# Up-Scaling Distinct Element Method Simulations of Discontinua

by

Michael Yetisir

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Applied Science
in
Civil Engineering

Waterloo, Ontario, Canada, 2017

## Abstract

Pre-existing fractures significantly influence the geomechanical response of the rock mass at the reservoir scale. For geomechanical applications, these natural fractures need to be considered in the mechanical response of the system. Distinct Element Methods (DEM) are often used to explicitly model the mechanics of Naturally Fractured Rock (NFR); however, they are often too computationally prohibitive for reservoir-scale problems. A DEM up-scaling framework is presented that facilitates estimating a representative parameter set for continuum constitutive models that capture the salient feature of Naturally Fractured Rock (NFR) behaviour.

Up-scaling is achieved by matching homogenized DEM stress-strain curves from multiple load paths to those of continuum constitutive models using a Particle Swarm Optimization (PSO) algorithm followed by a Damped Least-Squares (DLS) algorithm. The effectiveness of the framework is demonstrated by up-scaling a DEM model of a NFR to a Drucker-Prager damage-plasticity model; the up-scaled model is shown to capture well the effect of confinement on the the yielding and sliding of natural fractures in the rock mass.

The goal of this thesis is to present a framework to facilitate effective simulation of fine-scale behaviour in full-scale NFR systems while significantly reducing the computational demands associated with modelling these systems with DEM.

As such, four main research objectives have been identified and achieved: 1) Develop and implement stress and strain homogenization algorithms for DEM models with deformable blocks, 2) present a methodology to parameterize complex nonlinear continuum constitutive models, 3) develop and implement an automated modular software framework for up-scaling DEM simulations, and 4) demonstrate that the performance of the up-scaled continuum models are accurate and significantly more computationally efficient.

The up-scaling methodology is verified through a case study on a naturally fractured granite slope in which the top surface is loaded until failure. The up-scaled continuum model is shown to compare quite well to Direct Numerical Simulation (DNS) in a slope stability analysis and requires two orders of magnitude less computational effort.

**Acknowledgements**

This thesis, though perhaps at times a lonely affair, could not have been possible without the involvement and support of many individuals who helped and encouraged me at every step along the way. I wish to acknowledge the invaluable support and contributions that everyone provided.

First and foremost, I wish to thank my two research supervisors, Dr. Rob Gracie and Dr. Maurice Dusseault, for their endless technical insights, inspirational ideas, and moral support throughout the duration of this degree. Without their guidance, this research thesis would not have been nearly as comprehensive nor comprehensible.

Additionally, I would like to acknowledge Dr. L. Shawn Matott for allowing me to use his OSTRICH optimization software and providing some much needed support when I was getting started.

Furthermore, I should thank my partners in crime (and research), Endrina Rivas and Eleanor Mak, with whom I have shared many late nights at the office. These ladies have kept me motivated when my research has stagnated, while conversely making sure I never worked too hard.

Lastly, additional thanks goes to my coaches, Vinit Kudva, Clive Porter, and Jeff Muirhead for pushing me to be my best on and off the court, distracting me from my studies, and allowing me to be part of such a wonderfull team.

# Table of Contents

# List of Tables

# List of Figures

# List of Abbreviations

| | |
|---|---|
| API | Application Program Interface |
| APPSO | Asynchronous Parallel Particle Swarm Optimization |
| CDM | Continuum Damage Mechanics |
| CLI | Command Line Interface |
| DEM | Distinct Element Method |
| DFN | Discrete Fracture Network |
| DNS | Direct Numerical Simulation |
| FDM | Finite Difference Method |
| FEM | Finite Element Method |
| LMA | Levenberg-Marquardt Algorithm |
| NFR | Naturally Fractured Rock |
| PSO | Particle Swarm Optimization |
| REV | Representative Elementary Volume |
| RMSE | Root-Mean-Square Error |
| SRM | Selectively Refined Mesh |

# Chapter 1

# Introduction

Traditional modelling approaches tend to focus on single scale problems. For example, when considering the macroscale response of a system, the effect of the microscale mechanics is described implicitly by macroscale phenomenological constitutive relations. Conversely, when interested in the microscale behaviour, macroscale features are assumed to be homogeneous and irrelevant to the microscale response.

Macroscale constitutive relationships are generally obtained empirically based on simple methods such as linearization, Taylor series expansion, and symmetry [Weinan, 2011]. For simple systems, this empirical approach can yield sufficiently accurate approximations of the overall constitutive behaviour. However, this approach is often insufficient to capture an accurate constitutive response of highly non-linear and complex systems with materials exhibiting physical behaviour over multiple length scales. These materials are often referred to as multi-scale materials.

Alternatively, one can model the microscale behaviour explicitly throughout the domain in order to accommodate complex systems and materials. This approach is far more accurate; however, the degree of complexity can be orders of magnitude larger, making the solution difficult to find, and often finding the solution becomes computationally prohibitive [Xu et al., 2002].

To overcome the limitations of single-scale models, multi-scale approaches have been developed. Multi-scale methods attempt to model a system at both the microscale and the macroscale in such a way that shares the computational efficiency of the macroscale models as well as the accuracy of the microscale models.

The most common types of multiscale methods are hierarchical and concurrent [Gracie and Belytschko, 2011]. In concurrent multiscale models, different scales are used in different regions of the domain; the solution of the coupled model proceeds by solving both scales simultaneously. This approach is computationally expensive since the time step of the entire simulation is controlled by requirements of the fine-scale model; however, the solution is often more accurate. In hierarchical multiscale methods, the constitutive behavior at the coarser scale is determined by exercising a finer scale Representative Elementary Volume (REV) [Li et al., 2014]. The finer scale models vary from relatively simple models, as in micromechanics, to complex nonlinear models such as FE$^2$ models [Feyel, 2003]. This approach is much more efficient, but can be less accurate, and furthermore presents challenges when the REV loses stability [Belytschko et al., 2008]. Up-scaling in this investigation can be considered to be a hierarchical multiscale method using computational homogenization.

Many homogenization techniques have been developed and proposed in the past, but none have presented algorithms for homogenizing Distinct Element Method (DEM) simulations with deformable blocks. The homogenization algorithms presented here are based on the work of D'Addetta et al. [2004] and Wellmann et al. [2008] in which homogenization is applied to rigid body DEM simulations and focused on the computation of the homogenized stress and strain response.

## 1.1 Context and Research Motivation

Naturally Fractured Rock (NFR) is often modeled as a multiscale material because of the vastly different length and time scales involved in the deformation process [Zhou et al., 2003]. At the fracture scale (10$^{-2}$ m), the physics is dominated by brittle fracture propagation and fracture-to-fracture contact force interaction. However, one is normally interested in the reservoir scale (10$^3$ m) response as a result of the spatial extension of the fractures. Because these length scales of interest span approximately five orders of magnitude, multiscale methods may aid in assessing the overall response, as models with natural fracture scale resolution at the reservoir scale becomes computationally prohibitive.

When dealing with reservoir scale geomechanics problems, the complex behaviour of preexisting fractures become influential upon and indeed may dominate the constitutive response of the rock mass because of strain localization, stress redistribution, and damage-induced anisotropy [Petracca et al., 2015]. However, attempting to capture the constitutive response of the NFR in a laboratory context becomes impractical because of the prohibitively large samples required to obtain a representative response. Since natural fracture spacing in a stiff sedimentary rock can be between 0.1 m to 1m [Nelson, 2001], physical samples required to obtain a response representative of the macroscale could be as large as $1000m^3$, a nearly impossible scale for testing.

There exist methods to estimate constitutive parameters of rock masses, but their validity can be tenuous and methods exist only to estimate some of the parameters. These methods can either be based directly from in-situ geophysical measurements, or deduced indirectly through small scale laboratory testing and empirical correlations. A common way to estimate the elastic properties is through the use of seismic methods by comparing the seismic wave velocities measured in-situ to the seismic velocities of the intact rock mass [Sjogren et al., 1979]. These in-situ methods, however, do not have the capacity to estimate the plastic parameters very well, and it is well-known that seismically deduced elastic parameters invariably over-estimate the system stiffness exhibited in response to static stress changes [Barton, 2006]. Attempts to estimate the plastic properties of the rock mass through small scale lab testing and qualitative assessments of the rock mass have been presented, e.g., the Geological Strength Index (GSI) proposed by Hoek and Brown [1997]. These methods are limited by the necessarily qualitative aspect of the rock mass classification systems employed. More recently, others [Min and Jing, 2003, Chen et al., 2012, Bidgoli et al., 2013] have used numerical methods to estimate the elastic properties of the rock mass using prescribed fracture networks. These methods are again limited by the lack of plastic behaviour characterization.

To address the limitations of continuum models, Distinct Element Method (DEM) models are used commonly in geomechanics to explicitly model the mechanics of Naturally Fractured Rock (NFR) masses to capture the constitutive response of the rock mass indirectly [Jing, 2003].

DEM models, unlike standard continuum models, consider the fractures within the rock mass as a Discrete Fracture Network (DFN) which explicitly defines the geometry of the rock blocks. The physics of block interaction is then governed by the motion, contact forces and traction-separation laws between the rock blocks and the fractures [Thallak et al., 1990]. Because NFR behavior is complex, even sophisticated phenomenological constitutive relationships may be inadequate to describe the complete rock mass behavior. The DEM approach aims to address this deficiency by requiring only constitutive relations for the block interactions and the intact rock [Barbosa and Ghaboussi, 1990]. In this thesis, deformable DEM blocks are considered which require the constitutive parameters of the intact rock to be specified, but these parameters are more easily acquired from lab testing.

However, the main issue with DEM models is primarily the computational demands. Because of the large number of degrees of freedom in the models and the requirement for very small time steps — because of the constant need for contact detection between blocks — executing reservoir-scale models is computationally prohibitive. The intent of this thesis is to develop a framework that incorporates the response of DEM models while harnessing the computational speed of continuum models. The general goal of up-scaling is to formulate simplified coarse-scale governing equations that approximate the fine-scale behavior of a material [Geers et al., 2010]. In the case of the DEM simulations in this investigation, the aim of up-scaling is to identify the parameters of a continuum model that best mimic the response of the DEM model. Up-scaling is accomplished in this thesis by 'calibrating' a continuum model with DEM virtual experimental data using a combination of a heuristic optimization algorithm and an iterative least squares regression algorithm.

The goal of this thesis is to present a framework to facilitate effective simulation of fine-scale behaviour in full-scale NFR systems while significantly reducing the computational demands associated with modelling these systems with DEM.

## 1.2 Research Objectives

In this thesis, a multi-scale up-scaling framework is developed to address the computational demands of simulating microscale phenomenon in a macroscale domain in the context of NFR. The primary research objectives are:

1. **Deformable DEM Homogenization**: Develop and implement stress and strain homogenization algorithms for DEM models with deformable blocks.
2. **Parameterization Methodology**: Present a methodology to parameterize complex nonlinear continuum constitutive models.
3. **Up-Scaling Framework**: Develop and implement an automated modular software framework for up-scaling DEM simulations.
4. **Framework Verification**: Demonstrate that the performance of the up-scaled continuum models are accurate and significantly more computationally efficient.

## 1. Deformable DEM Homogenization

In order to extract the constitutive stress-strain curves from the DEM simulations, suitable homogenization algorithms are developed and applied to the DEM data. Here, since the DEM simulations use deformable bodies, application of homogenization algorithms that account for the block deformation are required. Additionally, it is also necessary to define an appropriate REV in order for the applied homogenization algorithms to be valid.

## 2. Parameterization Methodology

Running the parameter estimation algorithms requires the macroscale constitutive law to be parameterized in terms of a series of scalar parameters. Here, a parameterization methodology is presented to reduce complex constitutive laws to a manageable number of scalar parameters. The key to effective parameterization is minimizing the number of parameters, while maintaining the fidelity of the original model. This is done by assuming functional forms of curves normally provided by laboratory data. The constitutive relationships chosen aim to capture all the salient features of the discrete system.

## 3. Up-Scaling Framework

An up-scaling framework for DEM simulations is developed in this thesis. The aim of the framework is to generate a continuum parameter set that best captures the salient features of the DEM constitutive response. The framework developed is general, such that the DEM and continuum models are not directly part of the framework, but are rather plug-ins to the framework. Thus, my software implementation of the up-scaling framework, "**M**odular aut**O**mated **U**p-**S**caling softwar**E**" (MOUSE), primarily contains protocols for different modules to exchange information. The model components are wrapped as modules that plug-in to the framework so that the core up-scaling software is model independent.

## 4. Framework Verification

A series of verification studies need to be conducted in order to confirm the formulation and implementation of various aspects of the the up-scaling framework. Studies to verify the parameter estimation module, investigate the influence of the REV size on the estimated parameters, compare different macroscale constitutive models, and compare a DEM and up-scaled simulation in a Direct Numerical Simulation (DNS) are conducted. The results of these studies will show the accuracy and computational speed increase of the up-scaling.

## 1.3 Scope of Study and Research Limitations

The scope of this research is to present an up-scaling framework and a simple implementation as a proof of concept. Here, the goal is to provide all the necessary pieces using in-house and third-party software to show the up-scaling process from DEM simulations to optimal macroscale parameter set.

For simplicity, the DEM simulations considered here are large deformation, isotropic, two-dimensional, and purely mechanical (not thermo-hydro-mechanically coupled). These simplifications reduce the complexity of the implementation significantly. As a result, the homogenization algorithms presented are generalized to two-dimensional DEM simulations, though they can easily be adapted for three-dimensional problems.

The macroscale constitutive models that are used are also implemented with an isotropic response assumption and do not account for any thermal or hydraulic coupling effects. The macroscale constitutive models used are pre-implemented in ABAQUS$^{\text{TM}}$, but custom material subroutines are required for more sophisticated constitutive responses.

There are countless different optimization algorithms to investigate along with their respective optimization parameters. Exhaustively searching for the most efficient and accurate algorithm for this particular application in a quantitative capacity was not a focus of this thesis. As such, several different algorithms are presented and compared in their capacity to accurately and efficiently converge to the optimal parameter set.

Ultimately, the research presented in this thesis aims to show the efficacy of a modular up-scaling framework for DEM simulations that can facilitate the installation of upgraded modules in the future to account for more sophisticated physics.

# Chapter 2

# Up-Scaling Methodology

The goal of the following up-scaling methodology is to identify the parameters of a continuum constitutive model (macroscale model) that best emulates the average response in the DEM Representative Elementary Volume (REV) (microscale model). Let the displacement, strain and stress of the DEM REV (microscale) model be denoted by $\mathbf{u}^m$, $\boldsymbol{\epsilon}^m$, and $\boldsymbol{\sigma}^m$, respectively. Let the homogenized (averaged) strain and stress in the DEM REV model be denoted by $\langle\boldsymbol{\epsilon}\rangle$ and $\langle\boldsymbol{\sigma}\rangle$, respectively. Finally, let the strain and stress from the continuum (macroscale) constitutive model be denoted as $\boldsymbol{\epsilon}^M$ and $\boldsymbol{\sigma}^M$, respectively. The rate of macroscale stress, $\dot{\boldsymbol{\sigma}}^M = \dot{\boldsymbol{\sigma}}^M\left(\dot{\boldsymbol{\epsilon}}^M, \boldsymbol{\chi}, \mathbf{h}\right)$, is defined in terms of the rate of macroscale strain, $\dot{\boldsymbol{\epsilon}}^M$, a set of material parameters $\boldsymbol{\chi}$ and a set of internal history variables $\mathbf{h}$.

The up-scaling methodology has five steps:

1. Identify the DEM REV for the NFR.
2. Exercise the DEM REV using multiple load paths. Store $\mathbf{u}^m$, $\boldsymbol{\epsilon}^m$, and $\boldsymbol{\sigma}^m$ for each load path.
3. Apply homogenization algorithms to the microscale results ($\mathbf{u}^m$, $\boldsymbol{\epsilon}^m$, and $\boldsymbol{\sigma}^m$) to determine the average stress-strain response of the REV, i.e., $\langle\boldsymbol{\sigma}\rangle$-$\langle\boldsymbol{\epsilon}\rangle$, for each load path.
4. Identify a continuum constitutive model, $\dot{\boldsymbol{\sigma}}^M = \dot{\boldsymbol{\sigma}}^M\left(\dot{\boldsymbol{\epsilon}}^M, \boldsymbol{\chi}, \mathbf{h}\right)$, that captures the salient features of NFR mechanics.
5. Run parameter estimation algorithms to identify the parameters, $\boldsymbol{\chi}$, that minimize the difference between $\langle\boldsymbol{\sigma}\rangle$-$\langle\boldsymbol{\epsilon}\rangle$ and $\boldsymbol{\sigma}^M$-$\boldsymbol{\epsilon}^M$ over all load paths.

Once an optimal parameter set, $\boldsymbol{\chi}$, for the desired model, $\dot{\boldsymbol{\sigma}}^M = \dot{\boldsymbol{\sigma}}^M\left(\dot{\boldsymbol{\epsilon}}^M, \boldsymbol{\chi}, \mathbf{h}\right)$, has been identified, the newly established constitutive model can be used in Finite Element Method (FEM) models or with other suitable numerical or analytical simulations.

## 2.1 Up-Scaling Implementation Overview

The up-scaling framework consists of four main software components (Figure 2.1): a DEM simulator, a homogenization module, a Finite Element Method (FEM) simulator, and a parameter estimation module. In procedural order, the first software component involved is a DEM simulation package, which is used to directly model the NFR.

The DEM software accepts as inputs the geometry of the DFN, the material properties of the rock and the natural fractures, and the load paths. The DEM REV is exercised for different load-paths in a way that is akin to conducting multiple triaxial tests on physical specimens to characterize the full range of material behaviour. The DEM software outputs the microscale displacement, $\mathbf{u}^m$, and stress-strain, $\boldsymbol{\sigma}^m$-$\boldsymbol{\epsilon}^m$, responses for each load path. This microscale data is subsequently fed into the homogenization module to compute the average stress-strain response, $\langle\boldsymbol{\sigma}\rangle$-$\langle\boldsymbol{\epsilon}\rangle$, for each load path. Next, the homogenized stress-strain data, $\langle\boldsymbol{\sigma}\rangle$-$\langle\boldsymbol{\epsilon}\rangle$, is used by the parameter estimation software as observation data (i.e., laboratory/field data). The parameter estimation module iteratively executes a constitutive model, $\dot{\boldsymbol{\sigma}}^M = \dot{\boldsymbol{\sigma}}^M\left(\dot{\boldsymbol{\epsilon}}^M, \boldsymbol{\chi}, \mathbf{h}\right)$, embedded in the FEM simulator for each load path using different parameter sets, $\boldsymbol{\chi}^i$, while attempting to minimize the error between the homogenized microscale, $\langle\boldsymbol{\sigma}\rangle$-$\langle\boldsymbol{\epsilon}\rangle$, and macroscale, $\boldsymbol{\sigma}^M$-$\boldsymbol{\epsilon}^M$, stress-strain curves. Eventually, the algorithm converges to a near-optimal parameter set, $\boldsymbol{\chi}$, that can be viewed to be the best estimate of the NFR responses by the given continuum model.

In my implementation, UDEC$^{\text{TM}}$ was used as the DEM simulator and ABAQUS$^{\text{TM}}$ was used as the FEM simulator. In ABAQUS$^{\text{TM}}$, single element simulations were performed with a strain-history prescribed through displacement boundary conditions for a given set of material parameters and the stress is obtained as the output. There is nothing particularly special about the DEM or FEM simulators chosen; each could easily be replaced to overcome any inherent limitations. Moreover, a FEM simulator is not actually needed, since its inclusion in this framework is simply to gain access to the constitutive models within. The FEM simulator could easily be replaced by a Finite Difference Method (FDM) simulator or simply by a material subroutine. OSTRICH$^{\text{TM}}$, a model-independent

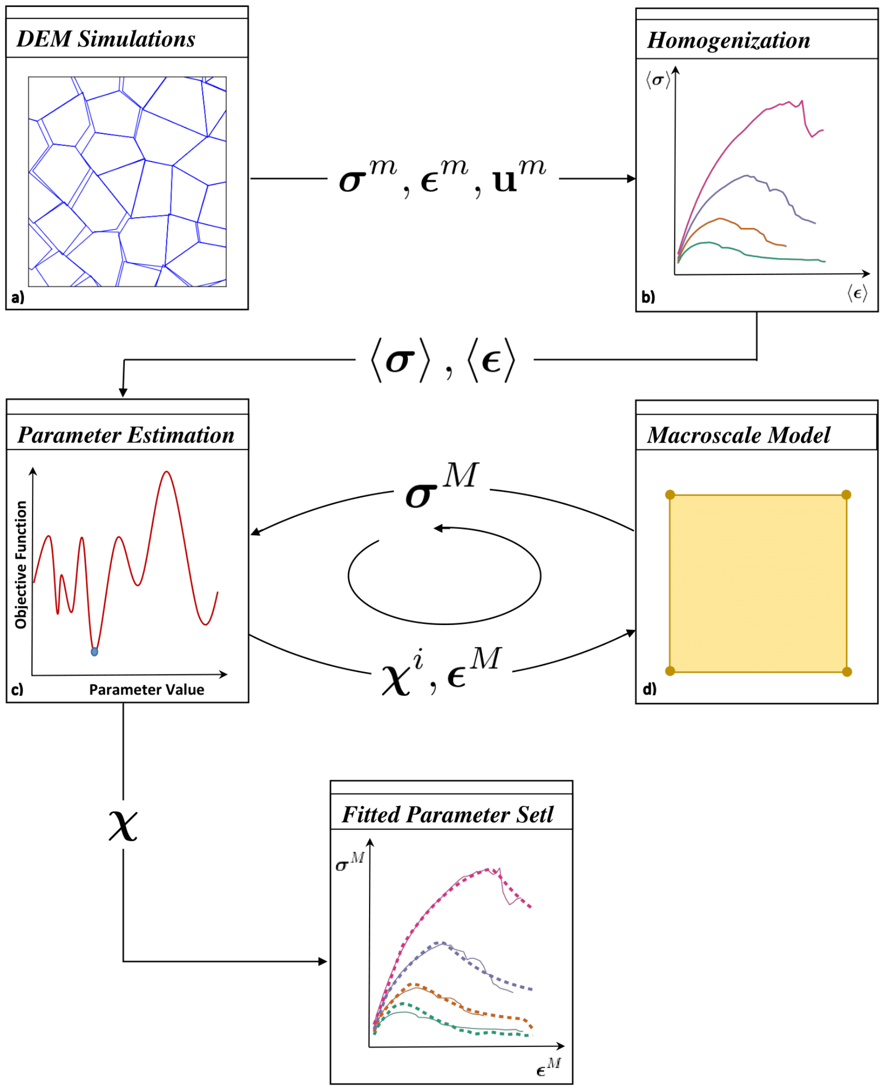


Figure 2.1: Up-scaling workflow used to estimate the optimal continuum model parameter set. The DEM software (a) produces a data set which is run through homogenization software (b) which in turn produces another dataset that is fed into the parameter estimation program (c). This parameter estimation program drives the macroscale simulations (d) iteratively in order to find an optimal parameter set for the fitted model.

optimization package [Matott, 2016], is used for the parameter estimation module. I implemented the homogenization module in Python$^{\text{TM}}$. I also used Python$^{\text{TM}}$ to develop interfaces drive the various components of the up-scaling framework.

## 2.2 Distinct Element Method

Discontinuous systems are characterized by the existence of discontinuities that separate discrete domains within the system. In order to effectively model a discontinuous system, it is necessary to represent two distinct types of mechanical behaviour: the behaviour of the discontinuities and the behaviour of the solid material.

There exists a set of methods, referred to as discrete element methods, which provide the capacity to explicitly represent the behaviour of multiple intersecting discontinuities. The methods allow for the modelling of finite displacements and rotations of discrete bodies, including contact detachment as well as automatic detection of new contacts. Within the set of Discrete Element Methods, there are four subsets [Cundall and Hart, 1992]: Modal Methods, Discontinuous Deformation Analysis Methods, Momentum Exchange Methods, and Distinct Element Method (DEM). In this thesis, the discrete element method used is DEM.

With DEM methods, the discontinuous system is represented as an assembly of deformable blocks such that the interfaces between the blocks represent the discontinuities. With respect to NFR, the blocks can be used to represent the intact rock while the discontinuities represent the joints in the rock mass.

Consider an arbitrary deformable domain, $\Omega$, with a boundary, $\Gamma$, that is subdivided by prescribed discontinuities into $i$ subdomains, each denoted by $\Omega_i$ (Figure 2.2). Let $\Gamma_{ij} = \Gamma_{ji}$ represent the boundary between $\Omega_i$ and $\Omega_j$. The motion of these subdomains (discrete elements) is governed by the conservation of momentum which relates the divergence of the stress field at a material point, $\nabla \cdot \boldsymbol{\sigma}^m$, to the element acceleration, $\ddot{\mathbf{u}}^m$, and density, $\rho$:

$$\rho \ddot{\mathbf{u}}^m = \nabla \cdot \boldsymbol{\sigma}^m \tag{2.1}$$

The interaction between $\Omega_i$ and $\Omega_j$ along $\Gamma_{ij}$ is the distinguishing feature in the DEM formulation, and is comprised of two main components: contact detection and the constitutive relationship. The contact detection algorithms are responsible for ensuring that $\Omega_i$

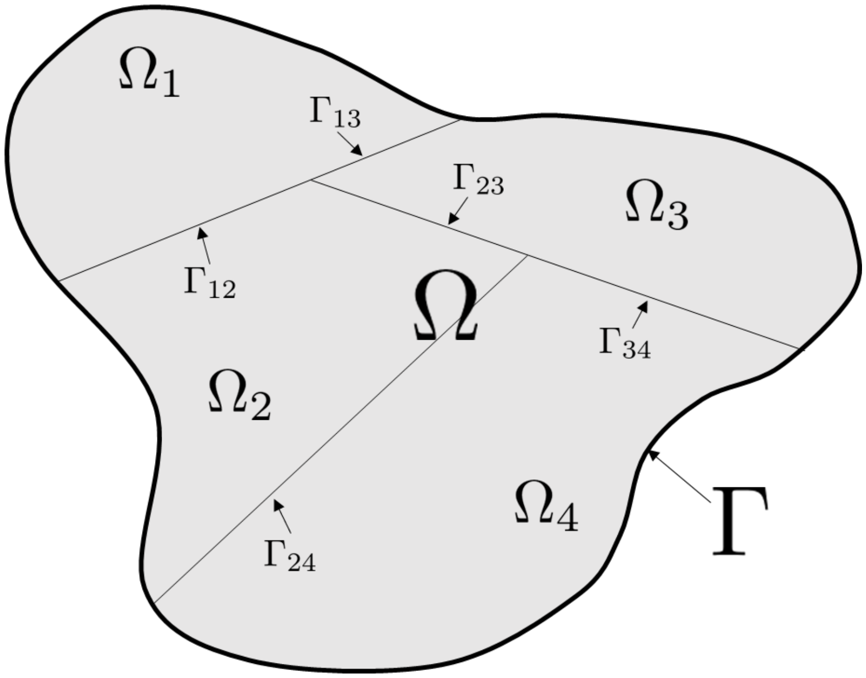


Figure 2.2: DEM block formulation for an arbitrary domain, $\Omega$, with a set of subdomains, $\Omega_i$.

and $\Omega_j$ do not penetrate each other and ensures that the appropriate contact forces are transferred between elements. These contact forces are governed by constitutive models of $\Gamma_{ij}$ which can be described in general by a shear stiffness, $k_s$, in a direction parallel to the $\Gamma_{ij}$, and a normal stiffness, $k_n$, in a direction normal to $\Gamma_i$. The normal stress over any $\Gamma_{ij}$, $\sigma^n$, can be expressed as a function of the normal elastic displacement jump across the interface, $u^n$, up until the tensile strength, $T$, is exceeded:

$$\sigma^n = \begin{cases} \sigma^n\left(k^n, u^n\right) & if \quad \sigma^n \geq -T \\ 0 & if \quad \sigma^n < -T \end{cases} \tag{2.2}$$

Futhermore, the shear stress, $\tau$, over any $\Gamma_{ij}$ can be written in terms of the elastic shear displacement jump across the interface, $u^s$, until the maximum shear strength, $\tau^{max}$ is reached. The point when the shear stress at a point on $\Gamma_{ij}$ exceeds the prescribed maximum shear stress, the discontinuity experiences plastic shear displacements in order not to exceed the maximum shear stress:

$$\tau = \begin{cases} \tau\left(k^s, u^s, \sigma^n\right) & if \quad |\tau| < \tau^{max} \\ \frac{u^s}{|u^s|}\tau^{max} & if \quad |\tau| \geq \tau^{max} \end{cases} \tag{2.3}$$

The microscale stress field, $\boldsymbol{\sigma}^m$, within $\Omega_i$ is described by standard continuum constitutive relationships. With the constitutive behaviour of the discontinuities and the constitutive behaviour of the continuum blocks, the overall behaviour of the rock mass can be characterized through material properties of the rock discontinuities and the intact rock.

## 2.3 Homogenization Approach

The main objective of up-scaling DEM simulations is to be able to describe the behavior of the discontinuous medium in terms of a more computationally efficient continuum model. The homogenization algorithms used herein to determine the average stress-strain behaviour, $\langle\boldsymbol{\sigma}\rangle$-$\langle\boldsymbol{\epsilon}\rangle$, of the REV from the microscale displacements $\mathbf{u}^m$, strain $\boldsymbol{\epsilon}^m$, and stresses $\boldsymbol{\sigma}^m$ are based on the methods developed by D'Addetta et al. [2004] and Wellmann et al. [2008]. In this homogenization process, the resultant inter-block contact forces and block displacement from the DEM simulations are converted to average stresses and strains.

For the homogenization procedure to yield meaningful results, it should be applied to a Representative Elementary Volume (REV). The exact size of the REV depends on the geometry and mechanical properties of the DEM model. For the homogenization approach to hold, the REV of size $d$ within a system with a characteristic length $D$ and consisting of blocks with a characteristic diameter $\delta$, must satisfy scale separation [Hill, 1963]:

$$D \gg d \gg \delta \tag{2.4}$$

In the following sections, all deformations are assumed to be small, such that there is no need to differentiate between the deformed and undeformed configurations.

Consider a $L \times L$ square DEM simulation domain over which mixed-boundary conditions will be applied. The REV is taken as a circular domain of radius $R$, $2R < L$. The REV is taken to be a subdomain of the actual DEM simulation domain to eliminate any boundary effects. As will be seen below, it is convenient to take the boundary of the domain used for homogenization as a slightly larger domain encompassing the REV boundary. The boundary of the homogenization domain, denoted as $\Gamma_h$, is defined by the outer edges of the deformable blocks, i.e., the cohesive/contact surfaces between deformable blocks, which intersect a circle of radius $R$ located in the center of the DEM simulation domain. Let the homogenization domain, the domain bounded by $\Gamma_h$, be denoted by $\Omega_h$. These definitions are illustrated in Figure 2.3 for a 10m $\times$ 10m DEM domain, where the deformable blocks are defined through a Voronoi tessellation. The radius of the REV domain is 2.5m. It can be seen that the actual domain used for homogenization is non-circular and larger than the REV domain.

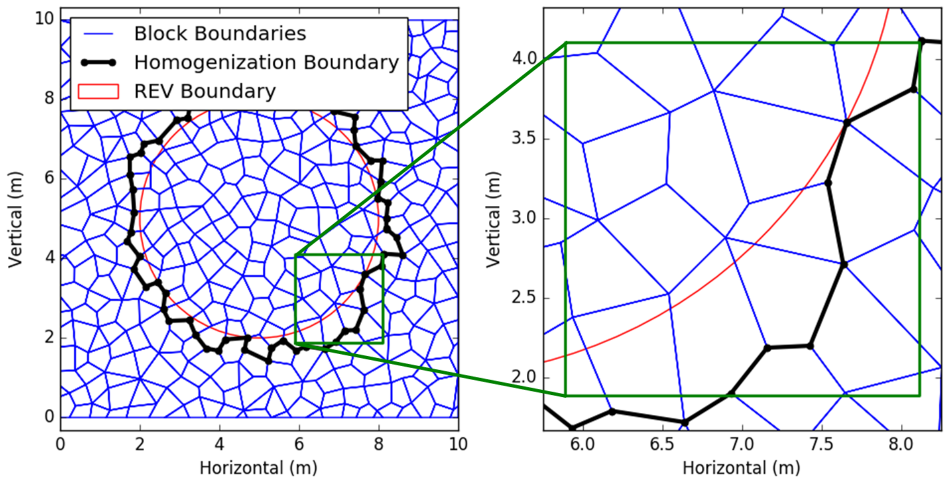


Figure 2.3: Assessment of the homogenization boundary given a circular REV.

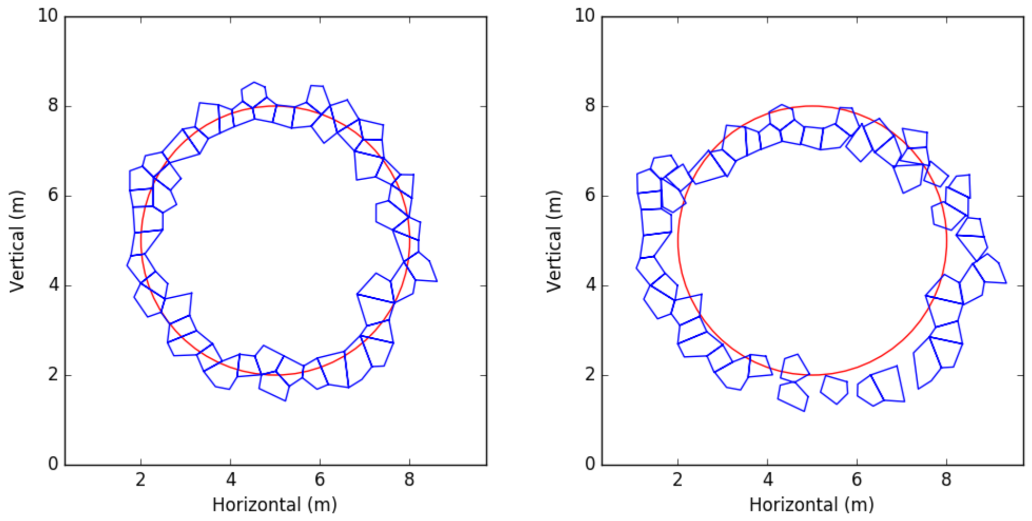
Given a circular REV, due to the discontinuous nature of the DEM simulations, the circular REV cannot be used directly. Because the calculated displacements and contact forces from the DEM are known at the block edges, the homogenization domain boundary must follow the block boundaries. The homogenization domain is characterized by a series of block corners which are identified at the initial (zero strain) state. These corners continue to define the homogenization domain once deformation occurs, allowing for a consistent homogenization domain definition as the model is deformed. An algorithm is developed here to assess the homogenization domain boundary:

1. Identify all blocks that intersect the circular REV boundary.
2. Identify all blocks that lie completely outside the REV boundary.

3. Find all contacts corresponding to the intersection between blocks in 1 and 2.
4. Find all corners corresponding to the contacts in 3.
5. Find all contacts of blocks in 1.
6. Find all blocks corresponding to the intersection of contacts in 3 and 5.
7. Find all corners corresponding to blocks in 6.
8. Find intersection of corners in 4 and 7.
9. Order corners from 8 such that they form the closed boundary of the homogenization area.

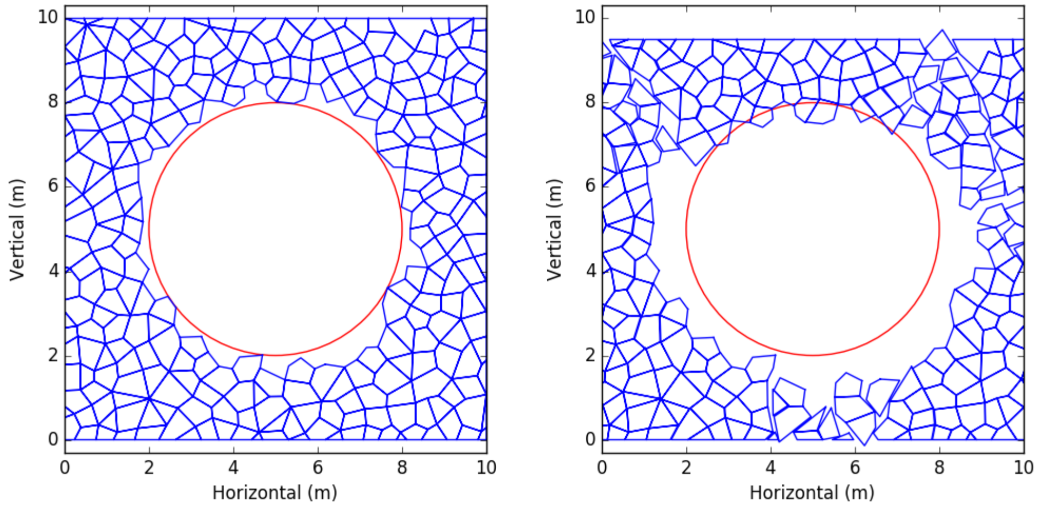
This algorithm for defining the homogenization domain is developed to provide an unambiguous method for assessing a unique homogenization domain for a given circular REV at a specific location. In steps 1 and 2, two mutually exclusive block sets are identified (Figures 2.4 and 2.5), which necessarily share contacts. It is the boundary between these two block sets that is the homogenization domain boundary which is characterized by this algorithm. In the diagrams accompanying each step, the model is shown in both the original and deformed configuration for clarity. In some cases, the diagram of the deformed configuration helps illustrate the reason for the steps, though for the calculations, the original configuration is used. Furthermore, this algorithm is formulated to be able to be used for homogenization in the deformed configuration as well.

Ultimately, the goal of this process is to identify the corners on this boundary that are on the blocks that intersect with the REV boundary and not the blocks that are outside the REV boundary. As such, the corners (step 4, Figure 2.7) corresponding to the contacts (step 3, Figure 2.6) between the two sets of blocks are determined. Step 5 (Figure 2.8) is necessary to help eliminate some blocks from the set of boundary block which intersect the REV boundary, but only have contact with blocks inside the homogenization domain boundary, and thus do not have any overall contribution to the definition. Step 6 (Figure 2.9) determines this resultant set of boundary blocks of which every member is connected to the boundary in some capacity. Finding the set of corners that is mutually shared by these blocks in step 7 (Figure 2.10) and the boundary contact corners in step 4 allow for the determination of the initial set of boundary corners (step 8, Figure 2.11). It is also necessary to order the corners (step 9, Figure 2.11) to define a closed set of boundary segments along which integration can be performed.



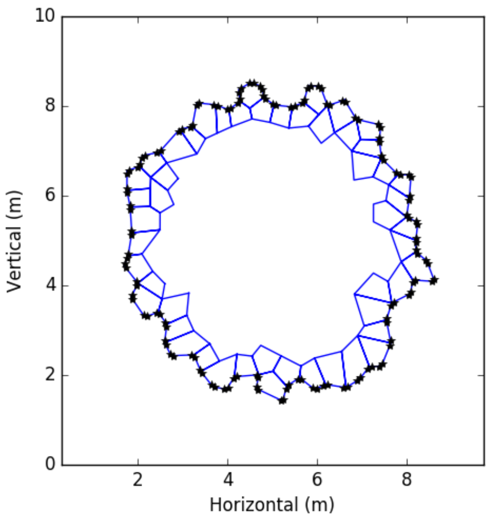
(a) Undeformed configuration (b) Deformed configuration

Figure 2.4: Homogenization boundary algorithm step 1 showing the set of blocks that lie on the REV boundary in the underformed configuration.

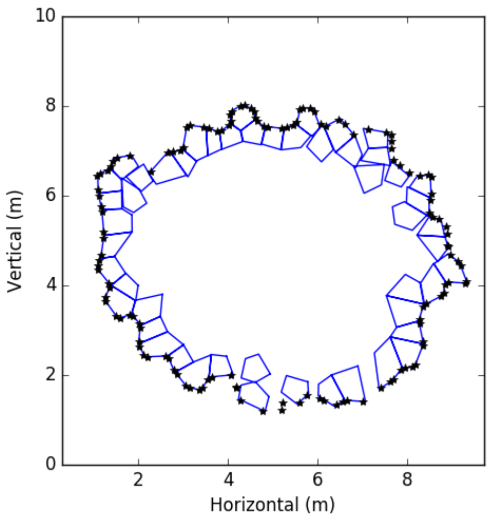


(a) Undeformed configuration (b) Deformed configuration

Figure 2.5: Homogenization boundary algorithm step 2 showing the set of blocks that lie outside the REV boundary in the undeformed configuration.



(a) Undeformed configuration



(b) Deformed configuration

Figure 2.6: Homogenization boundary algorithm step 3 showing the contacts along the homogenization boundary. Note how there are orphaned contacts not associated with the blocks defined in step 1 when deformed.

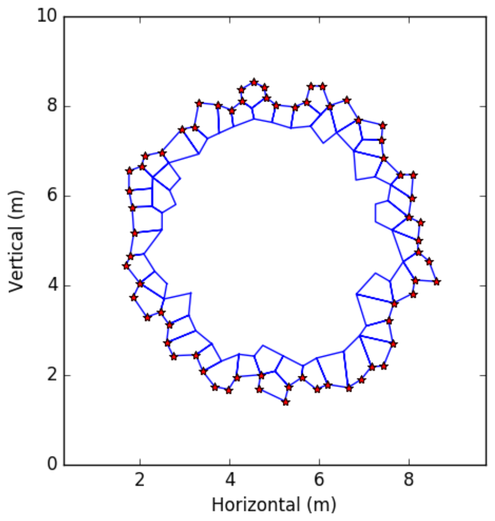
The homogenization boundary, $\Gamma_h$, can be described in terms of $n$ ordered boundary vertices, $V_i^h = (x_i^h, y_i^h)$, representing the $i$-th set of vertex coordinates along the boundary, such that the area of the homogenization domain, $A^h$, can be calculated using the following formulation for the area of an arbitrary, non-self-intersecting polygon [Zwillinger, 1995]:

$$A^h = \frac{1}{2} \sum_{i=1}^{n} x_i^h (y_{i+1}^h - y_{i-1}^h) \tag{2.5}$$

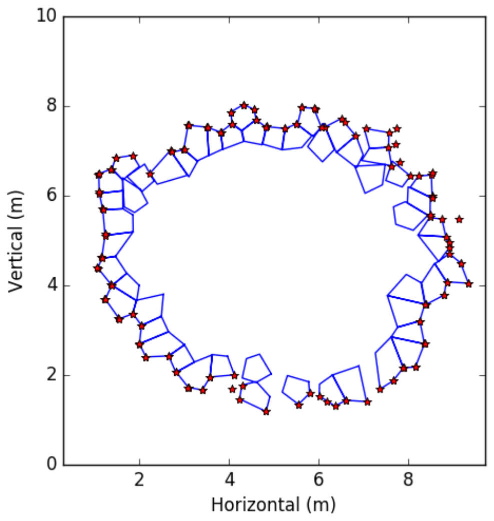
### 2.3.1 Stress Homogenization

The homogenized Cauchy stress, $\langle \boldsymbol{\sigma} \rangle$, is derived from the definition of the spatial average of the microscale stress of the deformable blocks of the DEM simulation, $\boldsymbol{\sigma}^m$, over the homogenization domain $\Omega^h$ [Hill, 1963].

$$\langle \boldsymbol{\sigma} \rangle = \frac{1}{A^h} \int_{\Omega^h} \boldsymbol{\sigma}^m dA \tag{2.6}$$



(a) Undeformed configuration



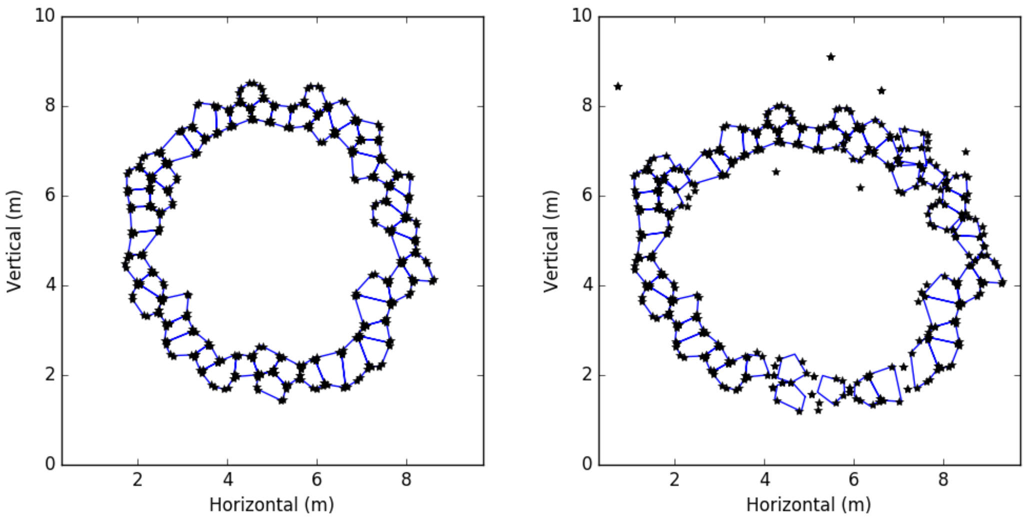
(b) Deformed configuration

Figure 2.7: Homogenization boundary algorithm step 4 showing corners along the homogenization boundary. Again, note how there are orphaned corners not associated with the blocks defined in step 1.

Since the block subdomain is inherently discretized, the integration of the stress over the homogenization domain in can be written as a summation of the average stress in each block. Let $\boldsymbol{\sigma}_I^m$ denote the average stress in block $I$ with an area $A_I$ and let $N^b$ represent the number of blocks in the homogenization domain. The homogenized stress from (2.6) can now be written as:

$$\langle \boldsymbol{\sigma} \rangle = \frac{1}{A^h} \sum_{I=1}^{N^b} \boldsymbol{\sigma}_I^m A_I \tag{2.7}$$

When each deformable block is discretized by constant stress triangles (elements/zones), the integration of the stress over each deformable block subdomain can be written as a summation in the form of a spatially weighted average of the zone stresses. Let $\boldsymbol{\sigma}_{IJ}^m$ denote the stress in zone $J$ of deformable block $I$, $N_I^z$ denote the number of zones within block $I$, and $A_{IJ}$ denote area of zone $J$ in block $I$. The average stress in a block is then defined as:



(a) Undeformed configuration (b) Deformed configuration

Figure 2.8: Homogenization boundary algorithm step 5 showing contacts within the homogenization boundary blocks. Again, Note how there are orphaned contacts not associated with the blocks defined in step 1.

$$\boldsymbol{\sigma}_I^m = \frac{1}{A_I} \sum_{J=1}^{N_I^z} \boldsymbol{\sigma}_{IJ}^m A_{IJ} \tag{2.8}$$

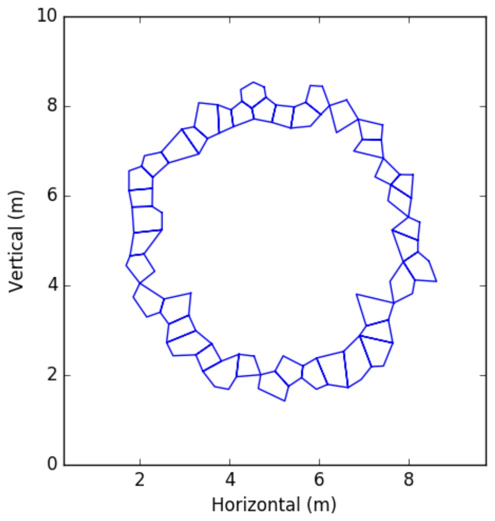
Here, substituting (2.8) into (2.7) allows us to write the homogenized stress in terms of the zone stresses:

$$\langle \boldsymbol{\sigma} \rangle = \frac{1}{A^h} \sum_{I=1}^{N^b} \sum_{J=1}^{N_I^z} \boldsymbol{\sigma}_{IJ}^m A_{IJ} \tag{2.9}$$

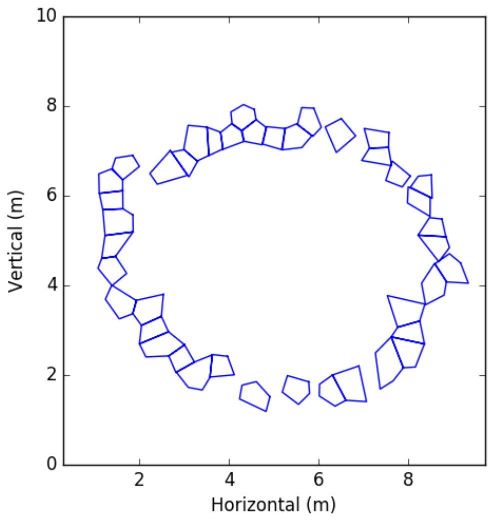
In my implementation, (2.9) is incorporated into the homogenization module.

### 2.3.2 Strain Homogenization

The derivation for the homogenized strain tensor, $\langle \boldsymbol{\epsilon} \rangle$, begins in a similar manner to the homogenized stress tensor derivation with the familiar definition of the spatial average:



(a) Undeformed configuration



(b) Deformed configuration

Figure 2.9: Homogenization boundary algorithm step 6 showing only the boundary blocks that are in contact with the blocks outside the REV domain. Note the differences between this set of blocks and the set of blocks in Figure 2.4.

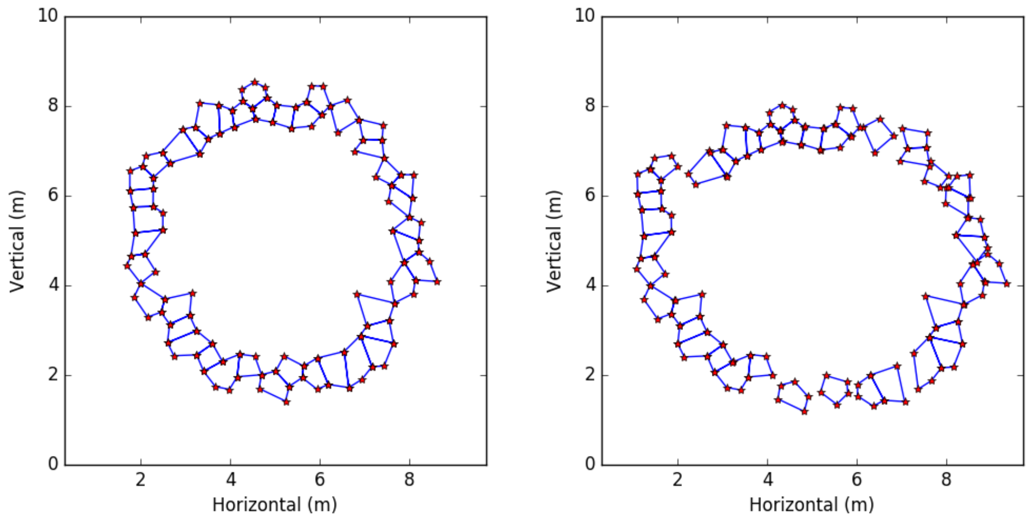
$$\langle \boldsymbol{\epsilon} \rangle = \frac{1}{A^h} \int_{\Omega^h} \boldsymbol{\epsilon}^m dA \tag{2.10}$$

At this point, it becomes convenient to assume a small displacement formulation of strain. This displacement assumption limits the applicability of the strain homogenization, but in the context of large scale geomechanics, this assumption remains reasonable. As such, the linear infinitesimal strain tensor can be written in terms of the displacement vector, $\mathbf{u}^m$:

$$\boldsymbol{\epsilon}^m = \frac{1}{2} \left[ \nabla \mathbf{u}^m + \left( \nabla^\top \mathbf{u}^m \right) \right] \tag{2.11}$$

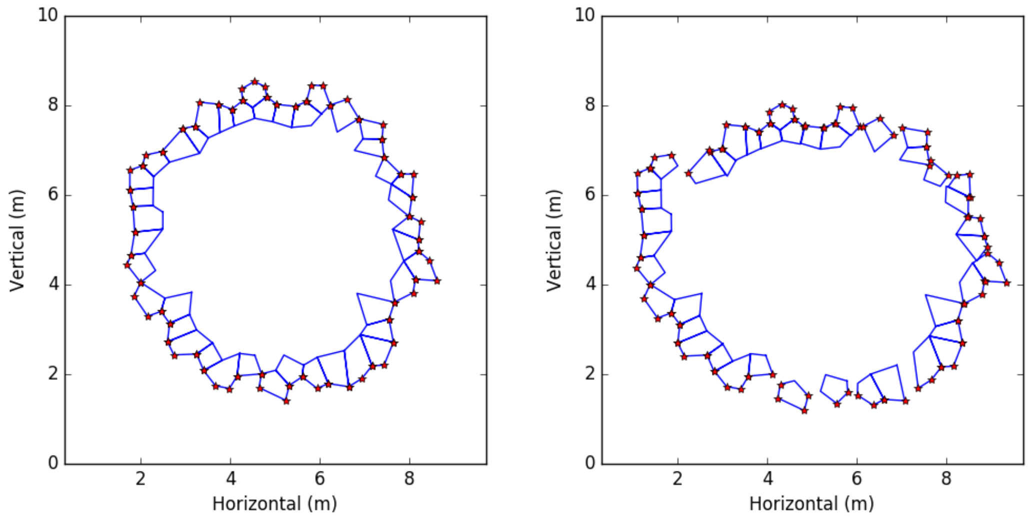
The above integral can be converted to the following boundary integral using the divergence theorem where $\mathbf{n}$ is the outward pointing normal to $\Gamma_h$. :

$$\langle \boldsymbol{\epsilon} \rangle = \frac{1}{2A^h} \oint_{\Gamma^h} \left[ \mathbf{u}^m \otimes \mathbf{n} + \mathbf{n} \otimes \mathbf{u}^m \right] d\Gamma \tag{2.12}$$



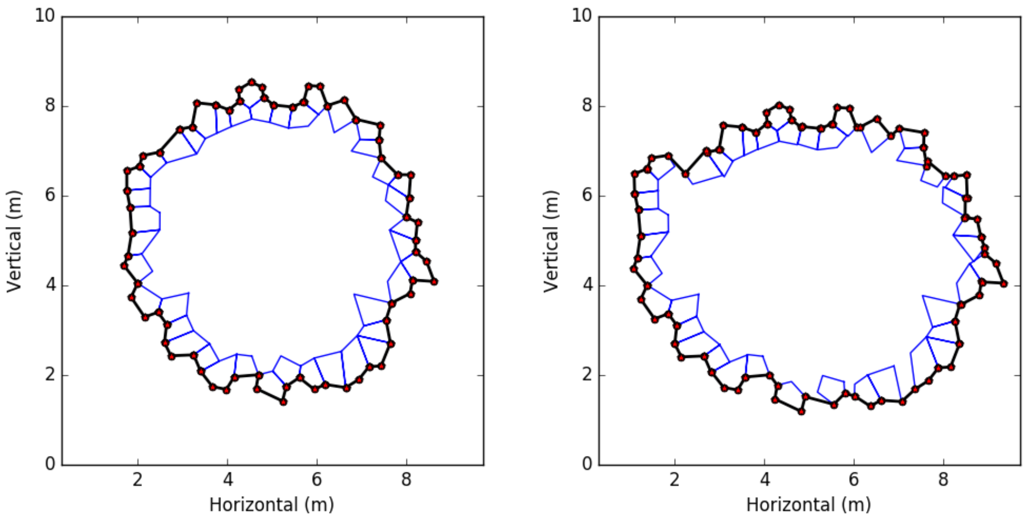
(a) Undeformed configuration (b) Deformed configuration

Figure 2.10: Homogenization boundary algorithm step 7 showing all the corners associated with the boundary blocks defined in step 6.



(a) Undeformed configuration (b) Deformed configuration

Figure 2.11: Homogenization boundary algorithm step 8 showing all the corners on the boundary blocks that lie on the homogenization boundary.



(a) Undeformed configuration

(b) Deformed configuration

Figure 2.12: Homogenization boundary algorithm step 9 showing all the corners from step 7 connected in sequence to form the homogenization domain.

When $\Gamma_h$ is defined by a set of line segments over which the displacement is also linear, the boundary integral can be rewritten as a summation over each of the $N$ boundary segments. Let $\bar{\mathbf{u}}_I^m$ denote the average displacement along the $I^{th}$ boundary segment of the homogenization boundary, which is calculated as the average of the two nodal displacements defining the boundary of each segment. Let $\mathbf{n}_I$ represent the outward pointing normal to the $I^{th}$ boundary segment on the homogenization boundary. Let the length of boundary segment $I$ be denoted by $L_I$. The homogenized strain can be rewritten as

$$\langle \boldsymbol{\epsilon} \rangle = \frac{1}{2A^h} \sum_{I=1}^{N} \left[ \bar{\mathbf{u}}_I^m \otimes \mathbf{n}_I + \mathbf{n}_I \otimes \bar{\mathbf{u}}_I^m \right] L_I \tag{2.13}$$

In my implementation, (2.13) is incorporated into the homogenization module.

## 2.4 Assessment of the REV Size

Homogenizing DEM simulations requires the existence and determination of the REV for the given medium. Generally, the REV of a given domain can be described as the smallest subdomain that is statistically representative of the entire domain [Kanit et al., 2003, Gitman et al., 2007]. This qualitative definition is insufficient to rigorously define an REV as it is subjective with respect to what "statistically representative" means. Hence, the assessment of the REV can be a contentious issue, fraught with ambiguity.

One can conceptualize an REV to be "statistically representative" in two primarily different ways [Drugan and Willis, 1996]. The classically cited means for characterizing an REV suggests that the micro-scale heterogeneities (e.g., fractures, voids, grains, etc.) should be statistically representative within the REV such that the REV should contain a sufficiently large sample of these heterogeneities. This characterization of the REV is potentially problematic when attempting to quantify the REV as the descriptions of these heterogeneities tend to be nominally qualitative, and at best, quasi-quantitative.

The alternative means of conceptualizing "statistically representative", and arguably a more pragmatic way, proposes that the constitutive response of the REV should be statistically representative of the domain. In other words, as one increases the size of a sample domain, the point at which the constitutive response within the domain becomes sensibly constant can be referred to as the REV. Thus, the subdomain constitutive response is quantifiable through resultant model properties and parameters. This REV interpretation has been widely used because of its quantifiability [Kanit et al., 2003, Gitman et al., 2005, Gusev, 1997, Müller et al., 2010], and is adopted for this work.

Here, the aim of this discussion is to present tools to help evaluate the size and existence of the REV for a given microscopically heterogeneous domain. The method of assessing the REV that is presented here is based on testing the statistical significance of the sample distribution parameters. In this method of REV assessment, only one numerical simulation is required if it is of sufficient size. Instead of running the numerical simulations for each realization, subsets of the larger simulation are extracted and analysed independently for their respective constitutive response. This method is far less computationally demanding and correspondingly much faster as a result.

For a given measurement of a material property (e.g., stress, strain, etc.) in an REV, one can assume that it is primarily variant within the heterogeneous domain with respect to two key parameters: the spatial location of the subdomain and the subdomain size. As such, these parameters can be modified to observe the changes in the measured properties of the subdomain. To clarify some terminology, the set of data that corresponds to a single location will be termed a realization. Whereas the set of data that corresponds to a single subdomain size will be referred to as a sample in the statistical sense. Therefore, the statistical population can be interpreted as the set of all possible material property measurements for a given subdomain size as the spatial location of the subdomain changes.

To assess the size of the REV for a given realization, a selection of subdomain sizes is chosen and the material property measurement is evaluated over each subdomain. The material property measurement can be plotted against the corresponding subdomain size as conceptually indicated in Figure 2.13. In this plot, one can see that as the subdomain size increases, the variability in the material property measurement decreases such that it converges to a single value. It is at this convergence that one can say that the subdomain is statistically representative of the whole domain, and therefore represents the REV for that realization. Each realization is spatially static such the spatial variability is not sampled. As such, in order to account for the spatial variability of the material property measurement, multiple realizations are required to accurately define the REV.

## 2.5 Macroscale Constitutive Model

In this section the macroscale stress-strain relationships, $\dot{\boldsymbol{\sigma}}^M = \dot{\boldsymbol{\sigma}}^M \left(\dot{\boldsymbol{\epsilon}}^M, \boldsymbol{\chi}, \mathbf{h}\right)$, used in the validation examples are described. The models are chosen to be complex enough to make the validation of the framework meaningful; however, these are not claimed to be the "best" macroscale models. The framework presented is general and the macroscale constitutive models described here can be replaced in particular applications by different ones. To simplify the discussion and notation in this section, the superscript "$M$" is omitted since all quantities defined describe macroscale behaviour.

Continuum Damage Mechanics (CDM) constitutive models are chosen to represent the NFR at the macroscale. CDM is a branch of continuum mechanics that is concerned with modeling the progressive failure and stiffness degradation in solid materials [Zhang and Cai, 2010]. CDM in this investigation is used to help describe the micro-mechanical degradation of the rock mass due to the nucleation and growth of cracks and voids. This micro-mechanical degradation is represented in a CDM model by using macroscopic state

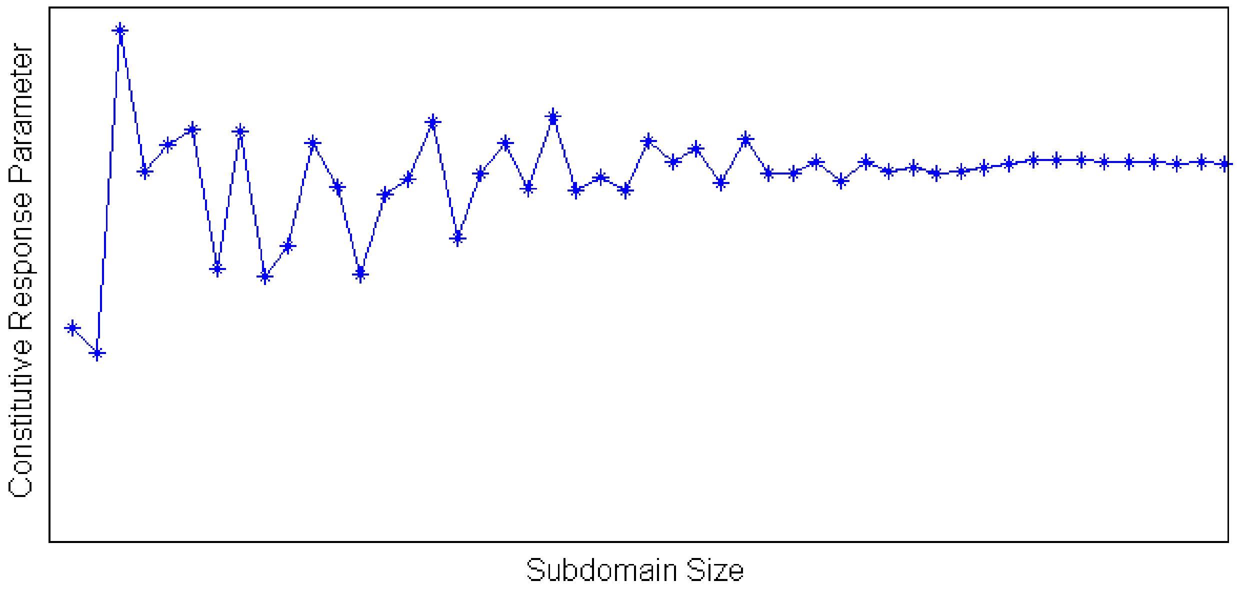


Figure 2.13: Assessing the REV of a heterogeneous domain for a single realization. For very small subdomain sizes, the material property measurement is very sensitive to small fluctuations, whereas at a sufficiently large subdomain size, the material property measurement becomes insensitive to small fluctuations and converges to a single value representative of the whole domain. It is at this convergence that one can say the REV exists.

variables to represent a spatial average of the effects of this degradation [Krajcinovic, 1989]. These state variables used in this context with respect to CDM are known as damage variables.

The damage variables in a CDM model can be described in different capacities. Often, for mathematical and physical simplicity, a single scalar damage variable is used to characterize the state of damage in the material. In this case, the damage variable, $D$, takes a value between 0 and 1 to represent the degree of damage to the material, where $D = 0$ represents a completely undamaged material (original stiffness) and $D = 1$ represents a completely damaged material with no stiffness. A scalar damage description limits the applicability of the CDM model to an isotropically damaged state, which may not be appropriate in some circumstances. More sophisticated CDM models use 2nd and 4th order tensorial representations of the damage variables as well as distinguishing between compressive damage and tensile damage states in order to more accurately characterize anisotropic damage evolution.

Consider the standard elastic relationship described by Hooke's law which relates the stress, $\boldsymbol{\sigma}$, and the elastic strain, $\boldsymbol{\epsilon}^{el}$ through an elastic stiffness tensor, $\mathbf{E}$:

$$\boldsymbol{\sigma} = \mathbf{E} : \boldsymbol{\epsilon}^{el} \tag{2.14}$$

Introducing a scalar damage variable, $D$, to Hooke's law using CDM to describe the stiffness degredation of the material can be shown as:

$$\boldsymbol{\sigma} = (1 - D)\, \mathbf{E} : \boldsymbol{\epsilon}^{el} \tag{2.15}$$

Here, the damaged stiffness of the material, $\mathbf{E}^d$, is described as follows:

$$\mathbf{E}^d = (1 - D)\, \mathbf{E} \tag{2.16}$$

So the constitutive elastic CDM relationship is:

$$\boldsymbol{\sigma} = \mathbf{E}^d : \boldsymbol{\epsilon}^{el} \tag{2.17}$$

In addition to damage, the elasto-plastic behaviour of the rock is also considered. Models that incorporate theories of plasticity and damage mechanics in a unified approach to damage evolution and constitutive relationships are often referred to as damage-plasticity models [Zhang and Cai, 2010]. In general, the constitutive relationship for these damage-plasticity models describes the relationship between the stress, $\boldsymbol{\sigma}$, and the strain, $\boldsymbol{\epsilon}$ as a function of the damage variable, the original elastic stiffness tensor, $\mathbf{E}$, and the plastic strain, $\boldsymbol{\epsilon}^{pl}$:

$$\boldsymbol{\sigma} = \mathbf{E}^d : \left(\boldsymbol{\epsilon} - \boldsymbol{\epsilon}^{pl}\right) \tag{2.18}$$

Here, an additive decomposition of the elastic and plastic strain associated with small deformations is assumed:

$$\boldsymbol{\epsilon} = \boldsymbol{\epsilon}^{el} + \boldsymbol{\epsilon}^{pl} \tag{2.19}$$

In CDM, the notion of effective stress, $\bar{\boldsymbol{\sigma}}$, becomes useful to describe the mechanics of the system, as it refers to the stress that the system would be experiencing without damage. This effective stress can be related to the actual Cauchy stress through the scalar damage variable:

$$\boldsymbol{\sigma} = (1 - D)\,\bar{\boldsymbol{\sigma}} \tag{2.20}$$

### 2.5.1 General Formulation and Assumptions

The plasticity models used are comprised of three key components: the yield function, the flow rule, and the hardening rule. The yield function, $F\left(\bar{\boldsymbol{\sigma}}, \bar{\epsilon}^{pl}\right)$ describes whether or not the material has experienced yield given a particular stress state. The yield function varies between the two models but can be written in general as a function of the effective stress, $\bar{\boldsymbol{\sigma}}$, and equivalent plastic strain, $\dot{\bar{\epsilon}}^{pl}$, expressed through three stress invariants: the Von-Mises equivalent stress, $p\left(\bar{\boldsymbol{\sigma}}\right)$, the hydrostatic stress, $q\left(\bar{\boldsymbol{\sigma}}\right)$, and the third invariant of deviatoric stress, $r\left(\bar{\boldsymbol{\sigma}}\right)$:

$$F = F\left(\bar{\boldsymbol{\sigma}}\right) = F\left(p, q, r\right) \tag{2.21}$$

The Von Mises equivalent stress is:

$$p\left(\bar{\boldsymbol{\sigma}}\right) = \frac{1}{3}\,\mathrm{trace}\left[\boldsymbol{\sigma}\right] \tag{2.22}$$

The hydrostatic stress is:

$$q\left(\bar{\boldsymbol{\sigma}}\right) = \sqrt{\frac{3}{2}}\left[\mathbf{S} : \mathbf{S}\right] \tag{2.23}$$

where $\mathbf{S}$ is the stress deviator and $\mathbf{I}$ is the identity matrix:

$$\mathbf{S} = \boldsymbol{\sigma} + p\left(\bar{\boldsymbol{\sigma}}\right)\mathbf{I} \tag{2.24}$$

The third invariant of the deviatoric stress is:

$$r(\bar{\boldsymbol{\sigma}}) = \left[\frac{9}{2}\mathbf{S}\cdot\mathbf{S}:\mathbf{S}\right]^{\frac{1}{3}} \tag{2.25}$$

In addition, the flow rule describes the amount of plastic deformation that the material exhibits given an applied stress. The flow rule is assumed to be of the following form:

$$\dot{\boldsymbol{\epsilon}}^{pl} = \dot{\lambda}\frac{\partial G(\bar{\boldsymbol{\sigma}})}{\partial \bar{\boldsymbol{\sigma}}} \tag{2.26}$$

Where $\dot{\boldsymbol{\epsilon}}^{pl}$ is the plastic strain rate, $\dot{\lambda}$ is referred to as the plastic consistency parameter, and $G(\bar{\boldsymbol{\sigma}}) = G(p, q, r)$ is the flow potential function. In addition to the yield function and the flow rule, the hardening rule is prescribed to govern the increase/decrease in yield stress as the plastic strain increases. More specifically, the hardening function, $\boldsymbol{h}\left(\bar{\boldsymbol{\sigma}}, \bar{\epsilon}^{pl}\right)$, in these models is used to relate the equivalent plastic strain, $\bar{\epsilon}^{pl}$, to the plastic strain in rate form:

$$\dot{\bar{\epsilon}}^{pl} = \boldsymbol{h}\left(\bar{\boldsymbol{\sigma}}, \bar{\epsilon}^{pl}\right) : \dot{\boldsymbol{\epsilon}}^{pl} \tag{2.27}$$

For the damage models, the damage initiation criteria and evolution equations are different for each material model. In general though, the damage initiation criteria for both material models is strain based and the nature of the damage evolution is assumed to be a function the equivalent plastic strain:

$$D = D\left(\bar{\boldsymbol{\sigma}}, \bar{\epsilon}^{pl}\right) \tag{2.28}$$

### 2.5.2 Drucker-Prager Plasticity Model with Ductile Damage

In this constitutive model, a ductile isotropic damage formulation is prescribed using a modified Johnson-Cook damage initiation criterion and a linear stiffness degradation model. In addition to damage, the elasto-plastic behaviour of the rock is also considered using an extended Drucker-Prager model with a linear yield criterion and a Barcelona hardening function.

The Drucker-Prager plasticity model was developed by Drucker [1950] for modelling frictional materials like granular soils and rock. An important aspect of this plasticity model is the use of a pressure dependent yield criterion to account for the increase in yield stress of geomaterials as the in-situ stresses increase. Specifically, the Drucker-Prager material model is formulated and used for materials with compressive yield strength much greater than the tensile yield strength such as one finds in soils and rocks. However, this material model is intended to simulate the material response under essentially monotonic loading which limits the capacity for modeling cyclic loading.

In addition, the Drucker-Prager model is suitable for using in conjunction with progressive damage and failure models. In this formulation, the Johnson-Cook Damage model is used to model the damage evolution of the rock mass [Johnson and Cook, 1985a]. At a sufficiently large scale, the damage response of NFR can be thought of as behaving in a ductile capacity.

Here, for the extended Drucker-Prager plasticity model, a linear yield function, $F\left(\bar{\boldsymbol{\sigma}}, \bar{\epsilon}^{pl}\right)$, is assumed to be a function of three stress invariants: the Von-Mises equivalent stress, $p\left(\bar{\boldsymbol{\sigma}}\right)$, the hydrostatic stress, $q\left(\bar{\boldsymbol{\sigma}}\right)$, and the third invariant of deviatoric stress, $r\left(\bar{\boldsymbol{\sigma}}\right)$. In addition, the yield function is written in terms of the compressive yield stress, $\sigma_c^y\left(\bar{\epsilon}^{pl}\right)$, which is defined by the hardening function and two material parameters: the friction angle, $\phi$, and a parameter $K$, defined as the ratio of the yield stress in triaxial tension to the yield stress in triaxial compression:

$$F\left(\bar{\boldsymbol{\sigma}}, \bar{\epsilon}^{pl}\right) = \frac{1}{2} q\left(\bar{\boldsymbol{\sigma}}\right)\left[1 + \frac{1}{K} - \left[1 - \frac{1}{K}\right]\left[\frac{r\left(\bar{\boldsymbol{\sigma}}\right)}{q\left(\bar{\boldsymbol{\sigma}}\right)}\right]^3\right] - p\left(\bar{\boldsymbol{\sigma}}\right)\tan\phi - \left[1 - \frac{1}{3}\tan\phi\right]\sigma_c^y\left(\bar{\epsilon}^{pl}\right) \quad (2.29)$$

The flow rule in this formulation is non-associated but the flow potential function, $G\left(\bar{\boldsymbol{\sigma}}\right)$, is written in a very similar form as the yield function with dilation angle, $\psi$, in place of the friction angle. As with the yield function, the flow potential function is written in terms of three stress invariants and two material parameters, dilation angle, and $K$:

$$G\left(\bar{\boldsymbol{\sigma}}\right) = \frac{1}{2} q\left(\bar{\boldsymbol{\sigma}}\right)\left[1 + \frac{1}{K} - \left[]1 - \frac{1}{K}\right]\left[\frac{r\left(\bar{\boldsymbol{\sigma}}\right)}{q\left(\bar{\boldsymbol{\sigma}}\right)}\right]^3\right] - p\left(\bar{\boldsymbol{\sigma}}\right)\tan\psi \quad (2.30)$$

In addition to the yield function and the flow rule, the hardening rule is assumed to take the form of the Barcelona model [Lubliner et al., 1989]. The Barcelona model allows for material hardening before softening and approaches a yield stress of 0 as the plastic strain increases. This form of the hardening function can be written in terms of three material parameters, initial compressive yield strength $\sigma_c^{iy}$, $\alpha$, and $\beta$:

$$\sigma_c = \sigma_c^{iy}\left[\left[1+\alpha\right]e^{-\beta\bar{\epsilon}^{pl}} - \alpha e^{-2\beta\bar{\epsilon}^{pl}}\right] \tag{2.31}$$

The damage initiation criterion for this material model is based on the Johnson-Cook model of ductile damage initiation [Johnson and Cook, 1985a]. The standard Johnson-Cook model assumes the equivalent plastic strain when damage is initiated, $\bar{\epsilon}_f^{pl}\left(\eta\right)$, is a function of triaxiality, $\eta$, and is written in terms of five material parameters:

$$\bar{\epsilon}_f^{pl}\left(\eta, \dot{\bar{\epsilon}}^{pl}, \hat{T}\right) = \left[D_1 + D_2 e^{D_3\eta}\right]\left[1 + D_4 \ln\left(\frac{\dot{\bar{\epsilon}}^{pl}}{\dot{\bar{\epsilon}}}\right)\right]\left[1 + D_5\hat{T}\right] \tag{2.32}$$

However, assuming isothermal conditions, neglecting rate effects, and assuming a simplified form of the exponential relationship, the initiation criterion can be reduced to two material parameters, $D_2$ and $D_3$:

$$\bar{\epsilon}_f^{pl}\left(\eta\right) = D_2 e^{D_3\eta} \tag{2.33}$$

After the material has experienced yield and material damage has occurred, the stress-strain relationship becomes strongly mesh-dependent because of strain localization due to the energy dissipation decreasing as the mesh is refined. As such, Hillerborg et al. [1976] proposed a stress-displacement response based on fracture energy after damage initiation assuming that evolving damage is a linear degradation of the material stiffness in compression. Assuming a linear form, the effective plastic displacement when the material is completely damaged, $\bar{u}_f^{pl}$, can be specified, and the damage evolution can then be written in terms of the effective plastic displacement, $\bar{u}^{pl}$:

$$\dot{D} = \frac{\dot{\bar{u}}^{pl}}{\bar{u}_f^{pl}} \tag{2.34}$$

### 2.5.3 Damage-Plasticity Model for Quasi-Brittle Materials

In this constitutive model, a quasi-brittle isotropic damage formulation is prescribed using a linear stiffness degradation model that accounts for cyclic loading. In addition to damage, the elasto-plastic behaviour of the rock is also considered using Lubliner's plasticity model with a linear yield criterion and a Barcelona hardening function [Lubliner et al., 1989].

This damage-plasticity model was developed by Lubliner et al. [1989] as a plasticity based damage model for non-linear analysis of concrete failure. Subsequently, Lee and Fenves [1998] further developed the model to facilitate cyclic loading by adding a second damage variable and introducing a new yield function to account for the additional damage variable.

This model was specifically formulated for modeling quasi-brittle materials under low confining stresses subject to cyclic loading. In addition to the separate damage variables governing the stiffness degradation, the stiffness recovery and material hardening/softening is also treated separately in both compression and tension. Because the formulation does not consider the effects of large hydrostatic stresses, the applicability of this plasticity model to in-situ geomechanics at depth may not be sufficiently accurate. As such, this model is more appropriate for shallow geological models that require cyclic loading paths to be considered.

The yield function for this model is based on the yield function proposed by Lee and Fenves [1998] which was developed to allow for differential hardening under tension and compression. The resultant yield function, $F\left(\bar{\boldsymbol{\sigma}}, \bar{\epsilon}^{pl}\right)$, can be expressed in terms of two stress invariants: the Von-Mises equivalent stress, $p\left(\bar{\boldsymbol{\sigma}}\right)$, the hydrostatic stress, $q\left(\bar{\boldsymbol{\sigma}}\right)$:

$$F\left(\bar{\boldsymbol{\sigma}}, \bar{\epsilon}^{pl}\right) = \frac{1}{1-A}\left[q\left(\bar{\boldsymbol{\sigma}}\right) - 3Ap\left(\bar{\boldsymbol{\sigma}}\right) + B\left(\bar{\epsilon}^{pl}\right)\left\langle \hat{\bar{\boldsymbol{\sigma}}} \right\rangle - \gamma\left\langle -\hat{\bar{\boldsymbol{\sigma}}} \right\rangle\right] - \bar{\sigma}_c\left(\bar{\epsilon}_c^{\,pl}\right) \tag{2.35}$$

Where $A$ and $\gamma$ are dimensionless material constants. Experimental testing has yielded values of $A$ between 0.08 and 0.12, as well as a typical $\gamma$ value of approximately 3 [Lubliner et al., 1989]. The hat notation, $\hat{\bar{\boldsymbol{\sigma}}}$, for an arbitrary stress tensor, $\bar{\boldsymbol{\sigma}}$, represents the algebraically maximum eigenvalue, or, the maximum principle stress. In addition, $\langle\rangle$ are Macauley brackets and can be defined as:

$$\langle x \rangle = \frac{1}{2}\left(|x| + x\right) \tag{2.36}$$

The flow rule in this formulation is non-associated which means that the flow potential function, $G(\bar{\boldsymbol{\sigma}})$, follows a different form than the yield function. Like with the yield function, the flow potential function, is written in terms of two stress invariants, $p(\bar{\boldsymbol{\sigma}})$ and $q(\bar{\boldsymbol{\sigma}})$, and two material parameters, dilation angle, $\psi$, and eccentricity $\varepsilon$:

$$G(\bar{\boldsymbol{\sigma}}) = \sqrt{\left[\varepsilon\sigma^{iy}\tan\psi\right]^2 + q(\bar{\boldsymbol{\sigma}})^2} - p(\bar{\boldsymbol{\sigma}})\tan\psi \tag{2.37}$$

In this formulation of damage-plasticity, the brittle nature of rock necessitates separate characterization of tensile and compressive damage. With quasi-brittle materials such as rock, it has been found that compressive stiffness can be recovered upon crack closure. Conversely, in these materials, tensile stiffness is not recovered after compressive cracks have developed. This behaviour implies that two separate scalar damage values should exist for the given system to account for both the compressive stiffness degradation and the tensile stiffness degradation. As such, the equivalent plastic strain is also considered separately for tension ($\bar{\epsilon}_t^{pl}$) and compression ($\bar{\epsilon}_c^{pl}$) and is represented as follows:

$$\bar{\epsilon}^{pl} = \begin{bmatrix} \bar{\epsilon}_t^{pl} \\ \bar{\epsilon}_c^{pl} \end{bmatrix} \tag{2.38}$$

The hardening rule for this model is slightly modified from (2.27) to accommodate two hardening variables (equivalent plastic strains) for tension and compression. The hardening rule can thus be written in matrix form:

$$\mathbf{h}\left(\bar{\boldsymbol{\sigma}}, \bar{\epsilon}^{pl}\right) = \begin{bmatrix} r\left(\hat{\bar{\sigma}}_{ij}\right)\frac{\sigma_t\left(\bar{\epsilon}_t^{pl}\right)}{g_t} & 0 & 0 \\ 0 & 0 & -\left(r\left(\hat{\bar{\sigma}}_{ij}\right) - 1\right)\frac{\sigma_c\left(\bar{\epsilon}_c^{pl}\right)}{g_c} \end{bmatrix} \tag{2.39}$$

Where $\sigma_t$ and $\sigma_c$ are the yield stresses in tension and compression as specified by the hardening curves which describe the evolution of the equivalent plastic strains. The compressive hardening rule is approximated here using the Barcelona model in a similar capacity as was done for the Drucker-Prager model in the previous section. The only difference here being that the hardening is defined in terms of the inelastic strain, $\bar{\epsilon}^{in}$, rather than the plastic strain before:

$$\sigma_c = \sigma_c^{iy}\left[\left[1+\alpha\right]e^{-\beta\bar{\epsilon}^{in}} - \alpha e^{-2\beta\bar{\epsilon}^{in}}\right] \tag{2.40}$$

There exists a subtle but important distinction between these two strain measurements when considering CDM. The plastic strain refers to all the strain that is non-elastic (i.e. the remaining strain after the applied stress is unloaded in the damaged state), while the inelastic strain refers to the theoretical plastic strain that would remain if the material was unloaded with the original material stiffness (i.e. in an undamaged state).

The tensile hardening function has a fundamentally different behavior than the compressive hardening function, and is therefore approximated using an exponential function. This function is described by the initial tensile yield stress, $\sigma_t^{iy}$, and a decay parameter, $\lambda$. These parameters describe the relationship between the tensile yield stress, $\sigma_t$, and the cracking strain, $\bar{\epsilon}^{ck}$ which is the tensile portion of inelastic strain:

$$\sigma_t\left(\bar{\epsilon}^{ck}\right) = \sigma_t^{iy} e^{\lambda \bar{\epsilon}^{ck}} \tag{2.41}$$

In addition, $g_t$ and $g_c$ from (2.39) represent the dissipated fracture energy density during micro-cracking. The use of the dissipated fracture energy density over fracture energy (a material property) stems from the fact that the strain softening part of the stress-strain curve cannot represent a local physical property of the material in addition to being highly mesh sensitive. The dissipated fracture energy densities are defined in terms of a characteristic length, $l$, associated with the mesh size and the fracture energy in tension, $G_t$, and compression, $G_c$:

$$g_t = \frac{G_t}{l} \tag{2.42}$$

$$g_c = \frac{G_c}{l} \tag{2.43}$$

Furthermore, the weighting function, $r\left(\hat{\bar{\boldsymbol{\sigma}}}\right)$, weights the hardening functions depending on the degree of tension or compression that the model is experiencing:

$$r\left(\hat{\bar{\boldsymbol{\sigma}}}\right) = \frac{\sum_{i=1}^{3}\left\langle \hat{\bar{\boldsymbol{\sigma}}}_i \right\rangle}{\sum_{i=1}^{3}\left| \hat{\bar{\boldsymbol{\sigma}}}_i \right|}, \qquad 0 \leq r\left(\hat{\bar{\boldsymbol{\sigma}}}\right) \leq 1 \tag{2.44}$$

Loading a quasi-brittle material in compression or tension causes damage in the material, which reduces the effective stiffness, weakening the unloading response. This damage is characterized by two damage variables, one of which represents the damage due to tensile loading, $D_t$, the other represents damage due to compressive loading, $D_c$.

$$D_t = D_t\left(\bar{\epsilon}_t^{pl}\right), \qquad 0 \leq D_t \leq 1 \tag{2.45}$$

$$D_c = D_c\left(\bar{\epsilon}_c^{pl}\right), \qquad 0 \leq D_c \leq 1 \tag{2.46}$$

The damage in both compression and tension is a necessarily increasing function of the equivalent plastic strains. For cyclic loading, both the compressive and tensile damage need to be considered. Two stiffness recovery factors are introduced, $s_t$ and $s_c$, which represent the stiffness recovery effects associated with stress reversals. The damage takes the form of:

$$[1 - D] = [1 - s_t D_c]\,[1 - s_c D_t], \qquad 0 \leq s_t, s_c \leq 1 \tag{2.47}$$

## 2.6 Parameter Estimation Algorithms

Parameter estimation involves a process of obtaining a parameter set $\boldsymbol{\chi}$ of a CDM model that minimizes the difference between the model response($\boldsymbol{\sigma}^M$-$\boldsymbol{\epsilon}^M$) and the measured system response ($\langle\boldsymbol{\sigma}\rangle$-$\langle\boldsymbol{\epsilon}\rangle$) for all load paths. Herein, the parameter estimation was conducted with optimization methods which attempts to minimize a least-squares objective function. Optimization algorithms are often described as either deterministic, which find the same optimum each time, or stochastic, which may find different answers each time because of introduced randomness. Additionally, optimization algorithms can also be described as either heuristic, which find an approximate optimum, or exact, which find the precise optimum.

Deterministic optimization algorithms are often used to focus on searching for the optima within the local parameter space by iteratively converging towards a consistent solution because finding a global optima deterministically can be computationally prohibitive, depending on the complexity of the problem. Stochastic optimization algorithms can allow for more computationally effective exploration of the global parameter space by introducing some randomness in the solution, allowing for arbitrary exploration of the parameter space.

Heuristic techniques are useful for highly non-linear problems, where there are numerous local optima within the parameter space. When searching the global parameter-space exactly becomes too computationally demanding, heuristic methods are used, at the cost of completeness and accuracy. A compromise between speed and accuracy can be obtained by strategically using different types of algorithms.

A combination of two optimization algorithms is used to assess the optimal parameter set. An initial stochastic heuristic algorithm is applied to search for the approximate global optima, followed by a deterministic algorithm as a local refinement of the optimal parameter set. Particle Swarm Optimization (PSO) is used for the global heuristic search, and the Levenberg-Marquardt Algorithm (LMA) is used for the local deterministic search.

Let $s_j$ represent any arbitrary scalar stress component at any given simulation time where $j$ represents the stress measurement number. Consider any component of a homogenized stress tensor, $\langle s \rangle_j$, which acts as the target solution for the macroscale model. The same stress measurement in the macroscale model, $s_j^M(\boldsymbol{\chi}, \langle e \rangle)$, can be expressed as a function of the parameter set for the continuum constitutive model, $\boldsymbol{\chi}$, containing $n$ number of parameters, and the homogenized strain at the given load step, $\langle e \rangle$. Here, the weighted least-squares objective function to be minimized, $\Psi$, can be written in terms of a weighting parameter, $w_j$, for $m$ number of stress measurements:

$$\Psi = \sum_{j=1}^{m} \left[ w_j \left[ \langle s \rangle_j - s_j^M(\boldsymbol{\chi}, \langle e \rangle) \right] \right]^2 \tag{2.48}$$

The aim of the optimization algorithms is to minimize $\Psi$ with respect to $\boldsymbol{\chi}$, subject to the constraint that all parameters values within $\boldsymbol{\chi}$ are realistic and fall within specified bounds.

The parameter estimation algorithms work by iteratively running a single element CDM model, subject to boundary conditions provided by the homogenized DEM simulations, with successive parameter sets that intelligently adapt in order to minimize $\Psi$. The approach used here is similar to the Least Squares method described briefly in Marquardt [1963].

### 2.6.1 Particle Swarm Optimization (PSO)

The PSO algorithm is a heuristic optimization algorithm that was developed by Kennedy and Eberhart [1995] as a by-product of modeling the cooperative-competitive nature of social behaviour in birds as they flocked searching for food. The PSO algorithm, in a conceptual sense, consists of a series of 'particles' (birds) which 'swarm' through the entire parameter space (sky) searching for the global optima (food) using a combination of individual 'particle' knowledge and global 'swarm' (flock) knowledge.

Consider a particle in an $n$ dimensional parameter space with an arbitrary velocity, $\vec{v}_i$, and position, $\vec{x}_i$. At each position in this parameter space, $\Psi$ can be evaluated with an attempt to find the position that minimizes $\Psi$. The motion of this particle allows the exploration of the parameter space to find a minimum value of $\Psi$ and the corresponding position. After a period of time, $\Delta t$, the new position of the particle, $\vec{x}_{i+1}$, can be written as a combination of the old position and the new velocity, $\vec{v}_{i+1}$.

$$\vec{x}_{i+1} = \vec{x}_i + \vec{v}_{i+1}\Delta t \tag{2.49}$$

Here, $\vec{v}_{i+1}$, is considered to be influenced by $\vec{v}_i$, the position of the current local optimum, $p_l$, and the position of the current global optimum, $p_g$. The local optimum refers to the minimum value of $\Psi$ observed by the individual particle, while the global optimum refers to the minimum value of $\Psi$ observed by all particles. As such, $\vec{v}_{i+1}$ is written as a linear combination of $\vec{v}_i$, the velocity required to move the particle back to the local optimum and the velocity required to move the particle back to global optimum. In order to prevent the algorithm from oscillating indefinitely in a predictable manner, a randomization vector, $\vec{U}\left(\phi\right)$, of length $n$ is introduced to provide coefficients between 0 and $\phi$ to the velocity vectors. The operator $\odot$ refers to the Hadamard product (element-wise multiplication):

$$\vec{v}_{i+1} = \vec{v}_i + \frac{\vec{U}_i\left(\phi_1\right) \odot \left[\vec{p}_l - \vec{x}_i\right] + \vec{U}_i\left(\phi_2\right) \odot \left[\vec{p}_g - \vec{x}_i\right]}{\Delta t} \tag{2.50}$$

As one would expect, the behaviour of the PSO is highly sensitive to the chosen values of $\phi_1$ and $\phi_2$. If these parameters are too small, then the optimization becomes "unresponsive" such that the initial velocity is maintained and successive iterations do not have the capacity to appreicably change their velocity to search the parameter space effectively. Alternatively, if these parameters are too large, the PSO has the capacity to become unstable such that the particle speeds keep increasing on successive iterations. Commonly accepted in most PSO algorithms is the assumption that $\phi_1 = \phi_2 = 2$. To overcome these limitations and control the scope of the search, Shi and Eberhart [1998] introduced the inertial weight term, $\omega$:

$$\vec{v}_{i+1} = \omega\vec{v}_i + \frac{\vec{U}_i(\phi_1) \odot [\vec{p}_l - \vec{x}_i] + \vec{U}_i(\phi_2) \odot [\vec{p}_g - \vec{x}_i]}{\Delta t} \tag{2.51}$$

This inertial weight acts as a scalar multiplier between 0 and 1 for $\vec{v}_i$, and can be interpreted as a measure of the fluidity of the system. A large inertial term allows the particle to maintain its current velocity to a higher degree indicating a system with low viscosity lending to a more explorative search, while a small inertial term dissipates the particles velocity more rapidly indicating a more viscous system which favours exploitative searching.

Additional damping can be provided in the form of a constriction coefficient which controls the convergence of the particle, by ensuring convergence and preventing explosion. Clerc and Kennedy [2002] noted that many means of constricting the velocity function exist, but provided a simple form of the constriction, using a constriction coefficient, $\zeta(\phi_1, \phi_2)$.

$$\vec{v}_{i+1} = \zeta(\phi_1, \phi_2)\left[\omega\vec{v}_i + \frac{\vec{U}_i(\phi_1) \odot [\vec{p}_l - \vec{x}_i] + \vec{U}_i(\phi_2) \odot [\vec{p}_g - \vec{x}_i]}{\Delta t}\right] \tag{2.52}$$

Here, $\zeta(\phi_1, \phi_2)$ is given by the following, where $\phi = \phi_1 + \phi_2$:

$$\zeta(\phi_1, \phi_2) = \frac{2}{\phi - 2 + \sqrt{\phi^2 - 4\phi}} \tag{2.53}$$

In addition, with the desire to give more credence to either the local optimum or the global optimum, two "trust" parameters are introduced. $c_1$ is referred to as the cognitive parameter as it weights the particles own experience, while the second parameter, $c_2$, is referred to as the social parameter as it weights the influence of the combined experience of the swarm:

$$\vec{v}_{i+1} = \zeta(\phi_1, \phi_2)\left[\omega\vec{v}_i + \frac{c_1\vec{U}_i(\phi_1) \odot [\vec{p}_l - \vec{x}_i] + c_2\vec{U}_i(\phi_2) \odot [\vec{p}_g - \vec{x}_i]}{\Delta t}\right] \tag{2.54}$$

In terms of the up-scaling application, the position of the particle can be taken to represent an estimate of the optimal parameter set, $\boldsymbol{\chi}_i$, while the velocity of the particle represents the direction and magnitude of the change in the parameter set estimate for the next iteration $\Delta\boldsymbol{\chi}_{i+1}$. Furthermore, the time step is considered to be a unit iteratation step, which allows for (2.54) to be abstracted as follows in the context of up-scaling DEM simulations.

$$\Delta\boldsymbol{\chi}_{i+1} = \zeta\left(\phi_1, \phi_2\right)\left[\omega\Delta\boldsymbol{\chi}_i + c_1\vec{U}_i\left(\phi_1\right)\odot\left[\boldsymbol{\chi}_l - \boldsymbol{\chi}_i\right] + c_2\vec{U}_i\left(\phi_2\right)\odot\left[\boldsymbol{\chi}_g - \boldsymbol{\chi}_i\right]\right] \tag{2.55}$$

Where $\boldsymbol{\chi}_l$ is the parameter set corresponding to the minimum value of $\Psi$ for the particle, and $\boldsymbol{\chi}_g$ is the parameter set corresponding to the minimum value of $\Psi$ for all particles. In addition, (2.49) can be rewritten in a similar capacity:

$$\boldsymbol{\chi}_{i+1} = \boldsymbol{\chi}_i + \Delta\boldsymbol{\chi}_{i+1} \tag{2.56}$$

For a particle swarm containing an arbitrary number of particles, the optimal parameter set of the system, $\boldsymbol{\chi}$, is considered to be $\boldsymbol{\chi}_g$ after a specified number of iterations, or once all the particles converge to a stable solution. The general PSO algorithm as described here is summarized as follows:

1. Assign particles random positions and velocities in the parameter space
2. Move each particle with (2.55) and (2.56)
3. For each particle, revise $\boldsymbol{\chi}_l$ if new local optimum found
4. Revise $\boldsymbol{\chi}_g$ if new global optimum found
5. If current iteration is greater than the maximum number of iterations or solution is stable, iteration is complete. Otherwise, go to step 2

### 2.6.2 Asynchronous Parallel PSO (APPSO)

The PSO described above possesses a large number of attractive qualities for parameter estimation which make it desirable for up-scaling. However, the main drawback of the PSO is the large computational costs in terms of total elapsed time primarily due to the fact that the algorithm was originally designed for a serial implementation. The serial implementation, although effective in its own right, can be dramatically improved through parallelization.

The nature of the PSO lends itself to a fairly trivial implementation of a synchronous parallelization scheme which does not require changing the nature of the algorithm. Here, since all the particles at each iteration are treated independently, the updated positions and corresponding objective functions can be computed in parallel. This parallelization scheme waits for all the particles to complete their analysis before moving on to the next iteration. As a result, the parallel efficiency is often compromised due to processors having to wait for the final particle(s) to finish their analysis. This idleness of the processors can be caused by having a swarm size that is not an integer multiple of the number of processors, having a heterogeneous computing environment where processors have different computational speeds, or having a numerical simulation that requires different amount of computational time depending on the input parameters [Venter and Sobieszczanski-Sobieski, 2006]. This inefficiency of this synchronous parallelization increases as the number of processors increases due to the increasing number of idle processors towards the end of the iteration.

To overcome these parallel inefficiencies, Venter and Sobieszczanski-Sobieski [2006] introduced an Asynchronous Parallel Particle Swarm Optimization (APPSO) algorithm to continuously use the available processors with the goal of having no idle processors from one iteration to the next. The key here, is to separate the update calculations associated with each point and those associated with the swarm as a whole. Normally, in the synchronous scheme, the update calculations (i.e., updating $\boldsymbol{\chi}_l$ and $\boldsymbol{\chi}_g$) are done at the end of each iteration. In this asynchronous scheme, updating $\boldsymbol{\chi}_l$ remains the same, but updating $\boldsymbol{\chi}_g$ uses the best position from the previous iteration instead of the current iteration in order perform the update calculations immediately and allow the analyses to proceed without waiting for the rest of the particles to complete their analysis.

### 2.6.3 Levenburg-Marquardt Algorithm (LMA)

Levenberg-Marquardt Algorithm (LMA) is a deterministic optimization algorithm that was proposed by Marquardt [1963] which builds on the work of Levenberg [1944]. This calibration algorithm combines a quasi-Newton approach with a conjugate gradient technique in order to efficiently minimize non-linear least-squares problems.

For notational simplicity (2.48) is rewritten in matrix form by considering a vector containing $m$ number of arbitrary homogenized stress measurements at any given time, $\langle \mathbf{s} \rangle$, and a corresponding vector of stress measurements for the macroscale model, $\mathbf{s}^M$:

$$\langle \mathbf{s} \rangle = \left[\langle s \rangle_1, \langle s \rangle_2, \ldots, \langle s \rangle_m\right]^T \tag{2.57}$$

$$\mathbf{s}^M = \left[s_1^M, s_2^M, \ldots, s_m^M\right]^T \tag{2.58}$$

This vectorization facilitates writing the weighted least-squares objective function in matrix form using $\mathbf{Q}$ as a weighting matrix:

$$\Psi = \left[\langle \mathbf{s} \rangle - \mathbf{s}^{\mathbf{M}}\right]^T \mathbf{Q} \left[\langle \mathbf{s} \rangle - \mathbf{s}^{\mathbf{M}}\right] \tag{2.59}$$

Where $\mathbf{Q}$ is written as a diagonal matrix containing the weighting parameters:

$$\mathbf{Q} = \begin{bmatrix} w_1^2 & 0 & \ldots & 0 \\ 0 & w_2^2 & & 0 \\ \vdots & & \ddots & \vdots \\ 0 & 0 & \ldots & w_m^2 \end{bmatrix} \tag{2.60}$$

The first step in the LMA formulation considers an arbitrary initial set of parameters, $\boldsymbol{\chi}_0$, and the corresponding macroscale stress measurements, $\mathbf{s}_0^M$. The relationship between $\boldsymbol{\chi}$ and $\mathbf{s}^M$ is generally highly non-linear, so the function is approximated with a taylor series expansion about $\boldsymbol{\chi}_0$, yielding the following linearization:

$$\mathbf{s}^M \approx \mathbf{s}_0^M + \mathbf{J}\left[\boldsymbol{\chi} - \boldsymbol{\chi}_0\right] \tag{2.61}$$

Where the partial derivatives of the given set of $s^M$ stress measurements with respect to the parameters in $\boldsymbol{\chi}$ are represented in the Jacobian matrix, $\mathbf{J}$:

$$\mathbf{J} = \begin{bmatrix} \frac{\partial s_1^M}{\partial \boldsymbol{\chi}_1} & \frac{\partial s_1^M}{\partial \boldsymbol{\chi}_2} & \cdots & \frac{\partial s_1^M}{\partial \boldsymbol{\chi}_n} \\ \frac{\partial s_2^M}{\partial \boldsymbol{\chi}_1} & \frac{\partial s_2^M}{\partial \boldsymbol{\chi}_2} & & \frac{\partial s_2^M}{\partial \boldsymbol{\chi}_n} \\ \vdots & & \ddots & \vdots \\ \frac{\partial s_m^M}{\partial \boldsymbol{\chi}_1} & \frac{\partial s_m^M}{\partial \boldsymbol{\chi}_2} & \cdots & \frac{\partial s_m^M}{\partial \boldsymbol{\chi}_n} \end{bmatrix} \tag{2.62}$$

Substituting (2.61) into (2.59) gives an linearized approximation of the objective function:

$$\Psi \approx \left[\langle \mathbf{s} \rangle - \left[\mathbf{s}_0^M + \mathbf{J}\left[\boldsymbol{\chi} - \boldsymbol{\chi}_i\right]\right]\right]^T \mathbf{Q}\left[\langle \mathbf{s} \rangle - \left[\mathbf{s}_0^M + \mathbf{J}\left[\boldsymbol{\chi} - \boldsymbol{\chi}_0\right]\right]\right] \tag{2.63}$$

Since (2.63) is still an approximation of the objective function, an iterative approach is required to converge to an optimal estimate of $\boldsymbol{\chi}$. Here, (2.63) can be modified and written in an iterative capacity such that the current parameter set estimate, $\boldsymbol{\chi}_i$, simulated stress measurements, $\mathbf{s}_i^M$, objective function value, $\Psi_i$, and Jacobian matrix, $\mathbf{J}_i$ can be used to estimate the next parameter set estimate $\boldsymbol{\chi}_{i+1}$:

$$\Psi_i = \left[\langle \mathbf{s} \rangle - \left[\mathbf{s}_i^M + \mathbf{J}_i\left[\boldsymbol{\chi}_{i+1} - \boldsymbol{\chi}_i\right]\right]\right]^T \mathbf{Q}\left[\langle \mathbf{s} \rangle - \left[\mathbf{s}_i^M + \mathbf{J}_i\left[\boldsymbol{\chi}_{i+1} - \boldsymbol{\chi}_i\right]\right]\right] \tag{2.64}$$

The vector $\left[\boldsymbol{\chi}_{i+1} - \boldsymbol{\chi}_0\right]$, which represents the difference in the current estimate of the parameter set and the next parameter set estimate is termed the upgrade vector. By taking the derivative of $\Psi$ with respect to $\boldsymbol{\chi}$, the upgrade vector can be written as:

$$\left[\boldsymbol{\chi}_{i+1} - \boldsymbol{\chi}_i\right] = \left[\mathbf{J}_i^T \mathbf{Q} \mathbf{J}_i\right]^{-1} \mathbf{J}_i^T \mathbf{Q}\left[\langle \mathbf{s} \rangle - \mathbf{s}_i^M\right] \tag{2.65}$$

Here, for convenience, the upgrade vector is written as $\mathbf{U}_i = \left[\boldsymbol{\chi}_{i+1} - \boldsymbol{\chi}_i\right]$. Since (2.65) is still an approximation of the upgrade vector, an iterative approach to finding $\boldsymbol{\chi}$ is required. The LMA further proposes that $\boldsymbol{U}_i$ be modified with the Marquardt parameter, $\alpha$:

$$\mathbf{U}_i = \left[\mathbf{J}_i^T \mathbf{Q} \mathbf{J}_i + \alpha \mathbf{I}\right]^{-1} \mathbf{J}_i^T \mathbf{Q}\left[\langle \mathbf{s} \rangle - \mathbf{s}_i^M\right] \tag{2.66}$$

Here, $\mathbf{I}$ is the $n \times n$ identity matrix. The Marquardt parameter in (2.66) allows $\mathbf{U}_i$ to approximate a Steepest-Descent Method (SDM) for large values of $\alpha$, while using a Taylor Series Approximation (TSA) for small values of alpha. This formulation allows for a smooth transition between a SDM when the parameter set estimate is far away from the optimal parameter set, and a TSA when the parameter set estimate is close to the optimal parameter set.

Another key development in the LMA notes that for many calibration and parameter estimation problems, elements within $\mathbf{s}_i^M$ and $\langle \mathbf{s} \rangle$ may differ by several orders of magnitude. Such large variations can lead to significant roundoff error during the calculation of $\mathbf{J}_i$. This error can be avoided by introducing a scaling matrix, $\mathbf{S}_i$:

$$\mathbf{S}_i = \begin{bmatrix} \frac{1}{J_{i_{11}} w_1} & 0 & \cdots & 0 \\ 0 & \frac{1}{J_{i_{22}} w_2} & & 0 \\ \vdots & & \ddots & \vdots \\ 0 & 0 & \cdots & \frac{1}{J_{i_{nn}} w_n} \end{bmatrix} \tag{2.67}$$

With (2.67), the upgrade vector in (2.66) can be rewritten in a mathematically identical way while avoiding numerical errors:

$$\mathbf{U}_i = \mathbf{S}_i \left[ \mathbf{S}_i^T \mathbf{J}_i^T \mathbf{Q} \mathbf{J}_i \mathbf{S}_i + \alpha \mathbf{S}_i^T \mathbf{S}_i \right]^{-1} \mathbf{S}_i^T \mathbf{J}_i^T \mathbf{Q} \left[ \langle \mathbf{s} \rangle - \mathbf{s}_i^M \right] \tag{2.68}$$

The final feature of the LMA is the introduction of the Marquardt Lambda, $\lambda$. The Marquardt Lambda is taken as the largest term in the matrix $\alpha \mathbf{S}^T \mathbf{S}$. Here, the adjustment of $\lambda$ provides control over the relative weighting of the SDM versus the TSA, such that for large values of $\lambda$, the SDM dominates, while for small values of $\lambda$, the TSA dominates. The iterative algorithm is summarized as follows [Matott, 2008]:

1. Choose initial value for $\lambda$
2. Compute $\mathbf{s}^M (\boldsymbol{\chi}_i, \langle e \rangle)$
3. Compute $\Psi_i$ with (2.64)
4. Compute $\mathbf{U}_i$ with (2.68)
5. Compute $\boldsymbol{\chi}_{i+1} = \boldsymbol{\chi}_i + \mathbf{U}_i$
6. Compute $\mathbf{s}^M (\boldsymbol{\chi}_{i+1}, \langle e \rangle)$
7. Compute $\Psi_{i+1}$ with (2.64)
8. Adjust $\lambda$

   (a) If number of $\lambda$ adjustments is optimal or exceeds maximum, go to step 9
   (b) If $\Psi_{i+1} < \Psi_i$, reduce $\lambda$, increment $i$, and go to step 2
   (c) If $\Psi_{i+1} \geq \Psi_i$, increase $\lambda$, and go to step 4

9. Test for convergence by comparing $\Psi_{i+1}$ and $\Psi_i$
   (a) If converged, iteration is complete and $\boldsymbol{\chi}_{i+1}$ represents the optimal solution.
   (b) If not converged, increment $i$, and go to step 2

## 2.7 Physically Meaningful Model Parameterization

To accelerate the process of finding a near-optimal set of parameters, it is important to limit the search space of the parameterization algorithm. This is especially important when the number of parameters is large. It was found to be beneficial for the parameters to have physical meaning in order to specify realistic bounds. This is the case when functional assumptions have to be made for curves used in the constitutive models.

Here, the parameterization of the two macroscale continuum models is presented. For both models, the approach to parameterization is the same, but the specific methods and resulting parameters are different due to the inherently different constitutive models. However, the elastic behaviour for both models is governed by the same constitutive relation. The elastic behaviour is parameterized by Young's modulus, $E$, and Poisson's ratio, $\nu$. Bounds on these quantities are well known.

### 2.7.1 Drucker-Prager Model with Ductile Damage

The yield function and flow potential function are parameterized in terms of the friction angle, dilation angle, and the stress ratio $K$. Bounds on these quantities are relatively well known.

The hardening function (2.31) is given in terms of two empirical coefficients $\alpha$ and $\beta$ and the initial compressive yield stress $\sigma_c^{iy}$. While it is possible to set bounds on $\sigma_c^{iy}$, it is less straightforward to set bounds for $\alpha$ and $\beta$ as they do not have obvious physical meaning. The hardening function for the Barcelona model is shown in Figure 2.14. The coefficients

$\alpha$ and $\beta$ can be rewritten in terms of the peak compressive yield strength, $\sigma_c^p$, the plastic strain at the peak compressive yield strength, $\epsilon_c^p$, and the initial compressive yield stress $\sigma_c^{iy}$:

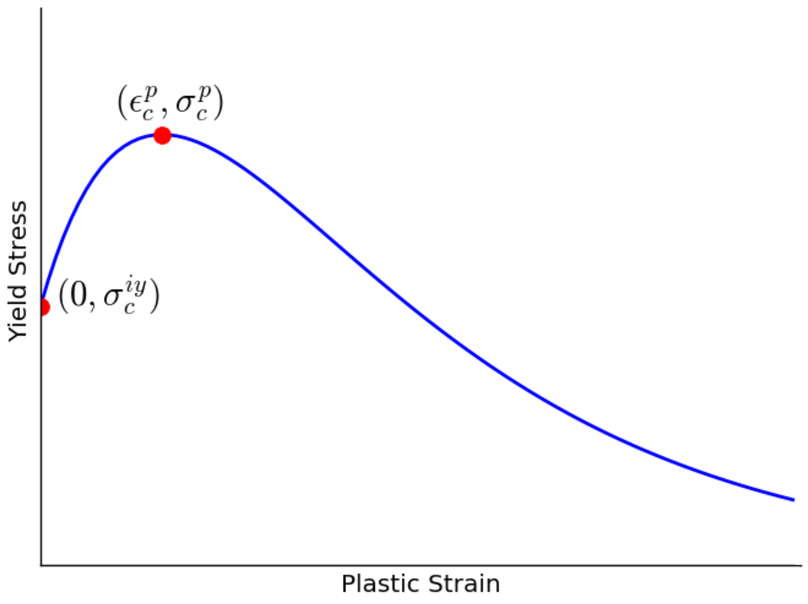


Figure 2.14: Compressive hardening/softening function from the Barcelona model. The curve is able to be parameterized using three parameters.

$$\beta = \frac{\ln\left[\frac{2\alpha}{1+\alpha}\right]}{\epsilon_c^p} \tag{2.69}$$

$$\alpha = \frac{2\sigma_c^p - \sigma_c^{iy} + 2\sqrt{-\sigma_c^p\left[\sigma_c^{iy} - \sigma_c^p\right]}}{\sigma_c^{iy}} \tag{2.70}$$

Thus, the hardening law is parameterized in terms of $\sigma_c^p$, $\epsilon_c^p$, and $\sigma_c^{iy}$ so that bounds can be more easily defined.

Similarly, the Johnson-Cook damage initiation criterion from (2.33) is described by two empirical coefficients ($D_2$, and $D_3$) which do not have intuitive physical meaning. To make setting the bounding limits during the parameter estimation simpler, the Johnson-Cook parameters were rewritten in terms of equivalent plastic strain at which damage is initiated at triaxialities of -0.5 and -0.6, $\bar{\epsilon}^{pl}_{y_{-0.5}}$ and $\bar{\epsilon}^{pl}_{y_{-0.6}}$, respectively:

$$D_2 = \frac{\left[\bar{\epsilon}^{pl}_{y_{-0.5}}\right]^6}{\left[\bar{\epsilon}^{pl}_{y_{-0.6}}\right]^5} \tag{2.71}$$

$$D_3 = 10 \ln \left[\frac{\bar{\epsilon}^{pl}_{y_{-0.5}}}{\bar{\epsilon}^{pl}_{y_{-0.6}}}\right] \tag{2.72}$$

In addition, the damage evolution was parameterized using only the plastic displacement at failure parameter. The goal of the parameter estimation module, in the verification examples, is therefore to estimate the 11 parameters $\boldsymbol{\chi} = \{E, \nu, \psi, K, \phi, \sigma_c^{iy}, \sigma_c^p, \epsilon_c^p, \bar{\epsilon}^{pl}_{y_{-0.5}}, \bar{\epsilon}^{pl}_{y_{-0.6}}, \bar{u}_f^{pl}\}$, which are summarized in Table 2.1.

Table 2.1: Parameter set for Drucker-Prager Material Model with Ductile Damage

| **Parameter Type** | | **Name** | **Symbol** |
|---|---|---|---|
| Elastic | | Young's Modulus | $E$ |
| | | Poisson's Ratio | $\nu$ |
| Plastic | Flow Rule/Yield Function | Dilation Angle | $\psi$ |
| | | Yield Stress Ratio | $K$ |
| | | Friction Angle | $\phi$ |
| | Hardening Rule | Initial Compressive Yield Strength | $\sigma_c^{iy}$ |
| | | Peak Compressive Yield Strength | $\sigma_c^p$ |
| | | Strain at Peak Compressive Yield | $\epsilon_c^p$ |
| Damage | Initiation | Yield Strain at $-0.5$ Triaxiality | $\bar{\epsilon}^{pl}_{y_{-0.5}}$ |
| | | Yield Strain at $-0.6$ Triaxiality | $\bar{\epsilon}^{pl}_{y_{-0.6}}$ |
| | Evolution | Plastic Displacement at Failure | $\bar{u}_f^{pl}$ |

### 2.7.2 Damage-Plasticity Model for Quasi-Brittle Materials

The yield function for this model, (2.35), is written in terms of three material parameters: $A$, $B$, and $\gamma$. These material parameters are not directly measurable, but can be expressed in terms of measurable parameters. $A$ is expressed here as a function of the ratio $f_0$, which is defined as the ratio of the initial biaxial compressive yield strength, $\sigma_{b0}$ to the initial uniaxial compressive strength, $\sigma_{c0}$:

$$f_0 = \frac{\sigma_{b0}}{\sigma_{c0}} \tag{2.73}$$

$$A = \frac{f_0 - 1}{2f_0 - 1} \tag{2.74}$$

Additionally, $\gamma$ is expressed in terms of the ratio $K_c$ which expresses the ratio of hydrostatic pressure at yield in tension to the hydrostatic pressure at yield in compression:

$$\gamma = \frac{3\left[1 - K_c\right]}{2K_c - 1} \tag{2.75}$$

Similarly, $B$ is expressed as a function of $A$ and the two yield stresses from the hardening rule in (2.39), $\bar{\sigma}_c\left(\bar{\epsilon}_c^{pl}\right)$ and $\bar{\sigma}_c\left(\bar{\epsilon}_t^{pl}\right)$.

$$B = \frac{\bar{\sigma}_c\left(\bar{\epsilon}_c^{pl}\right)}{\bar{\sigma}_t\left(\bar{\epsilon}_t^{pl}\right)}(1 - A) - (1 + A) \tag{2.76}$$

The compressive hardening function, similar to the hardening function in the Drucker-Prager model in the previous section, is also approximated using the Barcelona model as shown in Figure 2.14. The same parameterization scheme is also used to write $\alpha$ and $\beta$ in terms of the peak compressive yield strength, $\sigma_c^p$, and the plastic strain at the peak compressive yield strength, $\epsilon_c^{pp}$:

$$\beta = \frac{\ln\left[\frac{2\alpha}{1+\alpha}\right]}{\epsilon_c^{in}} \tag{2.77}$$

$$\alpha = \frac{2\sigma_c^p - \sigma_c^{iy} + 2\sqrt{-\sigma_c^p\left[\sigma_c^{iy} - \sigma_c^p\right]}}{\sigma_c^{iy}} \tag{2.78}$$

The tensile hardening rule has a fundamentally different behavior than the compressive hardening rule, and was therefore approximated using an exponential function (Fig 2.15). The exponential function required only two parameters to characterize the curve completely. The first parameter was the initial tensile yield stress, $\sigma_t^{iy}$, which defines the y-intercept of the curve, while the second parameter was the tensile yield stress decay parameter,$\lambda$.

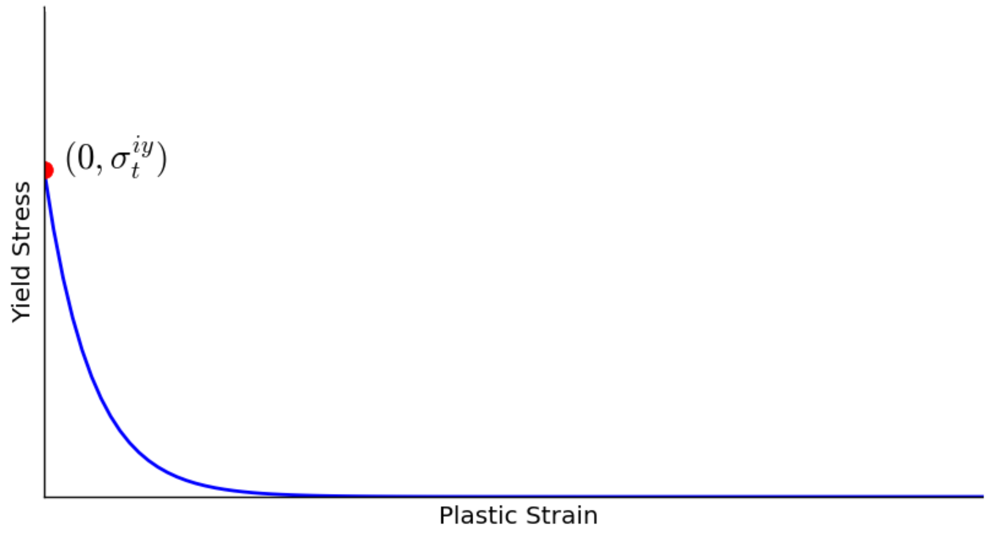


Figure 2.15: Tensile hardening/softening function. The curve is able to be parameterized using two parameters.

In addition to the hardening rules, the damage evolution equations are also parameterized. The compressive damage, $D_c$, is assumed to be a linear function of the inelastic strain through a compressive damage rate parameter, $m$:

$$D_c\left(\bar{\epsilon}^{in}\right) = \bar{\epsilon}^{in} m \tag{2.79}$$

The tensile damage ($D_t$) evolution is slightly less trivial, but can also be characterized by a single parameter due to some constraints imposed on the function by the nature of the damage parameter. In tension, the damage evolution curve starts at the origin and asymptotically approaches $D_t = 1$ as $\bar{\epsilon}^{ck} \to \infty$. As such, under this functional assumption, the only parameter required to describe this relationship is the tensile damage rate parameter, n:

$$D_t\left(\bar{\epsilon}^{ck}\right) = 1 - \frac{1}{\left[1 + \bar{\epsilon}^{ck}\right]^n} \tag{2.80}$$

Sample damage evolution curves for both tension and compression are illustrated in Fig 2.16, where one can see that the rate at which the tensile damage evolves is far larger than the rate at which the compressive damage evolves. The combination of the elastic parameters, the hardening rule parameters, and the damage evolution parameters, yield a total of 11 parameters that must be identified by experiments or through up-scaling to define the behavior of CDM model.

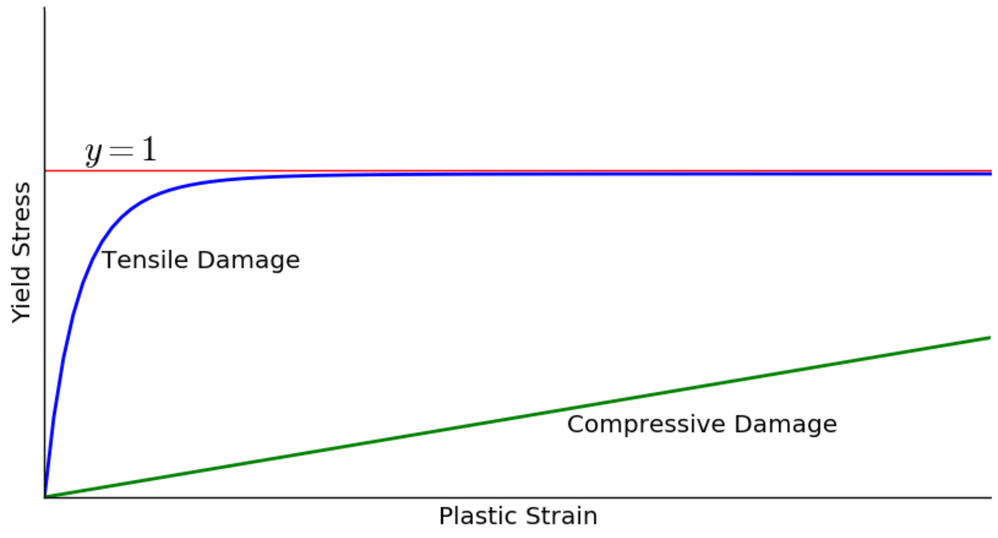


Figure 2.16: Tensile and compressive damage evolution curves.

At this point, there are some parameter constraints for the damage evolution that need to be considered for numerical stability. In this model, the damage curves are specified in terms of inelastic strain and cracking strain which need to be converted into plastic strain for the analysis. The inelastic and cracking strains represent the same strain component but refer to compression and tension respectively. This inelastic/cracking strain can be considered as the theoretical plastic strain given that the material is in an undamaged state. The conversion from inelastic/cracking strain to plastic strain is a function of the damaged state at every increment and can be expressed as:

$$\bar{\epsilon}_c^{pl} = \bar{\epsilon}^{in} - \frac{D_c}{1-D_c}\frac{\sigma_c^{iy}}{E} \tag{2.81}$$

$$\bar{\epsilon}_t^{pl} = \bar{\epsilon}^{ck} - \frac{D_t}{1-D_t}\frac{\sigma_t^{iy}}{E} \tag{2.82}$$

The numerical issues with this formulation arise due to the fact that it is very possible for the converted plastic strain to not be monotonically increasing with respect to the tensile damage. By having a damage evolution curve with a sufficiently steep slope, such that the second term in (2.81) increases faster than the first term, it becomes mathematically possible to have decreasing and/or negative plastic strains in the damage evolution definition. As such, the following conditions are applied to constrain the damage evolution to always yield monotonically increasing plastic strains as damage increases:

$$\frac{d\bar{\epsilon}_c^{pl}}{dD_c} > 0 \tag{2.83}$$

$$\frac{d\bar{\epsilon}_t^{pl}}{dD_t} > 0 \tag{2.84}$$

For the compressive damage evolution, (2.79) is substituted into (2.81) to get the following expression of plastic strain as a function of the compressive damage rate parameter:

$$\bar{\epsilon}_c^{pl} = \bar{\epsilon}^{in} - \frac{D_c}{1-D_c}\frac{\sigma_c^{iy}}{E} \tag{2.85}$$

Combining (2.85) and (2.83) yields the following expression governing the stability limit for the compressive damage rate parameter:

$$m < \frac{\sigma_c^{iy} + 2E\bar{\epsilon}^{in} - \sqrt{\sigma_c^{iy}\left[\sigma_c^{iy} + 4E\bar{\epsilon}^{in}\right]}}{2E\left[\bar{\epsilon}^{in}\right]^2} \tag{2.86}$$

However, since this upper bound for $m$ is functional on several of the parameterization parameters, which are not constant, the upper bound varies depending on the other input parameters. Because of this, a compressive damage scaling factor, $d_c$, is introduced. In addition, the upper bound for $m$ is dependent on the inelastic strain which is not constant throughout the model. Since only one value of $m$ can be specified for a given simulation, the chosen value of $m$ should be the smallest value over the range of the expected inelastic strain experienced. As can be seen from (2.86), as $\bar{\epsilon}^{in} \to \infty$, $m \to 0$ such that for very large inelastic strains the conversion to plastic strain becomes very unstable. Thus, the compressive damage rate parameter can be written as:

$$m = d_c \min_{\bar{\epsilon}^{in}} \left\{ \frac{\sigma_c^{iy} + 2E\bar{\epsilon}^{in} - \sqrt{\sigma_c^{iy}\left[\sigma_c^{iy} + 4E\bar{\epsilon}^{in}\right]}}{2E\left[\bar{\epsilon}^{in}\right]^2} \right\} \tag{2.87}$$

where the compressive damage scaling factor has the following limits:

$$0 < d_c < 1 \tag{2.88}$$

The tensile damage evolution curve has the same numerical constraints when converting form cracking strain to plastic strain. Substituting (2.80) into (2.81) yields the following expression for the plastic strain:

$$\bar{\epsilon}_t^{pl} = \bar{\epsilon}^{ck} - \left[\left[1 + \bar{\epsilon}^{ck}\right]^n - 1\right] \frac{\sigma_t^{iy}}{E} \tag{2.89}$$

Solving for $n$ with (2.83) and (2.89) yields the following inequality governing the upper bound of the tensile damage rate parameter:

$$n < \frac{W\left(\frac{E\left[\bar{\epsilon}^{ck}+1\right]}{\sigma_t^{iy}} \ln\left[\bar{\epsilon}^{ck} + 1\right)\right]}{\ln\left[\bar{\epsilon}^{ck} + 1\right]} \tag{2.90}$$

where $W(x)$ is the Lambert W function defined implicitly as [Corless et al., 1996]:

$$x = W(x)\, e^{W(x)} \tag{2.91}$$

Using the same methodology as was used to derive (2.87) in compression, the tensile damage scaling factor can be written as:

$$n = d_t \min_{\bar{\epsilon}^{ck}} \left\{ \frac{W\left( \frac{E\left[\bar{\epsilon}^{ck}+1\right]}{\sigma_t^{iy}} \ln\left[\bar{\epsilon}^{ck}+1\right] \right)}{\ln\left[\bar{\epsilon}^{ck}+1\right]} \right\} \tag{2.92}$$

Where the tensile damage scaling factor has the following limits:

$$0 < d_t < 1 \tag{2.93}$$

The goal of the parameter estimation module, is therefore to estimate the 11 parameters $\boldsymbol{\chi} = \{E, \nu, \psi, K_c, \phi, \varepsilon, \sigma_c^{iy}, \sigma_c^{p}, \epsilon_c^{p}, d_c, d_t\}$, which are summarized in Table 2.2.

Table 2.2: Parameters for Damage-Plasticity Model for Quasi-Brittle Materials

| Parameter Type | | Name | Symbol |
|---|---|---|---|
| Elastic | | Young's Modulus | $E$ |
| | | Poisson's Ratio | $\nu$ |
| Plastic | Flow Rule | Dilation Angle | $\psi$ |
| | | Flow Eccentricity | $\varepsilon$ |
| | Yield Function | Second Stress Invariant Ratio | $K_c$ |
| | | Initial Equibiaxial Stress Ratio | $f_0$ |
| | Hardening Rule | Initial Compressive Yield Strength | $\sigma_c^{iy}$ |
| | | Peak Compressive Yield Strength | $\sigma_c^{p}$ |
| | | Strain at Peak Compressive Yield | $\epsilon_c^{p}$ |
| Damage | Compressive | Compressive Damage Scaling Factor | $d_c$ |
| | Tensile | Tensile Damage Scaling Factor | $d_t$ |

# Chapter 3

# Framework Implementation

The software that I developed for the implementation of this up-scaling framework is called "**M**odular aut**O**mated **U**p-**S**caling softwar**E**" (MOUSE). MOUSE for DEM simulations was created and written in Python$^{TM}$ to provide an implementation of the up-scaling framework presented in the previous chapter using in house and third-party software modules. Additionally, I also developed the homogenization software, **H**omogenization **O**f **D**EM **S**imulations (HODS) to be used as the homogenization module. The MOUSE software itself aims to provide a platform through which the four software elements of the up-scaling framework (DEM, homogenization, parameter estimation, and macroscale model) can communicate with each other. The communication is facilitated by MOUSE through modules which wrap the third party software in such a way that the I/O routines to and from the modules are performed in a consistent way regardless of the third party software being used.

This chapter of the thesis aims to provide a very high level overview of the key aspects of the MOUSE and HODS software implementations. The programmatic structure of some of the important software components is presented to illustrate the software design, but a complete discussion on the construction of all the specific wrappers is beyond the scope of the discussion in this chapter. The modular implementation of MOUSE is summarized by Figure 3.1

The goal of the up-scaling framework implementation is that it is model independent. So, MOUSE was written in such a way that allowed for different models (both DEM and macroscale) to be implemented without rewriting the up-scaling algorithms. As such, a

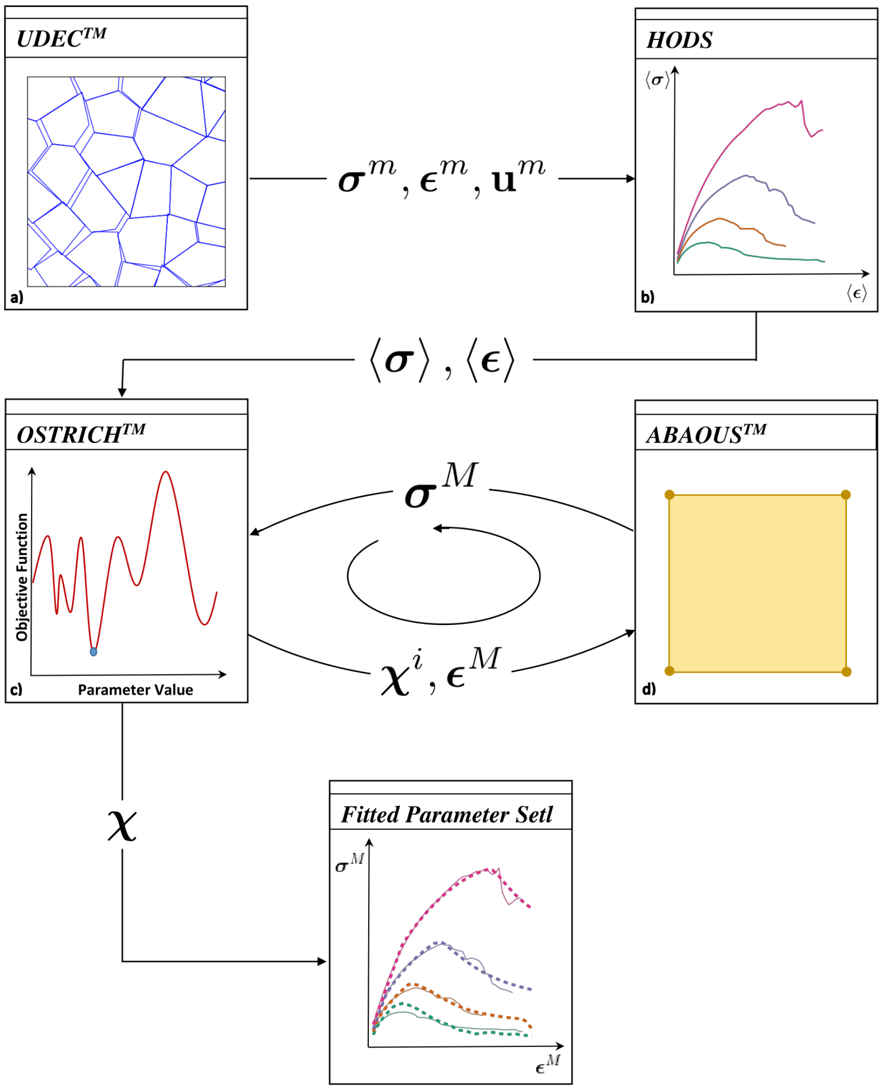


Figure 3.1: MOUSE module implementation schematic. The DEM software, UDEC™, produces a data set which is run through homogenization software, HODS, which in turn produces another dataset that is fed into the parameter estimation program, OSTRICH™. This parameter estimation program drives the macroscale simulations in ABAQUS™ iteratively in order to find an optimal parameter set for the fitted model.

modular approach was taken such that each module is isolated from the others and only communicates data through strict protocols set out by MOUSE.

There exist four base classes that are used as parents to the third-party software module classes. Because each software component has distinct I/O protocols, the base classes serve as a collection of useful methods which the software module can inherit to ensure compatibility with MOUSE. Each of these four component base classes inherit from a base module class. Figure 3.2 indicates this class hierarchy, as well as, where each method and attribute definition falls in the hierarchy.

## 3.1 Software Module Format

A base module class, **BaseModuleClass**, is implemented to provide a framework containing required methods and attributes for the MOUSE modules to inherit. The module class contains methods pertaining to I/O routines associated with the module so that each module that is written behaves in a consistent manner and to avoid reimplementation of certain methods.

### 3.1.1 Attributes

Attributes are assigned to **BaseModuleClass** through the constructor method (**__init__**) when the class is initialized. The following attributes are passed as arguments through instantiation of the class children.

The **program** attribute is a string that contains the name of the software executable associated with the given module. This parameter allows the module the capacity to run the program through a Command Line Interface (CLI) As such, the program must be able to accept command line arguments.

```
#String containing name of module software executable file.
self.program = program
```

The **parameters** attribute is a dictionary that contains all the command line parameters and corresponding arguments. Here, the parameters are the dictionary keys and the arguments are the dictionary values. If no arguments are required, then an empty dictionary is acceptable as well.

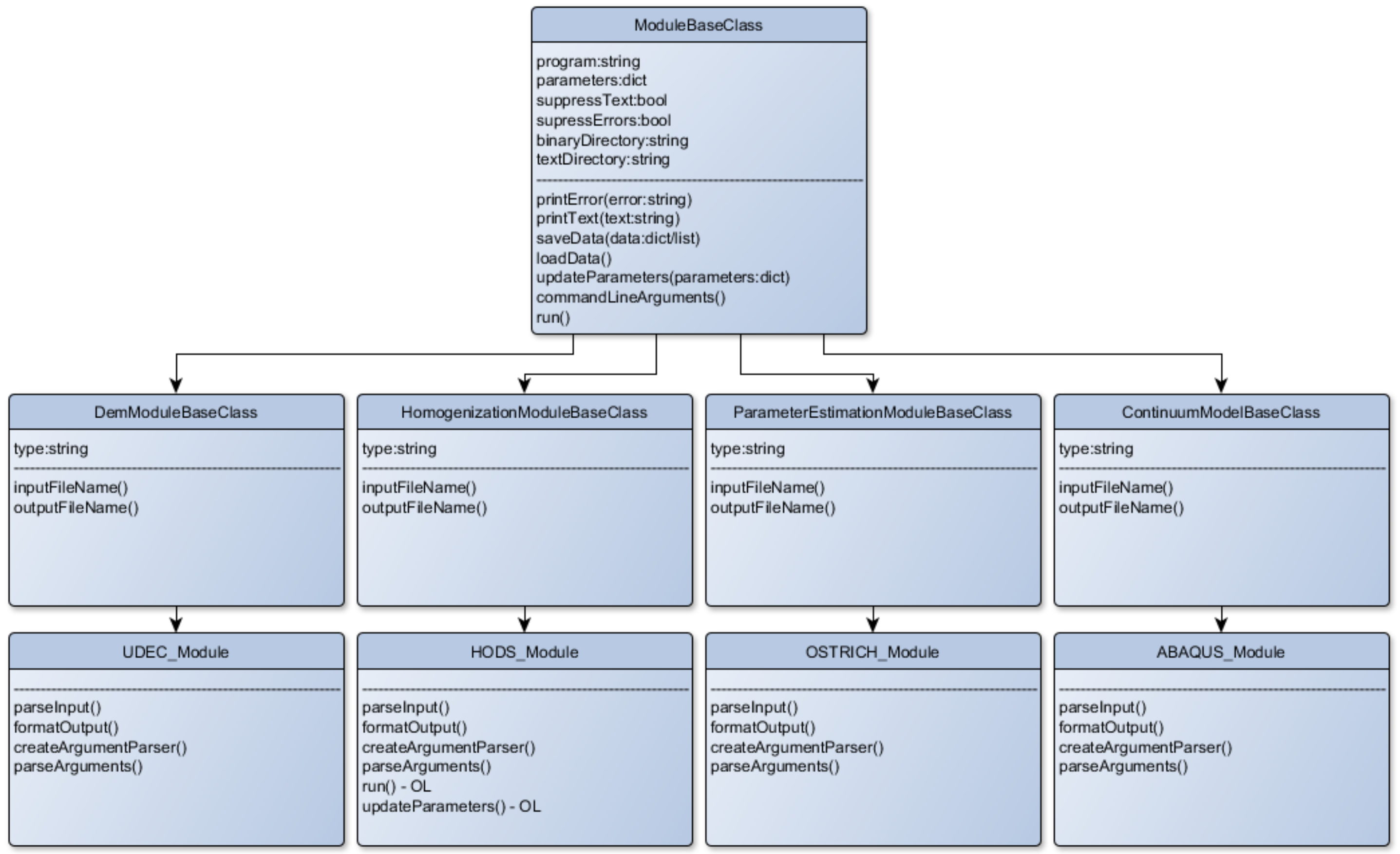


Figure 3.2: Module class inheritence structure.

```
#Dictionary of command line parameters and corresponding
    arguments
self.parameters = parameters
```

The **supressText** and **supressErrors** attributes are Boolean parameters that, when true, allow for the suppression of command line output of text and errors, respectively.

```
#Boolean parameter that allows text output to CMD to be supressed
self.suppressText = suppressText

#Boolean parameter that allows errors to be supressed
self.suppressErrors = suppressErrors
```

The **binaryDirectory** and **textDirectory** attributes are strings that refer to file folders in which the binary and text data are to be saved, respectively. These values are determined through relative directory parsing, rather than as an initialization argument.

```
#String that indicates where to store the binary data
self.binaryDirectory = os.path.join(dataDirectory, 'Binary')

#String that indicates where to store the text data
self.textDirectory = os.path.join(dataDirectory, 'Text')
```

### 3.1.2 Defined Methods

Methods in **BaseModuleClass** are divided into two types: defined and undefined. Defined methods are methods that are common to all the module classes and are thus able to be defined at the parent level. The undefined methods are methods that require overloading, so they are defined at the child level.

The **__init__** method is a built-in constructor method that is automatically called when the class is initialized. The only required argument for this method is **parameter**, with the other three being optional arguments.

```
#Contructor method that is called when the class is initialized
def __init__(self, program, parameters={}, suppressText=False,
   suppressErrors=True):
        """
        program:          string of program name
        parameters:       dictionary of parameter-argument pairs
        suppressText:     boolean
        suppressErrors:   boolean"""
```

A series of printing methods are included in **BaseModuleClass** to allow each module to route command line output through the parent module for consistent output formatting. The **printText**, **printTitle**, **printSection**, **printStatus**, **printDone**, and **printErrors** methods are implemented to give a large degree of flexibility in the displayed output.

```
#Method to print text to command line if text supression is off
def printText(self, text):
        """
        text:             string of text to print """

#Method to format primary title in command line output
def printTitle(self, title):
        """
        title:            string of primary title to print"""

#Method to format section title in command line output
def printSection(self, section):
        """
        section:          string of section title to print"""

#Method to format status update in command line output
def printStatus(self, status):
        """
        status:           string of status to print"""
```

```
#Method to print 'Done' when section is finished
def printDone(self):
        """
        """

#Method to control and print errors if error supression is off
def printErrors(self, error):
        """
        error:          error to print"""
```

The **saveData** and **loadData** methods use binary serialization methods to write and load specified data structures to and from file. Saving and loading serialized binary data is much faster and more compact than unicode.

```
#Method to save serialized data structures as output
def saveData(self, data):
        """
        data:           Data format specified by module type"""

#Method to load serialized data structures as input
def loadData(self):
        """
        """
```

The **updateParameters** method allows for the dynamic updating of module input parameters. This allows for the module to run the program multiple times without being re-instantiated.

```
#Method that allows dynamic updating of parameters.
def updateParameters(self, parameters):
        """
        parameters:     dictionary of parameter-argument pairs"""
```

The **commandLineArguments** method takes the **parameters** dictionary attribute and converts it to a string which can be passed to the command line when running the specified program.

```
#Method to create a string of command line arguments from
    parameters
def commandLineArguments(self):
        """
        """
```

The **run** method simply runs the specified program with the specified parameters.

```
#Method to run the program with the specified parameters
def run(self):
        """
        """
```

### 3.1.3 Undefined Methods

The undefined methods listed here are required to be defined in the child classes. These methods are different for each type of module or even each module implementation, which does not allow them to be written in **baseModuleClass**.

The **createArgumentParser** method creates an argument parser that allows the module to parse any command line arguments passed to it if the module is run in an isolated environment, While the **parseArguments** method parses the command line arguments derived from **createArgumentParser**.

```
#Creates a module specific argument parser specific to allow the
    module to run independantly
def createArgumentParser(self):
        """
        """
#Parses the command line arguments passed to the module
def parseArguments(self):
        """
        """
```

The **inputFileName** and **outputFileName** methods provide the formatting for the file names in a means that is consistent amongst modules.

```
#Name of input file
def inputFileName(self):
        """
        """
#Name of output file
def outputFileName(self):
        """
        """
```

The final two undefined methods **parseInput** and **formatOutput** are provided by the software specific modules. These two methods act as wrappers for the software by manipulating the MOUSE inputs into input files for the specified programs as well as manipulating the software output to a form which is consistent with the MOUSE data architecture.

```
#converts MOUSE input to input files accepted by the software
def parseInput(self):
        """
        """
#converts module software output to MOUSE consistent data
   structures
def formatOutput(self):
        """
        """
```

## 3.2 Data Architecture

This section aims to explain the data structures used to store and transfer data between the modules in the up-scaling framework presented in this thesis. Three fundamental data structures are used for the storage of data in conjunction with each other: hash tables, lists, and arrays. Using these three structures, three distinct types of data in the up-scaling framework are used to transfer between the modules in a consistent manner: constitutive parameters, DEM data, and continuum data.

#### Python Lists

The list in Python$^{TM}$ is one of the most versatile data structures. The list treats the data as a sequence such that each element in the list is assigned a number referring to its position or index. The implementation of the list structure in python contains a number of unique features. The main feature of python lists is that the elements within the lists are not required to be of the same data type. Additionally, these lists are mutable, allowing for more advanced manipulation of the data structure. Because of this flexibility, the list structure is slow and can be unsuitable for large data sets.

#### Numpy Arrays

Numpy arrays are a specific implementation of a Python$^{TM}$ list which are optimized for numerical operations on large datasets. Here the numpy arrays are much more compact by using a sequence of uniform values, rather than a sequence of pointers as is the case for the list structure.

#### Python Dictionaries (Hash Tables)

In general, a hash table is a data structure that maps keys to values. The implementation of hash tables in Python$^{TM}$, are referred to as dictionaries. Similar to a Python$^{TM}$ list, a Python$^{TM}$ dictionary is a mutable structure that does not require the elements to be of the same type. However, instead of accessing the element value through an index argument, the data is accessed through a key argument. This unordered structure is useful for storing non-sequential data.

### 3.2.1 Data Storage Structures

Three distinct types of data are used in the information transfer protocols between modules: constitutive parameter data, DEM data, and continuum data. These different types of data require different data structures which are explained in more detail below. The constitutive parameter data is stored in a hash table, the DEM data is stored in a three level nested hash table, and the continuum data is stored in a list of arrays.

## Hash Tables for Constitutive Parameters

Constitutive parameters are required as an input to the DEM and macroscale modules as well as an output from the parameter estimation module. Here, the constutive parameters are always stored in a Python$^{\text{TM}}$ dictionary with the constitutive parameter name as the key. To access the value of a specific constitutive parameter in a dictionary named 'constitutiveParameters', the value corresponding to any parameter name can be accessed by:

```
parameterValue = constitutiveParameters[parameterName]
```

For example, accessing the value of the elastic modulus from a hash table named 'constitutiveParameters' is performed as follows:

```
elasticModulus = constitutiveParameters['elasticModulus']
```

## Nested Hash Tables for DEM Data

The raw DEM Data is considered to be comprised of six distinct types: block data, contact data, corner data, domain data, grid point data, and zone data. Here, each block, contact, corner, domain, grid point, and zone is assigned a unique 7-digit numeric identifier (assuming here that the number of components in the system does not exceed 10 million) by which the associated data can be accessed. The same identifier may be repeated for different data types.

To allow for convenient access to the data, the DEM data is stored in six distinct nested hash tables corresponding to each of the DEM data types. Each DEM data hash table has three levels of nesting. The first level keys are the simulation times, which returns the second level of hash tables. The second level keys are the component identifiers, which returns a third level hash table. In this third level, the component attributes can be accessed using the attribute name as the key. In general, accessing an attribute of a specific component with a given ID at a specific time from a hash table called demDataHash is done as follows:

```
componentAttributeValue = demDataHash[time][ID][attribute]
```

For example, accessing the list of grid points associated with block ID 2543465 at a simulation time of 1.0 is performed as follows:

```
gridPoints = blockData[1.0][2543465]['gridPoints']
```

The keys of each of the third level dictionaries for each component are presented in Table 3.1, Table 3.2, Table 3.3, Table 3.4, Table 3.5, and Table 3.6 for the block data, contact data, corner data, domain data, gridpoint data, and zone data, respectively.

Table 3.1: Block data attributes in third level hash

| **Parameter** | **Description (Data Type)** |
|---|---|
| x | X coordinate of block centroid (float) |
| y | Y coordinate of block centroid (float) |
| xForce | Resultant forces at block centroid (float) |
| yForce | Resultant forces at block centroid (float) |
| corners | Corners associated with this block (list of corner IDs) |
| zones | Zones associated with this block (list of zone IDs) |
| gridPoints | Grid points associated with this block (list of corner IDs) |

### Nested List of Arrays for Continuum Data

The continuum data is stored in three lists: a normal list and two nested lists of arrays. The three lists contain the time data, stress data, and strain data. The time data is in the normal list, with each element representing a different time step. The other two lists contain two-dimensional nested arrays representing the stress and strain tensors respectively. Each array element within these lists represents the stress or strain tensors for the corresponding time step.

Table 3.2: Contact data attributes in third level hash

| Parameter | Description (Data Type) |
|---|---|
| x | X coordinate of contact point (float) |
| y | Y coordinate of contact point (float) |
| length | Length associated with contact point (float) |
| flowRate | Fluid flow rate through contact (float) |
| nForce | Resultant normal force on contact (float) |
| sForce | Resultant shear force on contact (float) |
| xCosine | X component of contact normal cosine (float) |
| yCosine | Y component of contact normal cosine (float) |
| blocks | Blocks associated with this contact (list of block IDs) |
| domains | Domains associated with this contact (list of domain IDs) |
| corners | Corners associated with this contact (list of corner IDs) |

Table 3.3: Corner data attributes in third level hash

| Parameter | Description (Data Type) |
|---|---|
| gridPoint | Grid points associated with this corner (list of grid point IDs) |

Table 3.4: Domain data attributes in third level hash

| Parameter | Description (Data Type) |
|---|---|
| x | X coordinate of domain centroid (float) |
| y | Y coordinate of domain centroid (float) |
| area | Area of domain (float) |
| porePressure | Average pore pressure in the domain (float) |

### 3.2.2 Binary Serialization

These data structures mentioned above are great for efficiently accessing the desired data, but cannot easily be saved to file in standard unicode. In the event that this is possible, formatting and parsing large data sets from text files is slow and increases the possibility of data corruption. To speed up the data access and increase the integrity of the data, binary serialization is employed.

Table 3.5: Gridpoint data attributes in third level hash

| **Parameter** | **Description (Data Type)** |
|---|---|
| x | X coordinate of grid point (float) |
| y | Y coordinate of grid point (float) |
| xDisp | X displacement of grid point (float) |
| yDisp | Y displacement of grid point (float) |
| xForce | Resultant X force on grid point (float) |
| yForce | Resultant Y force on grid point (float) |
| xVel | X velocity of grid point (float) |
| yVel | Y velocity of grid point (float) |
| block | Block associated with this grid point (block ID) |
| corner | Corner associated with this grid point (corner ID) |

Table 3.6: Zone data attributes in third level hash

| **Parameter** | **Description (Data Type)** |
|---|---|
| S11 | X component of stress in the zone (float) |
| S22 | Y component of stress in the zone (float) |
| S12 | Corresponding shear stress in the zone (float) |
| block | Block associated with this zone (block ID) |
| gridPoints | Grid points associated with this grid point (list of block IDs) |

Binary serialization stores the state of any given object by converting the public and private fields of the class to a stream of bytes, which is then written to the data stream. This data stream can easily be written to a binary file and subsequently read at any later date. The deserialization of this data stream takes the stream of bytes and reconstructs an exact clone of the original object whenever it is required. This method is useful when attempting to store large non-linear data structures to file.

## 3.3 Third Party Software Modules

Four software modules are required for the MOUSE software to be functionally complete. The DEM software, parameter estimation software, and macroscale modules were developed by wrapping third-party software in Python$^{\text{TM}}$ scripts. However, commercial homogenization software does not exist, so an in-house homogenization software (HODS) was created and implemented into the homogenization module.

Some of these module implementations are non-trivial and require a substantial amount of code to properly parse the output files and format the input files. In the interest of brevity, the majority of the details of these implementations have been omitted.

For the DEM module, the commercial software UDEC$^{\text{TM}}$ was used. Here, the binary output from the software was unable to be parsed easily without knowing the data structure of the serialization method or having access to an Application Program Interface (API). As such, a cycling script, using the built-in scripting language, FISH, was written to write the simulation data to text file as the simulations are running. These text files are subsequently consolidated and parsed with the Python$^{\text{TM}}$ wrapper to create a data file that is consistent with the MOUSE architecture.

The homogenization module, as previously stated, was written in-house as a 2D homogenization code, HODS. This development meant that full control over the I/O routines was possible. As such, the software was able to be directly written as a module with the exact same data structures to avoid and parsing and formatting issues.

The parameter estimation module was implemented using OSTRICH$^{\text{TM}}$ software, a model independent optimization software toolkit. Here, this parameter estimation module iteratively generates input parameters for the macroscale model in order to converge on a solution. The module accepts writes the DEM data and FEA data to text files for the OSTRICH program to read in. Here, multiple different optimization algorithms are implemented and easy to switch between.

The macroscale module was implemented using the commercial FEA software, ABAQUS$^{\text{TM}}$. Similarly to the DEM module, an internal script had to be written to describe the constitutive relationships as well as write the simulation data to text file since no API exists to parse the output binary data file. Subsequent to the simulation completion, the Python$^{\text{TM}}$ wrapper converts the text to the appropriate MOUSE format to be used in the parameter estimation module.

## 3.4 HODS Homogenization Software

Since no commercial homogenization software exists, a code is developed here to integrate into the homogenization module. The **H**omogenization **O**f **D**EM **S**imulations (HODS) software implements the stress and strain homogenization algorithms described in the previous chapter.

This section aims to describe the implementation of the homogenization algorithms in the HODS software. The main software is comprised of a single class, in which both the stress and the strain homogenization calculations are conducted. The instantiation of this class requires the subdomain location and radius, and the file containing the DEM binary data (the constructor function is also overloaded to be able to parse text files as well in the case that the binary files do not exist or are somehow corrupted). Once the class is instantiated, the homogenization boundaries are defined based on the subdomain radius and location. The homogenization algorithms can then be applied to this subset of the DEM data when requested. It is possible to modify the input parameters and rerun the homogenization algorithms in this implementation without re-instantiating the class. This structure makes efficient use of the memory when running multiple homogenization inquiries when trying to assess the REV. Some of the relevant attributes and methods used to implement the homogenization algorithms are presented below.

The class methods have been (roughly) divided into two subsets in order to clarify the process a bit more: data methods and homogenization methods. The data methods are methods that are used to manipulate or retrieve data while the homogenization methods are methods that are used to perform the actual homogenization calculations and return the resultant homogenized DEM results. This is not a complete, nor comprehensive list of the methods used in the implementation, but illustrate the structure of the software.

### 3.4.1 Class Attributes

There are three main subsets of class attributes used for the HODS class in this implementation. The first subset of attributes contains the subdomain parameters, **centre** and **radius**, which completely define a circular subdomain. In the case that the specified circular subdomain does not fully intersect with the full domain, the subdomain is chosen to be the intersection of these two areas. That being said, it is not recommended to specify such a subdomain as the influence of boundary effects will be large.

```
self.centre = centre
self.radius = radius
```

An additional set of class attributes, **blockData**, **contactData**, **cornerData**, **zoneData**, **gridPointData**, and **domainData**, contain their respective DEM data in nested hash tables as described above. Here, the data is loaded from a **parseDataFile** method, which is called in the overloaded constructor method to parse the DEM data from text files instead of binary data. The functionality of this method is explained in more detail below.

```
self.blockData = self.parseDataFile(blockFileName)
self.contactData = self.parseDataFile(contactFileName)
self.cornerData = self.parseDataFile(cornerFileName)
self.zoneData = self.parseDataFile(zoneFileName)
self.gridPointData = self.parseDataFile(gridPointFileName)
self.domainData = self.parseDataFile(domainFileName)
```

The third subset of attributes below relates to the definition of the homogenization boundary. These attributes represent the algorithmic steps required to define the homogenization domain as discussed in Chapter 3. Though perhaps not strictly necessary for these parameters to be class attributes, it becomes useful to retain these steps in the class memory to be recalled for plotting to verify and debug the algorithms for assessing the homogenization domain.

```
self.boundaryBlocks = self.blocksOnBoundary()
self.insideBlocks = self.blocksInsideBoundary()
self.outsideBlocks = self.blocksOutsideBoundary()
self.insideBoundaryBlocks = self.boundaryBlocks + self.
    insideBlocks
self.boundaryContacts = self.contactsBetweenBlocks(self.
    outsideBlocks, self.boundaryBlocks)
self.boundaryContactCorners = self.cornersOnContacts(self.
    boundaryContacts)
self.boundaryContactBlocks = self.blocksWithContacts(self.
    boundaryBlocks, self.boundaryContacts)
```

```
self.outsideCorners = self.cornersOutsideBoundary()
self.outsideContacts = self.contactsOutsideBoundary()
self.boundaryBlockCorners = self.cornersOnBlocks(self.
   boundaryContactBlocks)
self.boundaryCorners = common.listIntersection(self.
   boundaryContactCorners, self.boundaryBlockCorners)
self.allBoundaryCorners = self.boundaryCorners
self.boundaryBlocksOrdered = self.orderBlocks(self.
   boundaryContactBlocks, self.outsideContacts)
self.boundaryCornersOrdered = self.orderCorners(self.
   boundaryBlocksOrdered, self.allBoundaryCorners)
```

### 3.4.2 Class Data Methods

These data methods are class methods that are primarily used to manipulate and retrieve the DEM data. The constructor method, **__init__**, here requires 8 parameters to properly be instantiated: **centre**, **radius** and the 6 nested hash tables containing the DEM data. Additionally, the constructor method is overloaded to be able to parse the raw DEM data from text files.

```
#Constructor method
def __init__(self, centre, radius, blockData, contactData,
   cornerData, zoneData, gridPointData, domainData):
        """
        centre:         centre of subdomain to be homogenized
        radius:         radius of subdomain to be homogenized
        blockData:      Nested hash table of block data
        contactData:    Nested hash table of contact data
        cornerData:     Nested hash table of corner data
        zoneData:       Nested hash table of zone data
        gridPointData:  Nested hash table of gird point data
        domainData:     Nested hash table of domain data"""
```

```
#Overloded constructor method
def __init__(self, centre, radius, fileName):
        """
        centre: centre of subdomain to be homogenized
        radius: radius of subdomain to be homogenized
        fileName:       prefix of files that contain \acrshort{
           dem} data"""
```

The **parseDataFile** method takes a raw DEM text data file and returns a triple level nested hash table. Here the first column of the text data file is specified to be simulation time and the second column is the component ID. These two columns represent the first two levels of hash keys. For the third level hash keys, the remaining column headers are used.

```
#Method to parse the \acrshort{dem} raw text files.
def parseDataFile(self, fileName):
        """
        fileName:       Name of file that contains \acrshort{dem}
            data"""
        return data
```

The following series of methods are implemented here in order to provide the capacity to find the relational influence of different DEM components.

```
#Finds the list of contacts that are between the two sets of
   blocks
def contactsBetweenBlocks(self, blocks1, blocks2):
        """
        blocks1:        list of block IDs
        blocks2:        list of block IDs       """
        return contacts
#Finds the list of blocks that contain the specified contacts
def blocksWithContacts(self, blocks, contacts):
        """
        blocks:         list of block IDs
        contacts:       list of contact IDs"""
        return newBlocks

#Finds the list of blocks that contain the specified corners
def blocksWithCorners(self, blocks, corners):
        """
        blocks:         list of block IDs
        corners:        list of corner IDs"""
        return newBlocks
#Orders the blocks in a clockwise fashion.
def orderBlocks(self, blocks, relContacts):
        """
        blocks:         list of block IDs
        relContacts:    list of contact IDs"""
        return orderedBlocks
#Orders the corners in a clockwise fashion
def orderCorners(self, orderedBlocks, corners):
        """
        orderedBlocks:  list of block IDs
        corners:        list of corner IDs"""
        return orderedCorners
```

Additionally, the following set of methods relate to describing the homogenization boundary given a circular subdomain. These methods use the already defined class attributes (**radius**, **centre**, DEM hash tables) to perform the search rather than by passing arguments.

```
#Finds all the blocks intersecting the circular subdomain
def blocksOnBoundary(self):
        """
        """
        return blocks
#Finds all the blocks outside of the circular subdomain
def blocksOutsideBoundary(self):
        """
        """
        return blocks

#Finds all the blocks inside the circular subdomain
def blocksInsideBoundary(self):
        """
        """
        return blocks
#Finds all the corners outside of the circular subdomain
def cornersOutsideBoundary(self):
        """
        """
        return corners
#Finds all the corners inside the circular subdomain
def cornersInsideBoundary(self):
        """
        """
        return corners
#Finds all the contacts outside the circular subdomain
def contactsOutsideBoundary(self):
        """
        """
        return contacts
#Finds all the contacts inside the circular subdomain
def contactsInsideBoundary(self):
        """
        """
        return contacts
```

### 3.4.3 Class Homogenization Methods

The **calculateHomogenizationParameters** method implements the algorithm for assessing the homogenization domain and boundary. This method is called by the constructor function after the DEM data is loaded to calculate the homogenization boundary and associated parameters. When the subdomain parameters are modified, this method is called again in order to update the homogenization domain. The parameters calculated in this method are used in both the stress and the strain homogenization alogirithms, and are thus stored as class attributes to be accessed by the **stress** and **strain** methods.

```
#Calculates homogenization boundary and homogenization parameters
def calculateHomogenizationParameters(self):
        """
        """
```

The **stress** and **strain** methods are where the majority of the homogenization algorithms are implemented. These methods can be copmutationally expensive if the dataset is sufficiently large, and thus don't run automatically upon instantiation. These methods can be called independently and at any time the class is alive. When these classes are called, the homogenization algorithms are run and a nested list of homogenized tensor arrays are returned, in accordance with the MOUSE data architecture.

```
#Calculates and returns a nested list of stress tensors
def stress(self):
        """
        """
        return stressHistory
#Calculates and returns a nested list of strain tesnors
def strain(self):
        """
        """
        return strainHistory
```

The **time** method simply returns a list of time steps corresponding with each of the homogenized stress and strain tensors. Again, this format is consistent with the MOUSE architecture.

```
#Returns a list of simulation timesteps
def time(self):
        """
        """
        return timeHistory
```

# Chapter 4

# Verification and Application

In this chapter, two-dimensional DEM models are used to demonstrate the effectiveness of the up-scaling methodology. The framework is validated using three tests: 1) The homogenized stress and strain behaviour obtained from the DEM microscale response are compared to that of the macroscale response. This test verifies the effectiveness of the parameter estimation module and the ability of the chosen macroscale constitutive model to capture the salient features of NFR behaviour. 2) The homogenization and parameter estimation algorithms are rerun using the same data, but with different REV sizes to investigate the REV size effect has on the resultant parameter set. 3) Slope stability analyses carried out by both Direct Numerical Simulation (DNS) with a DEM model and with an up-scaled macroscale model are compared. This last test verifies the whole up-scaling methodology.

The up-scaling is conducted by running a series of four quasi-static DEM virtual 'triaxial' compression tests under different confining stresses. These are not true triaxial tests as simulations are in 2D, but illustrate the method regardless. Algorithms are rerexecuted using different REV sizes to determine an appropriate REV size. In a macroscopically homogenous domain, as the REV size increases, the parameter values will converge to a single value, when the REV is too small, local heterogeneities induce a variance into the optimal parameter set.

## 4.1 DEM Simulations

The DEM models used consist of a pseudo-random isotropic fracture network defined by a Voronoi tessellation. The average block size is specified to be 0.5m using 20 iterations of Lloyd's method [Lloyd, 1982] in order to achieve an even size distribution. A 10m x 10m domain was determined to be sufficiently large to represent the rock mass behaviour as an REV.

A Mohr-Coulomb plasticity model was used as the constitutive model to describe the plastic behaviour of the intact (deformable blocks) and the joint (natural fracture) behaviour was governed by a Coulomb area slip model. The parameters for the rock and joints summarized in Table 4.1 are representative of a fractured granitic rock mass. The joints are relatively weak compared to the blocks, so the blocks behave mostly elastically.

Table 4.1: Rock and joint properties for DEM Simulations

| **Property Type** | **Property** | **Value** |
|---|---|---|
| Rock | Young's Modulus | $65GPa$ |
| | Poisson's Ratio | 0.2 |
| | Density | $2.7g/cm^3$ |
| | Friction Angle | $51°$ |
| | DilationAngle | $0°$ |
| | Cohesion | $55.1MPa$ |
| | Tensile Strength | $11.7MPa$ |
| Joint | Friction Angle | $32°$ |
| | Dilation Angle | $5°$ |
| | Cohesion | $100kPa$ |
| | Tensile Strength | $100kPa$ |
| | Normal Stiffness | $10GPa/m$ |
| | Shear Stiffness | $1GPa/m$ |

The blocks are meshed with linear three-node triangular plane strain finite difference elements with an average side length of 0.5m. This discretization yielded 5-10 zones within each block. A rounding length of 10% of the average block edge length (0.05m) is applied to the blocks to prevent numerical instabilities in the contact algorithm. Quasi-static analysis is obtained through dynamic relaxation, in which the dynamic equations are integrated in time using velocity-proportional viscous damping and mass scaling. State data of the model is collected at 50 evenly spaced intervals.

The quasi-static loading of the DEM simulations is intended to imitate triaxial laboratory tests, so a constant confining stress was applied on the lateral boundaries of the DEM model. Loading is achieved by applying vertical displacements to the top boundary while fixing the bottom boundary, compressing the model to a vertical strain of 5% for four confining horizontal stresses: $0.5MPa$, $1MPa$, $2MPa$, and $4MPa$. These load paths capture key physical phenomena including the pressure dependent yield of the NFR, hardening, and the dependence of damage initiation on the triaxiality.

## 4.2 Verification of the Parameter Estimation Module

Using a PSO algorithm followed by an LMA optimization, the Drucker-Prager plasticity model with ductile damage is then fitted to the homogenized DEM simulation data in order to obtain an optimal parameter set. Each simulation is fit to 50 points defining the homogenized stress-strain curve resulting in a total of 200 data points for all four DEM simulations at different confining stresses. The PSO algorithm uses a swarm size of 24 for 100 generations which is found to be sufficient to converge to a consistent solution.

Here, the CDM model is confined laterally by the homogenized horizontal DEM stress and vertical displacements are prescribed by the homogenized vertical DEM strain with the parameter estimation algorithms programmed to match the horizontal strain and the vertical stress. Because of the large variation in observation magnitudes (between stress/strain and from different confining stresses), each curve is weighted with a normalization factor to prevent the large stress values from dominating parameter estimation. In addition, a linear weighting scheme is applied to each curve to give larger influence to the loading section and lesser influence to the post-damage section.

Parameter bounding limits are required by the optimization algorithms in order to limit the search space. These limits are chosen based on two criteria: physical limitations and numerical stability. If there exist physical limitations that prevent parameters from exceeding certain values or if there exists a range of realistic values that the parameter should not deviate from, then those physical limitations are specified as the bounds. In other cases, the parameter bounds come from numerical limitations such that beyond a certain capacity, certain parameter values would cause the simulations to become unstable. In these cases, a combination of the two bounding methods is used. The specified bounding limits for each parameter results can be seen in Table 4.2.

Table 4.2: Parameter estimation results for Drucker-Prager model with ductile damage

| **Parameter** | **Symbol** | **Units** | **Lower Bound** | **Upper Bound** | **Optimum** |
|---|---|---|---|---|---|
| Young's Modulus | $E$ | $GPa$ | 1 | 25 | 1.8 |
| Poisson's Ratio | $\nu$ | | 0.1 | 0.4 | 0.15 |
| Dilation Angle | $\psi$ | $^\circ$ | 5 | 15 | 22 |
| Flow Stress Ratio | $K$ | | 0.78 | 1 | 0.81 |
| Friction Angle | $\beta$ | $^\circ$ | 45 | 60 | 56 |
| Initial Compressive Yield Strength | $\sigma_c^{iy}$ | $kPa$ | 1 | 100 | 52 |
| Peak Compressive Yield Strength | $\sigma_c^p$ | $MPa$ | 0.5 | 5 | 3.1 |
| Strain at Peak Compressive Yield | $\epsilon_c^p$ | % | 0.5 | 5 | 1.7 |
| Yield Strain at -0.5 Triaxiality | $\bar{\epsilon}_{f_{-0.5}}^{pl}$ | % | 0.01 | 0.1 | 0.0078 |
| Yield Strain at -0.6 Triaxiality | $\bar{\epsilon}_{f_{-0.6}}^{pl}$ | % | 0.1 | 10 | 0.30 |
| Plastic Displacement at Failure | $\bar{u}_f^{pl}$ | $m$ | 0.01 | 1 | 0.12 |

The stress-strain curves from the DEM simulations used for the parameter estimation and the stress-strain curves of the CDM simulations using the optimal parameter set are presented in Figure 4.1. The CDM fit is good with a Root-Mean-Square Error (RMSE) of $1.03MPa$ and the pressure dependent yield function works well with this model as the error is not biased to curves of a certain confining stress. This fit implies a strong likelihood that the model will be valid under confining stresses outside of the range fitted. Also, the damage initiation points at the peak of the curve are well correlated and indicate that the triaxiality based damage initiation criterion is a good model for this problem. The majority of the error in the curves is found in the post-yield behaviour. This error results from limitations in the continuum constitutive model because the post-yield behaviour of the DEM simulations is discontinuous in nature (stick-slip response). The CDM model cannot accommodate for such oscillations and thus represents the post-yield response as an average.

The optimal parameter set in Table 4.2 represents the constitutive response of the rock mass. As expected, the elastic modulus of the rock mass ($1.9GPa$) is substantially less

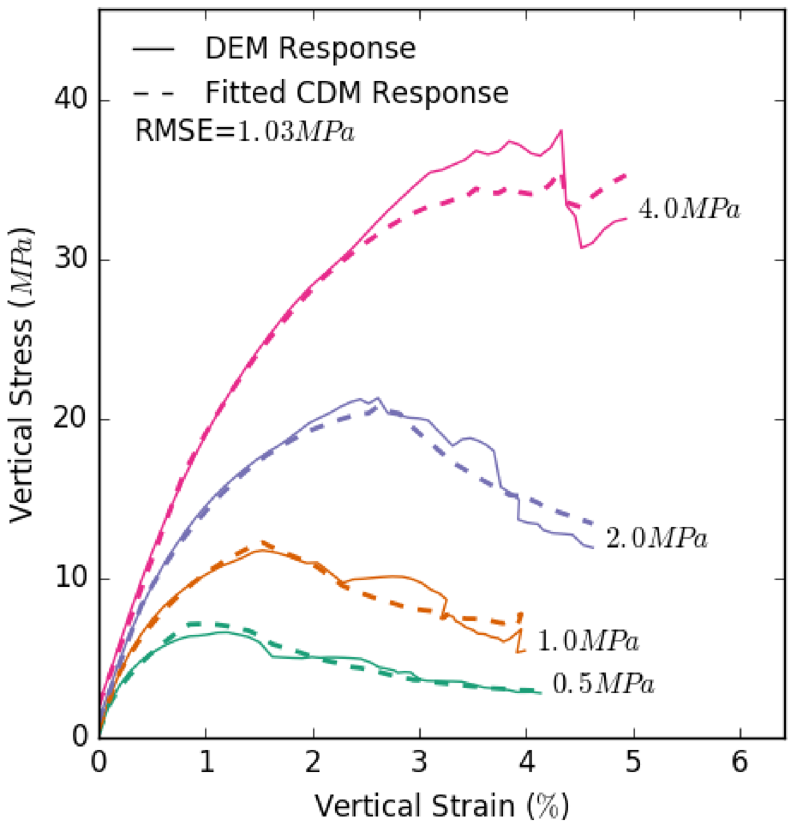


Figure 4.1: Axial Stress-Strain curves of the monotonically loaded DEM simulations used for estimating the Drucker-Prager CDM parameter set under different confining stresses.

than the elastic modulus of the intact rock ($65GPA$) because of yielding in the joints. Additionally, Poisson's ratio of the rock mass (0.15) is less than Poisson's ratio of the intact rock (0.2) because of the compliance of the joints before yield, which limits the lateral strain. After yielding however, substantial lateral strain is observed because of dilation of the joints, resulting in a large dilation angle ($22°$). This dilation response of the rock mass is larger than the the prescribed joint dilation ($5°$) because of block rotation and geometry.

There are additional minor sources of error from the homogenization algorithms that do not manifest themselves in this fitted relationship. In addition, if the REV is too small, it introduces its error in the DEM data rather than in the fitted response. Furthermore, the global fitting algorithms are not completely exhaustive, so it is possible they do not find the actual globally optimal parameter set, potentially leading to some error. With the given PSO parameters, up to 2400 sets of simulations are conducted for the global parameter estimation, and replicate optimization trials with different random seeds tend to give results within 1% deviation. This consistency and large search domain give confidence that the estimated parameter set is the globally optimal set.

In addition to the loading response under the specified confining stresses, DEM simulations under confining stresses of $3MPa$, $6MPa$, $8MPa$ and $10MPa$ are compared to the the CDM model using the previously estimated parameter set to see how well the constitutive behaviour is captured (Figure 4.2). These simulations demonstrate the interpolative ($3MPa$) and extrapolative ($6MPa$, $8MPa$ and $10MPa$) capacity of the fitted parameter set, and indeed a strong fit is obtained (RMSE of $2.83MPa$) for all confining stresses, with the error being more prominent for larger degrees of strain.

## 4.3 Comparison of CDM Constitutive Models

Here, the results from the damage plasticity model for quasi-brittle materials are compared to the results from the Drucker-Prager model with ductile damage presented above. Like with the Drucker-Prager model, a PSO algorithm followed by an LMA optimization is employed to fit the damage-plasticity model to the homogenized DEM simulation data in order to obtain an optimal parameter set. Again, 200 data points for all four DEM simulations are used and the PSO algorithm uses a swarm size of 24 for 100 generations. Additionally, parameter bounding limits are again required by the optimization algorithms in order to limit the search space. The specified bounding limits for each parameter results can be seen in Table 4.3.

The benefit of the damage plasticity model for quasi-brittle materials used here is its ability to differentiate between tensile and compressive damage, allowing for cyclic loading to be more accurately modelled. However, the fitted curves in Figure 4.3 show that the loading response is very poor for this material for larger confining stresses. The poor fit is attributed to deficiencies in the contitutive model rather than the the optimization algorithm becasue replicate optimization trials showed less than a 1% deviation. This behaviour is somewhat expected as the constitutive relationship is formulated primarily

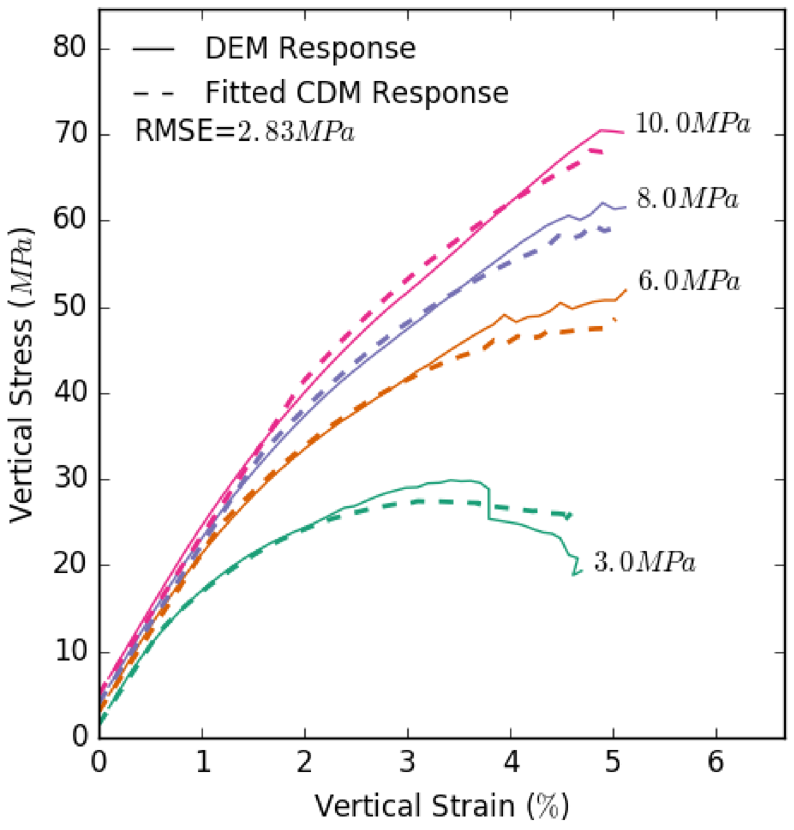


Figure 4.2: Axial Stress-Strain curves of the verification simulations for the fractured granite rock mass under different confining stresses for both the DEM simulations and the fitted Drucker-Prager CDM simulations.

with uniaxial loading conditions considered. Unfortunately, without a successful match on the loading curve, the cyclic modelling capacity of this model cannot be used effectively. The interpolated and extrapolated curves show an expectedly poor fit (Figure 4.4).

The optimal parameter set in Table 4.3 compares well in some respects to the optimal parameter set in Table 4.2. The estimated elastic modulus of the rock mass, compares well, but Poisson's ratio and dilation angle are quite different. Based on the fit and the RMSE of the two fitted parameter sets, the discrepancy in parameter values shown by the damage plasticity model can be attributed to the poor overall fit of the constitutive model for the dataset.

Table 4.3: Parameter estimation results for damage-plasticity model for quasi-brittle materials

| **Parameter** | **Symbol** | **Units** | **Lower Bound** | **Upper Bound** | **Optimum** |
|---|---|---|---|---|---|
| Young's Modulus | $E$ | $GPa$ | 1 | 25 | 1.4 |
| Poisson's Ratio | $\nu$ | | 0.1 | 0.4 | 0.32 |
| Dilation Angle | $\psi$ | $^\circ$ | 5 | 15 | 5 |
| Flow Eccentricity | $\varepsilon$ | | 0.05 | 0.5 | 0.11 |
| Second Stress Invariant Ratio | $K_c$ | | 0.51 | 1 | 0.61 |
| Initial Equibiaxial Stress Ratio | $f_0$ | $m$ | 1.05 | 1.3 | 1.1 |
| Initial Compressive Yield Strength | $\sigma_c^{iy}$ | $kPa$ | 1 | 100 | 28 |
| Peak Compressive Yield Strength | $\sigma_c^p$ | $MPa$ | 0.5 | 5 | 0.5 |
| Strain at Peak Compressive Yield | $\epsilon_c^p$ | % | 0.5 | 5 | 3.4 |
| Compressive Damage Scaling Factor | $d_c$ | | 0.4 | 0.95 | 0.85 |
| Tensile Damage Scaling Factor | $d_t$ | | 0.4 | 0.95 | 0.90 |

## 4.4 Impact of REV Size on Estimated Parameters

The appropriateness of the REV size was tested using eight different sample REV radii and running the homogenization and parameter estimation algorithms for each. The assumed REV radius for the parameter estimation simulations is $4m$, which corresponds to a homogenization area of $57.7m^2$. To validate this assumption, the REV radii is sampled at $0.5m$ intervals to see where the resultant parameters converge.

The convergence of three of the 11 parameters is shown in Figure 4.5 as a function of REV size. The material parameters apparently converge at different sizes, illustrating part of the challenge in defining an REV; some parameters require a larger REV than others and it is not obvious *a priori* which parameters will dominate. For the granite rock mass considered, an REV of radius $3m$ or with a homogenization area of $34.8m^2$ is chosen to be the minimum size based on the convergence of the dilation angle - the last parameter to converge. The suitability of the assumed REV size is confirmed since it is larger than the minimum REV size determined by the convergence study.

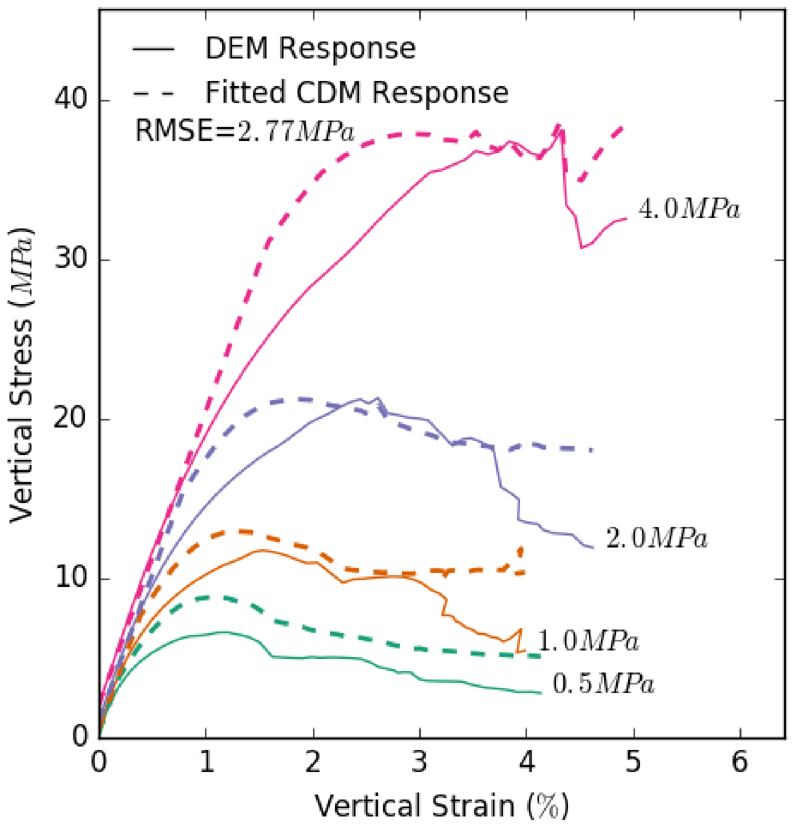


Figure 4.3: Axial Stress-Strain curves of the monotonically loaded DEM simulations used for estimating the CDM parameter set under different confining stresses using the damage-plasticity model for quasi-brittle materials.

## 4.5 Comparison to DNS - Application to Slope Stability Analysis

To validate the up-scaling methodology used, a simple 2-D slope problem is presented and loaded from the top until failure using both DEM and the up-scaled CDM model. Here, the resultant stress distributions are compared just as failure occurs.

In the DEM model, failure can be assessed based on the unbalanced forces in the model. Since the joints in the model have a stiffness and cohesion, when the slope fails, the explicit quasi-static solution becomes dynamic because of a sudden release of elastic energy and

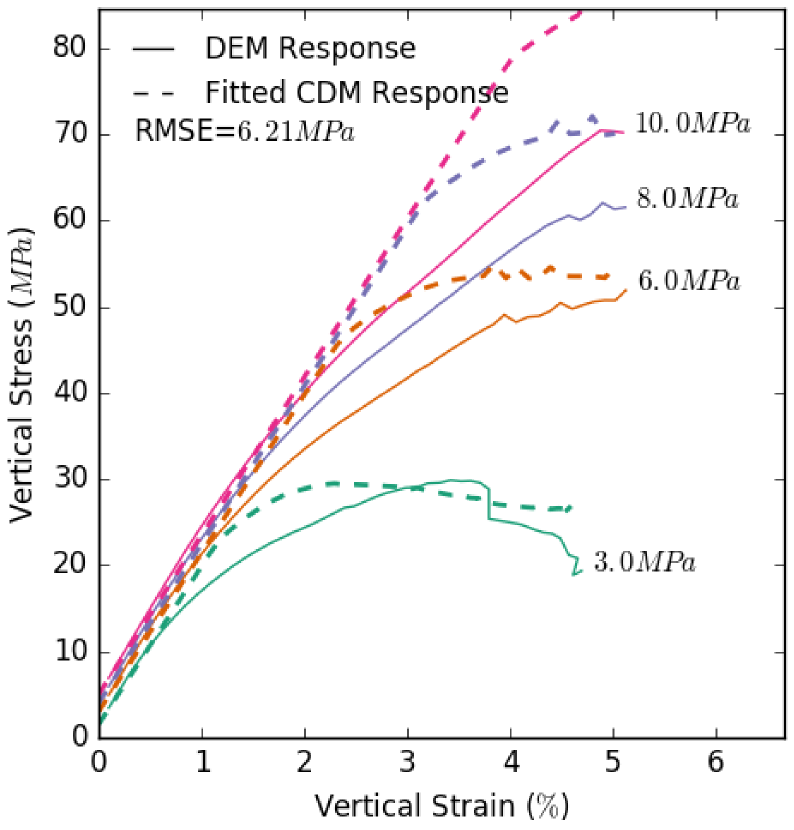


Figure 4.4: Axial Stress-Strain curves of the verification simulations using the damage-plasticity model for quasi-brittle materials under different confining stresses for both the DEM simulations and the fitted CDM simulations.

the inability of the applied damping to suppress it all. At this point, the total unbalanced forces in the model increase and the slope can be said to have failed.

For the CDM model, failure can be assessed based on non-convergence of the model when run as an implicit static simulation, which does not converge when the slope fails. The load step in which the CDM model fails to converge because the slope fails dynamically is considered the point of failure.

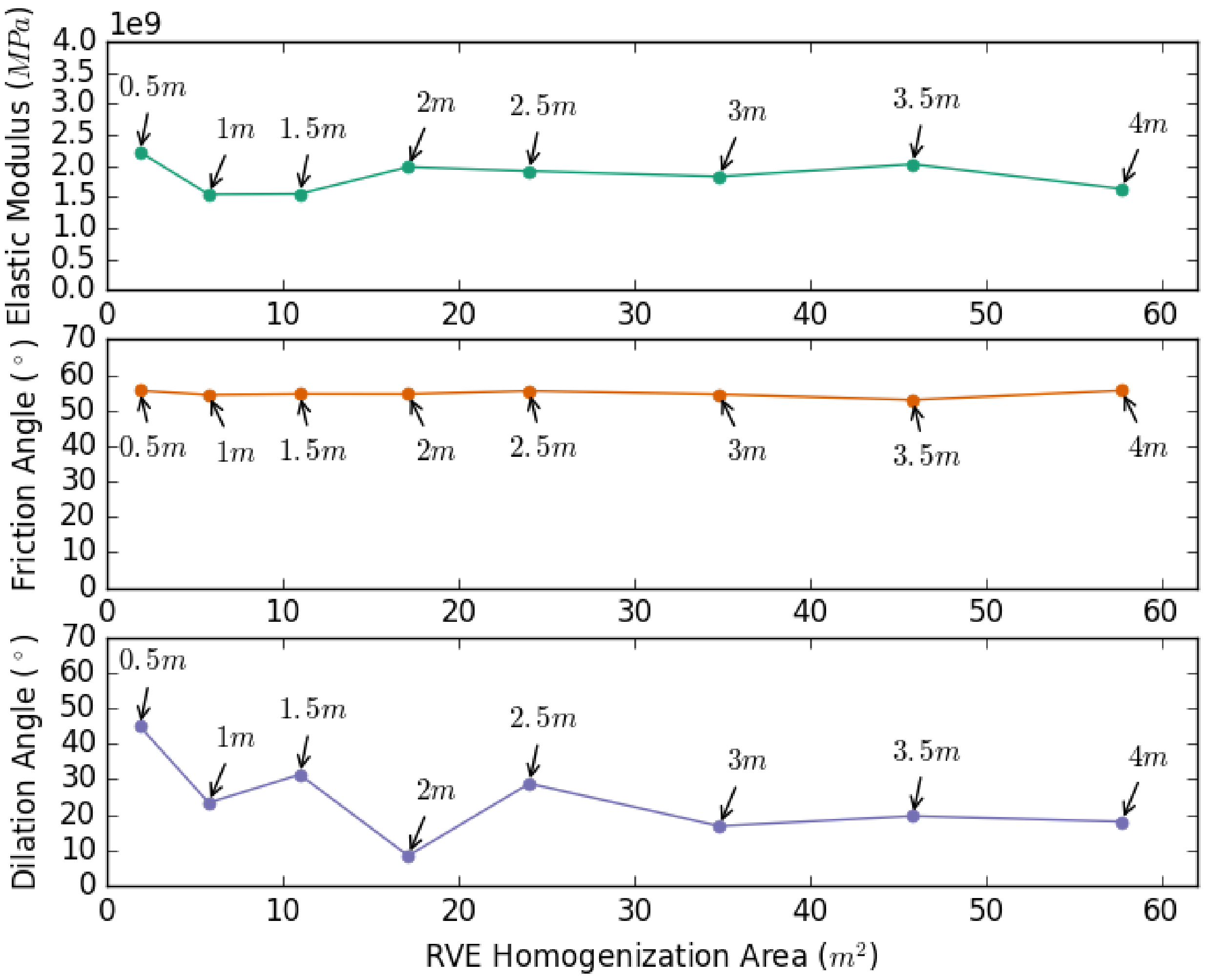


Figure 4.5: Convergence of three constitutive material parameters as the REV homogenization area is increased. Annotations indicate the specified radius of the circular REV.

### 4.5.1 Model Description

The plane-strain slope stability problem has a height of $50m$ and a depth of $80m$ with a $30m$ high slope with a grade of 300% (Figure 4.6). This geometry provides enough space for the failure mechanisms to occur with little influence from the boundaries. The lateral boundary conditions are zero displacement in the x-direction, and the bottom boundary conditions are zero displacement in the y-direction. The slope and top boundaries are free. A uniformly distributed load was applied over a 5m section on the backslope in a linear incremental fashion until failure.

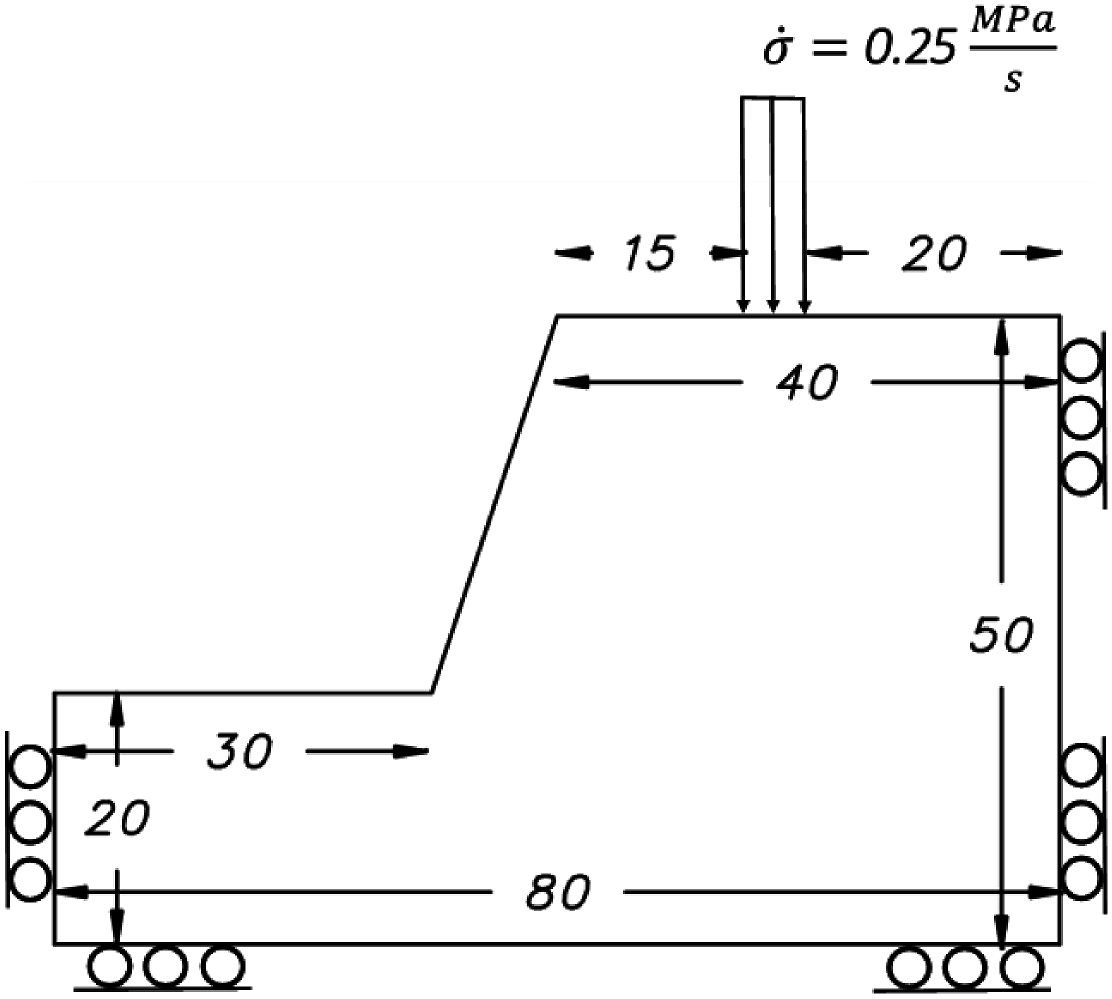


Figure 4.6: Schematic geometry and boundary conditions of the slope failure problem.

The meshing of the DEM simulations is identical to the REV simulations for the parameter estimation, while the CDM model is meshed using 4-node bilinear plane strain elements.

The models in both DEM and FEM are allowed to find a static equilibrium after the gravity force is applied, then a linearly increasing compressive stress along the top of the slope is applied until the slope fails. The load is increased from $0MPa$ to $25MPa$ over the course of $100s$ in the quasi-static/static simulations, knowing that the slope should fail long before $25MPa$ is reached.

### 4.5.2 DNS Comparison

A qualitative comparison of the DEM and the CDM model results uses the stress distribution and the surface deflection just before failure. For the DEM solution, since the stress field is discontinuous, the stress fields are smoothed using a cubic spline interpolation and subsequently run through a Gaussian filter with a standard deviation of 2 to reduce the noise in the data set. Figures 4.7, 4.8, and 4.9 show the horizontal stress distributions, the vertical stress distributions, and the shear stress distributions, respectively.

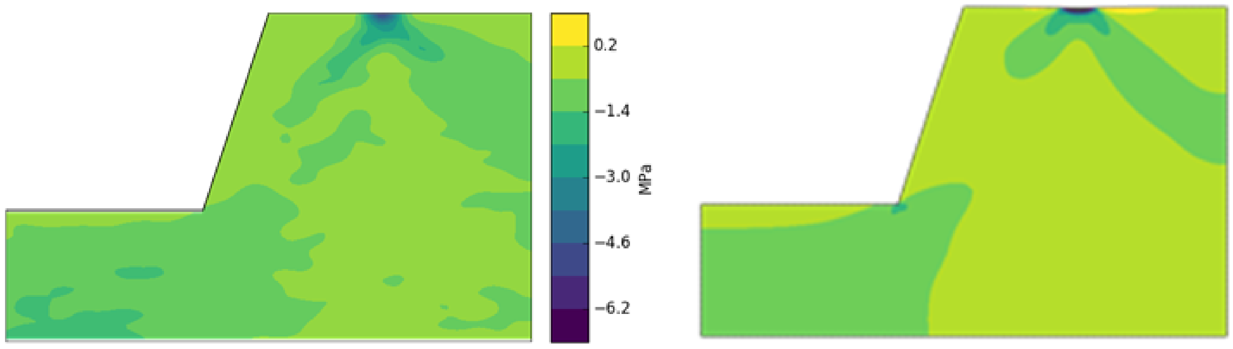


Figure 4.7: Comparison of DEM (left) and CDM (right) horizontal stress contours for the slope just before failure.

The continuum approximation of the stress fields shows a good match to the smoothed DEM stress fields. More importantly, the load at failure for the two models are quite close. The DEM simulation failed at $11.2MPa$, while the CDM simulation failed at $11.5MPa$, a 3% error considered to be not only negligible in the context of geological uncertainty but acceptable in terms of the computational savings. This agreement of the two models both in terms of the stress distribution and the failure load shows a high degree of success for the up-scaling framework.

An additional comparison of the surface deflection where the load was applied is presented in Figure 4.10. Again, the behaviour of the two models is similar, with downward displacement occurring where the load is applied, upwards displacement towards the slope on the

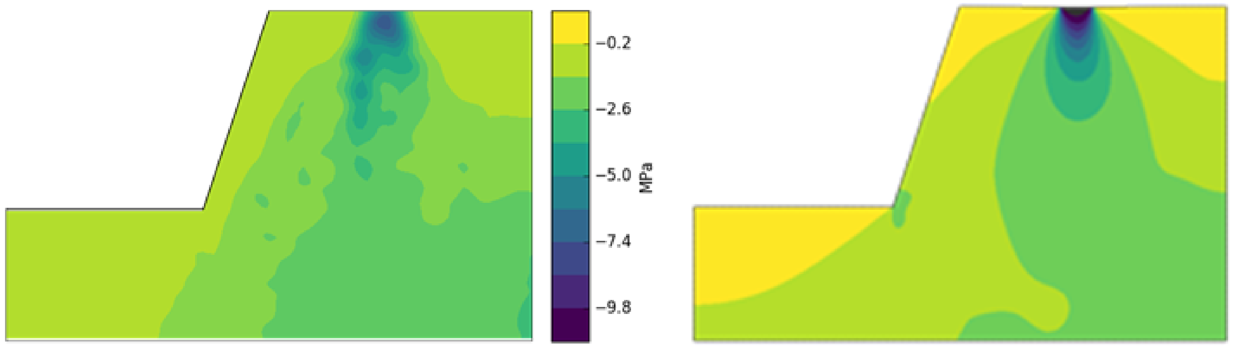


Figure 4.8: Comparison of DEM (left) and CDM (right) vertical stress contours for the slope just before failure.

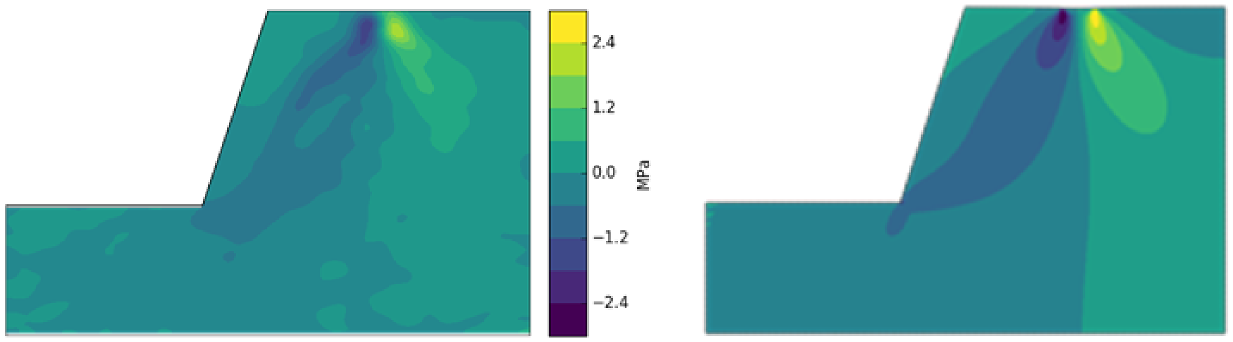


Figure 4.9: Comparison of DEM (left) and CDM (right) shear stress contours for the slope just before failure.

left and negligible displacement towards the right model boundary. Some divergence from the DEM results can be observed in the CDM approximation where sharp changes in the profile gradient occur; this arises partly from CDM model limitations and partly because the scale of the deviation is similar to the REV scale, which is the limiting case. For larger scales, the error will be smaller.

### 4.5.3 Up-Scaling Computational Efficiency

The CDM model for this case requires about two orders of magnitude less computational effort than the DEM model (Table 4.4). The CDM simulation uses a comparable number of continuum elements (29, 866) (Figure 4.12) as in the DEM simulation (25, 898) for

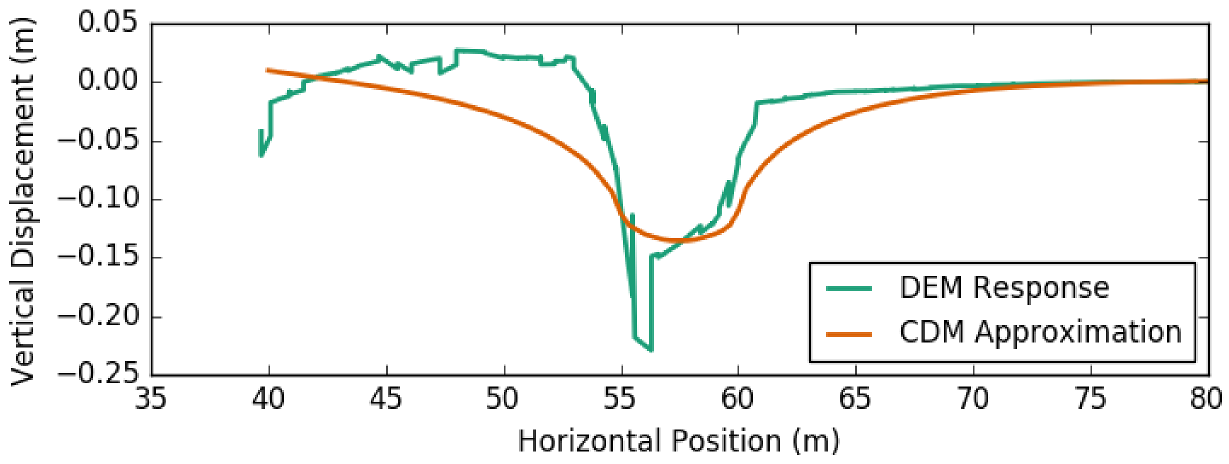


Figure 4.10: Comparison of DEM and CDM surface deflection profile for the slope just before failure.

comparison and adequate convergence. The CDM model efficiency can be improved by applying a Selectively Refined Mesh (SRM) (Figure 4.11) where only the areas with stress concentrations and large stress gradients have a strongly refined mesh. With the SRM, a converged CDM solution is achievable with only $3,577$ elements leading to another order of magnitude reduction in computational effort. The SRM is verified to be accurate by comparing the results to the fully converged solution with $29,866$ elements. To obtain the SRM, the number of elements is selectively reduced until an appreciable divergence from the actual solution is noted.

Table 4.4: Comparison of Computational Time for the DNS

| **Simulation Type** | **Continuum Elements** | **Processor Clock Speed** | **Slope Failure Load** | **Computational Time** |
|---|---|---|---|---|
| DEM | $25,898$ | $2.20GHz$ | $11.2MPa$ | $46.5hr$ |
| CDM | $29,866$ | $1.80GHz$ | $11.5MPa$ | $0.65hr$ |
| CDM - SRM | $3,577$ | $1.80GHz$ | $11.5MPa$ | $0.013hr$ |

The DEM simulation was run serially on a $2.2GHz$ CPU while the CDM simulation was run serially on a $1.8GHz$ CPU. Despite the CDM model having more continuum elements than the DEM model, and the CDM model running on a slower CPU, a decrease in computational time of the DEM simulation from $46.5hr$ to $0.65hr$ was observed. Running

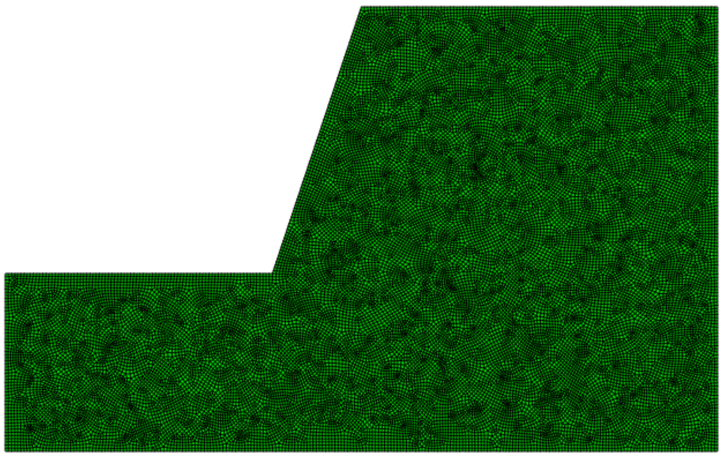


Figure 4.11: Mesh for converged Drucker-Prager CDM FEM slope failure simulation

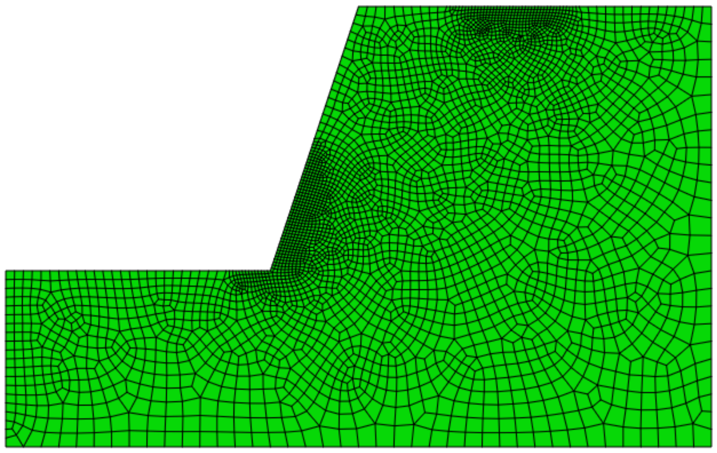


Figure 4.12: Selectively Refined Mesh (SRM) for Drucker-Prager CDM FEM slope failure simulation.

the CDM model with a SRM reduces the total computational time to $0.013hr$, or eight minutes instead of two days. This large increase in computational efficiency with marginal decrease in model accuracy can be immensely useful for large scale geomechanical problems in NFR.

# Chapter 5

# Conclusions and Future Considerations

A summary of the main conclusions from the development of the up-scaling framework as well as the implementation and testing of the framework are presented here. These conclusions represent a successful completion of the research objectives for the thesis. That being said, there are substantial limitations to this research. A series of recommendations are provided to address some of these limitations and to provide guidance on how to extend this research.

## 5.1 Conclusions

A multi-scale framework for up-scaling DEM simulations has been developed to address the computational demands of simulating microscale phenomenon in a macroscale domain in the context of NFR. Up-scaling is achieved by matching homogenized stress-strain curves from REV-scale DEM simulations to single element continuum models using PSO and LMA optimization algorithms. A Drucker-Prager plasticity model with ductile damage is implemented in the CDM model to empirically capture the effect of the degradation (damage) of the NFR as deformation takes place.

## 1. Deformable DEM Homogenization

Homogenization algorithms were developed for homogenizing DEM simulations with deformable bodies to assess the spatially averaged stress-strain behaviour of the REV from the microscale displacements, strains, and stresses. In this homogenization process, the resultant inter-block contact forces and block displacement from the DEM simulations are converted to average stresses and strains. To apply the homogenization algorithms, a method of automatically assessing a suitable REV given a sufficiently large domain was developed. These algorithms were implemented in Python$^{\text{TM}}$ as the HODS software, which was used as a module for MOUSE.

## 2. Parameterization Methodology

Two examples of the parameterization methodology are presented. Here, the key parameters required to capture the salient features of the model are isolated in order to be able to run the parameter estimation algorithms effectively. the main aspect of this parameterization methodology is the functional assumptions of the hardening/softening and damage evolution functions. In addition, the parameters are rewritten in term of physically meaningful parameters to provide more insight into the mechanics. The Drucker-Prager model with ductile damage is shown to be a reasonable CDM model approach to represent NFR in a continuum context, including effects of pressure dependent yield and the triaxiality based damage initiation criterion. Compared to a full DEM simulation, the CDM model shows a good fit pre-damage, but is unable to emulate the subtle post-yield oscillations arising from non-continuous yielding in the NFR.

## 3. Up-Scaling Framework

MOUSE software was created and written in Python$^{\text{TM}}$ to provide an implementation of the up-scaling framework presented in this thesis using in house and third-party software modules. The software itself provides a platform through which the four software elements of the up-scaling framework (DEM module, homogenization module, parameter estimation module, and macroscale module) can communicate with each other. The communication is facilitated by MOUSE through modules which wrap the third party software in such a way that the I/O routines to and from the modules are performed in a consistent capacity regardless of the third party software being used. A consistent set of data protocols were developed for the modules to effectively transfer data between them.

**4. Framework Verification**

The parameter estimation module was tested and yielded an appropriate parameter set that both matched the DEM data and provided realistic parameters. Additionally, the Drucker-Prager model with ductile damage was found to provide a far superior fit than the damage plasticity model for quasi-brittle materials. Most importantly, the DNS of the slope stability analysis showed that with this up-scaling framework, very significant computational gains can be had with an acceptable error. Very comparable results ($< 5\%$ error) to full DEM solutions were obtained with the CDM method but required two orders of magnitude less computational time. The computational demands were again able to be reduced by another order of magnitude by using a selectively refined continuum mesh at the locations in the domain with high stress gradients.

## 5.2 Recommendations

The main limitation of the presented up-scaling implementation is the macroscale constitutive model module. In the current ABAQUS$^{TM}$ module, the constitutive models consider only isotropic behaviour. In addition, the model does not consider the effects of pore pressure in the rock mass or fluid flow in any capacity. Though the isotropic assumptions for the elasto-plastic constitutive relationships are likely sufficiently accurate, future implementations of the macroscale constitutive model should consider anisotropic damage behaviour, as anisotropic implementations are found to be completely insufficient. In the case of the Drucker-Prager model with ductile damage, the exponential Johnson-Cook triaxiality based damage initiation criterion provides an excellent fit for monotonic loading, but does not model cyclic loading well. Here, it would be ideal for the cyclic loading capacity of the damage plasticity model for quasi-brittle materials to be incorporated as well. Ultimately, the available damage material subroutines in ABAQUS$^{TM}$ are insufficient for the key physical characteristics in the system to be captured. As such, a custom anisotropic damage implementation is recommended for the macroscale constitutive model.

Retrospectively, the functional form of the hardening curve is overly complex. Though it is often necessary to model the softening of the material in the plasticity model, with CDM the damage can implicitly model the softening behaviour. Here, it is observed that for the Barcelona model used for the hardening/softening curve, only the hardening portion of the curve is ever used. As such, for future implementations of the plasticity hardening functions, a simpler exponential function could be applied which would also have the benefit of decreasing the number of parameters that need to be estimated, leading to more consistent solutions and faster convergence of the optimization algorithms.

Furthermore, it is speculated by the author that portions of the parameterization methodology could be modified to yield more consistent solutions. In the parameterization formulations presented in this thesis, too much emphasis was placed on creating physically meaningful parameters rather than numerically consistent parameters. This inconsistency is the case with certain paired parameters if one of the parameters is highly sensitive.

Additionally, effectively searching a 11+ dimensional parameter space is computationally expensive. By dividing the problem and exploiting features of the curves and constitutive models, it may be possible to increase the convergence rate and effectively get a better, faster solution. Instead of searching the entire parameter space, it is possible to split the parameters into groups (e.g. pre-damage and post damage). Here, the damage parameters don't actually affect the plasticity calculations until damage is initiated. As such, the plasticity parameters can be estimated using the pre-damage curve and the damage parameters can subsequently be estimated using the post-damage curve.

A more rigorous examination of other available optimization routines and associated optimization parameters would be another way to potentially reduce the computational cost and increase the accuracy of the parameter estimation process. PSO and LMA were used in this research, but Dynamically Dimensioned Search (DDS), Real-Coded Genetic Algorithm (RGA) and Simulated Annealing (SA) were also briefly investigated. As previously stated, rigorously assessing the most effective algorithm was not a priority of this research, but the PSO + LMA combination was chosen for its simplicity and effectiveness. The other algorithms, if properly applied may provide a faster and more accurate solution.

For this up-scaling methodology to be more accurate, 3D DEM simulations are required to capture accurate physical responses of these complex systems. In addition to modifying the DEM simulations, the associated homogenization algorithms would have to be modified to provide the 3D stress and strain tensors.

Hydro-mechanically coupled DEM simulations are also an important consideration for more accurate simulations. Here, the homogenization module should be modified to assess the homogenized fluid properties such as pore pressure and flow velocity vectors and the macroscale model needs to be modified to simulate poroelastic physics.

Ultimately, rigorous validation studies should be done with real-world applications to show the viability of this approach. Incorporating some of the above reccomendations will allow this up-scaling framework to validly be applied to complex geomechanical problems in order to avoid the computational demands of DEM modelling.

# References

S. I. Aanonsen and D. Eydinov. A multiscale method for distributed parameter estimation with application to reservoir history matching. *Comput Geosci*, 10(1):97–117, mar 2006. doi: 10.1007/s10596-005-9012-4. URL http://dx.doi.org/10.1007/s10596-005-9012-4.

R. E. Barbosa and J. Ghaboussi. Discrete finite element method for multiple deformable bodies. *Finite Elements in Analysis and Design*, 7(2):145–158, nov 1990. doi: 10.1016/0168-874x(90)90006-z. URL http://dx.doi.org/10.1016/0168-874x(90)90006-z.

N. Barton. *Rock Quality Seismic Velocity, Attenuation and Anisotropy.* Informa UK Limited, oct 2006. doi: 10.1201/9780203964453. URL http://dx.doi.org/10.1201/9780203964453.

T. Belytschko, S. Loehnert, and J.-H. Song. Multiscale aggregating discontinuities: A method for circumventing loss of material stability. *International Journal for Numerical Methods in Engineering*, 73(6):869–894, feb 2008. doi: 10.1002/nme.2156. URL http://dx.doi.org/10.1002/nme.2156.

M. N. Bidgoli, Z. Zhao, and L. Jing. Numerical evaluation of strength and deformability of fractured rocks. *Journal of Rock Mechanics and Geotechnical Engineering*, 5(6):419–430, dec 2013. doi: 10.1016/j.jrmge.2013.09.002. URL http://dx.doi.org/10.1016/j.jrmge.2013.09.002.

S. Chen, J. He, and I. Shahrour. Estimation of elastic compliance matrix for fractured rock masses by composite element method. *International Journal of Rock Mechanics and Mining Sciences*, 49:156–164, jan 2012. doi: 10.1016/j.ijrmms.2011.11.009. URL http://dx.doi.org/10.1016/j.ijrmms.2011.11.009.

M. Clerc and J. Kennedy. The particle swarm - explosion stability, and convergence in a multidimensional complex space. *IEEE Transactions on Evolutionary Computation*, 6 (1):58–73, 2002. doi: 10.1109/4235.985692. URL http://dx.doi.org/10.1109/4235.985692.

R. M. Corless, G. H. Gonnet, D. E. G. Hare, D. J. Jeffrey, and D. E. Knuth. On the LambertW function. *Advances in Computational Mathematics*, 5(1):329–359, dec 1996. doi: 10.1007/bf02124750. URL http://dx.doi.org/10.1007/bf02124750.

P. A. Cundall. A discontinuous future for numerical modelling in geomechanics? *Proceedings of the ICE - Geotechnical Engineering*, 149(1):41–47, jan 2001. doi: 10.1680/geng.2001.149.1.41. URL http://dx.doi.org/10.1680/geng.2001.149.1.41.

P. A. Cundall and R. D. Hart. Numerical Modelling of Discontinua. *Engineering Computations*, 9(2):101–113, feb 1992. doi: 10.1108/eb023851. URL http://dx.doi.org/10.1108/eb023851.

P. A. Cundall and O. D. L. Strack. A discrete numerical model for granular assemblies. *Géotechnique*, 29(1):47–65, mar 1979. doi: 10.1680/geot.1979.29.1.47. URL http://dx.doi.org/10.1680/geot.1979.29.1.47.

G. D'Addetta, E. Ramm, S. Diebels, and W. Ehlers. A particle center based homogenization strategy for granular assemblies. *Engineering Computations*, 21(2/3/4):360–383, mar 2004. ISSN 0264-4401. doi: 10.1108/02644400410519839. URL http://www.emeraldinsight.com/doi/abs/10.1108/02644400410519839.

B. M. Das. *Principles of geotechnical engineering*. Cengage, Mason, OH, 7th ed edition, 2009. ISBN 0-495-41130-2.

D. C. Drucker. Some Implications of Work Hardening and Ideal Plasticity. *Quarterly of Applied Mathematics*, 7(4):411–418, 1950. ISSN 0033-569X. URL http://www.jstor.org/stable/43633751.

W. Drugan and J. Willis. A micromechanics-based nonlocal constitutive equation and estimates of representative volume element size for elastic composites. *Journal of the Mechanics and Physics of Solids*, 44(4):497–524, apr 1996. doi: 10.1016/0022-5096(96)00007-5. URL http://dx.doi.org/10.1016/0022-5096(96)00007-5.

F. Feyel. A multilevel finite element method (FE2) to describe the response of highly nonlinear structures using generalized continua. *Computer Methods in Applied Mechanics and Engineering*, 192(28-30):3233–3244, jul 2003. doi: 10.1016/s0045-7825(03)00348-7. URL http://dx.doi.org/10.1016/s0045-7825(03)00348-7.

M. Geers, V. Kouznetsova, and W. Brekelmans. Multi-scale computational homogenization: Trends and challenges. *Journal of Computational and Applied Mathematics*, 234 (7):2175–2182, aug 2010. doi: 10.1016/j.cam.2009.08.077. URL http://dx.doi.org/10.1016/j.cam.2009.08.077.

I. Gitman, H. Askes, and L. Sluys. Representative volume: Existence and size determination. *Engineering Fracture Mechanics*, 74(16):2518–2534, nov 2007. doi: 10.1016/j.engfracmech.2006.12.021. URL http://dx.doi.org/10.1016/j.engfracmech.2006.12.021.

I. M. Gitman, M. B. Gitman, and H. Askes. Quantification of stochastically stable representative volumes for random heterogeneous materials. *Archive of Applied Mechanics*, 75(2-3):79–92, dec 2005. doi: 10.1007/s00419-005-0411-8. URL http://dx.doi.org/10.1007/s00419-005-0411-8.

R. Gracie and T. Belytschko. An adaptive concurrent multiscale method for the dynamic simulation of dislocations. *International Journal for Numerical Methods in Engineering*, 86(4-5):575–597, feb 2011. doi: 10.1002/nme.3112. URL http://dx.doi.org/10.1002/nme.3112.

D. Griffiths and P. Lane. Slope Stability analysis by Finite Elments. *Géotechnique*, 49: 387, 1999.

A. A. Gusev. Representative volume element size for elastic composites: A numerical study. *Journal of the Mechanics and Physics of Solids*, 45(9):1449–1459, sep 1997. doi: 10.1016/s0022-5096(97)00016-1. URL http://dx.doi.org/10.1016/s0022-5096(97)00016-1.

R. Hill. Elastic properties of reinforced solids: some theoretical principles. *Journal of the Mechanics and Physics of Solids*, 11(5):357–372, 1963. URL http://www.sciencedirect.com/science/article/pii/002250966390036X.

A. Hillerborg, M. Modéer, and P.-E. Petersson. Analysis of crack formation and crack growth in concrete by means of fracture mechanics and finite elements. *Cement and Concrete Research*, 6(6):773–781, nov 1976. doi: 10.1016/0008-8846(76)90007-7. URL http://dx.doi.org/10.1016/0008-8846(76)90007-7.

E. Hoek and E. Brown. Practical estimates of rock mass strength. *International Journal of Rock Mechanics and Mining Sciences*, 34(8):1165–1186, dec 1997. doi: 10.1016/s1365-1609(97)80069-x. URL http://dx.doi.org/10.1016/s1365-1609(97)80069-x.

L. Jing. A review of techniques, advances and outstanding issues in numerical modelling for rock mechanics and rock engineering. *International Journal of Rock Mechanics and Mining Sciences*, 40(3):283–353, apr 2003. ISSN 13651609. doi: 10.1016/S1365-1609(03)00013-3. URL http://linkinghub.elsevier.com/retrieve/pii/S1365160903000133.

G. R. Johnson and W. H. Cook. Fracture characteristics of three metals subjected to various strains strain rates, temperatures and pressures. *Engineering Fracture Mechanics*, 21(1): 31–48, jan 1985a. doi: 10.1016/0013-7944(85)90052-9. URL http://dx.doi.org/10.1016/0013-7944(85)90052-9.

G. R. Johnson and W. H. Cook. Fracture characteristics of three metals subjected to various strains, strain rates, temperatures and pressures. *Engineering Fracture Mechanics*, 21 (1):31–48, jan 1985b. ISSN 0013-7944. doi: 10.1016/0013-7944(85)90052-9. URL http://www.sciencedirect.com/science/article/pii/0013794485900529.

T. Kanit, S. Forest, I. Galliet, V. Mounoury, and D. Jeulin. Determination of the size of the representative volume element for random composites: statistical and numerical approach. *International Journal of Solids and Structures*, 40(13-14):3647–3679, jun 2003. doi: 10.1016/s0020-7683(03)00143-4. URL http://dx.doi.org/10.1016/s0020-7683(03)00143-4.

J. Kennedy and R. Eberhart. Particle swarm optimization. In *Proceedings of ICNN'95 - International Conference on Neural Networks*. Institute of Electrical & Electronics Engineers (IEEE), 1995. doi: 10.1109/icnn.1995.488968. URL http://dx.doi.org/10.1109/icnn.1995.488968.

D. Krajcinovic. Damage mechanics. *Mechanics of materials*, 8(2):117–197, 1989. URL http://www.sciencedirect.com/science/article/pii/0167663689900112.

J. Lee and G. L. Fenves. Plastic-Damage Model for Cyclic Loading of Concrete Structures. *Journal of Engineering Mechanics*, 124(8):892–900, 1998. ISSN 0733-9399. doi: 10.1061/(ASCE)0733-9399(1998)124:8(892). URL http://dx.doi.org/10.1061/(ASCE)0733-9399(1998)124:8(892).

K. Levenberg. A method for the solution of certain non–linear problems in least squares. In *Meeting of the American Math Society in Chicago*, 1944. URL http://en.journals.sid.ir/ViewPaper.aspx?ID=53617.

X. Li, Y. Liang, Q. Duan, B. Schrefler, and Y. Du. A mixed finite element procedure of gradient Cosserat continuum for second-order computational homogenisation of granular materials. *Computational Mechanics*, 54(5):1331–1356, jul 2014. doi: 10.1007/s00466-014-1062-9. URL http://dx.doi.org/10.1007/s00466-014-1062-9.

S. Lloyd. Least squares quantization in PCM. *IEEE Trans. Inform. Theory*, 28(2):129–137, mar 1982. doi: 10.1109/tit.1982.1056489. URL http://dx.doi.org/10.1109/tit.1982.1056489.

S. Loehnert and P. Wriggers. Aspects of computational homogenisation of microheterogeneous materials including decohesion at finite strains. *Proc. Appl. Math. Mech.*, 5 (1):427–428, dec 2005. doi: 10.1002/pamm.200510190. URL http://dx.doi.org/10.1002/pamm.200510190.

J. Lubliner, J. Oliver, S. Oller, and E. Onate. A plastic-damage model for concrete. *International Journal of solids and structures*, 25(3):299–326, 1989. URL http://www.sciencedirect.com/science/article/pii/0020768389900504.

D. W. Marquardt. An algorithm for least-squares estimation of nonlinear parameters. *Journal of the Society for Industrial & Applied Mathematics*, 11(2):431–441, 1963. URL http://epubs.siam.org/doi/pdf/10.1137/0111030.

L. S. Matott. OSTRICH: An Optimization Software Tool; Documentation and User's Guide, 2008. URL http://www.eng.buffalo.edu/~lsmatott/Ostrich/OstrichMain.html.

L. S. Matott. OSTRICH: An Optimization Software Tool; Documentation and User's Guide, 2016. URL http://www.eng.buffalo.edu/~lsmatott/Ostrich/OstrichMain.html.

K.-B. Min and L. Jing. Numerical determination of the equivalent elastic compliance tensor for fractured rock masses using the distinct element method. *International Journal of Rock Mechanics and Mining Sciences*, 40(6):795–816, sep 2003. doi: 10.1016/s1365-1609(03)00038-8. URL http://dx.doi.org/10.1016/s1365-1609(03)00038-8.

C. Müller, S. Siegesmund, and P. Blum. Evaluation of the representative elementary volume (REV) of a fractured geothermal sandstone reservoir. *Environ Earth Sci*, 61(8):1713–1724, mar 2010. doi: 10.1007/s12665-010-0485-7. URL http://dx.doi.org/10.1007/s12665-010-0485-7.

R. Nelson. Detecting and Predicting Fracture Occurrence and Intensity. In *Geologic Analysis of Naturally Fractured Reservoirs*, pages 125–162. Elsevier BV, 2001. doi: 10.1016/b978-088415317-7/50006-3. URL http://dx.doi.org/10.1016/b978-088415317-7/50006-3.

M. Petracca, L. Pelà, R. Rossi, S. Oller, G. Camata, and E. Spacone. Regularization of first order computational homogenization for multiscale analysis of masonry structures. *Computational Mechanics*, 57(2):257–276, dec 2015. doi: 10.1007/s00466-015-1230-6. URL http://dx.doi.org/10.1007/s00466-015-1230-6.

A. Prantl, J. Ruzicka, M. Spaniel, M. Moravec, J. Dzugan, and P. Konopík. Identification of Ductile Damage Parameters. In *SIMULIA Community Conference*, 2013.

Y. Shi and R. Eberhart. A modified particle swarm optimizer. In *1998 IEEE International Conference on Evolutionary Computation Proceedings. IEEE World Congress on Computational Intelligence (Cat. No.98TH8360)*. Institute of Electrical & Electronics Engineers (IEEE), 1998. doi: 10.1109/icec.1998.699146. URL http://dx.doi.org/10.1109/icec.1998.699146.

B. Sjogren, A. Ofsthus, and J. Sandberg. Seismic Classification of Rock Mass Qualities. *Geophysical Prospecting*, 27(2):409–442, jun 1979. doi: 10.1111/j.1365-2478.1979.tb00977.x. URL http://dx.doi.org/10.1111/j.1365-2478.1979.tb00977.x.

I. Temizer and P. Wriggers. A Contact Homogenization Framework for Granular Interfaces. *Proc. Appl. Math. Mech.*, 9(1):417–418, dec 2009. doi: 10.1002/pamm.200910182. URL http://dx.doi.org/10.1002/pamm.200910182.

S. Thallak, L. Rothenburg, M. Dusseault, and R. Bathurst. Numerical Simulation of Hydraulic Fracturing in a Discrete Element System. In *ECMOR II - 2nd European Conference on the Mathematics of Oil Recovery*. EAGE Publications, sep 1990. doi: 10.3997/2214-4609.201411126. URL http://dx.doi.org/10.3997/2214-4609.201411126.

G. Venter and J. Sobieszczanski-Sobieski. Parallel Particle Swarm Optimization Algorithm Accelerated by Asynchronous Evaluations. *Journal of Aerospace Computing Information, and Communication*, 3(3):123–137, mar 2006. doi: 10.2514/1.17873. URL http://dx.doi.org/10.2514/1.17873.

B. L. Wahalathantri, D. P. Thambiratnam, T. H. T. Chan, and S. Fawzia. A material model for flexural crack simulation in reinforced concrete elements using ABAQUS. In *Proceedings of the First International Conference on Engineering, Designing and Developing the Built Environment for Sustainable Wellbeing*, pages 260–264. Queensland University of Technology, 2011. URL http://eprints.qut.edu.au/41712.

R. E. Walpole, editor. *Probability & statistics for engineers & scientists*. Pearson Prentice Hall, Upper Saddle River, NJ, 8th ed edition, 2007. ISBN 978-0-13-204767-8.

E. Weinan. *Principles of multiscale modeling*. Cambridge University Press, 2011.

C. Wellmann, C. Lillie, and P. Wriggers. Homogenization of granular material modeled by a three-dimensional discrete element method. *Computers and Geotechnics*, 35(3): 394–405, may 2008. ISSN 0266352X. doi: 10.1016/j.compgeo.2007.06.010. URL http://linkinghub.elsevier.com/retrieve/pii/S0266352X07000754.

M. Xu, R. Gracie, and T. Belytschko. Multicale Modeling with Extended Bridging Domain Method. In *Multiscale modeling with extended bridging domain method*. Oxford University Press, 2002. URL http://www.civil.uwaterloo.ca/rgracie/papers/2009/(2009a%20Gracie)%20Concurrent%20Coupling%20of%20Atomistic%20and%20Contiuum%20Models.pdf.

W. Zhang and Y. Cai. *Continuum Damage Mechanics and Numerical Applications*. Advanced Topics in Science and Technology in China. Zhejiang Univeristy Press, Hangzhou, 2010. ISBN 978-7-308-06589-4 978-3-642-04707-7.

Q. Zhou, H.-H. Liu, G. S. Bodvarsson, and C. M. Oldenburg. Flow and transport in unsaturated fractured rock: effects of multiscale heterogeneity of hydrogeologic properties. *Journal of Contaminant Hydrology*, 60:1–30, jan 2003. ISSN 0169-7722. doi: 10.1016/S0169-7722(02)00080-3. URL http://www.sciencedirect.com/science/article/pii/S0169772202000803.

D. Zwillinger. *CRC Standard Mathematical Tables and Formulae 30th Edition*. Informa UK Limited, dec 1995. doi: 10.1201/noe0849324796. URL http://dx.doi.org/10.1201/noe0849324796.