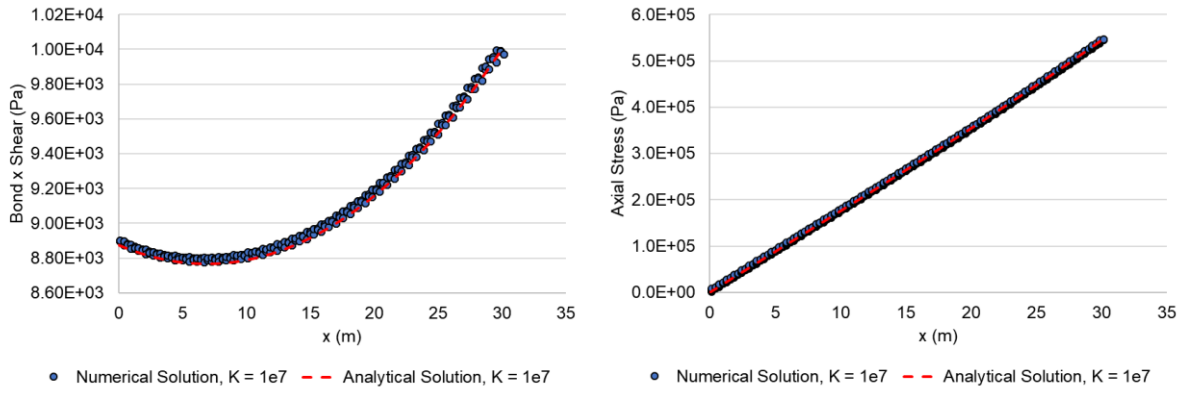

Figure 4-24: Interfacial bond shear stress (left) and fibre axial stress (right) for an interfacial shear stiffness of 1e7 Pa. NRMSE (shear stress) - 0.003, NRMSE (axial stress) - 0.004

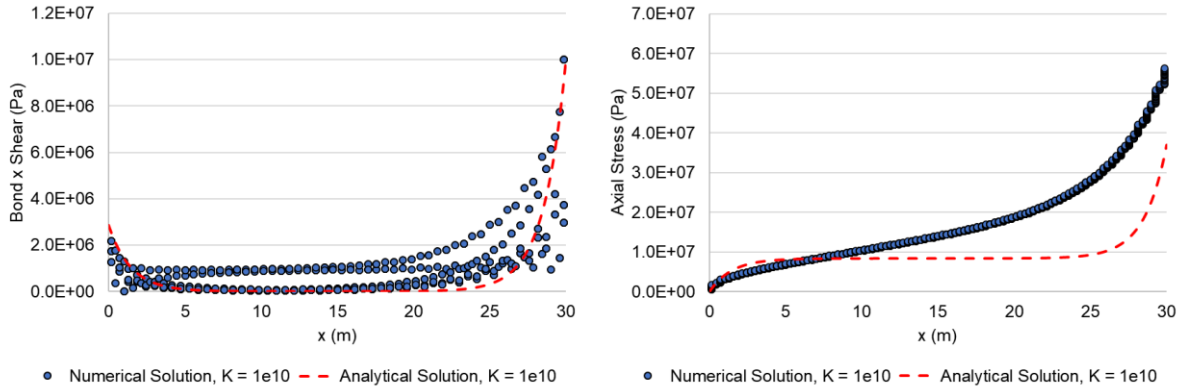

Figure 4-25: Interfacial bond shear stress (left) and fibre axial stress (right) for an interfacial shear stiffness of 1e10 Pa. NRMSE (shear stress) - 0.108, NRMSE (axial stress) - 0.197

4.5.2.5 DISCUSSION

From the simulated results, the linear elastic stage of fibre debonding can be simulated using a DEM model as used in this study. The simulated results closely match the analytical solution. Variation in the simulated results from the analytical solution is evident for high interfacial stiffnesses and significant differences between the matrix and fibre stiffnesses (specifically when the fibre is stiffer than the matrix) which also results in the largest strains in the matrix and fibre. It would appear then that large strains in the particle model result in a larger deviation from the analytical solution.

The model shows reduced accuracy for scenarios where the fibres are significantly stiffer than the matrix. The combination of stiff fibres and a compliant matrix is a common occurrence for typical fibre reinforced materials. The high strains that occur in the matrix when the fibre is significantly stiffer may amplify the non-physical preferred force transfer directions of the regular packing. Using smaller particles might not mitigate this preferred force transfer direction property. Using a range of particle sizes and a dense random packing should mitigate this issue and may allow stiff fibres in a compliant matrix to be modelled more effectively.

4.5.3 De-bonding phase

During the debonding stage, the shear slip at the interface increases until bonds begin to soften and ultimately fail. Interaction between the particles is limited to only bonded behaviour, and no new particle contact pairs form during the pull-out process. The debonding stage experiences more dynamic behaviour than the linear elastic phase due to the breakage of bonds. Accurately resolving the forces in the system needs to take into account possible dynamic behaviour. To isolate possible inaccuracies in the measurement of forces in the fibre two approaches were followed: i) summing the interfacial bond shear forces along the fibre in the direction of the fibre and ii) summing the unbalanced forces in the end particles of the fibre.

For all debonding investigations, the fibre and matrix parameters tabulated in Table 4-3, are used. A particle diameter of 0.5m is used unless otherwise stated. The LSDP contact model

is employed unless otherwise stated. The DEM results are compared to analytical and FEM results presented by Chen and Yan (2015).

As part of the investigation of the debonding behaviour, the effect of the bond model, particle size, extraction speed and fibre length are modelled. The fibre end stress and interfacial slip relationship are also investigated, the simulations are undertaken using either displacement control or a load/unload displacement control method.

Table 4-3: Parameters used for the debonding simulation.

Parameter	Magnitude	Unit
Particle Diameter	0.5, 0.7, 1.0	Meter
Matrix Youngs Modulus	5e9	Pascal
Fibre Youngs Modulus	50e9	Pascal
Matrix Poissons Ratio	0.3	Dimensionless
Fibre Poissons Ratio	0.2	Dimensionless
Fibre Length	30	Meter
Fibre Radius	1	Meter
Matrix Radius	6	Meter
Interfacial bond K value	1e9	Pascal
Interfacial bond yield slip	1e-3	Meter
Interfacial bond failure slip	3e-3	Meter
Extraction Speed	0.01	Meter/Second
Timestep	1e-5	Second

4.5.3.1 EFFECT OF FIBRE PULL-OUT SPEED

The debonding stage experiences more dynamic behaviour than the linear elastic phase due to the breakage of the bonds. The effect of the extraction speed on the pull-out curve was investigated by extracting the fibre at different velocities. The pull-out curve and the shear stress in the interfacial bonds for different extraction speeds are shown in Figure 4-25 and Figure 4-26 (the simulations are undertaken using displacement control only). The fibre end stress is calculated here by measuring the force acting on the end particles of the fibre (where the boundary condition is applied). From the results, the fibre extraction speed has a significant effect on the extraction behaviour. As the extraction speed increases the interfacial slip at failure increases (Figure 4-25). From Figure 4-26, it is apparent that for increasing extraction velocities, the damage is concentrated closer to the free end of the fibre. This can be attributed to dynamic effects within the particle ensemble. The effect of decreasing velocities reduces below 0.01m/s. For this system, therefore, the maximum extraction velocity before dynamic behaviour begins influencing the results is 0.01m/s.

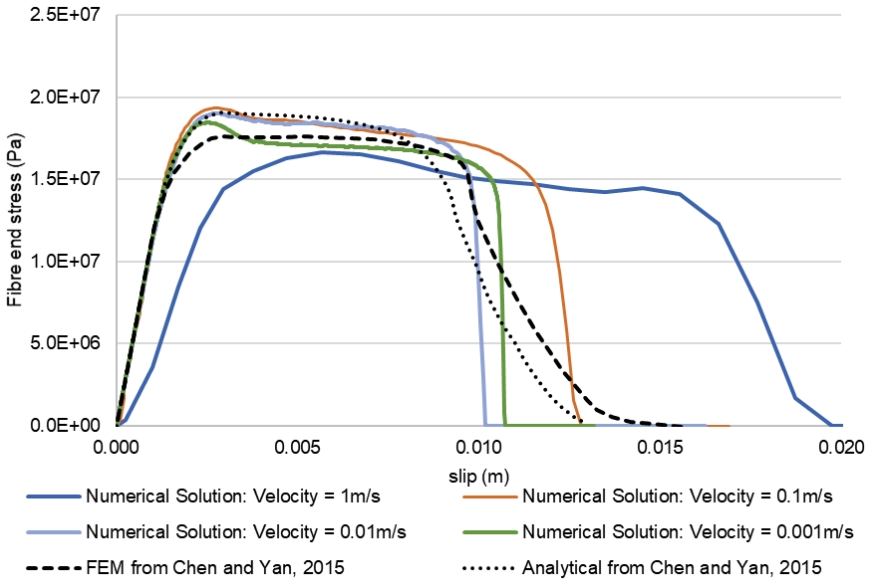

Figure 4-26: Pull out curves for different extraction speeds (0.001m/s to 1m/s)

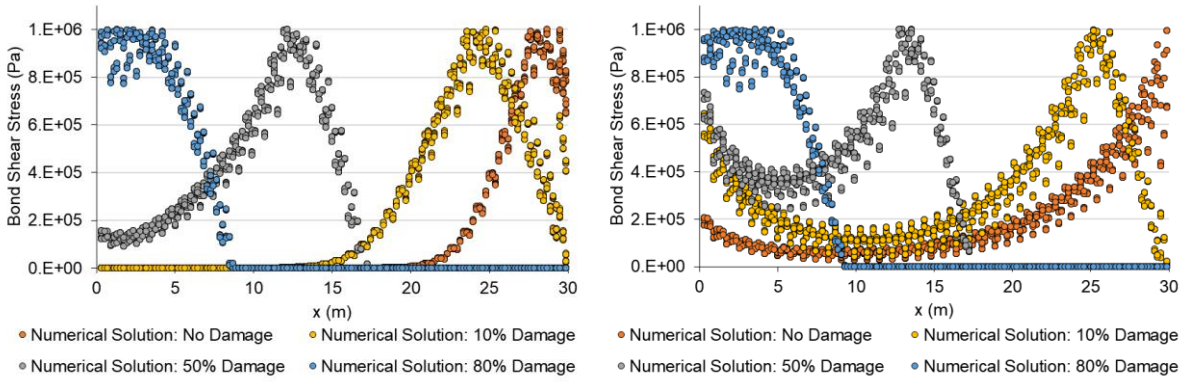

Figure 4-27: Bond shear stresses (in the x direction) plotted against the bond centroid location along the interface length for an extraction velocity of 1m/s (left) and 0.1m/s (right). The bond shears are plotted for four points during the pull-out curve.

4.5.3.2 INTERFACIAL SHEAR DURING DEBONDING

The interfacial shear stresses for several levels of total interfacial damage during the debonding process are shown in Figure 4-27 (the simulations are undertaken using displacement control only). Good agreement to the analytical solution is achieved with an NRMSE of between 0.058 and 0.196 (the NRMSE increases as the extent of damage increases). The most substantial difference between the analytical and the numerical solution is close to the embedded end after debonding has occurred. This suggests that the assumption of the analytical model used by Chen and Yan (2015), that the full bonded length that includes the damaged area is used to describe the distribution of shear stress at the interface is incomplete.

Describing the interfacial shear stress using a fibre length between the length at which yield occurs and the length at which failure occurs would reduce the discrepancy but extending the Chen and Yan (2015) model does not form part of this study.

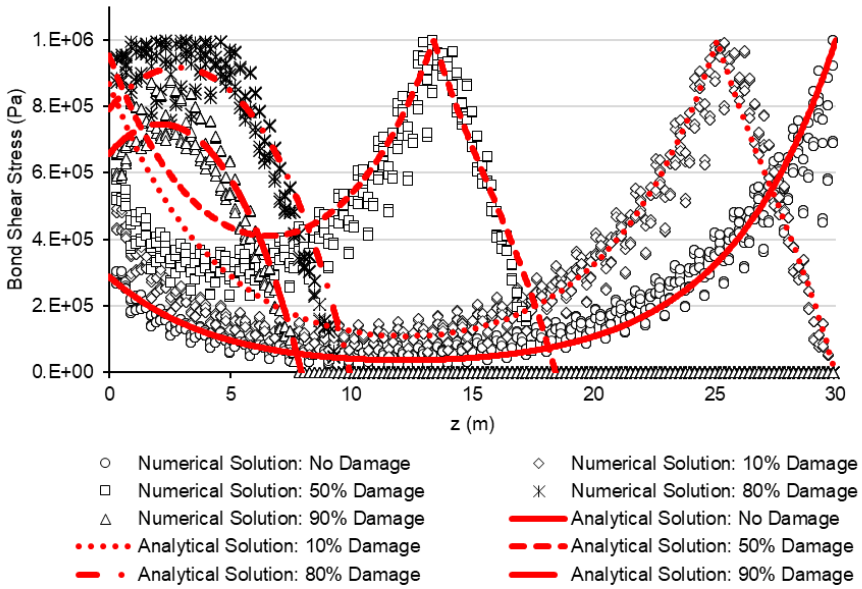

Figure 4-28: Bond shear stresses (in the z-direction) plotted against the bond centroid location along the interface length for a particle diameter of 0.5m. The bond shears are plotted for five points during the pull-out curve. The analytically predicted shear forces are also plotted. NRMSE 0% damage - 0.058, 10% damage - 0.124, 50% damage - 0.122, 80% damage - 0.153, 90% damage - 0.196.

4.5.3.3 COMPARISON BETWEEN PARALLEL BONDS AND LSDP

An embedded fibre with parameters as given in Table 4-2 was modelled using the parallel bonds and LSDP models (the simulations are undertaken using displacement control only). The difference between the LSDP and parallel bonds models are the most significant when relative rotation between particles is large. For primarily normal interaction between particles in the absence of relative rotation, the models are indistinguishable.

For the uniaxial fibre pull-out problem, the relative particle rotation is expected to be limited, it is expected to see limited differences between the two models. Figure 4-28 and Figure 4-29 present the results of the interface at the point of yield and 80% interfacial damage, respectively. As expected, there is no significant difference in the bond shear stresses predicted using the parallel bonds and LSDP models.

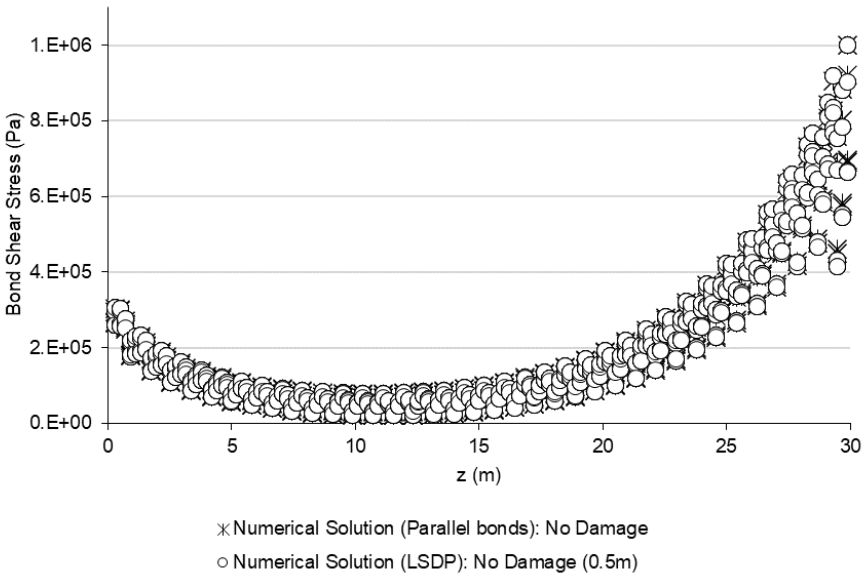

Figure 4-29:Comparison in the pull-out behaviour for the DEM model using linear dash pot and parallel bond contact models no damage. NRMSE = 0.004

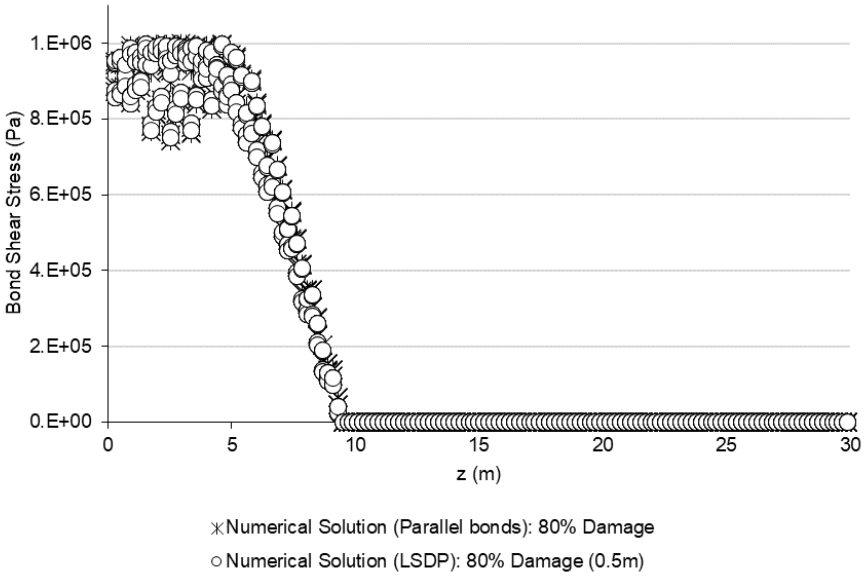

Figure 4-30: Comparison in the pull-out behaviour for the DEM model using linear dash pot and parallel bond contact models 80% damage. NRMSE = 0.009

Notably, there is no significant difference between parallel bonds and LSDP bonds for both the fully bonded scenario (Figure 4-28) as well as the scenario where significant damage has occurred (Figure 4-29). Recall the demonstration in Section 4.3.4 that showed significant differences when a chain of particles was considered. Difference between the parallel bonded and LSDP models are expected when moments need to be transferred, and the number of

particles through the fibre thickness is reduced. As the number of particles through the fibre thickness increases the LSDP resolves the ability to transfer moments.

4.5.3.4 EFFECT OF PARTICLE SIZE – DE-BONDING STAGE

Three particle sizes are used to model the fibre pull-out using the parameters tabulated in Table 4-3 (the simulations are undertaken using displacement control only). The bond model employed here was the LSDP; it was previously shown that there is no appreciable difference between the bond models for this problem. The pull-out curve plots the sum of interfacial shear stresses (Figure 4-30) and axial stress measured at the end of the fibre (Figure 4-31) against the ratio between the maximum measured interfacial slip (at the point where the fibre enters the matrix) and the slip at which bonds failure (1e-3m). The curves are only plotted while at least some bonds in the interface are not entirely broken.

The load-displacement relationship follows an initial linear increase in stress with slip. Once the end slip reaches the point at which bond failure begins the load-displacement relationship departs from linear. When the maximum slip reaches the point of bond failure, the interface begins failing from the end nearest the free end of the fibre. The interfacial failure occurs at a constant fibre axial end stress (the horizontal portion of the load-displacement relationship). Once a certain amount of the interface has failed, the remaining portion fails suddenly causing a sudden drop in the fibre axial end stress.

The pull-out curves compare well with both the analytical and the FEM results. The significant deviation is however noted on the descending section of the pull-out curve. The differing behaviour is due to the complex dynamic behaviour of the final stages of the fibre pull-out which is investigated later. The FEM model that was used by the authors operated under displacement control to extract the fibre. It is likely that the stress in the fibre was measured at the end of the fibre and that once the interface had failed entirely the dynamic behaviour of the now wholly free fibre was erroneously interpreted as the axial stress for a fibre still bonded to the matrix. The reason for the deviation of the analytical solution from the DEM could not be determined.

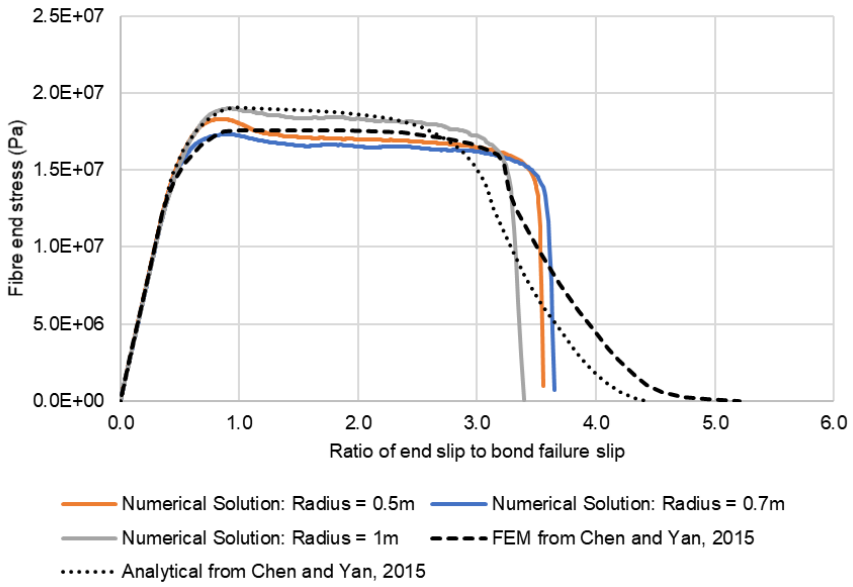

Figure 4-31: Fibre end stress plotted against the ratio of the slip between the fibre and the matrix and the slip at which bond failure occurs. The force is calculated by summing shear in the bonds at the interface. Analytical and Finite Element solutions as given by Chen and Yan (2015) are also plotted.

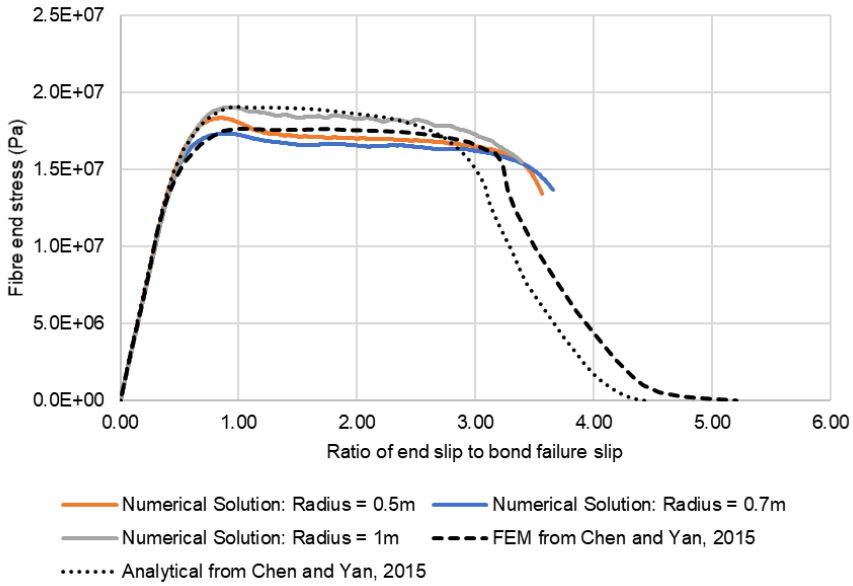

Figure 4-32: Fibre end stress plotted against the ratio of the slip between the fibre and the matrix and the slip at which bond failure occurs. The force is measured at the fibre end. Analytical and Finite Element solutions as given in Chen and Yan are also plotted

4.5.3.5 EFFECT OF FIBRE LENGTH

An essential aspect of the design of fibre reinforcing is the length of the fibres. Analytical and experimental investigations have demonstrated that as the embedded fibre length increases the maximum axial stress in the fibre during pull-out asymptotically approaches a limiting value. This allows the minimum required fibre strength to be determined such that the fibre itself does not fail before the interface. This investigation is to determine whether the DEM model can replicate this trend using the parameters tabulated in Table 4-3. The fibre embedded lengths range from 5m to 50m.

The pull-out curves, as well as the relationship between maximum force and fibre length, are shown in Figure 4-32 and Figure 4-33 (the simulations are undertaken using displacement control only). It can be seen that the maximum fibre axial stress versus fibre length closely matches the analytical result as given by Chen and Yan (2015).

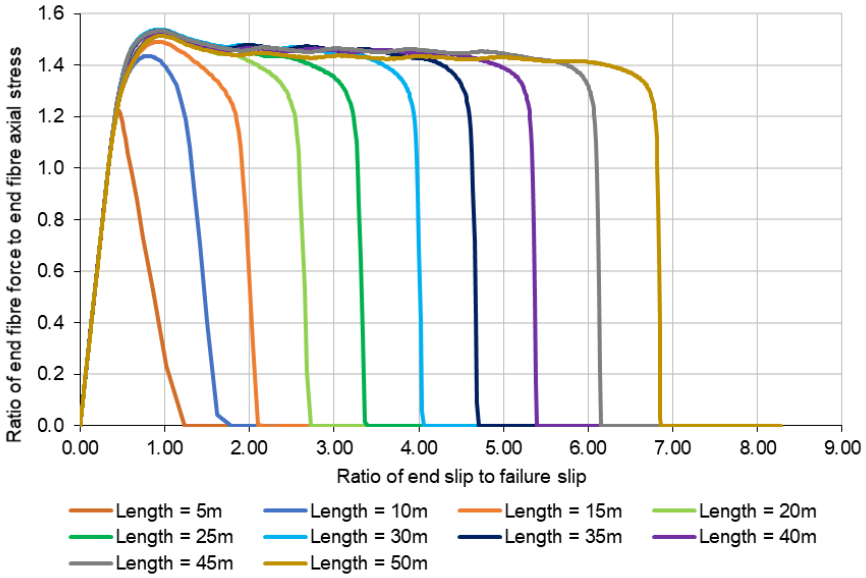

Figure 4-33: Pull out curves for different fibre lengths

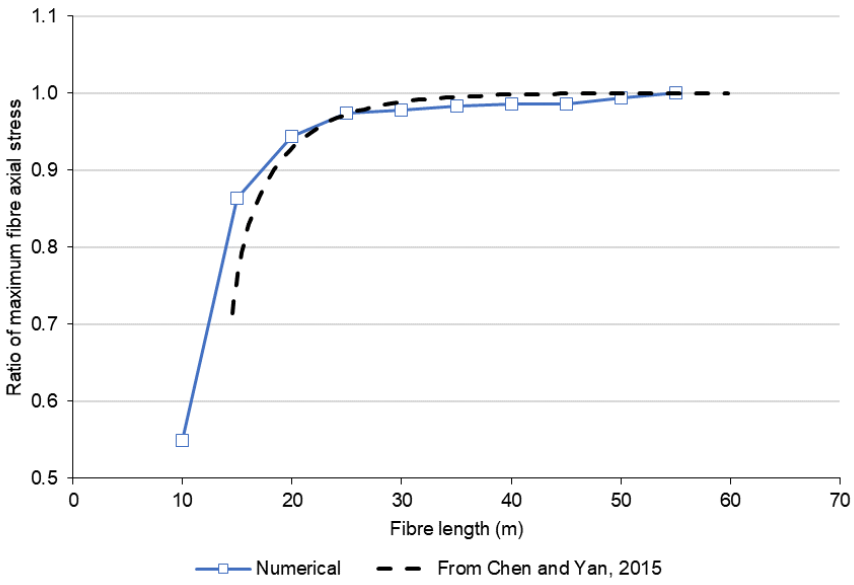

Figure 4-34: Relationship between maximum force and fibre length

4.5.3.6 FULL FAILURE SURFACE

From Figure 4-30, Figure 4-31 and Figure 4-32 it can be seen that the final stages of the debonding phase occur rapidly. The equilibrium stress condition during this final debonding stage cannot be recovered using a displacement-controlled methodology. An investigation was carried out using particles bonded using the LSDP contact model to track the failure surface. The previous investigations have revealed that for this model, the difference between parallel bonds and LSDP is expected to be small.

The full failure surface was tracked by loading the fibre until an increment of damage was accumulated in the interface and then returning the fibre end to the initial position. During the process of returning the fibre end to the starting position all further damage accumulation in the interface is arrested. The results, as well as the failure surface, are shown in Figure 4-34. A point of maximum interfacial slip is reached at approximately 0.0095m. At this point, sufficient damage has occurred at the interface that any further damage occurs at reducing interfacial slips and forces since the elastic energy in the fibre is sufficient to drive damage. Hence, displacement control would result in a fully dynamic debonding of the remaining interface driven by the elastic energy in the fibre, similar to what was observed by Chen and Yan (2015). However, by employing our load/unload approach, we can trace the failure surface to reveal a distinct result from that presented by Chen and Yan (2015).

The previous history of damage to the interface plays a role in the shape of the failure surface. Beyond a specific maximum interfacial slip, further damage to the interface occurs at reducing load and slip. The failure curve could be extrapolated both from Figure 4-34 and through argumentation to end at zero fibre stress and a maximum interfacial slip of 0.003m (the failure slip of the interface bonds). This would match the softening curve employed for the contact models which fail at zero stress and a shear displacement of 0.003m. The failure of the last bond along the interface necessitates therefore that the point of zero stress and an interfacial slip of 0.003m is the final point on the failure surface. This point cannot be

recovered numerically due to the very low stiffness of the interface close to the point of failure.

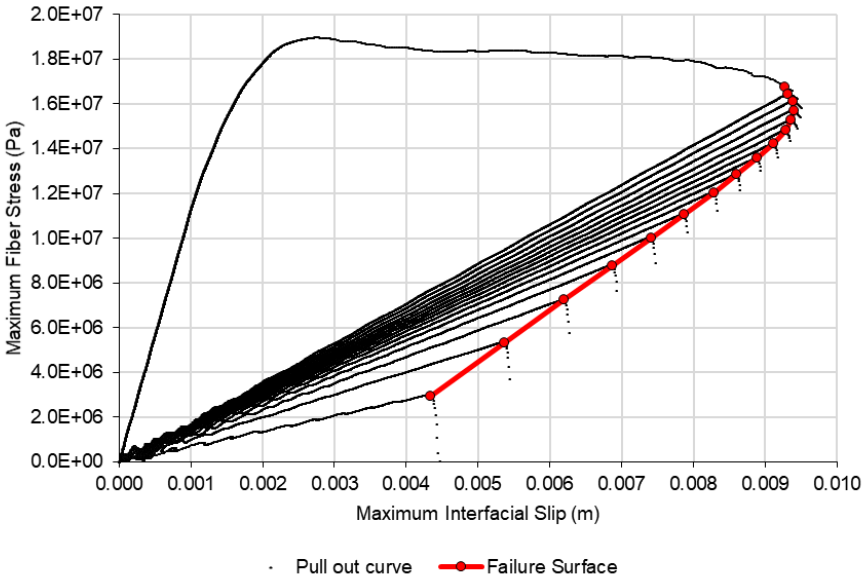

Figure 4-35: Full failure surface. The failure surface is defined as the minimum combination of interfacial slip and fibre stress which results in the remaining bonds accruing damage.

4.5.3.7 DISCUSSION OF DEBONDING PHASE

The developed DEM was able to simulate the debonding phase of a fibre pull-out. Notably, the full pull-out curve displayed a maximum interfacial shear slip. Beyond the maximum interfacial slip, the interfacial shear slip needs to reduce. The full pull-out curve cannot be recovered using displacement control, which is notably the reason why it has not been reported from numerical investigations to the best of the authors' knowledge. The reported failure surface presented by Chen and Yan (2015) was most likely obtained using displacement control and therefore was unable to isolate the entire failure surface. Displacement control is unable to allow for the reduction of strength by reducing the interfacial shear slip, resulting in a sudden dynamic debonding of the remaining interface. The full failure surface is recovered by intermittently loading and unloading the fibre.

4.6 Conclusion

The developed DEM code was used to simulate the stress growth phase and debonding phase of a uniaxial fibre pull-out. It was found that both phases could be modelled to a high level of accuracy using DEM with appropriate particle size, which has not been replicated elsewhere in the literature for this form of problem. The bond softening model was able to replicate the softening behaviour and the interfacial shear stress distribution as predicted analytically. The fibre pull-out loading-displacement response is sensitive to the size of the particles used in the simulation. The size of the particle should be at most 25% of the fibre diameter to capture the appropriate form of the pull-out curve.

The full debonding curve was found to be complex with the final stages showing reducing strength with reducing interfacial slip. Tracking this debonding behaviour required a load/unload strategy to be used. The full shape of the debonding curve has not been reported before.

There was no difference in the predicted pull-out curves, and interfacial shear stresses for both the LSDP and the parallel bond contact models. This is due to the LSDP bonds effectively setting up couple moments when sufficient particles are present are therefore replicating the behaviour of parallel bonded models. It should be noted though that for a single string of particles the parallel bonded models are able to transmit moments and the LSDP bonds are not. For the uniaxial pull-out problem, the response is insensitive to the contact model used.

Damping is an essential aspect of maintaining stability for DEM simulations. Two different damping methods were investigated: viscous local and non-viscous local damping. It was found that for a pair of particles where one has perturbed both damping methods quickly return the system to equilibrium. For the string of particles pulled at one end, the effects of the two damping methods differ significantly. The viscous damping approach produced the expected behaviour while the non-viscous damping approach resulted in non-physical behaviour and did not seem to apply to this type of problem.

Future study is recommended to investigate the minimum number of particles through the thickness of the fibre that would be required to model the fibre debonding behaviour accurately. Also, to simulate a fibre under various loading and for various geometric shapes would induce bending moments and amplify the differences between the two models when limiting the number of particles that describe the fibre. Such scenarios with fibres exhibiting hooks or waves are common in practice. Future investigations should consider the effect of various packings on the predicted response. Breakage of the matrix and fibre, as well as the interaction between multiple fibres, should be investigated.

5 Conclusion

The research carried out in this dissertation is composed of two sections: the development of a Python-based, GPU accelerated, DEM program, and the investigation of a fibre pull-out using the developed program. The DEM program was verified against some example problems, and the accuracy of the fibre pull-out simulation was compared against research published by Chen and Yan (2015).

The programming language Python, leveraging modules Numpy and Numba, was used to develop a DEM program. The parallel nature of DEM was exploited by writing the DEM code to execute on the GPU. The DEM was written to be able to model bonded particles and progressive damage of the bonds during loading. The performance of the DEM was investigated and found to provide almost 2 orders of magnitude improvement on similar code running in serial on CPU. Two examples were investigated using the developed code.

A minimal physical model of the chain fountain was devised and modelled using two different bond models. The bond models differed by whether they transmitted moments between particles or not. The excess reaction force, predicted by published investigations into the chain fountain, that occurs as the chain is picked up during the siphoning of the chain was recovered for moment transmitting bonds only. This suggests that the bending stiffness of the chain plays an essential role in the dynamics observed. The minimal physical example devised here and the method of investigating the chain fountain itself show promise for further investigation. The elastic interaction between a sphere and a plane was also successfully demonstrated as further verification of the developed code.

The developed code was used to replicate the fibre pull-out experiment, as described by Chen and Yan (2015). The authors describe the full pull-out curve and interfacial stresses for a fictitious embedded fibre using both an analytical and FEM model with good agreement. The developed DEM code was used to perform a simulation with the same geometry and parameters as used by Chen and Yan in their investigation. Two bond models: Linear Spring Dash Pot (LSDP) and Parallel Bond models were individually calibrated and compared. Different particle sizes were calibrated for and compared. A single particle packing, hexagonal close packing, was used.

Damping is a necessary component of any DEM simulation. Two methods to apply damping were investigated, viscous and non-viscous damping. It was found that both methods of damping effectively damped particle motion when particles are at rest. It was, however, found that if several particles are experiencing rigid body motion, non-viscous damping results in non-physical behaviour. Viscous damping was able to damp excess particle oscillation for the same scenario without non-physical behaviour.

Two phases of the fibre pull-out process were simulated: The stress growth phase where the interface between the fibre and the matrix is intact, and the debonding phase where the interface accumulates damage and eventually fails in its entirety. The frictional phase, where stress transfer between the matrix and fibre is via friction, was not simulated in this study.

For the stress growth phase, the accuracy of the DEM was investigated for various combinations of the problem geometry (fibre length) and material parameters (fibre, matrix,

and interfacial stiffness). The interfacial shear stresses and fibre axial stress from the simulation were found to closely match those predicted by the analytical model for most parameters investigated. The most significant deviation from the analytical solution occurred when the deformation of the fibre and matrix was high, such as when the fibre is significantly stiffer than the matrix and when the interfacial stiffness is high. The interfacial shear stresses and axial fibre stress were accurately predicted for all the particle radii investigated. The implication, therefore, is that the minimum particle size is dependent on the fibre radius and the sample packing used. It was also found that the choice of bond model (LSDP or Parallel) did not influence the result. This is due to the small relative rotations and moments between particles for the problem geometry here. Further investigations for more complex fibre geometries, such as hooked ends, which result in larger moments in the fibre and matrix may result in differences between the bond models.

During the debonding phase, the bonds across the interface yield and fail. The effect of fibre extraction speed, fibre length and bond model were investigated for this phase. The maximum fibre extraction speed such that the dynamic effect on the interfacial shear stress is eliminated was determined to be 0.01m/s. The analytically predicted relationship between fibre length and maximum force was recovered. As for the stress growth phase, no difference between the bond models was noted. During the debonding investigation, it became apparent that dynamic effects dominated the behaviour during the final phases of debonding. To eliminate the dynamic effects a method, involving successive loading and unloading, was employed to recover the load-displacement relationship. The simulated load/ displacement curve matched the analytical and FEM results of Chen and Yan (2015) for the early portion of the fibre pull-out where dynamic effects are not present. The last portion of the pull-out curve varies from the results presented by Chen and Yan (2015), with a reduction in maximum load corresponding to a reduction in displacement. By inspection, it can be demonstrated that the final portion of the load/displacement curve recovered here is of the correct form. The discrepancy between the research undertaken here and the results presented by Chen and Yan (2015) is most likely due to the authors of that paper misinterpreting the dynamic loads of a fully debonded fibre/matrix system as static loads of a bonded matrix/fibre system.

The developed program has proven to be a useful tool for investigating fibre pull-out. The program could be modified to account for dynamic effects by implementing an arc-length control method, which would have been challenging to do with a commercial DEM package. The flexibility of this package could be used to further research into fibre/matrix debonding. Some further research possibilities are:

- Investigate the stresses and debonding for multiple embedded fibres. Interaction between individual fibres result in complex stress fields, specifically as debonding occurs in some of the embedded fibres.
- Incorporate fibre/matrix friction and matrix strength into the simulation. This is an essential step towards approaching the behaviour of physical models where friction and failure of the matrix can influence the macroscopic behaviour of the composite.
- The final research stage is to demonstrate the modelling of a full fibre-reinforced composite material incorporating possible matrix failure, friction and increasing the number of particles in the simulation than used in the research presented here.

The Python programming language is generally accepted to be easier to learn than compiled languages such as C++. Python is also often taught to engineering students. The developed code can be used for the teaching of DEM, parallel computing on GPU and general programming.

6 References

Amada, T. et al. *(2004) 'Particle-Based Fluid Simulation on GPU',* ACM Workshop on General-Purpose Computing on Graphics Processors and SIGGRAPH*, pp. 1–2.*

Arain, M. F. et al. *(2019) 'Experimental and numerical study on tensile behavior of surface modified PVA fiber reinforced strain-hardening cementitious composites (PVA-SHCC)',* Construction and Building Materials*. Elsevier Ltd, 217, pp. 403–415.*

Aslani, F. and Nejadi, S. (2011) 'Evaluation and Comparison of the Analytical Models to Determine Tensile Strength of Self-Compacting Concrete and Conventional Concrete', (1), pp. 1–8.

Avery, G. T, "Sensitivity studies on the shear lag parameter β using analytical and numerical techniques" (2016). Theses and Dissertations. Paper 2021.

Azevedo, N. M. and Lemos, J. V. (2006) 'Hybrid discrete element/finite element method for fracture analysis', Computer Methods in Applied Mechanics and Engineering*, 195(33–36), pp. 4579–4593.*

Biggins, J. S. and Warner, M. (2014) 'Understanding the chain fountain', Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 470(2163).*

Bouhala, L. et al. *(2013) 'Modelling of failure in long fibres reinforced composites by X-FEM and cohesive zone model',* Composites Part B: Engineering*. Elsevier Ltd, 55(January), pp. 352–361.*

Budarapu, P. R. et al. *(2019) 'Stress transfer through the interphase in curved-fiber pullout tests of nanocomposites',* Composites Part B: Engineering*. Elsevier, 165(December 2018), pp. 417–434.*

Chen, J. Y. (2009) 'GPU technology trends and future requirements', Technical Digest - International Electron Devices Meeting, IEDM*, pp. 3–8.*

Chen, X., Beyerlein, I. J. and Brinson, L. C. (2009) 'Curved-fiber pull-out model for nanocomposites. Part 1: Bonded stage formulation', Mechanics of Materials*. Elsevier Ltd, 41(3), pp. 279–292.*

Chen, Z. and Yan, W. (2015) 'A shear-lag model with a cohesive fibre-matrix interface for analysis of fibre pull-out', Mechanics of Materials*. Elsevier Ltd, 91, pp. 119–135.*

Cundall, P. A. (1971) 'A Computer Model for Simulating Progressive Large-Scale Movements in Blocky Rock Systems', Proceesings of the Symposium of the International Society of Rock Mechanics*, 1, pp. 11–8.*

Cundall, P. A. and Strack, O. D. L. (1979) 'A discrete numerical model for granular assemblies', Geotechnique*, 29(1), pp. 47–65.*

Denneman, E. et al. *(2011) 'Discrete fracture in high performance fibre reinforced concrete materials',* Engineering Fracture Mechanics*, 78(10), pp. 2235–2245.*

Dowding, C. H., Dmytryshyn, O. and Belytschko, T. B. (1999) 'Parallel processing for a discrete element program', Computers and Geotechnics*, 25(4), pp. 281–285.*

Elsen, E. et al. *(2006) 'N-Body simulation on GPUs',* Proceedings of the 2006 ACM/IEEE Conference on Supercomputing, SC'06*, (June).*

Flekkøy, E. G., Moura, M. and Måløy, K. J. (2018) 'Mechanisms of the flying chain fountain', Frontiers in Physics*, 6(AUG), pp. 1–7.*

Fu, Z. et al. *(2014) 'Architecting the finite element method pipeline for the GPU',* Journal of Computational and Applied Mathematics*. Elsevier B.V., 257, pp. 195–211.*

Ghorpade, J. (2012) 'GPGPU Processing in CUDA Architecture', Advanced Computing: An International Journal*, 3(1), pp. 105–120.*

Govender, N., Wilke, D. N. and Kok, S. (2015) 'Blaze-DEMGPU: Modular high performance DEM framework for the GPU architecture', SoftwareX*. Elsevier B.V., 5, pp. 62–66.*

Guo, G. and Zhu, Y. (2015) 'Cohesive-Shear-Lag Modeling of Interfacial Stress Transfer between a Monolayer Graphene and a Polymer Substrate', Journal of Applied Mechanics, Transactions ASME*, 82(3), pp. 1–7.*

Heidarhaei, M., Shariati, M. and Eipakchi, H. R. (2018) 'Effect of interfacial debonding on stress transfer in graphene reinforced polymer nanocomposites', International Journal of Damage Mechanics*, 27(7), pp. 1105–1127.*

Heidarhaei, M., Shariati, M. and Eipakchi, H. R. (2019) 'Analytical investigation of interfacial debonding in graphene-reinforced polymer nanocomposites with cohesive zone interface', Mechanics of Advanced Materials and Structures*. Taylor & Francis, 26(12), pp. 1008–1017.*

Herman, A. (2015) 'Discrete-Element bonded particle Sea Ice model Design, version 1.3 – model description and implementation', Geoscientific Model Development Discussions*, 8(7), pp. 5481–5533.*

Hertz, H. (1881) 'H. Hertz, Über die Berührung fester elastischer Körper, Journal für die reine und angewandte Mathematik 92, 156-171 (1881)', Journal für die reine und angewandte Mathematik*, 171, pp. 156–171.*

Ho, N. M. and Wong, W. F. (2017) 'Exploiting half precision arithmetic in Nvidia GPUs', in 2017 IEEE High Performance Extreme Computing Conference, HPEC 2017*.*

Hofmann, T. (2011) Evaluation of Time Integrations Schemes in DEM-Simulations*. Technische Universitat Munchen. Available at: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.396.7738&rep=rep1&type=pdf.*

Hsueh, C. H. (1988) 'Elastic load transfer from partially embedded axially loaded fibre to matrix', Journal of Materials Science Letters*, 7(5), pp. 497–500.*

Ismail, Y., Yang, D. and Ye, J. (2016) 'A DEM model for visualising damage evolution and predicting failure envelope of composite laminae under biaxial loads', Composites Part B: Engineering*. Elsevier Ltd, 102, pp. 9–28.*

Jansson, A. (2011) Effects of Steel Fibres on Cracking in Reinforced Concrete, International Journal of Engineering Research & Technology (IJERT)*. Available at: http://publications.lib.chalmers.se/publication/140703-effects-of-steel-fibres-on-cracking-in-reinforced-concrete.*

Jd, R. (2014) 'Effect of Single Fiber Pull Out Test Result on Flexural Performance of ECC', 4(2), pp. 2–7.

Kang, J. et al. *(2014) 'Modeling of fiber-reinforced cement composites: Discrete representation of fiber pullout',* International Journal of Solids and Structures*. Elsevier Ltd, 51(10), pp. 1970–*

1979.

Koval, G., Danh, L. B. and Chazallon, C. (2014) 'Discrete element model for brittle fracture', Particles 2013*, (September 2013).*

Lam, S. K., Pitrou, A. and Seibert, S. (2015) 'Numba: A LLVM-based python JIT compiler', Proceedings of the Second Workshop on the LLVM Compiler Infrastructure in HPC - LLVM '15*, pp. 1–6.*

Lau, K. tak et al. *(2018) 'Properties of natural fibre composites for structural engineering applications',* Composites Part B: Engineering*. Elsevier, 136(September 2017), pp. 222–233.*

Leclerc, W. et al. *(2017) 'On a Discrete Element Method to simulate the mechanical behavior of heterogeneous media'. Available at: https://cfm2017.sciencesconf.org/129976/document.*

Leon, S. E. et al. *(2011) 'A unified library of nonlinear solution schemes',* Applied Mechanics Reviews*, 64(4).*

Liu, X., Duddu, R. and Waisman, H. (2012) 'Discrete damage zone model for fracture initiation and propagation', Engineering Fracture Mechanics*. Elsevier Ltd, 92, pp. 1–18.*

Ma, Z. et al. *(2011) 'A GPU accelerated continuous-based Discrete Element Method for elastodynamics analysis',* Advanced Materials Research*, 320, pp. 329–334.*

Maheo, L. et al. *(2015) 'A promising way to model cracks in composite using Discrete Element Method',* Composites Part B: Engineering*, 71, pp. 193–202.*

Maknickas, A. et al. *(2006) 'Parallel DEM software for simulation of granular media',* Informatica*, 17(2), pp. 207–224.*

Marcon, M. et al. *(2017) 'Modeling adhesive anchors in a discrete element framework',* Materials*, 10(8), pp. 1–23.*

Millman, K. J. and Aivazis, M. (2011) 'Python for scientists and engineers', Computing in Science and Engineering*, 13(2), pp. 9–12.*

Moës, N., Dolbow, J. and Belytschko, T. (1999) 'A finite element method for crack growth without remeshing', International Journal for Numerical Methods in Engineering*, 46(1), pp. 131–150.*

Monteiro Azevedo, N. (2003) 'A rigid particle discrete element model for the fracture analysis of plain and reinforced concrete', PhD Thesis, Heriot-Watt University, Scotland*.*

Mould, S. (2013) Self siphoning beads*. Available at: http://stevemould.com/siphoning-beads/.*

Naaman, A.E.; Namur, G.; Najm, H.; Alwan, J. Bond Mechanisms in Fiber Reinforced Cement-Based Composites; Report UMCE 89-9; Department of Civil Engineering, University of Michigan: Ann Arbor, MI, USA, 1989; pp. 1–233.

Nairn, J. A. (1997) 'On the use of shear-lag methods for analysis of stress transfer in unidirectional composites', Mechanics of Materials*, 26(2), pp. 63–80.*

Nairn, J. A. (2000) 'Analytical fracture mechanics analysis of the pull-out test including the effects of friction and thermal stresses', Advanced Composites Letters*, 9(6), pp. 373–383.*

Nairn, J. A. et al. *(2001) 'Fracture Mechanics Analysis of the Single-Fiber Pull-Out Test and the Microbond Test Including The Effects of Friction and Thermal Stresses',* Proc. 16th Ann. Tech. Conf. of the Amer. Soc. Composites*, pp. 9–12.*

Nickolls, J., Buck, I. and Garland, M. (2008) 'Scalable Parallel with CUDA', Queue*, 6(April), pp. 40–53. Available at: http://portal.acm.org/citation.cfm?id=1365490.1365500.*

Nickolls, J. and Kirk, D. (2009) 'Graphics and Computing GPUs BT - Computer Organization and Design', Computer Organization and Design*, pp. 1–76. Available at: https://www.google.co.kr/%5Cnpapers3://publication/uuid/4287E443-C8E2-4311-9F78-F578547AA694.*

Nvidia (2010) OpenCL Programming Guide for the CUDA Architecture, Version 3.2*,* Cuda SDK*. Available at: http://scholar.google.com/scholar?hl=en&btnG=Search&q=intitle:OpenCL+Programming+Guide+for+the+CUDA+Architecture#7%5Cnhttp://scholar.google.com/scholar?hl=en&btnG=Search&q=intitle:OpenCL+Programming+Guide+for+the+CUDA+Architecture,+Version+3.2#0.*

Oliphant, T. E. (2007) 'Python for Scientific Computing Python Overview', Computing in Science and Engineering*, pp. 10–20.*

Orlando, N. and Benvenuti, E. (2016) 'Advanced XFEM simulation of pull-out and debonding of steel bars and FRP-reinforcements in concrete beams', American Journal of Engineering and Applied Sciences*, 9(3), pp. 746–754.*

Owens, J. D. et al. *(2008) 'GPU computing',* Proceedings of the IEEE*, 96(5), pp. 879–899.*

Potapov, S., Faucher, V. and Daudeville, L. (2012) 'Advanced simulation of damage of reinforced concrete structures under impact', European Journal of Environmental and Civil Engineering*, 16(9), pp. 1090–1101.*

Potyondy, D. O. and Cundall, P. A. (2004) 'A bonded-particle model for rock', International Journal of Rock Mechanics and Mining Sciences*, 41(8 SPEC.ISS.), pp. 1329–1364.*

Qi, J. et al. *(2015) 'GPU-accelerated DEM implementation with CUDA Ji Qi Kuan-Ching Li Hai Jiang Qingguo Zhou Lei Yang *',* International Journal of Computational Science and Engineering*, 11(3), pp. 330–337.*

Reissner, E. (1946) 'Analysis of shear lag in box beams by the principle of minimum potential energy', Quarterly of Applied Mathematics*, 4(3), pp. 268–278.*

Rojek, J. et al. *(2012) 'Comparative study of different discrete element models and evaluation of equivalent micromechanical parameters',* International Journal of Solids and Structures*. Elsevier Ltd, 49(13), pp. 1497–1517.*

Greenlaw, R., Hoover, H. and Ruzzo, W., 1995. Limits To Parallel Computation*. New York, N.Y.: Oxford University Press.*

Sadek, M. A. et al. *(2014) 'Simulation of tensile tests of hemp fibre using discrete element method',* Agricultural Engineering International: CIGR Journal*, 16(2), pp. 126–135.*

Shang, J. et al. *(2018) 'DEM simulation of mortar-bolt interface behaviour subjected to shearing',* Construction and Building Materials*, 185(November), pp. 120–137.*

Sheng, Y. et al. *(2010) 'Microstructure effects on transverse cracking in composite laminae by DEM',* Composites Science and Technology*. Elsevier Ltd, 70(14), pp. 2093–2101.*

Shigeto, Y. and Sakai, M. (2011) 'Parallel computing of discrete element method on multi-core processors', Particuology*. Chinese Society of Particuology, 9(4), pp. 398–405.*

Šmilauer, V. and Chareyre, B. (2015) 'DEM Formulation, Release Yade documentation', Yade

Documentation*, (January 2015), pp. 137–160.*

Syed, Z. I. (2017) 'Development and calibration of discrete element method inputs to mechanical responses of granular materials'. Available at: https://lib.dr.iastate.edu/etd/16226.

Villaggio, P. (1996) 'The rebound of an elastic sphere against a rigid wall', Journal of Applied Mechanics, Transactions ASME*, 63(2), pp. 259–263.*

Virtanen, P. et al. *(2019) 'SciPy 1.0--Fundamental Algorithms for Scientific Computing in Python', pp. 1–22. Available at: http://arxiv.org/abs/1907.10121.*

van der Walt, S. and Aivazis, M. (2011) 'The NumPy Array: A Structure for Efficient Numerical Computation, Computing in Science & Engineering', Computing in Science and Engineering*, 13(2), pp. 22–30. Available at: http://aip.scitation.org/doi/abs/10.1109/MCSE.2011.37.*

Wang, G. et al. *(2017) 'Macro-Micro Failure Mechanisms and Damage Modeling of a Bolted Rock Joint',* Advances in Materials Science and Engineering*, 2017.*

Wolff, M. F. H. et al. *(2013) 'Three-dimensional discrete element modeling of micromechanical bending tests of ceramic-polymer composite materials',* Powder Technology*. The Authors, 248, pp. 77–83.*

Xu, J. et al. *(2011) 'Quasi-real-time simulation of rotating drum using discrete element method with parallel GPU computing',* Particuology*, 9(4), pp. 446–450.*

Yang, D. et al. *(2010) 'Discrete element modeling of the microbond test of fiber reinforced composite',* Computational Materials Science*. Elsevier B.V., 49(2), pp. 253–259.*

Yue, X. et al. *(2014) 'Parallelization of a DEM Code Based on CPU-GPU Heterogeneous Architecture', in Li, Kenli et al. (eds)* Parallel Computational Fluid Dynamics*. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 149–159.*

Zhan Y. und Bui, H. G. und N. J. und M. S. A. und M. G. (2014) 'Numerical modeling of steel fiber reinforced concrete on the meso-and macro-scale', Computational Modelling of Concrete Structures (Proceedings of EURO-C 2014)*, 1(March), pp. 579–586.*

Zhang, H. et al. *(2017) 'Poseidon: An Efficient Communication Architecture for Distributed Deep Learning on GPU Clusters', in* Proceedings of the 2017 USENIX Annual Technical Conference (USENIX ATC '17). *Available at: http://arxiv.org/abs/1706.03292.*

Zhang, X. Y. and Xie, X. T. (2017) 'Research on discrete element simulation of anchor frame beam reinforcement in bedding shale slope', IOP Conference Series: Earth and Environmental Science*, 94(1).*

Zheng, J., An, X. and Huang, M. (2012) 'GPU-based parallel algorithm for particle contact detection and its application in self-compacting concrete flow simulations', Computers and Structures*. Elsevier Ltd, 112–113, pp. 193–204.*